

0 0 W
A :
1
KTJ11-B DIAGNOSTIC

MACRO Y05.02 Friday 29-Mar-85 12:53 Page 2

.REM 6

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43

IDENTIFICATION

PRODUCT CODE: AC T875C-MC
PRODUCT NAME: COKTACO KTJ11 B DIAGNOSTIC
PRODUCT DATE: MARCH, 1985
MAINTAINER: SMALL SYSTEMS DIAGNOSTIC ENGINEERING

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

NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS AFFILIATED COMPANIES.

COPYRIGHT (C): 1984, 1985 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	DECX/11

45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

TABLE OF CONTENTS

- 1. ABSTRACT
- 2. RUN-TIME REQUIREMENTS
- 3. STARTING PROCEDURE
- 4. ERROR REPORTS
- 5. EXECUTION TIME
- 6. UBA REGISTER DEFINITIONS
- 7. TEST LIST

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

1. ABSTRACT

The following diagnostic tests the Unibus Adapter (UBA) module, KTJ11-B. The functionality of the module is: Unibus - PMI bus adapter (where PMI is a faster version of a Q22-bus), M9312 compatible boot facility, and the Unibus Map logic. The module also has a DMA cache store, utilised for doing DMA transfers from memory to Unibus devices. The UBA can be programmed to do diagnostic cycles to verify some of the functionality without requiring any of the peripherals to be actually connected to the Unibus.

2. RUN-TIME REQUIREMENTS

This diagnostic is the only one written specifically for the UBA module. Therefore, depending on the environment in which the program is run, different devices can be used.

Minimum hardware needed to run the diagnostic:

- 1) KDJ11-B CPU module
- 2) at least 28K of memory
- 3) KTJ11-B UBA module
- 4) console terminal
- 5) load media

To do further functional verification of the module 2 Unibus Exercisers (UBE) are required. This should be done in manufacturing or any other environment that needs verification of ALL functions of the UBA.

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

3. STARTING PROCEDURE

The diagnostic is a standart XXDP program with APT interface. Therefore, in stanalone mode, after booting the system, typing in:

R OKTAB?

will start the program, which will then prompt the operator for the software register switch setting desribed below.

COKTABO KTJ11-B DIAGNOSTIC

SWR = XXXXXX NEW =

Where "XXXXXX" correspond to the old setting of the software switch register. At this point an operator can either type in a carriage return, which would leave the software switch register as it was, or change it, according to the following parameters.

OPERATIONAL SWITCH SETTINGS

THE SWITCH SETTINGS ARE:

	OCTAL	MEANING
	--	----
SW<15>=1...	100000	HALT ON ERROR
SW<14>=1...	40000	LOOP ON CURRENT TEST
SW<13>=1...	20000	INHIBIT ERROR TYPEOUTS
SW<11>=1...	4000	INHIBIT ITERATIONS
SW<10>=1...	2000	RING TTY BELL ON ERROR
SW<9>=1...	1000	LOOP ON ERROR
SW<8>=1...	400	LOOP ON TEST SPECIFIED IN SW<5> THRU SW<0>

For example:

SWR = 000000 NEW = 100000

In this case the old software switch register didn't set any flags. The new setting will make the diagnostic halt on error.

If ran from a UFD chain file, the diagnostic is fully under control of the UFD monitor that will report only PASS/FAIL messages.

146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177

4. ERROR REPORTS

If ran in standalone mode, the diagnostic reports all errors to a functional level and then continues. Failing program counter, test number, and error number are printed out for all errors. Where possible expected and received data are also provided. Refer to OPERATIONAL SWITCH SETTINGS if anything different is required.

Example of error printout:

```
ERROR IN THE M9312 BOOT ROM SECTION
TEST   ERROR   ERROR
#      PC      #
33    12650    31
```

5. EXECUTION TIME

The diagnostic runs a full pass in less than a minute.

6. UBA REGISTER DEFINITION

DDR The Diagnostic Data register is a buffer that provides the ability of reading and writing the data to and from memory in diagnostic mode.

DCSR The Diagnostic Control and Status register provides means of going in and out of standalone mode and also of initiating diagnostic DATA from memory cycles.

KMCR The KTJ11-B Memory Configuration register identifies the amount of Unibus memory present in the system. It is also responsible for controlling and describing the status of the DMA cache.

179	7.	TESTS LIST
180		
181		TEST 1 - UNIBUS MAP REGISTER TESTS
182		TEST 2 - UNIBUS MAP REGISTER BIT PATTERN
183		TEST 3 - UNIBUS MAP REGISTER ADDRESS UNIQUENESS
184		TEST 4 - UNIQUENESS BETWEEN LOW MAP REGISTERS AND HIGH MAP REGISTERS
185		TEST 5 - DCSR REGISTER RESPONSE TEST
186		TEST 6 - KMCR BITS TEST
187		TEST 7 - UNIBUS TIMEOUT TEST
188		TEST 8 - DATA OUT WITHOUT RELOCATION
189		TEST 9 - DATA IN WITHOUT RELOCATION
190		TEST 10 - CONTENT OF DDR
191		TEST 11 - INDIRECT ACCESSING OF UNIBUS MAP REGISTERS
192		TEST 12 - DISABLING OF THE MAPPING REGISTERS
193		TEST 13 - NXM MEMORY TIMEOUT
194		TEST 14 - CARRY PROPOGATION TEST
195		TEST 15 - EXTENSIVE CARRY PROPOGATION TEST
196		TEST 16 - ALU TEST
197		TEST 17 - MAIN MEMORY DISABLE
198		TEST 18 - CACHE PRESENCE
199		TEST 19 - CACHE DISABLED AND KMCR
200		TEST 20 - AVAILABILITY OF SETS
201		TEST 21 - DEALLOCATION OF SETS
202		TEST 22 - CACHE WITH RELOCATION DISABLED
203		TEST 23 - WRITE CYCLES AND CACHE
204		TEST 24 - DMA READ WITH INDEX NOT ZERO
205		TEST 25 - TAG REGISTERS
206		TEST 26 - CACHE RAM BIT PATTERN TEST
207		TEST 27 - BOOT ROMS TEST
208		TEST 28 - UNIBUS MEMORY TEST
209		TEST 29 - UBE AUTOSIZING ROUTINE
210		TEST 30 - NPG ARBITRATION
211		TEST 31 - NO BUS GRANTS WITH PROCESSOR AT HIGHER PRIORITY
212		TEST 32 - BR7-BR4 ARBITRATION
213		TEST 33 - ARBITRATION BETWEEN INTERRUPTS AND PIRQ'S
214		TEST 34 - ARBITRATION BETWEEN INTERRUPTS FROM 2 UBE
215		TEST 35 - POWER DOWN TEST
216		TEST 36 - WRONG PARITY TEST
217		TEST 37 - NO SACK TIMEOUT
218		TEST 38 - NO INTERRUPT TEST
219		TEST 39 - UNIBUS DEVICE DATO CYCLE
220		TEST 40 - UNIBUS DEVICE DATI CYCLE
221		TEST 41 - UNIBUS DEVICE DATO CYCLE WITH RELOCATION ENABLED
222		TEST 42 - UNIBUS DEVICE DATI CYCLE WITH RELOCATION ENABLED
223		TEST 43 - ALU TEST USING UBE
224		TEST 44 - CARRY PROPOGATION TEST USING UBE
225		TEST 45 - NXM TEST USING UBE
226		TEST 46 - RELOCATION WITH UNIBUS MEMORY
227		TEST 47 - MAIN MEMORY DISABLE THRU UBE
228		TEST 48 - UNIBUS DEVICE DATOB CYCLE
229		TEST 49 - UNIBUS DEVICE DATIP CYCLE
230		TEST 50 - UNIBUS DEVICE I/O PAGE READ CYCLE
231		TEST 51 - UNIBUS DEVICE I/O PAGE WRITE CYCLE
232		TEST 52 - MAPPING REGISTERS TEST USING UBE
233		TEST 53 - UNIBUS DEVICE DATI CYCLE WITH CACHE ENABLED
234		TEST 54 - WRONG PARITY AND CACHE
235		

237
238
260
271
272
273
274

167400
000300

```

$SWR=167400
$SWRMK=300
.TITLE KTJ11-B DIAGNOSTIC
;*COPYRIGHT (C) MAY 83
;*DIGITAL EQUIPMENT CORP.
;*MAYNARD, MASS. 01754
;*
;*PROGRAM BY DIAG. ENG.
;*
;*THIS PROGRAM WAS ASSEMBLED USING THE PDP 11 MAINDEC SYSMAC
;*PACKAGE (MAINDEC-11-DZQAC-C8), OCT, 1982.

```

275

000001

```

$TN=1
.SBTTL OPERATIONAL SWITCH SETTINGS
;*
;* SWITCH USE
;* -----
;* 15 HALT ON ERROR
;* 14 LOOP ON TEST
;* 13 INHIBIT ERROR TYPEOUTS
;* 11 INHIBIT ITERATIONS
;* 10 BELL ON ERROR
;* 9 LOOP ON ERROR
;* 8 LOOP ON TEST IN SWR<5:0>

```

277

001100
104000
000004

```

.SBTTL BASIC DEFINITIONS
;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
STACK= 1100
ERROR= EMT ;;BASIC DEFINITION OF ERROR CALL
SCOPE= IOT ;;BASIC DEFINITION OF SCOPE CALL
;*MISCELLANEOUS DEFINITIONS
HT= 11 ;;CODE FOR HORIZONTAL TAB
LF= 12 ;;CODE FOR LINE FEED
CR= 15 ;;CODE FOR CARRIAGE RETURN
CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED
PS= 177776 ;;PROCESSOR STATUS WORD
PSW= PS
STKLMT= 177774 ;;STACK LIMIT REGISTER
PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
DSWR= 177570 ;;HARDWARE SWITCH REGISTER
DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
;*GENERAL PURPOSE REGISTER DEFINITIONS
R0= #0 ;;GENERAL REGISTER
R1= #1 ;;GENERAL REGISTER
R2= #2 ;;GENERAL REGISTER
R3= #3 ;;GENERAL REGISTER
R4= #4 ;;GENERAL REGISTER
R5= #5 ;;GENERAL REGISTER
R6= #6 ;;GENERAL REGISTER
R7= #7 ;;GENERAL REGISTER
SP= #6 ;;STACK POINTER
PC= #7 ;;PROGRAM COUNTER
;*PRIORITY LEVEL DEFINITIONS
PRO= 0 ;;PRIORITY LEVEL 0

```

000011
000012
000015
000200
177776
177776
177774
177772
177570
177570
000000
000001
000002
000003
000004
000005
000006
000007
000006
000007
000000

BASIC DEFINITIONS

```

000040      PR1=   40      ;;PRIORITY LEVEL 1
000100      PR2=   100    ;;PRIORITY LEVEL 2
000140      PR3=   140    ;;PRIORITY LEVEL 3
000200      PR4=   200    ;;PRIORITY LEVEL 4
000240      PR5=   240    ;;PRIORITY LEVEL 5
000300      PR6=   300    ;;PRIORITY LEVEL 6
000340      PR7=   340    ;;PRIORITY LEVEL 7
                ;*"SWITCH REGISTER" SWITCH DEFINITIONS
100000      SW15= 100000
040000      SW14=  40000
020000      SW13=  20000
010000      SW12=  10000
004000      SW11=  4000
002000      SW10=  2000
001000      SW09=  1000
000400      SW08=   400
000200      SW07=   200
000100      SW06=   100
000040      SW05=   40
000020      SW04=   20
000010      SW03=   10
000004      SW02=   4
000002      SW01=   2
000001      SW00=   1
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

```


BASIC DEFINITIONS

278

```

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      IOTVEC= 20        ;; INPUT/OUTPUT TRAP (IOT) **SCOPE**
000024      PWRVEC= 24        ;; POWER FAIL
000030      EMTVEC= 30        ;; EMULATOR TRAP (EMT) **ERROR**
000034      TRAPVEC=34        ;; "TRAP" TRAP
000060      TKVEC=  60         ;; TTY KEYBOARD VECTOR
000064      TPVEC=  64         ;; TTY PRINTER VECTOR
000240      PIRQVEC=240       ;; PROGRAM INTERRUPT REQUEST VECTOR

.SBTTL MEMORY MANAGEMENT DEFINITIONS
; *KT11 VECTOR ADDRESS
000250      MMVEC=  250

; *KT11 STATUS REGISTER ADDRESSES
177572      SR0=   177572
177574      SR1=   177574
177576      SR2=   177576
172516      SR3=   172516

; *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      KIPAR7= 172356

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

```

MEMORY MANAGEMENT DEFINITIONS

172370
172372
172374
172376

KDPA4= 172370
KDPA5= 172372
KDPA6= 172374
KDPA7= 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'

279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331

170200
170202
170204
170206
170210
170212
170214
170216
170220
170222
170224
170226
170230
170232
170234
170236
170240
170242
170244
170246
170250
170252
170254
170256
170260
170262
170264
170266
170270
170272
170274
170276
170300
170302
170304
170306
170310
170312
170314
170316
170320
170322
170324
170326
170330
170332
170334

MAPL00 = 170200
MAPH00 = 170202
MAPL01 = 170204
MAPH01 = 170206
MAPL02 = 170210
MAPH02 = 170212
MAPL03 = 170214
MAPH03 = 170216
MAPL04 = 170220
MAPH04 = 170222
MAPL05 = 170224
MAPH05 = 170226
MAPL06 = 170230
MAPH06 = 170232
MAPL07 = 170234
MAPH07 = 170236
MAPL10 = 170240
MAPH10 = 170242
MAPL11 = 170244
MAPH11 = 170246
MAPL12 = 170250
MAPH12 = 170252
MAPL13 = 170254
MAPH13 = 170256
MAPL14 = 170260
MAPH14 = 170262
MAPL15 = 170264
MAPH15 = 170266
MAPL16 = 170270
MAPH16 = 170272
MAPL17 = 170274
MAPH17 = 170276
MAPL20 = 170300
MAPH20 = 170302
MAPL21 = 170304
MAPH21 = 170306
MAPL22 = 170310
MAPH22 = 170312
MAPL23 = 170314
MAPH23 = 170316
MAPL24 = 170320
MAPH24 = 170322
MAPL25 = 170324
MAPH25 = 170326
MAPL26 = 170330
MAPH26 = 170332
MAPL27 = 170334

UNIBUS MAP REGISTER DEFINITIONS

332	170336	MAPH27 = 170336
333	170340	MAPL30 = 170340
334	170342	MAPH30 = 170342
335	170344	MAPL31 = 170344
336	170346	MAPH31 = 170346
337	170350	MAPL32 = 170350
338	170352	MAPH32 = 170352
339	170354	MAPL33 = 170354
340	170356	MAPH33 = 170356
341	170360	MAPL34 = 170360
342	170362	MAPH34 = 170362
343	170364	MAPL35 = 170364
344	170366	MAPH35 = 170366
345	170370	MAPL36 = 170370
346	170372	MAPH36 = 170372
347	170374	MAPL37 = 170374
348	170376	MAPH37 = 170376

349		
350	170200	MAPL0 = MAPL00
351	170202	MAPH0 = MAPH00
352	170204	MAPL1 = MAPL01
353	170206	MAPH1 = MAPH01
354	170210	MAPL2 = MAPL02
355	170212	MAPH2 = MAPH02
356	170214	MAPL3 = MAPL03
357	170216	MAPH3 = MAPH03
358	170220	MAPL4 = MAPL04
359	170222	MAPH4 = MAPH04
360	170224	MAPL5 = MAPL05
361	170226	MAPH5 = MAPH05
362	170230	MAPL6 = MAPL06
363	170232	MAPH6 = MAPH06
364	170234	MAPL7 = MAPL07
365	170236	MAPH7 = MAPH07

.SBTTL UBA SPECIFIC REGISTERS

366			
367			
368			
369	177572	MMR0 = 177572	; MEMORY MANAGEMENT REGISTER DEFINITIONS
370	177574	MMR1 = 177574	;
371	177576	MMR2 = 177576	;
372	172516	MMR3 = 172516	;
373	000001	UFDSET = 1	; FLAG FOR UFD MODE
374	177520	BCSR = 177520	; BOOT/DIAGNOSTIC STATUS REGISTER
375	177730	DCSR = 177730	; DIAGNOSTIC CONTROLLER STATUS REGISTER
376	177732	DDR = 177732	; DIAGNOSTIC DATA REGISTER
377	177734	KMCR = 177734	; KTJ11-B MEMORY CONFIGURATION REGISTER
378	177746	CCR = 177746	; CACHE CONTROL REGISTER FOR CPU
379	177522	PCR = 177522	; PAGE CONTROL REGISTER
380	120001	POLY = 120001	; POLYNOMIAL USED FOR CRC ROUTINES
381	170014	SIMLGO = 170014	; SIMULTANEOUS GO ADDRESS FOR MULTIPLE UBE'S

.SBTTL TRAP CATCHER

```

000000
.=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
000174
.=174

```

TRAP CATCHER

```

000174 000000      DISPREG: .WORD 0          ;;SOFTWARE DISPLAY REGISTER
000176 000000      SWREG: .WORD 0           ;;SOFTWARE SWITCH REGISTER
384      000200      . =200
385 000200 005037 001160      CLR $TMPO
386 000204 000137 002530      JMP @*START
387      000220      . =220
388 000220 012737 000777 001160  MOV #777,$TMPO
389 000226 000137 002530      JMP @*START
390
391      .SBTTL ACT11 HOOKS
      ;;*****
      ;HOOKS REQUIRED BY ACT11
      $SVPC= .          ;SAVE PC
      . =46
      $ENDAD           ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
      . =52
      .WORD 0          ;;2)SET LOC.52 TO ZERO
      .=$SVPC         ;; RESTORE PC
392      .SBTTL APT PARAMETER BLOCK
      ;;*****
      ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
      ;;*****
      . $X= .          ;;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.
      $APTHDR         ;;POINT TO APT HEADER BLOCK
      . =.$X          ;;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         ;;RUN TIM OF LONGEST TEST
      $PASTM: .WORD        ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
      $UNITH: .WORD        ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
      .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)

```

COMMON TAGS

393

.SBTTL COMMON TAGS

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

001100 001100 #CMTAG: .-1100 ;; START OF COMMON TAGS
001100 000000 #TSTNM: .WORD 0 ;; CONTAINS THE TEST NUMBER
001102 000 #ERFLG: .BYTE 0 ;; CONTAINS ERROR FLAG
001103 000 #ICNT: .WORD 0 ;; CONTAINS SUBTEST ITERATION COUNT
001104 000000 #LPADR: .WORD 0 ;; CONTAINS SCOPE LOOP ADDRESS
001106 000000 #LPERR: .WORD 0 ;; CONTAINS SCOPE RETURN FOR ERRORS
001110 000000 #ERTTL: .WORD 0 ;; CONTAINS TOTAL ERRORS DETECTED
001112 000000 #ITEMB: .BYTE 0 ;; CONTAINS ITEM CONTROL BYTE
001114 000 #ERMAX: .BYTE 1 ;; CONTAINS MAX. ERRORS PER TEST
001115 001 #ERRPC: .WORD 0 ;; CONTAINS PC OF LAST ERROR INSTRUCTION
001116 000000 #GDADR: .WORD 0 ;; CONTAINS ADDRESS OF 'GOOD' DATA
001120 000000 #BDADR: .WORD 0 ;; CONTAINS ADDRESS OF 'BAD' DATA
001122 000000 #GDDAT: .WORD 0 ;; CONTAINS 'GOOD' DATA
001124 000000 #BDDAT: .WORD 0 ;; CONTAINS 'BAD' DATA
001126 000000 .WORD 0 ;; RESERVED--NOT TO BE USED
001130 000000 .WORD 0
001132 000000 .WORD 0
001134 000 #AUTOB: .BYTE 0 ;; AUTOMATIC MODE INDICATOR
001135 000 #INTAG: .BYTE 0 ;; INTERRUPT MODE INDICATOR
001136 000000 .WORD 0
001140 177570 SWR: .WORD DSWR ;; ADDRESS OF SWITCH REGISTER
001142 177570 DISPLAY: .WORD DDISP ;; ADDRESS OF DISPLAY REGISTER
001144 177560 #TKS: 177560 ;; TTY KBD STATUS
001146 177562 #TKB: 177562 ;; TTY KBD BUFFER
001150 177564 #TPS: 177564 ;; TTY PRINTER STATUS REG. ADDRESS
001152 177566 #TPB: 177566 ;; TTY PRINTER BUFFER REG. ADDRESS
001154 000 #NULL: .BYTE 0 ;; CONTAINS NULL CHARACTER FOR FILLS
001155 002 #FILLS: .BYTE 2 ;; CONTAINS # OF FILLER CHARACTERS REQUIRED
001156 012 #FILLC: .BYTE 12 ;; INSERT FILL CHARS. AFTER A "LINE FEED"
001157 000 #TPFLG: .BYTE 0 ;; "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
. REPT 2
001160 000000 #TMP0: .WORD 0 ;; USER DEFINED
001162 000000 #TMP1: .WORD 0 ;; USER DEFINED
001164 000000 #TIMES: 0 ;; MAX. NUMBER OF ITERATIONS
001166 000000 #ESCAPE: 0 ;; ESCAPE ON ERROR ADDRESS
001170 207 377 377 #BELL: .ASCIZ <207><377><377> ;; CODE FOR BELL
001173 000
001174 077 #QUES: .ASCII /?/ ;; QUESTION MARK
001175 015 #CRLF: .ASCII <15> ;; CARRIAGE RETURN
001176 012 000 #LF: .ASCIZ <12> ;; LINE FEED

.SBTTL APT MAILBOX-ETABLE

.EVEN
\$MAIL: ;; APT MAILBOX
001200 000000 \$MSGTY: .WORD AMSGTY ;; MESSAGE TYPE CODE
001202 000000 \$FATAL: .WORD AFATAL ;; FATAL ERROR NUMBER
001204 000000 \$TESTN: .WORD ATESTN ;; TEST NUMBER
001206 000000 \$PASS: .WORD APASS ;; PASS COUNT
001210 000000 \$DEVCT: .WORD ADEVCT ;; DEVICE COUNT
001212 000000 \$UNIT: .WORD AUNIT ;; I/O UNIT NUMBER
001214 000000 \$MSGAD: .WORD AMSGAD ;; MESSAGE ADDRESS

APT MAILBOX-ETABLE

```

001216 000000 $MSGLG: .WORD AMSGLG ;;MESSAGE LENGTH
001220 $ETABLE: ;;APT ENVIRONMENT TABLE
001220 000 $ENV: .BYTE AENV ;;ENVIRONMENT BYTE
001221 000 $ENVM: .BYTE AENVM ;;ENVIRONMENT MODE BITS
001222 000000 $SWREG: .WORD ASWREG ;;APT SWITCH REGISTER
001224 000000 $USWR: .WORD AUSWR ;;USER SWITCHES
001226 000000 $CPUOP: .WORD ACPUOP ;;CPU TYPE,OPTIONS
;*
;* 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
001230 000 $MAMS1: .BYTE AMAMS1 ;;HIGH ADDRESS,M.S. BYTE
001231 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
001232 000000 $MADR1: .WORD AMADR1 ;;HIGH ADDRESS,BLK#1
;*
;* MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
001234 000 $MAMS2: .BYTE AMAMS2 ;;HIGH ADDRESS,M.S. BYTE
001235 000 $MTYP2: .BYTE AMTYP2 ;;MEM. TYPE,BLK#2
001236 000000 $MADR2: .WORD AMADR2 ;;MEM.LAST ADDRESS,BLK#2
001240 000 $MAMS3: .BYTE AMAMS3 ;;HIGH ADDRESS,M.S.BYTE
001241 000 $MTYP3: .BYTE AMTYP3 ;;MEM. TYPE,BLK#3
001242 000000 $MADR3: .WORD AMADR3 ;;MEM.LAST ADDRESS,BLK#3
001244 000 $MAMS4: .BYTE AMAMS4 ;;HIGH ADDRESS,M.S.BYTE
001245 000 $MTYP4: .BYTE AMTYP4 ;;MEM. TYPE,BLK#4
001246 000000 $MADR4: .WORD AMADR4 ;;MEM.LAST ADDRESS,BLK#4
001250 000000 $VECT1: .WORD AVECT1 ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
001252 000000 $VECT2: .WORD AVECT2 ;;INTERRUPT VECTOR#2BUS PRIORITY#2
001254 000000 $BASE: .WORD ABASE ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
001256 000000 $DEVN: .WORD ADEVN ;;DEVICE MAP
001260 000000 $CDW1: .WORD ACDW1 ;;CONTROLLER DESCRIPTION WORD#1
001262 000000 $CDW2: .WORD ACDW2 ;;CONTROLLER DESCRIPTION WORD#2
001264 000000 $DDW0: .WORD ADDW0 ;;DEVICE DESCRIPTOR WORD#0
001266 000000 $DDW1: .WORD ADDW1 ;;DEVICE DESCRIPTOR WORD#1
001270 000000 $DDW2: .WORD ADDW2 ;;DEVICE DESCRIPTOR WORD#2
001272 000000 $DDW3: .WORD ADDW3 ;;DEVICE DESCRIPTOR WORD#3
001274 000000 $DDW4: .WORD ADDW4 ;;DEVICE DESCRIPTOR WORD#4
001276 000000 $DDW5: .WORD ADDW5 ;;DEVICE DESCRIPTOR WORD#5
001300 000000 $DDW6: .WORD ADDW6 ;;DEVICE DESCRIPTOR WORD#6
001302 000000 $DDW7: .WORD ADDW7 ;;DEVICE DESCRIPTOR WORD#7
001304 000000 $DDW8: .WORD ADDW8 ;;DEVICE DESCRIPTOR WORD#8
001306 000000 $DDW9: .WORD ADDW9 ;;DEVICE DESCRIPTOR WORD#9
001310 000000 $DDW10: .WORD ADDW10 ;;DEVICE DESCRIPTOR WORD#10
001312 000000 $DDW11: .WORD ADDW11 ;;DEVICE DESCRIPTOR WORD#11
001314 000000 $DDW12: .WORD ADDW12 ;;DEVICE DESCRIPTOR WORD#12
001316 000000 $DDW13: .WORD ADDW13 ;;DEVICE DESCRIPTOR WORD#13
001320 000000 $DDW14: .WORD ADDW14 ;;DEVICE DESCRIPTOR WORD#14
001322 000000 $DDW15: .WORD ADDW15 ;;DEVICE DESCRIPTOR WORD#15
001324 $ETEND:

```

ERROR POINTER TABLE

```

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

```

001324

```

394
395
396
397
398 001324 027234
399 001326 031422
400 001330 032414
401 001332 000000
402
403
404 001334 027300
405 001336 031471
406
407 001340 032426
408 001342 000000
409
410
411 001344 027342
412 001346 031471
413
414 001350 032426
415 001352 000000
416
417
418 001354 027406
419 001356 031563
420
421 001360 032444
422 001362 000000
423
424
425 001364 027464
ERS 426 001366 031653
427
428 001370 032460
429 001372 000000
430
431
432 001374 027562
433 001376 031731
434
435 001400 032460
436 001402 000000
437
438
439 001404 027625

```

```

.SBTTL ERROR DEFINITIONS
;
;      ERROR 1
;      EM1          ; TIMEOUT ON ACCESSING A MAP REGISTER
;      DH1          ; TEST # ERROR PC ERROR # ADDRESS
;      DT1          ; TEST, $ERRPC, ERRNUM, $BDADR
;      0
;      ERROR 2
;      EM2          ; MAP REGISTER COULD NOT BE CLEARED
;      DH2          ;                GOOD BAD
;                  ; TEST # ERROR PC ERROR # DATA DATA ADDRESS
;                  ; TEST, $ERRPC, ERRNUM, $GDDAT, $BDDAT, $BDADR
;      DT2
;      0
;      ERROR 3
;      EM3          ; MAP REGISTER COULD NOT HOLD PATTERN
;      DH2          ;                GOOD BAD
;                  ; TEST # ERROR PC ERROR # DATA DATA ADDRESS
;                  ; TEST, $ERRPC, ERRNUM, $GDDAT, $BDDAT, $BDADR
;      DT2
;      0
;      ERROR 4
;      EM4          ; MAP REGISTER HAS NOT BEEN ADDRESSED CORRECTLY
;      DH3          ;                GOOD BAD
;                  ; TEST # ERROR PC ERROR # ADDRESS ADDRESS
;                  ; TEST, $ERRPC, ERRNUM, $GDADR, $BDADR
;      DT3
;      0
;      ERROR 5
;      EM5          ; THERE WAS NO DIFFERENCE FOUND BETWEEN HI AND LO MAP REGIST
;                  ;                HI LOW
;                  ; TEST # ERROR PC ERROR # MAP MAP
;                  ; TEST, $ERRPC, ERRNUM, $GDDAT, $BDDAT
;      DH4
;      DT4
;      0
;      ERROR 6
;      EM6          ; ERROR IN BITS 3-6,9-14 IN THE DCSR
;      DH5          ;                GOOD BAD
;                  ; TEST # ERROR PC ERROR # DATA DATA
;                  ; TEST, $ERRPC, ERRNUM, $GDDAT, $BDDAT
;      DT4
;      0
;      ERROR 7
;      EM7          ; DCSR DID RESPOND PROPERLY ON RESET

```

ERROR DEFINITIONS

```

440 001406 031731          DHS                    ;
441                                ; TEST # ERROR PC ERROR # DATA DATA
442 001410 032460          DT4                    ; TEST, $ERRPC, ERRNUM, $GDDAT, $BDDAT
443 001412 000000          0
444                                ;
445                                ; ERROR 10
446 001414 027674          EM10                   ; TIMEOUT HAS OCCURED ON ACCESS TO THE DCSR
447 001416 031422          DH1                    ; TEST # ERROR PC ERROR # ADDRESS
448 001420 032414          DT1                    ; TEST, $ERRPC, ERRNUM, $BDADR
449 001422 000000          0
450                                ;
451                                ; ERROR 11
452 001424 027746          EM11                   ; KMCR BITS 0-5,8 DID NOT GET SET CORRECTLY
453 001426 031471          DH2                    ;
454                                ; TEST # ERROR PC ERROR # DATA DATA ADDRESS
455 001430 032426          DT2                    ; TEST, $ERRPC, ERRNUM, $GDDAT, $BDDAT, $BDADR
456 001432 000000          0
457                                ;
458                                ; ERROR 12
459 001434 030020          EM12                   ; TIMEOUT HAS OCCURED ON ACCESS TO THE KMCR
460 001436 031422          DH1                    ; TEST # ERROR PC ERROR # ADDRESS
461 001440 032414          DT1                    ; TEST, $ERRPC, ERRNUM, $BDADR
462 001442 000000          0
463                                ;
464                                ; ERROR 13
465 001444 030072          EM13                   ; ERROR IN DATA PATH
466 001446 032013          DH13                   ; TEST # ERROR PC ERROR # PATTERN DDR
467 001450 032460          DT4                    ; TEST, $ERRPC, ERRNUM, $GDDAT, $BDDAT
468 001452 000000          0
469                                ;
470                                ; ERROR 14
471 001454 030141          EM14                   ; ERROR IN DATA OUT
472 001456 031731          DHS                    ;
473                                ; TEST # ERROR PC ERROR # DATA DATA
474 001460 032460          DT4                    ; TEST, $ERRPC, ERRNUM, $GDDAT, $BDDAT
475 001462 000000          0
476                                ;
477                                ; ERROR 15
478 001464 030171          EM15                   ; ERROR IN DATA IN
479 001466 032013          DH13                   ; TEST # ERROR PC ERROR # PATTERN DDR
480 001470 032460          DT4                    ; TEST, $ERRPC, ERRNUM, $GDDAT, $BDDAT
481 001472 000000          0
482                                ;
483                                ; ERROR 16
484 001474 030220          EM16                   ; DDR NOT ZERO WHEN DCSR SELECTS UNIBUS LINES
485 001476 031731          DHS                    ;
486                                ; TEST # ERROR PC ERROR # DATA DATA
487 001500 032460          DT4                    ; TEST, $ERRPC, ERRNUM, $GDDAT, $BDDAT
488 001502 000000          0
489                                ;
490                                ; ERROR 17
491 001504 030302          EM17                   ; ERROR IN SETTING DCSR<7>
492 001506 032066          DH16                   ; TEST # ERROR PC ERROR #
493 001510 032474          DT16                   ; TEST, $ERRPC, ERRNUM
494 001512 000000          0
495                                ;
496                                ; ERROR 20

```


ERROR DEFINITIONS

```

497 001514 030335      EM20      ; ERROR IN UNIQUE ADDRESSING OF REGISTERS
498 001516 032125      DH20      ; TEST # ERROR PC ERROR # KMCR PAIR FAILFD
499 001520 032504      DT20      ; TEST, $ERRPC, ERRNUM, KMCR, $BDADR
500 001522 000000      0
501      ; ERROR 21
502
503 001524 030413      EM21      ; REG. PAIR 31. PERFORMS RELOCATION
504 001526 032066      DH16      ; TEST # ERROR PC ERROR #
505 001530 032474      DT16      ; TEST, $ERRPC, ERRNUM
506 001532 000000      0
507      ; ERROR 22
508
509 001534 030460      EM22      ; NXM CONDITION COULDN'T BE CREATED THRU ALU
510 001536 032066      DH16      ; TEST # ERROR PC ERROR #
511 001540 032474      DT16      ; TEST, $ERRPC, ERRNUM
512 001542 000000      0
513      ; ERROR 23
514
515 001544 030534      EM23      ; ALU ERROR
516 001546 032205      DH23      ;
517      ; TEST # ERROR PC ERROR # DATA DATA ADDRESS
518 001550 032530      DT23      ; TEST, $ERRPC, ERRNUM, $GDDAT, $BDDAT, $KIPAR6, $BDADR
519 001552 000000      0
520      ; ERROR 24
521
522 001554 030546      EM24      ; CPU CACHE ERROR
523 001556 032066      DH16      ; TEST # ERROR PC ERROR #
524 001560 032474      DT16      ; TEST, $ERRPC, ERRNUM
525 001562 000000      0
526      ; ERROR 25
527
528 001564 030566      EM25      ; KMCR<4-0> DOESN'T DISABLE MAIN MEMORY
529 001566 032066      DH16      ; TEST # ERROR PC ERROR #
530 001570 032474      DT16      ; TEST, $ERRPC, ERRNUM
531 001572 000000      0
532      ; ERROR 26
533
534 001574 030636      EM26      ; KMCR DOES NOT REFLECT EXPECTED STATUS OF THE CACHE/
535 001576 031731      DH5       ;
536      ; TEST # ERROR PC ERROR # DATA DATA
537 001600 032460      DT4       ; TEST, $ERRPC, ERRNUM, $GDDAT, $BDDAT
538 001602 000000      0
539      ; ERROR 27
540
541 001604 030721      EM27      ; ERROR IN CACHE TAG REGISTERS
542 001606 032316      DH27      ; TEST # ERROR PC ERROR # ADDRESS
543 001610 032550      DT27      ; TEST, $ERRPC, ERRNUM, MAPH01, MAPL01
544 001612 000000      0
545      ; ERROR 30
546
547 001614 030756      EM30      ; ERROR IN THE DMA CACHE DATA RAMS
548 001616 031422      DH1       ; TEST # ERROR PC ERROR # ADDRESS
549 001620 032564      DT30      ; TEST, $ERRPC, ERRNUM, MAPLO0
550 001622 000000      0
551      ; ERROR 31
552
553 001624 031017      EM31      ; ERROR IN THE M9312 BOOT ROM SECTION

```

ERROR DEFINITIONS

```

554 001626 032066          DH16          ; TEST # ERROR PC ERROR #
555 001630 032474          DT16          ; TEST, $ERRPC, ERRNUM
556 001632 000000          0
557          ;          ERROR 32
558
559 001634 031063          EM32          ; ERROR IN ARBITRATION LOGIC THRU UBE
560 001636 032066          DH16          ; TEST # ERROR PC ERROR #
561 001640 032474          DT16          ; TEST, $ERRPC, ERRNUM
562 001642 000000          0
563          ;          ERROR 33
564
565 001644 031130          EM33          ; ERROR TRYING TO DO DMA CYCLES THRU UBE
566 001646 032066          DH16          ; TEST # ERROR PC ERROR #
567 001650 032474          DT16          ; TEST, $ERRPC, ERRNUM
568 001652 000000          0
569          ;          ERROR 34
570
571 001654 031204          EM34          ; ERROR IN THE UNIBUS MEMORY TEST
572 001656 032066          DH16          ; TEST # ERROR PC ERROR #
573 001660 032474          DT16          ; TEST, $ERRPC, ERRNUM
574 001662 000000          0
575          ;          ERROR 35
576
577 001664 031244          EM35          ; UNEXPECTED TIMEOUT HAS OCCURED
578 001666 032066          DH16          ; TEST # ERROR PC ERROR #
579 001670 032520          DT21          ; TEST, $BDADR, ERRNUM
580 001672 000000          0
581          ;          ERROR 36
582
583 001674 031303          EM36          ; UNIBUS TIMEOUT DID NOT OCCUR
584 001676 032066          DH16          ; TEST # ERROR PC ERROR #
585 001700 032474          DT16          ; TEST, $ERRPC, ERRNUM
586 001702 000000          0
587          ;          ERROR 37
588
589 001704 031351          EM37          ; DCSR<3> DIDN'T DISABLE UBA ROM'S
590 001706 032066          DH16          ; TEST # ERROR PC ERROR #
591 001710 032474          DT16          ; TEST, $ERRPC, ERRNUM
592 001712 000000          0

```

```

593
594          .SBTTL GLOBAL VARIABLES
595 001714 000000          ERRNUM: .WORD 0          ; ERROR NUMBER FOR REPORT
596 001716 000000          TEST: .WORD 0          ; TEST NUMBER FOR REPORT
597 001720 000000          PMIS: .WORD 0          ; PMI MEMORY SIZE
598 001722 000000          UBECT: .WORD 0          ; NUMBER OF UBE'S
599 001724 000000          BE1INT: .WORD 0          ; UBE #1 INTERRUPT FLAG
600 001726 000000          BE2INT: .WORD 0          ; UBE #2 INTERRUPT FLAG
601 001730 000000          TOUT: .WORD 0          ; TIMEOUT FLAG
602 001732 172100          MCSR: .WORD 172100          ; POINTER TO MEMORY CSR
603 001734          WRTBUF: .BLKW 20          ; WRITE BUFFER
604 001774 000377 007417 031463 PTRN16: .WORD 377,7417,31463,52525,125252 ; 16-BIT BINARY DIVIDE
        002002 052525 125252
605 002006 145454 104210 155555          .WORD 145454,104210,155555,021042
        002014 021042
606 002016 053346 146314 000000          .WORD 053346,146314,0
607 002024 000017 000014 000025 PTRN6: .WORD 17,14,25,52          ; 6-BIT BINARY DIVIDE
        002032 000052

```

GLOBAL VARIABLES

```

608 002034 000026 000042 000010      .WORD  26,42,10,55,77,70,7,0
      002042 000055 000077 000070
      002050 000007 000000
609
610      .SBTTL  CACHE RAM IMAGE TABLE
611
612      ;
613      ; THE FOLLOWING BLOCK OF DATA WILL BE USED IN THE CACHE RAM BIT TEST
614      ; THE EXACT ADDRESSES WILL BE FIGURED OUT BY THE PROGRAM DEPENDING
615      ; ON THE LOCATION OF THIS TABLE.
616      ;
617
618 002054      CTBLE:  .BLKW  47      ; THE LARGEST # OF LOCATIONS NEEDED
619
620 002172 000000      OBADR:  .WORD  0      ; FIRST ADDRESS OF A FIELD IN TABLE
621 002174 000000      .WORD  0      ; FIRST ADDRESS OF B FIELD IN TABLE
622 002176 000000      .WORD  0      ; FIRST ADDRESS OF C FIELD IN TABLE
623 002200 000000      .WORD  0      ; FIRST ADDRESS OF D FIELD IN TABLE
624
625      .SBTTL  UNIBUS EXERCISER REGISTER TABLES
626
627      ;
628      ;      UBE #1
629      ;
630
631 002202 000000      BE1DB:  0      ; UBE #1 DATA REGISTER
632 002204 000000      BE1CC:  0      ; UBE #1 CYCLE COUNT REGISTER
633 002206 000000      BE1BA:  0      ; UBE #1 ADDRESS REGISTER
634 002210 000000      BE1CR1: 0      ; UBE #1 CONTROL REGISTER 1
635 002212 000000      BE1CLR: 0      ; UBE #1 CLEAR ERROR REGISTER ADDRESS
636 002214 000000      BE1CR2: 0      ; UBE #1 CONTROL REGISTER 2
637 002216 000000      BE1VEC: 0      ; UBE #1 VECTOR PC
638 002220 000000      BE1PSW: 0      ; UBE #1 VECTOR PSW
639
640      ;
641      ;      UBE #2
642      ;
643
644 002222 000000      BE2DB:  0      ; UBE #2 DATA REGISTER
645 002224 000000      BE2CC:  0      ; UBE #2 CYCLE COUNT REGISTER
646 002226 000000      BE2BA:  0      ; UBE #2 ADDRESS REGISTER
647 002230 000000      BE2CR1: 0      ; UBE #2 CONTROL REGISTER 1
648 002232 000000      BE2CLR: 0      ; UBE #2 CLEAR ERROR REGISTER ADDRESS
649 002234 000000      BE2CR2: 0      ; UBE #2 CONTROL REGISTER 2
650 002236 000000      BE2VEC: 0      ; UBE #2 VECTOR PC
651 002240 000000      BE2PSW: 0      ; UBE #2 VECTOR PSW
652      .SBTTL  SUBROUTINE - DIAGNOSTIC_DATA_OUT SUBROUTINE
653
654      ;* INPUTS: PATTERN TO BE STORED IN DDR AND THEN WRITTEN TO MEMORY $GDDAT
655      ;*      TEST_LOCATION TO BE WRITTEN TO (16 OR 22 BITS) (R1)
656      ;
657      ;* ON RETURN FROM SUBROUTINE TEST_LOCATION SHOULD HAVE THE SAME DATA
658      ;* AS DDR AND THE SAME AS PATTERN
659      ;* THE PROGRAM HAS TO BE RUNNING IN DIAGNOSTIC MODE WITH DIAGNOSTIC
660      ;* NPR REGISTER SELECTED
661      ;
662      ;      BGNROUTINE

```

SUBROUTINE - DIAGNOSTIC_DATA_OUT SUBROUTINE

```

663 ;
664 ; MOVE PATTERN TO DDR
665 ; DO EXTERNAL WRITE FROM TEST_LOCATION
666 ; RETURN
667 ;
668 ; ENDRoutine
669 ;
670 ;-----
671 ;
672 002242 013737 001124 177732 DDOUT: MOV $GDDAT,DDR ; STORE PATTERN IN DDR
673 002250 011111 1$: MOV (R1),(R1) ; EXTENAL READ TO PROVIDE ADDRESS
674 002252 000207 RTS PC
675 ;

```

```

676 ;
677 .SBTTL SUBROUTINE DIAGNOSTIC_DATA_IN SUBROUTINE
678 ;
679 ;* INPUTS: PATTERN TO BE WRITTEN TO MEMORY AND THEN TO DDR $GDDAT
680 ;* TEST_LOCATION TO READ FROM (16 OR 22 BITS) (R1)
681 ;
682 ;* ON RETURN FROM SUBROUTINE DDR SHOULD HAVE THE SAME DATA AS
683 ;* SPECIFIED MEMORY LOCATION AND THE SAME AS THE PATTERN
684 ;* THE PROGRAM HAS TO BE RUNNING IN DIAGNOSTIC MODE WITH DIAGNOSTIC
685 ;* NPR REGISTER SELECTED.
686 ;
687 ; BGNROUTINE
688 ;
689 ; LET DCSR<0> = #1
690 ; DO EXTERNAL WRITE FROM TEST_LOCATION
691 ; RETURN
692 ;
693 ; ENDRoutine
694 ;
695 ;-----

```

```

696 ;
697 002254 052737 000001 177730 DDIN: BIS #BIT00,DCSR ; SET GO BIT
698 002262 011111 2$: MOV (R1),(R1) ; PROVIDE ADDRESS FOR DMA
699 002264 000207 3$: RTS PC
700 ;

```

```

701 ;
702 .SBTTL SUBROUTINE - TIMEOUT_ROUTINE
703 ;* THIS ROUTINE IS USED TO FLAG AN UNEXPECTED TIMEOUT.
704 ;
705 ; BGNROUTINE
706 ;
707 ; STORE ADDRESS THAT CAUSED TIMEOUT
708 ; ERROR
709 ; RETURN
710 ;
711 ; ENDRoutine
712 ;
713 ;-----

```

```

714 ;
715 002266 011637 001122 TIMOUT: MOV (SP), $BDADR ; STORE ADDRESS THAT TIMED OUT
716 002272 104035 ERROR +35 ; UNEXPECTED TIMEOUT
717 002274 000002 RTI
718 ;
719 ;

```

SUBROUTINE - MAP_PROGRAM_AREA

```

720 .SBTTL SUBROUTINE MAP PROGRAM AREA
721
722 ;* THIS ROUTINE MAPS THE PROGRAM AREA TO THE FIRST 32K
723 ;
724 ; BGNROUTINE
725 ;
726 ; MAP PROGRAM AREA THRU KIPAR'S TO FIRST 32 K
727 ; NO CACHE BYPASS
728 ; RETURN
729 ;
730 ; ENDROUTINE
731 ;
732 ;-----
733

```

```

734 002276 010046 MAPPR: MOV R0,-(SP) ;;PUSH R0 ON STACK
      002300 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
      002302 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
735 002304 012700 172300 MOV #172300,R0 ; R0 POINTS TO FIRST KIPDR
736 002310 012701 000010 MOV #8,R1 ; DO FOR ALL 8 REGISTERS
737 002314 012720 077406 1$: MOV #77406,(R0)+ ; . 4K PAGE, CACHE ON, READ/WRITE
738 002320 077103 SOB R1,1$ ; . CONTINUE TILL DONE
739 002322 012700 172340 MOV #172340,R0 ; R0 POINTS TO FIRST KIPAR
740 002326 012701 000006 MOV #6,R1 ; DO FOR ALL 8 REGISTERS
741 002332 012702 000000 MOV #0,R2 ; START WITH ADDRESS 0
742 002336 010220 2$: MOV R2,(R0)+ ; . LOAD WITH ADDRESS
743 002340 062702 000200 ADD #200,R2 ; . NEXT 4K
744 002344 077104 SOB R1,2$ ; . CONTINUE TILL DONE
745 002346 005020 CLR (R0)+ ; CLEAR KIPAR6
746 002350 012710 177600 MOV #177600,(R0) ; I/O PAGE TO KIPAR7
747 002354 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
      002356 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
      002360 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
748 002362 000207 RTS PC
749
750

```

```

751 .SBTTL SUBROUTINE - MEMORY SIZE
752
753 ;* THIS ROUTINE SIZES MAIN (PMI) MEMORY IN 4K WORDS.
754 ;
755 ; BGNROUTINE
756 ;
757 ; REMAP TIMEOUT VECTOR AND PROGRAM AREA
758 ; ENABLE MEMORY MANAGEMENT UNIT
759 ; DO UNTIL TIMEOUT OR ALL 2M CHECKED
760 ; . CHECK LOCATION 0 IN EACH 4K PAGE THRU KIPAR6
761 ; ENDDO
762 ;
763 ; ENDROUTINE
764 ;
765 ;-----
766

```

```

767 002364 013702 000004 MEMSIZ: MOV ERRVEC,R2 ; SAVE TIME OUT VECTOR
768 002370 012737 002444 000004 MOV #3$,ERRVEC ; POINT NEW TO PROGRAM
769 002376 012737 000340 000006 MOV #340,ERRVEC+2 ; AT PRIORITY 7
770
771 ; SIZE MEMORY IN 4K WORDS

```

SUBROUTINE - MEMORY SIZE

```

772
773 002404 012737 000200 172354 ;      MOV      #200,KIPAR6      ; FIRST 4K
774 002412 000403                ;      BR       2$             ; GO TRY TO ACCESS
775 002414 062737 000200 172354 1$:    ADD      #200,KIPAR6      ; . NEXT 4K
776 002422 005737 140000                2$:    TST      #0140000         ; . ACCESS THRU KIPAR6
777 002426 022737 170000 172354        ;      CMP      #170000,KIPAR6 ; . LAST PAGE?
778 002434 001367                ;      BNE     1$             ; . IF NOT, BRANCH
779 002436 012701 000040                ;      MOV     #40,R1         ; IF 2M PRESENT, ONLY 21ST MASKED
780 002442 000402                ;      BR       4$             ; BRANCH
781 002444 005726                3$:    TST      (SP)+          ; RESTORE STACK
782 002446 005726                ;      TST     (SP)+          ;
783 002450 010237 000004                4$:    MOV     R2,ERRVEC      ; RESTORE TIMEOUT VECTOR
784 002454 000207                ;      RTS      PC           ; RETURN
785

```

```

786      .SBTTL SUBROUTINE - SIZE UNIBUS MEMORY FROM KMCR
787

```

```

788      ;* ON RETURN FROM THIS SUBROUTINE R2 WOULD HAVE PAR VALUE
789      ;* OF UNIBUS MEMORY
790

```

```

791 002456 013701 177734      UMSIZ:  MOV     KMCR,R1          ; SAVE KMCR
792 002462 042701 000040      1$:    BIC     #BIT05,R1         ; LEAVE ONLY # OF PAGES
793 002466 012702 007600                ;      MOV     #7600,R2       ; START W/O UNIBUS MEMORY
794 002472 162702 000200      100$:  SUB     #200,R2           ; . SUBTRACT 4K
795 002476 077103                ;      SOB     R1,100$       ; . FOR ALL PAGES PRESENT
796 002500 000207      RTS      PC
797

```

```

798      .SBTTL SUBROUTINE - INITIALIZE THE UBE'S
799

```

```

800      ;* THIS ROUTINE IS USED TO INITIALIZE THE UBE'S
801

```

```

802      ; BGNROUTINE
803
804      ;
805      ;   SAVE R0,R1
806      ;   LET R0 := FIRST UBE ADDRESS
807      ;   DO FOR (R0) := BE1DB TO BE1CR1
808      ;     LET (R0) := 0
809      ;
810      ;   ENDDO
811      ;   RESTORE R1,R0
812      ;   RETURN
813

```

```

814      ; ENDROUTINE
815

```

```

816
817 002502      IUUBE:
      002502 010046      MOV     R0,-(SP)          ;;PUSH R0 ON STACK
      002504 010146      MOV     R1,-(SP)          ;;PUSH R1 ON STACK
818 002506 013700 002202      MOV     BE1DB,R0         ; POINT R0 TO UBE REGISTERS
819 002512 012701 000010      MOV     #10,R1          ; SET UP A LOOP COUNTER
820 002516 005020      5$:    CLR     (R0)+          ; CLEAR OUT A REGISTER
821 002520 077102                ;      SOB     R1,5$         ; HAVE WE INITIALIZED THE UBE ?
822 002522 012601                ;      MOV     (SP)+,R1      ;;POP STACK INTO R1
      002524 012600                ;      MOV     (SP)+,R0      ;;POP STACK INTO R0
823 002526 000207      RTS      PC
824
825

```

SUBROUTINE INITIALIZE THE UBE'S

```

826 ;
827 ; STARTING POINT OF PROGRAM
828 ;
829 ;
830 002530 START:
      ; LCP/ORION ROUTINE TO SAVE EMTULATOR AND PRIORITY
EMTSAV: TST SAV30 ; FIRST TIME THROUGH ?
          BNE VMKOR ; BRANCH IF BEEN HERE ALREADY
          BIT #BITS5,@#52 ; ARE WE IN UFD MODE ?
          BEQ VMKOR ; LEAVE IF NOT
          MOV #-1,UFDPLG ; SET UFD FLAG
          BIT #BIT6,@#52 ; ARE WE IN QUIET MODE ?
          BEQ 1$ ; BR IF NOT
          MOV #-1,UQUIET ; SET QUIET MODE
1$: EMT 42 ; GET ADDRESS OF XXDP DCA TABLE
      CLR 42(R0) ; CLR XXDP+ "DRSERR"
      MOV 30,SAV30 ; SAVE EMULATOR ADDRESS
      MOV 32,SAV32 ; SAVE EMULATOR PRIORITY LEVEL
      BR VMKOR ; GET AROUND TAG AREA
      SAV30: .WORD 0 ; PUT EMULATOR INFO HERE
      SAV32: .WORD 0 ; PUT PRIORITY LOCATION HERE
      UFDPLG: .WORD 0 ; USER FRIENDLY MODE FLAG
      UQUIET: .WORD 0 ; UFD QUIET MODE FLAG
      VMKOR:

```

```

831 ;*****
      .SBTTL INITIALIZE THE COMMON TAGS
      ;CLEAR THE COMMON TAGS ($CMTAG) AREA
      MOV #CMTAG,R6 ;FIRST LOCATION TO BE CLEARED
      CLR (R6)+ ;CLEAR MEMORY LOCATION
      CMP #SWR,R6 ;;DONE?
      BNE -.6 ;;LOOP BACK IF NO
      MOV #STACK,SP ;;SETUP THE STACK POINTER
      ;INITIALIZE A FEW VECTORS
      MOV #SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
      MOV #340,@#IOTVEC+2 ;;LEVEL 7
      MOV #ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
      MOV #340,@#EMTVEC+2 ;;LEVEL 7
      MOV #TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
      MOV #340,@#TRAPVEC+2;LEVEL 7
      MOV #PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
      MOV #340,@#PWRVEC+2 ;;LEVEL 7
      MOV #ENDCT,#EOPCT ;;SETUP END-OF-PROGRAM COUNTER
      CLR #TIMES ;;INITIALIZE NUMBER OF ITERATIONS
      CLR #ESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
      MOV #1,#ERMAX ;;ALLOW ONE ERROR PER TEST
      MOV #,$LPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
      MOV #,$LPERR ;;SETUP THE ERROR LOOP ADDRESS
      ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
      ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
      MOV @#ERRVEC,-(SP) ;;SAVE ERROR VECTOR
      MOV #30000@,#ERRVEC ;;SET UP ERROR VECTOR
      MOV #DSWR,SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
      MOV #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
      CMP #-1,@SWR ;;TRY TO REFERENCE HARDWARE SWR
      BNE 30002$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
      ;;AND THE HARDWARE SWR IS NOT = 1
      BR 30001$ ;;BRANCH IF NO TIMEOUT

```

INITIALIZE THE COMMON TAGS

```

003026 012716 003034      30000$: MOV    #30001$, (SP)    ;;SET UP FOR TRAP RETURN
003032 000002
003034 012737 000176 001140 30001$: MOV    #SWREG,SWR    ;;POINT TO SOFTWARE SWR
003042 012737 000174 001142      MOV    #DISPREG,DISPLAY
003050 012637 000004      30002$: MOV    (SP)+, @#ERRVEC    ;;RESTORE ERROR VECTOR
003054 005037 001206      CLR    $PASS                ;;CLEAR PASS COUNT
003060 132737 000200 001221      BITB   #APTSIZE, $ENVM      ;;TEST USER SIZE UNDER APT
003066 001403      BEQ    30003$              ;;YES, USE NON-APT SWITCH
003070 012737 001222 001140      MOV    #SWREG,SWR          ;;NO, USE APT SWITCH REGISTER
003076
832 003076 005737 002624      30003$: TST    UQUIET          ; ARE WE UNDER UFD QUIET MODE ?
833 003102 001056      BNE    1$                  ; YES, THEN SKIP THE DIAGNOSTIC TITLE PRINTOUT
834 003104 122737 000001 001220      CMPB   #APTENV, $ENV       ; APT?
835 003112 001452      BEQ    1$                  ; IF YES, SKIP PRINTOUT
836
.SBTTL TYPE PROGRAM NAME
;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
003114 005227 177777      INC    #-1                 ;;FIRST TIME?
003120 001047      BNE    30004$              ;;BRANCH IF NO
003122 022737 023340 000042      CMP    #ENDAD, @#42        ;;ACT-11?
003130 001443      BEQ    30004$              ;;BRANCH IF YES
003132 104401 003200      TYPE   ,30005$            ;;TYPE ASCIZ STRING
.SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
003136 005737 000042      TST    @#42                ;;ARE WE RUNNING UNDER XXDP/ACT?
003142 001012      BNE    30006$              ;;BRANCH IF YES
003144 123727 001220 000001      CMPB   $ENV, #1            ;;ARE WE RUNNING UNDER APT?
003152 001406      BEQ    30006$              ;;BRANCH IF YES
003154 023727 001140 000176      CMP    SWR, #SWREG         ;;SOFTWARE SWITCH REG SELECTED?
003162 001005      BNE    30007$              ;;BRANCH IF NO
003164 104406      GTSWR
003166 000403      BR     30007$              ;;GET SOFT-SWR SETTINGS
003170 112737 000001 001134 30006$: MOVB   #1, $AUTOB        ;;SET AUTO-MODE INDICATOR
003176 000420      30007$: BR     30004$          ;;GET OVER THE ASCIZ
003240      ;;30005$: .ASCIZ <CRLF>* COKTACO KTJ11-B DIAGNOSTIC *<CRLF>
837 003240 012737 002266 000004 30004$: 1$: MOV    #TIMOUT, @#4    ; SETUP TIMEOUT VECTOR FOR UNEXPECTED TIMEOUT
838 003246 012737 000340 000006      MOV    #340, @#6          ;
839 003254 005037 001716      CLR    TEST                ; TEST # FOR ERROR REPORTING
840
; SIZE MEMORY IN 4K PAGES
841 003260 004737 002276      JSR    PC, MAPPR           ; REMAP PROGRAM AREA
842 003264 052737 000060 172516      BIS    #BIT05!BIT04, MMR3  ; ENABLE RELOCATION AND 22BITS
843 003272 005237 177572      INC    MMRO                ; ENABLE MMU
844 00 276 004737 002364      JSR    PC, MEMSIZ          ; GET HIGHEST 128K OF MEMORY
845 00 302 005037 177572      CLR    MMRO                ; DISABLE MMU
846 00 306 005037 172516      CLR    MMR3                ;
847 00 112 013737 172354 001720      MOV    KIPAR6, PMIS        ; STORE HIGHEST PAGE
848 003 20
849 003 20 012706 001100      RESTART: MOV    #1100, SP    ; SET UP THE STACK POINTER
850
851
852

```


TEST UNIBUS MAP REGISTER TESTS

```

854 .SBTTL TEST - UNIBUS MAP REGISTER TESTS
855
856 ; * THIS TEST WILL TRY TO ACCESS ALL THE MAP
857 ; * REGISTERS WITH BOTH MAPPING ENABLED AND
858 ; * DISABLED
859 ;
860 ; BGNTST
861 ;
862 ; LET 4:= ADDRESS OF ERROR ROUTINE
863 ; LET 6:= PRIORITY OF 7
864 ; DO FOR BOTH UNIBUS MAP ENABLED AND DISABLED
865 ; . DO FOR ADDRESS := 170200 TO 170376
866 ; . . READ ADDRESS
867 ; . . IF ADDRESS CAN'T BE READ THEN
868 ; . . . TRAP THRU VECTOR 4
869 ; . . . ENDDO
870 ; . ENDDO
871 ; ENDDO
872 ; EXIT TEST
873 ;
874 ; ERROR:
875 ; REPORT WHICH MAP REGISTER TIMED OUT
876 ;
877 ; ENDTST
878 ;
879 ; -----
880 ; *****
881 ; *TEST 1 UNIBUS MAP REGISTER RESPONSE TEST
882 ; *****
883 ;
884 ; TST1: SCOPE
885 ;
886 ; SET UP MEMORY TIMEOUT VECTOR TO ERROR ROUTINE
887 ;
888 ; MOV @#4,R5 ; SAVE THE TIMEOUT VECTOR
889 ; MOV @20$,@#4 ; SET UP TIMEOUT VECTOR TO ERROR ROUTINE
890 ; MOV @340,@#6 ; SET UP TIMEOUT PRIORITY
891 ;
892 ; DISABLE UNIBUS MAPPING
893 ;
894 ; BIC @#BITS,@#SR3 ; DISABLE UNIBUS MAPPING
895 ;
896 ; TRY TO ACCESS ALL THE UNIBUS MAP REGISTERS
897 ;
898 ; CLR R1 ; INITIALIZE LOOP COUNT
899 5$: MOV @MAPL00,R0 ; . RO POINTS TO MAP REGISTER #0
900 10$: TST (R0)+ ; . . TRY TO READ MAP REGISTER
901 12$: CMP R0,@170400 ; . . HAVE WE READ ALL THE MAP REGISTERS
902 ; BNE 10$ ; . . NO THEN GO READ THE NEXT ONE
903 ; TST R1 ; . HAVE WE DONE THIS TWICE ?
904 ; BEQ 15$ ; . GO TURN ON UNIBUS MAP AND TRY AGAIN
905 ; BIC @#BITS,SR3 ; . TURN OFF UNIBUS MAPPING
906 ; MOV R5,@#4 ; RESTORE THE TIMEOUT VECTOR
907 ; BR TST2 ; . GO TO NEXT TEST
908 15$: INC R1 ; . INCREMENT LOOP COUNT
909 ; BIS @#BITS,SR3 ; . TURN ON UNIBUS MAP

```

T1 UNIBUS MAP REGISTER RESPONSE TEST

```

908 003422 000755                    BR      5+                    ; . GO TRY TO ACCESS THE REGISTERS
909                                    ;
910                                    ; MAP REGISTER TIMEOUT ROUTINE
911                                    ;
912 003424 005726                    20+: TST      (SP)+                    ; CLEAN UP THE STACK
913 003426 005726                    TST      (SP)+                    ;
914 003430 162700 000002             SUB      #2,RO                    ; GET ADDRESS THAT TIMED OUT
915 003434 010037 001122             MOV      RO,$BDADR                ; PUT IT IN $BDADR
916 003440 062700 000002             ADD      #2,RO                    ; GET NEXT ADDRESS
917 003444 104001                    ERROR    +1                    ; REPORT THAT ACCESS HAS TIMED OUT
918 003446 000746                    BR      12+                    ;
919

```

TEST UNIBUS MAP REGISTER BIT PATTERN

```

921 .SBTTL TEST - UNIBUS MAP REGISTER BIT PATTERN
922
923 ;* THIS TEST IS DESIGNED TO DETECT ANY STUCK
924 ;* AT BITS OR SHORTED BITS IN THE UNIBUS MAP
925 ;* REGISTERS
926 ;
927 ; BGNTST
928 ;
929 ; CLEAR ALL MAP REGISTERS
930 ; DO FOR ADDRESS := 170200 TO 170376
931 ;*
932 ;* CHECK LOW MAPPING REGISTERS
933 ;*
934 ; . READ ADDRESS
935 ; . IF ADDRESS NE 0 THEN
936 ; . . ERROR MAP REGISTER DID NOT CLEAR
937 ; . ENDF
938 ; . DO FOR PATTERNS := 377,7417,1463,52525,125252
939 ; . . WRITE PATTERN INTO ADDRESS
940 ; . . READ PATTERN
941 ; . . IF CONTENTS OF ADDRESS NE PATTERN THEN
942 ; . . . ERROR IN BITS OF ADDRESS
943 ; . . ENDF
944 ; . ENDDO
945 ; . LET ADDRESS := ADDRESS * 2
946 ;*
947 ;* CHECK HIGH MAPPING REGISTERS
948 ;*
949 ; . READ ADDRESS
950 ; . IF ADDRESS NE 0 THEN
951 ; . . ERROR MAP REGISTER DID NOT CLEAR
952 ; . ENDF
953 ; . DO FOR PATTERNS := 17,14,25,52
954 ; . . WRITE PATTERN INTO ADDRESS
955 ; . . READ PATTERN
956 ; . . IF CONTENTS OF ADDRESS NE PATTERN THEN
957 ; . . . ERROR IN BITS OF ADDRESS
958 ; . . ENDF
959 ; . ENDDO
960 ; ENDDO
961 ;
962 ; ENDTST
963 ;
964 ;-----
965 ;*****
966 ;*TEST 2 UNIBUS MAP REGISTER BIT PATTERN TEST
;*****

```

003450 000004

TST2: SCOPE

```

967
968
969
970
971 003452 012700 170200
972 003456 005020
973 003460 020027 170400
974 003464 001374

```

```

; CLEAR ALL MAP REGISTERS
;
5$: MOV #MAPL00,R0 ; GET ADDRESS TO 1ST MAP REGISTER
CLR (R0)+ ; . CLEAR THE MAP REGISTER
CMP R0,#170400 ; . HAVE WE CLEARED ALL THE MAP REGISTERS ?
BNE 5$ ; . NO THEN GO CLEAR THE NEXT ONE

```

T2 UNIBUS MAP REGISTER BIT PATTERN TEST

```

975 ;
976 ; DO A READ,WRITE,READ ON THE REGISTERS TO
977 ; VERIFY ALL THE BITS OF THE LOW MAP REGISTERS
978 ;
979 003466 012700 170200          ; MOV #MAPL00,R0 ; GET ADDRESS TO 1ST MAP REGISTER
980 003472 005710          10$: TST (R0) ; . IS THE MAP REGISTER CLEAR ?
981 003474 001410          BEQ 15$ ; . YES,THEN GO CHECK THE BITS
982 003476 011037 001126      MOV (R0),#BDDAT ; . NO,THEN GET THE BAD DATA
983 003502 012737 000000 001124 MOV #0,#GDDAT ; . GET THE GOOD DATA
984 003510 010037 001122      MOV R0,#BDADR ; . GET THE ADDRESS THAT DID NOT CLEAR
985 003514 104002          ERROR +2 ; . ERROR - REGISTER DID NOT CLEAR
986 003516 012701 001774      15$: MOV #PTRN16,R1 ; . GET POINTER TO PATTERN TABLE
987 003522 011110          20$: MOV (R1),(R0) ; . MOVE PATTERN TO MAP REGISTER
988 003524 011102          MOV (R1),R2 ; . MOVE COPY OF PATTERN TO R2
989 003526 042702 000001      BIC #BIT0,R2 ; . CLEAR BIT0 OF THE PATTERN
990 003532 020210          CMP R2,(R0) ; . DID PATTERN GET WRITTEN CORRECTLY ?
991 003534 001407          BEQ 25$ ; . YES,THEN GO SEE IF ALL PATTERNS HAVE BEEN WRIT

TEN
992 003536 011037 001126      MOV (R0),#BDDAT ; . . NO,GET THE BAD DATA
993 003542 010237 001124      MOV R2,#GDDAT ; . . GET THE GOOD DATA
994 003546 010037 001122      MOV R0,#BDADR ; . . GET THE ADDRESS OF DATA FAULT
995 003552 104003          ERROR +3 ; . . ERROR - REGISTER COULD NOT HOLD PATTERN
996 003554 005721          25$: TST (R1)+ ; . . ARE WE THROUGH THE TABLE ?
997 003556 001361          BNE 20$ ; . . NO THEN GO WRITE NEXT PATTERN
998 ;
999 ; DO A READ,WRITE,READ ON THE REGISTERS TO
1000 ; VERIFY ALL THE BITS OF THE HIGH MAP REGISTERS
1001 ;
1002 003560 062700 000002      ; ADD #2,R0 ; . POINT TO THE HIGH MAPPING REGISTER
1003 003564 005710          30$: TST (R0) ; . IS THE MAP REGISTER CLEAR ?
1004 003566 001410          BEQ 35$ ; . YES,THEN GO CHECK THE BITS
1005 003570 011037 001126      MOV (R0),#BDDAT ; . NO,THEN GET THE BAD DATA
1006 003574 012737 000000 001124 MOV #0,#GDDAT ; . GET THE GOOD DATA
1007 003602 010037 001122      MOV R0,#BDADR ; . GET THE ADDRESS THAT DID NOT CLEAR
1008 003606 104002          ERROR +2 ; . ERROR - REGISTER DID NOT CLEAR
1009 003610 012701 002024      35$: MOV #PTRN6,R1 ; . GET POINTER TO PATTERN TABLE
1010 003614 011110          40$: MOV (R1),(R0) ; . MOVE PATTERN TO MAP REGISTER
1011 003616 011102          MOV (R1),R2 ; . MOVE COPY OF THE PATTERN TO R2
1012 003620 042702 177700      BIC #177700,R2 ; . MASK OUT THE UPPER 10 BITS
1013 003624 020210          CMP R2,(R0) ; . DID PATTERN GET WRITTEN CORRECTLY ?
1014 003626 001407          BEQ 45$ ; . YES,THEN GO SEE IF ALL PATTERNS HAVE BEEN WRIT

TEN
1015 003630 011037 001126      MOV (R0),#BDDAT ; . . NO,GET THE BAD DATA
1016 003634 010237 001124      MOV R2,#GDDAT ; . . GET THE GOOD DATA
1017 003640 010037 001122      MOV R0,#BDADR ; . . GET THE ADDRESS OF DATA FAULT
1018 003644 104003          ERROR +3 ; . . ERROR - REGISTER COULD NOT HOLD PATTERN
1019 003646 005721          45$: TST (R1)+ ; . . ARE WE THROUGH THE TABLE ?
1020 003650 001361          BNE 40$ ; . . NO THEN GO WRITE NEXT PATTERN
1021 ;
1022 ; SEE IF WE HAVE CHECKED ALL THE MAP REGISTERS
1023 ;
1024 003652 062700 000002      ; ADD #2,R0 ; . POINT TO NEXT REGISTER
1025 003656 020027 170400      CMP R0,#170400 ; . ARE WE THROUGH ALL THE MAP REGISTERS ?
1026 003662 001303          BNE 10$ ; . GO CHECK THE NEXT REGISTER
1027
1028

```

TEST - UNIBUS MAP REGISTER ADDRESS UNIQUENESS

```

1030 .SBTTL TEST UNIBUS MAP REGISTER ADDRESS UNIQUENESS
1031 ;
1032 ;
1033 ; * THIS TEST IS DESIGNED TO DETECT ANY DUAL ADDRESSING
1034 ; * IN THE UNIBUS MAP ADDRESSING LOGIC
1035 ;
1036 ; BGNTST
1037 ;
1038 ; DO FOR ADDRESS := 170200 TO 170376
1039 ; . WRITE #ADDRESS TO ADDRESS
1040 ; ENDDO
1041 ; DO FOR ADDRESS := 170200 TO 170376
1042 ; . READ ADDRESS
1043 ; . IF CONTENTS OF ADDRESS NE ADDRESS THEN
1044 ; . . REPORT ERROR IN ADDRESSING
1045 ; . ENDF
1046 ; ENDDO
1047 ;
1048 ; ENDTST
1049 ;
1050 ;
1051 ;
1052 ;

```

```

;*****
;*TEST 3 UNIBUS MAP REGISTER UNIQUENESS TEST
;*****
TST3: SCOPE

```

003664 000004

```

1053 ;
1054 ;
1055 ; WRITE ADDRESS INTO ALL MAP REGISTERS
1056 ;
1057 003666 012700 170200 MOV #MAPL00,R0 ; GET ADDRESS OF 1ST MAP REGISTER
1058 003672 012701 170200 MOV #MAPL00,R1 ; GET ADDRESS OF 1ST MAP REGISTER
1059 003676 010021 5$: MOV R0,(R1)+ ; . WRITE ADDRESS TO ITSELF
1060 003700 062700 000002 ADD #2,R0 ; . GET THE NEXT ADDRESS
1061 003704 022700 170400 CMP #170400,R0 ; . HAVE WE WRITTEN TO ALL MAP REGISTERS ?
1062 003710 001372 BNE 5$ ; . NO,THEN GO WRITE TO NEXT ADDRESS
1063 ;
1064 ; MAKE SURE EACH LOW MAP REGISTER CONTAINS ITS ADDRESS
1065 ;
1066 003712 012700 170200 MOV #MAPL00,R0 ; GET ADDRESS TO 1ST MAP REGISTER
1067 003716 012701 170200 MOV #MAPL00,R1 ; GET ADDRESS TO 1ST MAP REGISTER
1068 003722 020021 10$: CMP R0,(R1)+ ; . WAS THE ADDRESS WRITTEN TO THE MAP REGISTER ?
1069 003724 001406 BEQ 15$ ; . YES,THEN GO CHECK THE HIGH MAP REGISTER
1070 003726 010037 001120 MOV R0,#GDADR ; . GET THE GOOD ADDRESS
1071 003732 016137 177776 001122 MOV -2(R1),#BDADR ; . GET THE BAD ADDRESS
1072 003740 104004 ERROR +4 ; . ERROR IN ADDRESSING OF MAP REGISTERS
1073 ;
1074 ; MAKE SURE EACH HIGH MAP REGISTER CONTAINS ITS ADDRESS
1075 ;
1076 003742 062700 000002 15$: ADD #2,R0 ; . POINT R0 TO HIGH MAP REGISTER
1077 003746 010002 MOV R0,R2 ; . PUT A COPY OF IT INTO R2
1078 003750 042702 177700 BIC #177700,R2 ; . MASK OUT THE UPPER 10 BITS
1079 003754 020221 CMP R2,(R1)+ ; . WAS THE ADDRESS WRITTEN TO THE HIGH MAP REGISTER
?
1080 003756 001406 BEQ 20$ ; . YES,THEN GO CHECK IF WE ARE THROUGH ALL THE MAP
REGISTERS
1081 003760 010237 001120 MOV R2,#GDADR ; . GET THE GOOD ADDRESS
1082 003764 016137 177776 001122 MOV -2(R1),#BDADR ; . GET THE BAD ADDRESS
1083 003772 104004 ERROR +4 ; . ERROR IN ADDRESSING OF MAP REGISTERS

```

T3 UNIBUS MAP REGISTER UNIQUENESS TEST

```

1084
1085
1086
1087 003774 062700 000002
1088 004000 020027 170400
1089 004004 001346
1090
1091
1092

```

```

;
; CHECK TO SEE IF WE HAVE CHECKED ALL MAP REGISTERS
;
20$:    ADD    #2,R0                    ; . GET THE NEXT REGISTER
          CMP    R0,#170400            ; . HAVE WE CHECKED ALL THE MAPPING REGISTERS
          BNE    10$                    ; . NO, THEN GO CHECK THE NEXT REGISTER

```

TEST - UNIQUENESS BETWEEN LOW MAP REGISTERS AND HIGH MAP REGIST

.SBTTL TEST UNIQUENESS BETWEEN LOW MAP REGISTERS AND HIGH MAP REGISTERS

1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118

```

;
;* THIS TEST IS DESIGNED TO TEST THE UNIQUENESS BETWEEN THE LOW
;* AND HIGH MAP REGISTERS
;
; BGNST
;
; DO FOR ADDRESS := 170200 TO 170374
; . WRITE #77777 TO ADDRESS
; ENDDO
; DO FOR ADDRESS := 170200 TO 170374 BY 4
; . LET R2 := CONTENTS OF ADDRESS
; . LET R3 := CONTENTS OF ADDRESS * 2
; . LET R3 := R3 - R2
; . IF R3 EQ 0 THEN
; . ERROR IN DIFFERENTIATING BETWEEN HI AND LO REGISTERS
; . ENDIF
; ENDDO
;
; ENDTST

```

```

;*****
;*TEST 4 HIGH AND LOW MAP REGISTER UNIQUENESS TEST
;*****
TST4: SCOPE

```

004006 000004

; WRITE #77777 TO ALL ADDRESSES

```

;
; MOV #MAPLO0,R0 ; GET ADDRESS OF FIRST MAP REGISTER
5$: MOV #77777,(R0)+ ; . WRITE 77777 TO MAP REGISTER
; CMP R0,#170400 ; . HAVE WE WRITTEN TO ALL REGISTERS ?
; BNE 5$ ; . NO,THEN GO WRITE TO THE NEXT REGISTER ?

```

; MAKE SURE THAT HIGH MAP REGISTER BITS 6-15 ARE CLEARED

```

;
; MOV #MAPLO0,R0 ; GET ADDRESS OF FIRST MAP REGISTER
10$: MOV (R0)+,R2 ; . GET CONTENTS OF LOW MAP REGISTER
; MOV (R0)+,R3 ; . GET CONTENTS OF HIGH MAP REGISTER
; SUB R3,R2 ; . DID THE REGISTERS GET ADDRESSED CORRECTLY ?
; BGT 15$ ; . YES,THEN GO SEE IF WE HAVE CHECKED ALL THE MAP R

```

EGISTERS

```

;
; CMP -(R0),-(R0) ; . NO, POINT TO THE LOW MAP REGISTER
; MOV (R0)+,$GDDAT ; . GET CONTENTS OF LOW MAP REGISTER
; MOV (R0)+,$BDDAT ; . GET CONTENTS OF HIGH MAP REGISTER
; ERROR +5 ; . ERROR - MAP REGISTERS WERE NOT ADDRESSED CORREC

```

TLY

; SEE IF WE HAVE CHECKED ALL THE MAP REGISTER PAIRS

```

;
15$: CMP R0,#170400 ; . HAVE WE CHECKED ALL THE REGISTERS ?
; BNE 10$ ; . NO,THEN GO CHECK THE NEXT PAIR !!!

```

1139
1140
1141
1142 004056 020027 170400
1143 004062 001363
1144

TEST - DCSR REGISTER RESPONSE TEST

```

1146 .SBTTL TEST - DCSR REGISTER RESPONSE TEST
1147 ;
1148 ;
1149 ; * THIS TEST IS DESIGNED TO SEE IF WE CAN ACCESS THE
1150 ; * DIAGNOSTIC CONTROLLER STATUS REGISTER. IF WE CAN IT
1151 ; * TESTS OUT BITS 3-6,9-14 OF THE DCSR AND MAKES SURE
1152 ; * THAT A BUS INIT HAS A PROPER RESPONSE IN THE DCSR.
1153 ;
1154 ;
1155 ; BGNTST
1156 ;
1157 ; LET 4 := ADDRESS OF ERROR ROUTINE
1158 ; LET 6 := PRIORITY OF 7
1159 ; LET ADDRESS := 177730
1160 ; READ ADDRESS
1161 ;
1162 ; CHECK BITS 3-6,9 14 OF THE DCSR REGISTER
1163 ;
1164 ; LET DCSR := DCSR .OR. #77170
1165 ; READ DCSR
1166 ; IF DCSR .AND. #77160 NE 0 THEN
1167 ; . REPORT ERROR IN BITS 3-6,9-14 OF THE DCSR REGISTER
1168 ; . ENDF
1169 ;
1170 ; CHECK OUT THE DCSR RESPONSE TO A RESET
1171 ;
1172 ; CACHE THE RESET INSTRUCTION
1173 ; LET DCSR := DCSR .OR. #206
1174 ; DO A RESET TO INITIALIZE THE BUS
1175 ; IF LOWER BYTE OF DCSR GE 0 THEN
1176 ; . REPORT ERROR (BIT 7 DID NOT GET SET BY BUS INIT)
1177 ; . ELSE
1178 ; . LET DCSR := DCSR .AND. #6
1179 ; . IF DCSR NE 0 THEN
1180 ; . . REPORT ERROR (BITS 1,2 DIDN'T CLEAR ON BUS INIT)
1181 ; . . ENDF
1182 ; . ENDF
1183 ; EXIT TEST
1184 ; ERROR:
1185 ; REPORT THAT ACCESS TO DCSR HAS TIMED OUT
1186 ;
1187 ; ENDTST
1188 ;
1189 ;-----
1190 ;*****
1191 ;*TEST 5 DCSR REGISTER TEST
1192 ;*****

```

004064 000004

```

1192 TST5: SCOPE
1193 ;
1194 ; SET UP MEMORY TIMEOUT VECTOR
1195 ;
1196 004066 012737 004240 000004 MOV #20$,@#4 ; SET UP VECTOR PC TO ERROR ROUTINE
1197 004074 012737 000340 000006 MOV #340,@#6 ; SET UP VECTOR PSW TO PRIORITY 7
1198 ;
1199 ; READ ADDRESS OF DCSR TO SEE IF IT TIMES OUT

```


T5 DCSR REGISTER TEST

```

1200
1201 004102 005737 177730      ;      TST      @#DCSR      ; WILL THE DCSR TIMEOUT ?
1202 004106 012737 002266 000004      ;      MOV      #TIMOUT,#4      ; RESTORE TIMEOUT VECTOR
1203
1204      ; TRY TO WRITE ONES TO BITS 3-6 AND 9-14 ( BIT 03 SHOULD STAY WRITTEN)
1205
1206 004114 052737 077170 177730      ;      BIS      #77170,@#DCSR      ; TRY TO WRITE TO READ ONLY BITS
1207 004122 032737 077160 177730      ;      BIT      #77160,@#DCSR      ; DID THEY GET WRITTEN TO ?
1208 004130 001412      ;      BEQ      5$      ; NO,THEN GO CHECK THE EFFECT OF A RESET ON THE DCSR
1209 004132 013701 177730      ;      MOV      @#DCSR,R1      ; GET COPY OF DCSR INTO R1
1210 004136 0427C1 100607      ;      BIC      #100607,R1      ; MASK OUT UNTESTED BITS
1211 004142 010137 001126      ;      MOV      R1,#BDDAT      ; PUT THEM INTO BAD DATA
1212 004146 012737 000010 001124      ;      MOV      #10 #GDDAT      ; GOOD DATA
1213 004154 104006      ;      ERROR    +6      ; ERROR IN BITS 3-6,9 14 OF DCSR
1214
1215      ; CACHE THE RESET INSTRUCTION
1216
1217 004156 012737 000200 001124 5$:      ;      MOV      #BIT7,$GDDAT      ; ONLY 7 SHOULD BE SET AFTER RESET
1218 004164 053737 177730 001124      ;      BIS      DCSR,$GDDAT      ; SET BOOT BITS
1219 004172 005737 004204      ;      TST      10$      ; CACHE THE RESET
1220
1221      ; SET UP DCSR,DO A RESET MAKE SURE PROPER RESPONSE OCCURS
1222
1223 004176 052737 000006 177730      ;      BIS      #6,@#DCSR      ; SET BIT 7 AND CLEAR BIT 1,2
1224 004204 000005      ;      RESET      ; DO A BUS INIT
1225 004206 032737 000200 177730 10$:      ;      BIT      #BIT7,@#DCSR      ; DID BIT 7 GET SET ?
1226 004214 001001      ;      BNE      15$      ; YES,THEN GO CHECK IF BITS 1,2 GOT CLEAR
1227 004216 104007      ;      ERROR    +7      ; NO,THEN ERROR - BIT 7 DIDN'T GET SET
1228 004220 032737 000006 177730 15$:      ;      BIT      #6,@#DCSR      ; DID BIT 1,2 GET CLEARED ?
1229 004226 001410      ;      BEQ      TST6      ; YES,THEN GO TO NEXT TEST
1230 004230 013737 177730 001126      ;      MOV      @#DCSR,#BDDAT      ; NO,THEN GET CONTENTS OF THE DCSR
1231 004236 104007      ;      ERROR    +7      ; ERROR BIT 1,2 DID NOT GET CLEAR
1232
1233      ; ERROR ROUTINE
1234
1235 004240 012737 177730 001122 20$:      ;      MOV      #DCSR,#BDADR      ; GET ADDRESS OF THE DCSR
1236 004246 104010      ;      ERROR    +10      ; ERROR - DCSR REGISTER ACCESS HAS TIMED OUT
1237
1238

```

TEST - KMCR BITS TEST

1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260

```

.SBTTL TEST - KMCR BITS TEST
;
;* THIS TEST CHECKS THAT OUT OF DIAGNOSTIC MODE (DCSR<8>=0), WRITE ACCESS
;* TO KMCR<5-0> IS DISABLED.
;
; BGNTST
;
;     LET 4 := ADDRESS OF TIMEOUT ROUTINE
;     LET 6 := PRIORITY OF TIMEOUT ROUTINE
;     READ THE MEMORY CONFIGURATION FROM THE EAROM
;     CHECK THAT IF DCSR<8>=0 WRITE ACCESS TO KMCR<5-0> IS DISABLED
;     IF NOT THEN
;         . ERROR
;     ENDIF
;
; ENDTST
;
;-----

```

```

;*****
;*TEST 6      KMCR MEMORY CONFIGURATION BITS TEST
;*****
TST6:  SCOPE

```

004250 000004

1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277

1266	004252	012737	004416	000004
1267	004260	012737	000340	000006
1268	004266	013737	177734	001160
1269	004274	012737	002266	000004
1270	004302	042737	177700	001160
1271	004310	022737	000077	001160
1272	004316	001003		
1273	004320	005037	001720	
1274	004324	000440		
1278	004326	013701	177734	
1279	004332	005101		
1280	004334	042701	177700	
1281	004340	042737	000400	177730
1282	004346	042701	000010	
1283	004352	050137	177734	
1284	004356	020137	177734	
1285	004362	001014		
1286	004364	013737	177734	001124
1287	004372	010137	001126	
1288	004376	012737	177734	001122
1289	004404	013737	001160	177734
1290	004412	104011		

```

;
; SETUP TIMEOUT VECTOR, TO ERROR ROUTINE
; VERIFY THAT AT LEAST SOME PMI MEMORY PRESENT
;
;     MOV     #10$, @#4      ; SET VECTOR PC TO ERROR ROUTINE
;     MOV     #340, @#6     ; SET VECTOR PSW TO ERROR ROUTINE
;     MOV     KMCR, $TMPO   ; STORE KMCR
;     MOV     @TIMOUT, @#4  ; RESTORE TIMEOUT VECTOR
;     BIC     #177700, $TMPO ; CLEAR ALL BUT U.MEMORY
;     CMP     #77, $TMPO    ; NO PMI MEMORY?
;     BNE     2$           ; IF SOME, GO TO THE NEXT TEST
;     CLR     PMIS         ; OTHERWISE FLAG NO PMI MEMORY
;     BR     TST7         ;; AND EXIT TEST
;
; CHECK THAT NOT IN DIAGNOSTIC MODE WRITE ACCESS TO KMCR<5-0> IS DISABLED
;
2$:  MOV     K1 7, R1      ; STORE PATTERN FROM KMCR TO R1
;     COM     R1          ; COMPLEMENT THE PATTERN
;     BIC     #177700, R1 ; CLEAR BITS <15-6>
;     BIC     #BIT08, DCSR ; MAKE SURE THAT NOT IN DIAGN. MODE
;     BIC     #BIT03, R1  ; ENABLE 32K, IN CASE OF ERROR
;     BIS     R1, KMCR    ; TRY TO WRITE TO KMCR<5-0>
;     CMP     R1, KMCR    ; WRITTEN OK?
;     BNE     1$         ; IF DIDN'T WRITE, BRANCH
;     MOV     KMCR, $GDDAT ; NO, THEN GET THE GOOD DATA
;     MOV     R1, $BDDAT  ; GET THE BAD DATA
;     MOV     #KMCR, $BDADR ; GET THE ADDRESS OF THE KMCR
;     MOV     $TMPO, KMCR ; RESTORE KMCR
;     ERROR   +11        ; NO, THEN ERROR BITS DID NOT GET SET CORRECTLY IN
;
;
1$:  BR     TST7         ;; GO TO THE NEXT TEST
;

```

THE KMCR

1291 004414
004414 000404
1292

T6 KMCR MEMORY CONFIGURATION BITS TEST

```
1293                                               ; TIMEOUT ROUTINE
1294                                               ;
1295 004416   012737   177734   001122   10+:   MOV    #KMCR,$BDADR      ; GET THE ADDRESS OF THE KMCR
1296 004424   104012                            ERROR   +12                 ; ERROR   MEMORY TIMEOUT HAS OCCURED
```

TEST - UNIBUS TIMEOUT TEST

```

1298 .SBTTL TEST UNIBUS TIMEOUT TEST
1299
1300 ;* THIS TEST CHECKS OUT THE UNIBUS TIMEOUT LOGIC
1301 ;* BEFORE IT IS USED IN ANY TESTS
1302 ;
1303 ; BGNST
1304 ;
1305 ; LET 4 := THE ADDRESS 10$
1306 ; LET 6 := #340
1307 ; MAKE SURE THAT UBA IS IN DIAGNOSTIC MODE,DCSR<8> IS SET
1308 ; READ UNIBUS LOCATION
1309 ; IF NO TIMEOUT THEN
1310 ; ERROR - UBA DIDN'T TIMEOUT ON UNIBUS ACCESS IN DIAGNOSTIC MODE
1311 ; ENDIF
1312 ; 10$: CLEAN UP THE STACK
1313 ;
1314 ; ENDTST
1315 ;
1316 ;-----
1317 ;*****
1318 ;*TEST 7 UNIBUS TIMEOUT TEST
1319 ;*****
004426 000004 TST7: SCOPE
1319 ;
1320 ; SET UP MEMORY TIMEOUT VECTOR AND PUT UBA INTO DIAGNOSTIC MODE
1321 ;
1322 ;
1323 004430 005737 001720 TST PMIS ; ANY PMI MEMORY?
1324 004434 001425 BEQ TST10 ;; IF NONE, EXIT TEST
1325 004436 012737 004470 000004 MOV #10$,@#4 ; SET UP TIMEOUT VECTOR TO END OF TEST
1326 004444 012737 000340 000006 MOV #340,@#6 ;
1327 004452 052737 000400 177730 BIS #BIT8,DCSR ; PUT UBA INTO DIAGNOSTIC MODE
1328 ;
1329 ; READ A UNIBUS LOCATION
1330 ;
1331 ;
1332 ;
1333 004460 005737 170002 TST @#170002 ; TRY TO READ UBE REGISTER ?
1334 004464 104036 ERROR +36 ; ERROR - SHOULD TIMEOUT HERE
1335 004466 000402 BR 11$ ; GO TO THE NEXT TEST
1336 ;
1337 ;
1338 ; TIMEOUT ROUTINE
1339 ;
1340 ;
1341 004470 005726 10$: TST (SP)+ ; CLEAN UP THE STACK
1342 004472 005726 TST (SP)+ ;
1343 004474 012737 002266 000004 11$: MOV #TIMOUT,@#4 ; RESTORE TIMEOUT VECTOR
1344 004502 042737 000400 177730 BIC #BIT8,DCSR ;
1345 ;
1346 ;

```

TEST - DATA OUT WITHOUT RELOCATION

1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374

```
.SBTTL TEST - DATA OUT WITHOUT RELOCATION
;* THIS TEST WILL PERFORM DIAGNOSTIC DATA OUT CYCLES WITHOUT ENABLING
;* RELOCATION. THE CYCLES WILL BE VERIFIED TO WORK CORRECTLY AND
;* DIAGNOSTIC NPR REGISTER WILL BE CHECKED TO BE ABLE TO STORE PROPER
;* DATA.
;
; BGNTST
;
; MAKE SURE THAT DIGNOSTIC MODE IS ON DCSR<8>=1
; SELECT DIAGNOSTIC NPR THRU DDR BY DCSR<2-1>=<0,0>
; LET TEST_LOCATION = 0
; DO FOR PATTERN = 377,7417,31463,52525,125152 (BINARY DIVIDE)
; . CALL DIAGNOSTIC_DATA_OUT <TEST_LOCATION,PATTERN>
; . IF DDR NE PATTERN THEN
; . . ERROR IN PATH
; . . ENDF
; . IF TEST_LOCATION NE PATTERN THEN
; . . ERROR PERFORMINDG DATA OUT
; . . ENDF
; ENDDO
;
; ENDTST
```

```
-----
;*****
;*TEST 10 DATA OUT WITHOUT RELOCATION
;*****
TST10: SCOPE
```

004510	000004								
1375									
1376	004512	005737	001720			TST	PMIS		; ANY PMI MEMORY?
1377	004516	001451				BEQ	TST11	::	IF NONE, EXIT TEST
1378	004520	052737	000400	177730		BIS	#BIT08,DCSR		; MAKE SURE DIAGNOSTIC MODE IS ON
1379	004526	042737	000006	177730		BIC	#BIT02:BIT01,DCSR		; MAKE SURE NPR REGISTER IS SELECTED
1380	004534	012702	001774			MOV	#PTRN16,R2		; POINT TO BINARY DIVIDE PATTERN
1381	004540	012701	001160			MOV	#\$TMPO,R1		; STORE TEST LOCATION
1382	004544	005011				CLR	(R1)		; CLEAR TEST LOCATION
1383	004546	032737	000200	177730		BIT	#BIT07,DCSR		; READY BIT SET?
1384	004554	001001				BNE	1\$; IF SO, BRANCH
1385	004556	104017				ERROR	+17		; DCSR<7> NOT SET
1386	004560	011237	001124		1\$:	MOV	(R2),#GDDAT		; . STORE PATTERN
1387	004564	004737	002242			JSR	PC,DDOUT		; . PERFORM DIAGNOSTIC DATA OUT
1388	004570	023737	177732	001124		CMP	DDR,#GDDAT		; . DDR HAS PROPER PATTERN
1389	004576	001404				BEQ	2\$; . IF YES, BRANCH
1390	004600	013737	177732	001126		MOV	DDR,#BDDAT		; . STORE DDR FOR ERROR REPORTS
1391	004606	104013				ERROR	+13		; . ERROR IN DATA PATH
1392	004610	023737	001160	001124	2\$:	CMP	#\$TMPO,#GDDAT		; . DATA OUT OK?
1393	004616	001404				BEQ	3\$; . BRANCH, IF YES
1394	004620	013737	001160	001126		MOV	#\$TMPO,#BDDAT		; . STORE MEMORY FOR ERROR REPORTS
1395	004626	104014				ERROR	+14		; . ERROR IN DATA OUT
1396	004630	005722			3\$:	TST	(R2)+		; . LAST PATTERN?
1397	004632	001352				BNE	1\$; . IF NOT, BRANCH
1398	004634	042737	000400	177730		BIC	#BIT8,DCSR		

TEST - DATA IN WITHOUT RELOCATION

1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420

```

.SBTTL TEST - DATA IN WITHOUT RELOCATION
;* THIS TEST WILL PERFORM DIAGNOSTIC DATA IN CYCLE AND VERIFY ITS
;* OPERATION
;
;   BGNTST
;
;   MAKE SURE THAT DIGNOSTIC MODE IS ON DCSR<8>=1
;   SELECT DIAGNOSTIC NPR THRU DDR BY DCSR<2-1>=<0,0>
;   LET PATTERN = 52525
;   LET DCSR<0> = #1
;   WRITE PATTERN TO TEST_LOCATION
;   IF DDR NE PATTERN THEN
;     . ERROR PERFORMING DATA IN
;   ENDIF
;
;   ENDTST

```

```

;*****
;*TEST 11      DATA IN WITHOUT RELOCATION
;*****
TST11: SCOPE

```

```

004642 000004
1421
1422 004644 005737 001720
1423 004650 001435
1424 004652 052737 000400 177730
1425 004660 042737 000006 177730
1426 004666 012702 001774
1427 004672 011237 001124
1428 004676 052737 000001 177730
1429 004704 013737 001124 001160
1430
1431
1432
1433 004712 023737 177732 001124
1434 004720 001404
1435 004722 013737 177732 001126
1436 004730 104015
1437 004732 005722
1438 004734 001356
1439 004736 042737 000400 177730

```

```

; ANY PHI MEMORY?
; ; IF NONE, EXIT TEST
; MAKE SURE DIAGNOSTIC MODE IS ON
; MAKE SURE NPR REGISTER IS SELECTED
; GET TEST PATTERN TABLE
; USE TEST PATTERN
; SET GO BIT
; STORE PATTERN IN TEST LOCATON

; NOW COMPARE THE RESULTS
;
; DDR GOT DATA OK?
; IF NO, ERROR
; STORE DDR FOR ERROR REPORTS
; ERROR IN DATA IN
; ARE WE DONE?
; NO

TST PHIS
BEQ TST12
BIS #BIT08,DCSR
BIC #BIT02!BIT01,DCSR
MOV #PTRN16,R2
5$: MOV (R2),#GDDAT
BIS #BIT00,DCSR
MOV #GDDAT,#TMPO

CMP DDR,#GDDAT
BEQ 10$
MOV DDR,#BDDAT
ERROR +15
10$: TST (R2)+
BNE 5$
BIC #BIT08,DCSR

```

TEST - CONTENT OF DDR

1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459

.SBTTL TEST - CONTENT OF DDR

;* THIS TEST VERIFIES THAT IF DCSR<2-1> ARE NOT ZERO, DDR IS ALWAYS
;* READ AS 0, MEANING THAT NONE OF THE UNIBUS LINES ARE STUCK.

```
;
; BGNST
;
; DO FOR DCSR<2-1> FROM <0,1> TO <1,1>
; DO DATI
; IF DDR NE 0 THEN
; . ERROR IN UNIBUS LINES
; ENDOIF
; ENDDO
;
; ENDTST
;
```

;*TEST 12 CONTENT OF DDR

004744 000004
1460 004746 005737 001720
1461 004752 001445
1462 004754 052737 000400 177730
1463 004762 052737 000002 177730
1464 004770 012703 000003
1465 004774 005037 001124
1466 005000 005001
1467 005002 004737 002254
1468 005006 022703 000001
1469 005012 001403
1470 005014 005737 177732
1471 005020 001413
1472 005022 012737 177400 001124
1473 005030 032737 000373 177732
1474 005036 001404
1475 005040 013737 177732 001126
1476 005046 104016
1477 005050 062737 000002 177730
1478 005056 077330
1479 005060 042737 000400 177730
1480

```
TST12: SCOPE
; ANY PMI MEMORY?
TST PMIS
BEQ TST13 ;; IF NONE, EXIT TEST
BIS #BIT08,DCSR ; MAKE SURE IN DIAG. MODE
BIS #BIT01,DCSR ; START WITH 1 IN DCSR
MOV #3,R3 ; DO 3 TIMES
CLR #GDDAT ; RECIEVED DATA 0
1#: CLR R1 ; . ADDRESS 0
JSR PC,DDIN ; . DO DIAGNOSTIC DATI
CMP #1,R3 ; . CONTROL LINES SELECTED?
BEQ 2# ; . IF YES, CHECK IT
TST DDR ; . ALL ZEROES?
BEQ 3# ; . IF YES, BRANCH
2#: MOV #177400,#GDDAT ; . EXPECTED PATTERN
BIT #373,DDR ; . SELECT ONLY USED
BEQ 3# ; . IF ALL ZEROES, BRANCH
MOV DDR,#BDDAT ; . STORE RECIEVED DATA
ERROR +16 ; . ERROR IN UNIBUS LINES
3#: ADD #BIT01,DCSR ; . GET NEXT SET OF UNIBUS LINES
SOB R3,1# ; . DO FOR ALL COMBINATIONS
BIC #BIT8,DCSR ; .
```

TEST - INDIRECT ACCESSING OF UNIBUS MAP REGISTERS

```

1482      .SBTTL TEST - INDIRECT ACCESSING OF UNIBUS MAP REGISTERS
1483
1484      ; * THIS TEST WILL CHECK THAT EACH OF THE UNIBUS MAP REGISTER PAIRS CAN BE
1485      ; * ACCESSED INDIRECTLY AND THAT RELOCATION WORKS PROPERLY
1486      ; * DATA OUT CYCLES WILL BE PERFORMED. EACH REGISTER EXCEPT THE ONE
1487      ; * UNDER TEST WILL POINT TO NXM 17760000. THE REGISTER PAIR UNDER TEST
1488      ; * WILL POINT TO 0, AND DIAGNOSTIC DATA OUT CYCLES WILL BE PERFORMED
1489      ; * TO PHYSICAL LOCATION 0. IN UFD MODE THIS TEST WILL RUN ONLY IF
1490      ; * KMCR<5>=0, I.E. 22 BIT MODE.
1491      ;
1492      ; BGNTST
1493      ;
1494      ;     IF UFD MODE AND KMCR<5> EQ #1 THEN
1495      ;     . EXIT TEST (RELOCATIO.) DOES NOT HAVE TO BE PERFORMED IN 18 BIT)
1496      ;     ENDIF
1497      ;     MAKE SURE THAT DIGNOSTIC MODE IS ON DCSR<8>=1
1498      ;     SELECT DIAGNOSTIC NPR THRU DDR BY DCSR<2-1>=<0,0>
1499      ;     DO FOR 32. MAPPING REGISTERS
1500      ;     . LET LO REGISTER = #160000
1501      ;     . LET HI REGISTER = #77
1502      ;     ENDDO (ALL REGISTERS POINT TO NXM 17760000)
1503      ;     POINT TIMEOUT_ROUTINE TO VECTOR 4
1504      ;     CALL MAP_PROGRAM_AREA
1505      ;     LET MMR0<0> = 1 TO ENABLE MMU
1506      ;     LET MMR3<5> = 1 TO ENABLE RELOCATION
1507      ;     POINT R0 TO ADDRESS OF MAP REGISTER 0
1508      ;     LET KIPAR6 = #0 TO ACCESS REGISTER 0
1509      ;     LET (R0) = #0
1510      ;     LET (R0)*2 = #0 TO USE ONLY 16 BITS
1511      ;     LET R1 = #140000 TO ACCESS THRU KIPAR6
1512      ;     IF NOT UFD THEN
1513      ;     . SAVE KMCR
1514      ;     . CLEAR KMCR<4-0>
1515      ;     . LET R4 = #31. (DO FOR ALL REGISTERS)
1516      ;     ELSE
1517      ;     . LET R4 = 31. - (KMCR<4-1>) ONLY NOT DISABLED REGISTERS
1518      ;     ENDIF
1519      ;     DO FOR R3 FROM #0 TO R4 FOR ALL REGISTERS
1520      ;     . CLEAR #00 (THIS LOCATION IS ACTUALLY USED)
1521      ;     . LET PATTERN = R0 (ADDRESS OF LOW REGISTER)
1522      ;     . CALL DIAGNOSTIC_DATA_OUT <(R1),PATTERN>
1523      ;     . IF TIMEOUT THEN
1524      ;     .     ERROR IN UNIQUE ADDRESSING OF REGISTERS
1525      ;     .     ENDIF
1526      ;     . IF #00 NE PATTERN THEN
1527      ;     .     ERROR PERFORMING DATA OUT
1528      ;     .     ENDIF
1529      ;     . LET (R0)* = #160000
1530      ;     . LET (R0)* = #77 TO MAP JUST USED PAIR TO NXM
1531      ;     . LET (R0) = #0
1532      ;     . LET (R0)*2 = #0 TO MAP NEXT PAIR TO TEST_LOCATION
1533      ;     . LET KIPAR6 = KIPAR6 + #200 TO ACCESS NEXT PAIR
1534      ;     ENDDO
1535      ;     DISABLE MMU
1536      ;
1537      ; ENDTST
1538

```


TEST - INDIRECT ACCESSING OF UNIBUS MAP REGISTERS

```

1539 ;-----
1540 ;*****
1541 ;*TEST 13      INDIRECT ACCESSING OF UNIBUS MAP REGISTERS
;*****
005066 000004 TST13: SCOPE
1542
1543 005070 005737 001720      TST      PMIS          ; ANY PMI MEMORY?
1544 005074 001576      BEQ      TST14          ;; IF NONE, EXIT TEST
1545 005076 032737 000040 000052  BIT      #BIT05,@#52    ; UFD MODE ?
1546 005104 001406      BEQ      1$            ; BRANCH, IF NOT
1547 005106 032737 000040 177734  BIT      #BIT05,KMCR    ; 18 BIT MODE ?
1548 005114 001402      BEQ      1$            ; BRANCH, IF NOT
1549 005116 000137 005452      JMP      10$           ; EXIT TEST
1550 005122 052737 000400 177730 1$:  BIS      #BIT08,DCSR   ; SET DIAGNOSTIC MODE
1551 005130 042737 000006 177730  BIC      #BIT02:BIT01,DCSR ; SELECT NPR THRU DDR
1552
1553 ; POINT EACH OF MAPPING REGISTER PAIRS TO 17760000
1554 ;
1555 005136 012701 000040      MOV      #32.,R1       ; DO FOR 32. REGISTER PAIRS
1556 005142 012702 170200      MOV      #MAPL00,R2    ; POINT TO ADDRESS OF THE FIRST REGISTER
1557 005146 012722 160000 2$:  MOV      #160000,(R2)+ ; . 160000 -> LO REGISTER
1558 005152 012722 000077      MOV      #77,(R2)+    ; . 77 -> HI REGISTER
1559 005156 077105      SOB     R1,2$         ; . CONTINUE UNTILL ALL DONE
1560 ;
1561 ; INITIALISE TO DO RELOCATION IN 22 BIT MODE
1562 ;
1563 005160 012737 002266 000004  MOV      #TIMOUT,@#4   ; POINT TIMEOUT ROUTINE
1564 005166 004737 002276      JSR     PC,MAPPR      ; REMAP PROGRAM AREA
1565 005172 005237 177572      INC     MMR0          ; TURN ON MMU
1566 005176 052737 000040 172516  BIS      #BIT05,MMR3   ; ENABLE RELOCATION
1567 005204 012700 170200      MOV      #MAPL00,R0   ; POINT R0 TO FIRST REGISTER
1568 005210 012737 000000 172354  MOV      #0, KIPAR6    ; LET KIPAR6 SELECT MAP PAIR 0
1569 005216 012710 000000      MOV      #0,(R0)     ; MOVE 0 TO LOW ADDRESS
1570 005222 012760 000000 000002  MOV      #0,2(R0)     ; MOVE 0 TO HI ADDRESS
1571 005230 012701 140000      MOV      #140000,R1   ; TO ACCESS THRU KIPAR6
1572 ;
1573 ; STORE # OF REGISTERS TO TEST IN R4
1574 ;
1575 005234 032737 000040 000052  BIT      #BIT05,@#52   ; UFD MODE ?
1576 005242 001010      BNE     3$            ; IF YES, BRANCH
1577 005244 013737 177734 001162  MOV      KMCR,$TMP1    ; SAVE KMCR
1578 005252 005037 177734      CLR     KMCR          ; DO NOT DISABLE ANY REGISTERS
1579 005256 012704 000036      MOV      #30.,R4     ; DO FOR ALL REGISTERS
1580 005262 000407      BR     4$            ; BRANCH AROUND
1581 005264 113703 177734 3$:  MOV     KMCR,R3       ; STORE # OF DISABLED REGISTERS
1582 005270 042703 177740      BIC     #177740,R3    ; LEAVE JUST BITS <4-0>
1583 005274 012704 000036      MOV      #30.,R4     ; STORE MAX. # OF REGISTERS
1584 005300 160304      SUB     R3,R4        ; GET # OF ACCESSABLE REGISTERS
1585 ;
1586 ; DO DATA OUT FOR EACH REGISTER PAIR UNDER TEST
1587 ;
1588 005302 005003 4$:  CLR     R3           ; START WITH REGISTER PAIR 0
1589 005304 005037 000000 5$:  CLR     @#0         ; . CLEAR LOCATION UNDER TEST
1590 005310 005037 001730      CLR     TOUT        ; . CLEAR TIMEOUT FLAG
1591 005314 010037 001124      MOV     R0,$GDDAT   ; . PATTERN IS ADDRESS OF LO REGISTER
1592 005320 004737 002242      JSR     PC,DDOUT    ; . DO DIAGN. DATA OUT

```

T13 INDIRECT ACCESSING OF UNIBUS MAP REGISTERS

1593	005324	005737	001730		TST	TOUT		; . TIMEOUT FLAG SET?
1594	005330	001411			BEQ	64		; . IF NOT BRANCH
1595	005332	013737	172354	001122	MOV	KIPAR6,\$BDADR		; . STORE # OF PAIR FOR ERRORS
1596	005340	012705	000007		MOV	#7,R5		; . PREPARE TO SHIFT TO GET TO LOWER BITS
1597	005344	006237	001122	114:	ASR	\$BDADR		; . . SHIFT ONCE
1598	005350	077503			SOB	R5,114		; . . SHIFT 7 TIMES
1599	005352	104020			ERROR	+20		; . IN UNIQUE ADDRESSING OF REGISTERS
1600	005354	023737	000000	001124	64:	CMP	#0,\$GDDAT	; . DATA OUT OK?
1601	005362	001404			BEQ	74		; . IF YES, BRANCH
1602	005364	013737	000000	001126	MOV	#0,\$BDDAT		; . STORE FOR ERROR REPORTS
1603	005372	104013			ERROR	+13		; . PERFORMING DATA OUT
1604	005374	012720	160000	74:	MOV	#160000,(R0)+		; . POINT JUST USED REGISTER
1605	005400	012720	000077		MOV	#77,(R0)+		; . TO NXM
1606	005404	012710	000000		MOV	#0,(R0)		; . POINT NEXT REGISTER PAIR
1607	005410	012760	000000	000002	MOV	#0,2(R0)		; . TO LOCATION 0
1608	005416	062737	000200	172354	ADD	#200,KIPAR6		; . ACCESS NEXT REGISTER PAIR
1609	005424	062703	000001		ADD	#1,R3		; . INCREMENT COUNT
1610	005430	020304			CMP	R3,R4		; . ALL AVAILABLE PAIRS DONE?
1611	005432	003724			BLE	54		; . BRANCH IF NOT
1612	005434	032737	000040	000052	BIT	#BIT05,#52		; CHECK TO SEE IF KMCR WAS SAVED
1613	005442	001003			BNE	104		; NO, DON'T RESTORE KMCR
1614	005444	013737	001162	177734	MOV	\$TMP1,KMCR		; RESTORE KMCR
1615	005452	005037	177572	104:	CLR	MMR0		; DISABLE MMU
1616	005456	042737	000040	172516	BIC	#BIT05,MMR3		; DISABLE MAPPING
1617	005464	042737	000400	177730	BIC	#BIT8,DCSR		; EXIT DIAGNOSTIC MODE

TEST - DISABLING OF THE MAPPING REGISTERS

```

1619          .SBTTL TEST - DISABLING OF THE MAPPING REGISTERS
1620
1621          ;* THIS TEST WILL CHECK THAT NONE OF THE REGISTERS DISABLED BY KMCR<4-0>
1622          ;* PERFORM RELOCATION. IN UFD MODE ONLY THE BITS ACTUALLY SET IN THE KMCR WILL
1623          ;* BE CHECKED, OTHREWISE ALL OF THEM EXCEPT THE FIRST 4 ARE CHECKED.
1624          ;* IN UFD MODE THIS TEST WILL RUN ONLY IF KMCR<5>=0. (IN 22 BIT MODE)
1625          ;
1626          ; BGNTST
1627          ;
1628          ; IF UFD MODE AND KMCR<5> EQ #1 THEN
1629          ; . EXIT TEST (RELOCATION DOES NOT HAVE TO BE PERFORMED IN 18 BIT)
1630          ; ENDF
1631          ; MAKE SURE THAT DIGNOSTIC MODE IS ON DCSR<8>=1
1632          ; SELECT DIAGNOSTIC NPR THRU DDR BY DCSR<2-1>=<0,0>
1633          ; CALL MAP_PROGRAM_AREA
1634          ; LET MMR3<5> = #1 TO ENABLE RELOCATION
1635          ; LET MMRO<0> = #1 TO ENABLE MMU
1636          ; LET R1 = #140000 TO ACCESS THRU KIPAR6
1637          ; LET KIPAR6 = #7600 TO ACCESS MAP REG. 32.
1638          ; SET 4 TO IGNORE TIMEOUT
1639          ; LET PATTERN = #125252
1640          ; LET @#0 = #0
1641          ; LET MAPL37 = #0
1642          ; LET MAPH37 = #0
1643          ; CALL DIAGNOSTIC_DATA_OUT<(R0),PATTERN>
1644          ; IF @#0 EQ PATTERN THEN
1645          ; . ERROR REG. PAIR 37 PERFORMS RELOCATION
1646          ; ENDF
1647          ;
1648          ; ENDTST
1649          ;
1650          ;-----
1651          ;:*****
1652          ;*TEST 14      DISABLING REGISTERS
1653          ;:*****

```

```

005472 000004
1653
1654 005474 005737 001720          TST      PMIS          ; ANY PMI MEMORY?
1655 005500 001473          BEQ      TST15          ;; IF NONE, EXIT TEST
1656 005502 032737 000040 000052          BIT      #BIT05,@#52          ; UFD MODE ?
1657 005510 001404          BEQ      1$            ; BRANCH, IF NOT
1658 005512 032737 000040 177734          BIT      #BIT05,KMCR          ; 18 BIT MODE ?
1659 005520 001063          BNE      TST15          ;;EXIT TEST IF 18 BIT MODE
1660
1661          ; INITIALISE MMU AND DIAGNOSTIC REGISTERS
1662          ;
1663 005522 052737 000400 177730 1$:      BIS      #BIT08,DCSR          ; SET DIAGNOSTIC MODE
1664 005530 042737 000006 177730          BIC      #BIT02!BIT01,DCSR    ; SELECT NPR THRU DDR
1665 005536 004737 002276          JSR      PC,MAPPR           ; REMAP PROGRAM AREA
1666 005542 005237 177572          INC      MMRO              ; TURN ON MMU
1667 005546 052737 000040 172516          BIS      #BIT05,MMR3         ; ENABLE RELOCATION
1668
1669          ; TRY TO DO RELOCATION THRU REGISTER PAIR 31.
1670          ;
1671 005554 012737 005636 000004          MOV      #2$,@#4            ; INITIALISE TIMEOUT ROUTINE
1672 005562 012701 140000          MOV      #140000,R1         ; ACCESS THRU KIPAR6

```

T14 DISABLING REGISTERS

1673	005566	012737	007600	172354	MOV	#7600,KIPAR6	; ACCESS REG. PAIR 31.
1674	005574	012737	125252	001124	MOV	#125252,\$GDDAT	; STORE PATTERN
1675	005602	005037	000000		CLR	@#0	; CLEAR LOCATION UNDER TEST
1676	005606	012737	000000	170374	MOV	#0,MAPL37	; LET MAP PAIR 31.
1677	005614	012737	000000	170376	MOV	#0,MAPH37	; POINT TO 0
1678	005622	013737	001124	177732	MOV	\$GDDAT,DDR	; STORE PATTERN IN DDR
1679	005630	005011			CLR	(R1)	; EXTENAL READ TO PROVIDE ADDRESS
1680	005632	104021			ERROR	+21	; RELOCATION PERFORMED THRU REG. 31.
1681	005634	000402			BR	3#	; EXIT TEST
1682	005636	005726			2#:	TST	(SP)+
1683	005640	005726				TST	(SP)+
1684	005642	005037	177572		3#:	CLR	MMRO
1685	005646	042737	000040	172516	BIC	#BIT05,MMR3	; DISABLE MMU
1686	005654	012737	002266	000004	MOV	#TIMOUT,@#4	; DISABLE MAPPING
1687	005662	042737	000400	177730	BIC	#BIT8,DCSR	; RESTORE TIMEOUT
							; EXIT DIAGNOSTIC MODE

TEST - NXM MEMORY TIMEOUT

```

1689 .SBTTL TEST - NXM MEMORY TIMEOUT
1690
1691 ;* THIS TEST WILL VERIFY THAT WHENEVER LOCATION 17760000 IS ACCESSED
1692 ;* DIAGNOSTIC NPR CYCLES RESULT IN A TIMEOUT SETTING BIT <15> IN DCSR.
1693 ;
1694 ; BGNSTST
1695 ;
1696 ; MAKE SURE THAT DIGNOSTIC MODE IS ON DCSR<8>=1
1697 ; SELECT DIAGNOSTIC NPR THRU DDR BY DCSR<2-1>=<0,0>
1698 ; CALL MAP_PROGRAM_AREA
1699 ; LET MMRO<0> = 1 TO ENABLE MMU
1700 ; IF NOT UFD MODE OR KMCR<5> EQ #0
1701 ; . LET TEST_LOCATION = #140000 TO ACCESS THRU KIPAR6
1702 ; . LET KIPAR6 = #0 TO ACCESS REGISTER PAIR 0
1703 ; . LET LO REGISTER = #160000 TO SET REG. 0 TO NXM
1704 ; . LET HI REGISTER = ##77
1705 ; . LET MMR3<5> = 1 TO ENABLE RELOCATION
1706 ; ELSE
1707 ; . IF KMCR<5> EQ #1 THEN(18 BIT MODE)
1708 ; . . LET TEST_LOCATION = #140000 TO ACCESS THRU KIPAR6
1709 ; . . LET KIPAR6 = #177600 TO ACCESS NXM
1710 ; . ENDF
1711 ; ENDF
1712 ; START DIAGNOSTIC_DATA_OUT TO @#0
1713 ; IF DCSR<15> NE 0 THEN
1714 ; . ERROR NXM BIT SET
1715 ; ENDF
1716 ; PUT ACTUAL EXTERNAL ADDRESS ON THE BUS
1717 ; IF DCSR<15> NE 1 THEN
1718 ; . ERROR NXM MEMORY DOES NOT SET DCSR<15>
1719 ; ENDF
1720 ;
1721 ; ENDTST
1722 ;
1723 -----
1724 ;*****
1725 ;*TEST 15 NXM MEMORY TIMEOUT
;*****

```

```

005670 000004
1726
1727 005672 005737 001720 TST PMIS ; ANY PMI MEMORY?
1728 005676 001476 BEQ TST16 ;; IF NONE, EXIT TEST
1729 005700 052737 000400 177730 BIS #BIT8,DCSR ; SELECT DIAGNOSTIC MODE
1730 005706 004737 002276 JSR PC,MAPPR ; REMAP PROGRAM AREA
1731 005712 005237 177572 INC MMRO ; ENABLE MMU
1732
1733 ; DECIDE WHETHER TO ACCESS WITH RELOCATION OR NOT
1734 ;
1735 005716 012701 140000 MOV #140000,R1 ; ACCESS THRU KIPAR6
1736 005722 032737 000040 000052 BIT #BIT05,@#52 ; UFD MODE?
1737 005730 001410 BEQ 1$ ; IF NOT, BRANCH
1738 005732 032737 000040 177734 BIT #BIT05,KMCR ; 18 BIT MODE?
1739 005740 001412 BEQ 2$ ; IF NOT BRANCH
1740 005742 012737 177600 172354 MOV #177600,KIPAR6 ; KIPAR6 HAS NXM
1741 005750 000422 BR 3$ ; GO DO IT
1742 005752 013737 177734 001162 1$: MOV KMCR,$TMP1 ; SAVE KMCR

```

T15 NXM MEMORY TIMEOUT

```

1743 005760 042737 000040 177734      BIC  #BIT05,KMCR      ; IN NOT UFD, DO IN 22 BITS
1744 005766 012737 000000 172354 2$:  MOV  #0,KIPAR6      ; KIPAR6 ACCESS MAP REG. 0
1745 005774 012737 160000 170200      MOV  #160000,MAPL00  ; MAP PAIR HAS NXM
1746 006002 012737 000077 170202      MOV  #77,MAPH00
1747 006010 052737 000040 172516      BIS  #BIT05,MMR3     ; ENABLE RELOCATION
1748
1749      ; DO ACTUAL DATA OUT CYCLE
1750
1751 006016 005037 177732      3$:  CLR  DDR          ; PROMT DIAG. DATO
1752 006022 005011      CLR  (R1)          ; WRITE TO NXM
1753 006024 032737 100000 177730      BIT  #BIT15,DCSR    ; NXM SET?
1754 006032 001001      BNE  5$           ; IF YES, BRANCH
1755 006034 104022      ERROR +22         ; NXM ACCESS DOES NOT SET DCSR<15>
1756 006036 005037 177572      5$:  CLR  MMRO        ; DISABLE MMU
1757 006042 042737 000040 172516      BIC  #BIT05,MMR3    ; AND MAPPING TOO
1758 006050 032737 000040 000052      BIT  #BIT05,#52     ; UFD MODE
1759 006056 001006      BNE  TST16        ; IF NOT UFD, GO TO NEXT TEST
1760 006060 013737 001162 177734      MOV  $TMP1,KMCR    ; OTHERWISE, RESTORE KMCR
1761 006066 042737 000400 177730      BIC  #BIT8,DCSR    ; EXIT DIAGNOSTIC MODE
1762

```

TEST - CARRY PROPOGATION TEST

```

1764 .SBTTL TEST - CARRY PROPOGATION TEST
1765
1766 ;* THIS TEST CHECKS THE CARRY PROPOGATION THRU THE ALU. MAPPING REGISTER PAIR 0
1767 ;* IS LOADED WITH ALL 1'S. LOCATION 1 IS ACCESSED. THE RESULT OF THE DATO
1768 ;* CYCLE IS SUPPOSED TO BE LOADED IN LOCATION 0.
1769 ;
1770 ; BGNTST
1771 ;
1772 ; IF UFD MODE AND KMCR<5> EQ #1 THEN
1773 ; . EXIT TEST (RELOCATION DOES NOT HAVE TO BE PERFORMED IN 18 BIT)
1774 ; ENDF
1775 ; MAKE SURE THAT DIGNOSTIC MODE IS ON DCSR<8>=1
1776 ; SELECT DIAGNOSTIC NPR THRU DDR BY DCSR<2-1>=<0,0>
1777 ; POINT TIMEOUT_ROUTINE TO VVECTOR 4
1778 ; CALL MAP_PROGRAM_AREA
1779 ; LET MMRO<0> = 1 TO ENABLE MMU
1780 ; LET MMR3<5> = 1 TO ENABLE RELOCATION
1781 ; LET KIPAR6 = #0 TO ACCESS THRU MAP REG.0
1782 ; LET R1 = #140001 TO ACCESS THRU KIPAR6
1783 ; LET LO MAP REG. = #177777
1784 ; LET HI MAP REG. = #77
1785 ; LET PATTERN = #125252
1786 ; CALL DIAGNOSTIC_DATA_OUT<(R1),PATTERN>
1787 ; LET MMRO<0> = #0 TO DISABLE MMU
1788 ; IF #0 NE #125252 THEN
1789 ; . ERROR IN CARRY PROPOGATION
1790 ; ENDF
1791 ;
1792 ; ENDTST
1793 ;
1794 ;-----
1795 ;*****
1796 ;*TEST 16 CARRY PROPOGATION TEST
;*****
TST16: SCOPE

```

```

006074 000004
1797
1798 006076 005737 001720 TST PMIS ; ANY PMI MEMORY?
1799 006102 001470 BEQ TST17 ;; IF NONE, EXIT TEST
1800 006104 032737 000040 000052 BIT #BIT05,#52 ; UFD MODE ?
1801 006112 001404 BEQ 1# ; BRANCH, IF NOT
1802 006114 032737 000040 177734 BIT #BIT05,KMCR ; 18 BIT MODE ?
1803 006122 001060 BNE TST17 ;; GO TO NEXT TEST, IF YES
1804 ;
1805 ; INITIALISE MMU AND DIAGNOSTIC REGISTERS
1806 ;
1807 006124 052737 000400 177730 1#: BIS #BIT08,DCSR ; SET DIAGNOSTIC MODE
1808 006132 042737 000006 177730 BIC #BIT02!BIT01,DCSR ; SELECT NPR THRU DDR
1809 006140 004737 002276 JSR PC,MAPPR ; REMAP PROGRAM AREA
1810 006144 005237 177572 INC MMRO ; TURN ON MMU
1811 006150 052737 000040 172516 BIS #BIT05,MMR3 ; ENABLE RELOCATION
1812 ;
1813 ; ACCESS LOCATION 0 THRU RELOCATION REGISTERS
1814 ;
1815 006156 005037 172354 CLR KIPAR6 ; CLEAR KIPAR6
1816 006162 012701 140002 MOV #140002,R1 ; ACCESS THRU KIPAR6
1817 006166 012737 177777 170200 MOV #177777,MAPL00 ; ALL ONE'S TO LO REG

```

T16 CARRY PROPOGATION TEST

```

1818 006174 012737 000077 170202      MOV    #77,MAPH00      ; ALL ONE'S TO HI REG
1819 006202 012737 125252 001124      MOV    #125252,$GDDAT ; THE PATTERN TO BE STORED
1820 006210 005037 000000                CLR    @#0             ; CLEAR TEST LOCATION
1821 006214 004737 002242                JSR    PC,DDOUT        ; DO DATA OUT
1822 006220 005037 177572                CLR    MMRO            ; DISABLE MMU
1823 006224 042737 000040 172516      BIC    #BIT05,MMR3     ; AND MAPPING
1824 006232 042737 000400 177730      BIC    #BIT8,DCSR      ; EXIT DIAGNOSTIC MODE
1825 006240 022737 125252 000000      CMP    #125252,@#0     ; DATA OUT OK?
1826 006246 001406                BEQ    TST17           ;; IF YES, GO TO NEXT TEST
1827 006250 013737 000000 001126      MOV    @#0,$BDDAT      ; STORE FOR ERROR REPORTS
1828 006256 005037 001122                CLR    $BDADR          ; ADDRESS 0
1829 006262 104023                ERROR  +23             ; IN CARRY PROPOGATION
1830

```


TEST - EXTENSIVE CARRY PROPOGATION TEST

```

1832 .SBTTL TEST EXTENSIVE CARRY PROPOGATION TEST
1833
1834 ;* THIS TEST PERFORMS EXTENSIVE CHECKING OF CARRY PROPOGATION. FIRST 1K IS
1835 ;* CHECKED IN WORD INCREMENTS, AFTER THAT EACH 1K UP TO 4K. DIAGNOSTIC DATI
1836 ;* CYCLES ARE PERFORMED AND A MAPPING REGISTER PAIR USED IS ALWAYS AT ALL 1'S.
1837 ;
1838 ; BGNST
1839 ;
1840 ; IF UFD MODE AND KMCR<5> EQ #1 THEN
1841 ; . EXIT TEST (RELOCATION DOES NOT HAVE TO BE PERFORMED IN 18 BIT)
1842 ; ENDF
1843 ; CALL MAP_PROGRAM_AREA
1844 ; ENABLE MAPPING
1845 ; INITIALISE ALL MAPPING REGISTERS TO POINT TO ALL 1'S
1846 ; DO FOR 4K OF MEMORY
1847 ; . IF MEMORY ACCESSED LESS THEN 1K
1848 ; . . DO WORD INCREMENTS
1849 ; . ELSE
1850 ; . . DO 1K INCREMENTS
1851 ; . ENDF
1852 ; . DO DATI INCREMENTS
1853 ; . IF DDR NE TO LOCATION IN MEMORY THEN
1854 ; . . ALU ERROR
1855 ; . ENDF
1856 ; ENDDO
1857 ;
1858 ; ENDTST
1859 ;
1860 ;
1861 ;
1862 ;

```

```

;*****
;*TEST 17 EXTENSIVE CARRY PROPOGATION TEST
;*****
TST17: SCOPE

```

```

006264 000004
1863
1864 006266 005737 001720 TST PMIS ; ANY PMI MEMORY?
1865 006272 001521 BEQ TST20 ;; IF NONE, EXIT TEST
1866 006274 032737 000040 000052 BIT #BIT05,#52 ; UFD MODE
1867 006302 001404 BEQ 1# ; IF NOT, BRANCH
1868 006304 032737 000040 177734 BIT #BIT05,KMCR ; 18 BIT MODE
1869 006312 001111 BNE TST20 ;; IF 18 BIT MODE, DON'T DO THIS TEST
1870 ;
1871 ; INITIALISE MMU AND RELOCATION
1872 ;
1873 006314 052737 000400 177730 1# : BIS #BIT08,DCSR ; SET DIAGNOSTIC MODE
1874 006322 042737 000006 177730 BIC #BIT02!BIT01,DCSR ; SELECT NPR THRU DDR
1875 006330 004737 002276 JSR PC,MAPPR ; REMAP PROGRAM AREA
1876 006334 052737 000060 172516 BIS #BIT05!BIT04,MMR3 ; ENABLE RELOCATION AND 22BITS
1877 006342 005237 177572 INC MMRO ; ENABLE MMU
1878 ;
1879 ; POINT ALL MAPPING REGISTERS TO ALL 1'S
1880 ;
1881 006346 012701 170200 MOV #MAPLOO,R1 ; START WITH 0
1882 006352 012703 000036 MOV #36,R3 ; DO FOR ALL REGISTERS
1883 006356 012721 177776 2# : MOV #177776,(R1)+ ; LOW MAP REGISTER
1884 006362 012721 000077 MOV #77,(R1)+ ; HIGH MAP REGISTER
1885 006366 077305 SOB R3,2# ; DO FOR ALL

```

T17 EXTENSIVE CARRY PROPOGATION TEST

```

1886
1887 ; START WITH DATI ON A WORD BOUNDARY
1888 ;
1889 006370 005037 172354 CLR KIPAR6 ; ACCESS REGISTER PAIR 0
1890 006374 012701 140002 MOV #140002,R1 ; START WITH 0 THRU KIPAR6
1891 006400 000401 BR 4# ;
1892 006402 005721 3#: TST (R1)+ ; . DO IN WORD INCREMENTS
1893 006404 004737 002254 4#: JSR PC,DDIN ; . DO DATI
1894 006410 010137 001122 MOV R1,#BDADR ; . STORE JUST ACCESSED ADDRESS
1895 006414 162737 000002 001122 SUB #2,#BDADR ; . GET RID OF OVERFLOW
1896 006422 027737 172474 177732 CMP #BDADR,DDR ; . PROPER DATA?
1897 006430 001412 BEQ 7# ; . IF EQUAL, BRANCH
1898 006432 017737 172464 001124 MOV #BDADR,#GDDAT ; . STORE RECEIVED DATA
1899 006440 013737 177732 001126 MOV DDR,#BDDAT ; . STORE EXPECTED DATA
1900 006446 042737 160000 001122 BIC #BIT15!BIT14!BIT13,#BDADR ; . STRIP OF PAR BITS
1901 006454 104023 ERROR +23 ; . ERROR IN RELOCATION
1902
1903 ; DECIDE WHICH CYCLE SHOULD OCCUR NEXT
1904 ;
1905 006456 005737 172354 7#: TST KIPAR6 ; . FIRST TIME 1K BOUNDARY?
1906 006462 001011 BNE 10# ; . IF NOT, BRANCH
1907 006464 022701 144000 CMP #144000,R1 ; . LESS THAN 1K DONE?
1908 006470 103744 BLO 3# ; . DO IN WORD INCREMENTS
1909 006472 012701 140002 MOV #140002,R1 ; . NOW DO INCREMENTING THRU PAR
1910 006476 012737 000040 172354 MOV #40,KIPAR6 ; . FIRST 1K
1911 006504 000737 BR 4# ; . GO DO COMPARE
1912 006506 062737 000040 172354 10#: ADD #40,KIPAR6 ; . ACCESS NEXT 1K
1913 006514 022737 000200 172354 CMP #200,KIPAR6 ; . OVER 4K BOUNDARY?
1914 006522 003330 BGT 4# ; . IF NOT, GO DO DATI
1915 006524 005037 177572 CLR MMRO ; DISABLE MMU
1916 006530 042737 000400 177730 BIC #BIT8,DCSR ; EXIT DIAGNOSTIC MODE
1917

```

TEST - ALU TEST

1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949

```

.SBTTL TEST ALU TEST
;* THIS TEST PERFORMS EXTENSIVE CHECKING OF ALU BY USING BINARY COUNT PATTERN
;* FOR EACH OF THE 5 ADDERS AT THE SAME TIME. HIGHEST LOCATIONS CHECKED
;* CORRESPOND TO MAXIMUM AMOUNT OF MEMORY AVAILABLE.
;
; BGNTST
;
; IF UFD MODE AND KMCR<5> EQ #1 THEN
; . EXIT TEST (RELOCATION DOES NOT HAVE TO BE PERFORMED IN 18 BIT)
; ENDF
; CALL MAP_PROGRAM_AREA
; ENABLE MAPPING AND RELOCATION
; SIZE MEMORY AND MAKE UP MASK FOR NXM
; DO FOR BINARY COUNT PATTERNS THRU ALL FOR ADDERS
; . MASK OUT NXM
; . IF LESS THAN 32K THEN
; . . DO DATI
; . ELSE
; . . DO DATO
; . ENDF
; . IF RESULT OF THE CYCLE IS NOT PROPER THEN
; . . ERROR IN ALU
; . ENDF
; ENDDO
;
; ENDTST

```

```

;*****
;*TEST 20 ALU TEST
;*****
TST20: SCOPE

```

006536 000004

1950
1951 006540 005737 001720
1952 006544 001574
1953 006546 032737 000040 177734
1954 006554 001170
1955
1956
1957
1958 006556 052737 000400 177730
1959 006564 042737 000006 177730
1960 006572 004737 002276
1961 006576 052737 000060 172516
1962 006604 005237 177572
1963 006610 013702 001720
1964 006614 010237 001160
1965 006620 012703 004200
1966 006624 040203
1967 006626 072227 177766
1968
1969
1970
1971 006632 012701 140000
1972 006636 005037 172354

```

;
; ANY 22-BIT PMI MEMORY?
; IF NONE, EXIT TEST
; 18 BIT MODE
; IF 18 BIT MODE, DON'T DO THIS TEST
;
; INITIALISE MMU AND RELOCATION
;
; SET DIAGNOSTIC MODE
; SELECT NPR THRU DDR
; REMAP PROGRAM AREA
; ENABLE RELOCATION AND 22BITS
; ENABLE MMU
; STORE SIZE OF PMI MEMORY
; CREATE A PATTERN
; CONSTANT FOR PAR
; MASK OUT NXM
; FOR MAPH00
;
; DO FOR ALL COMBINATIONS POSSIBLE, 4 BITS AT A TIME
;
; MOV #140000,R1 ; ACCESS THRU KIPAR6
; CLR KIPAR6 ; ACCESS REGISTER PAIR 0

```

T20 ALU TEST

```

1973 006642 005037 170200          CLR      MAPLOO          ; CLEAR MAP REGISTERS
1974 006646 005037 170202          CLR      MAPHOO          ;
1975 006652 005037 001162          CLR      $TMP1          ; CLEAR STORAGE FOR PAR
1976 006656 005037 001122          CLR      $BDADR         ; FIRST ADDRESS USED
1977 006662 005037 001124          CLR      $GDDAT         ; CLEAR PATTERN
1978 006666 004737 002254          2$: JSR      PC,DDIN      ; . DO DATI
1979 006672 013737 001162 172354  MOV      $TMP1,KIPAR6   ; . GET ADDED PAR
1980 006700 023777 177732 172214  CMP      DDR,@$BDADR    ; . DATI OK?
1981 006706 001435                    BEQ      10$           ; . IF SO, BRANCH
1982 006710 013737 177732 001126  MOV      DDR,$BDDAT     ; . STORE RECIEVED DATA
1983 006716 017737 172200 001124  MOV      @$BDADR,$GDDAT ; . STORE EXPECTED DATA
1984 006724 042737 160000 001122  BIC      $BIT15:$BIT14:$BIT13,$BDADR ; . STRIP OF PAR BITS
1985 006732 104023                    ERROR    +23          ;
1986 006734 005037 001124          CLR      $GDDAT         ;
1987 006740 000420                    BR       10$           ;
1988 006742 004737 002242          5$: JSR      PC,DDOUT    ; . DO DATO
1989 006746 013737 001162 172354  MOV      $TMP1,KIPAR6   ; . GET ADDED PAR
1990 006754 023777 001124 172140  CMP      $GDDAT,@$BDADR ; . DATI OK?
1991 006762 001407                    BEQ      10$           ; . IF SO, BRANCH
1992 006764 017737 172132 001126  MOV      @$BDADR,$BDDAT ; . STORE EXPECTED DATA
1993 006772 042737 160000 001122  BIC      $BIT15:$BIT14:$BIT13,$BDADR ; . STRIP OF PAR BITS
1994 007000 104023                    ERROR    +23          ;
1995                                ;
1996                                ; DECIDE WHAT ACCESS NEXT
1997                                ;
1998 007002 005037 172354          10$: CLR      KIPAR6      ; . CLEAR PAR
1999 007006 062701 001042          ADD      $1042,R1      ; . START INCREMENTING
2000 007012 062737 021042 170200  ADD      $21042,MAPLOO ; . EVERY 4TH BIT
2001 007020 005537 170202          ADC      MAPHOO        ; . DON'T FORGET CARRY
2002 007024 062737 000002 170202  ADD      $2,MAPHOO     ; . FOR ALL REGISTERS
2003 007032 010137 001122          MOV      R1,$BDADR     ; . SAVE ADDRESS USED
2004 007036 063737 170200 001122  ADD      MAPLOO,$BDADR ; . ADD UNIBUS MAP REG.
2005 007044 052737 140000 001122  BIS      $140000,$BDADR ; . MAKE SURE KIPAR6 SELECTED
2006 007052 042737 020000 001122  BIC      $20000,$BDADR ;
2007 007060 022701 150420          CMP      $150420,R1    ; . OVERFLOW FROM ADDITION?
2008 007064 001003                    BNE     15$           ; . IF NO, BRANCH
2009 007066 062737 000200 001162  ADD      $200,$TMP1    ; . ADD OVERFLOW FOR PAR
2010 007074 060337 001162          15$: ADD      R3,$TMP1    ; . ADD INCR. TO PAR
2011 007100 022737 002000 001162  CMP      $2000,$TMP1   ; . LESS THAN 32K?
2012 007106 003267                    BGT     2$            ; . IF YES, DO DATI
2013 007110 005237 001124          INC      $GDDAT        ; . INCREMENT PATTERN
2014 007114 023737 001160 001162  CMP      $TMP0,$TMP1   ; . OVER AVAILABLE MEMORY?
2015 007122 003307                    BGT     5$            ; . IF NOT, DO ANOTHER DATO
2016 007124 005037 177572          25$: CLR      MMRO      ; DISABLE MMU
2017 007130 042737 000400 177730  BIC      $BIT8,DCSR    ; EXIT DIAGNOSTIC MODE

```

TEST - MAIN MEMORY DISABLE

2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062

```

.SBTTL TEST - MAIN MEMORY DISABLE
; * THIS TEST WILL CHECK THAT A MAIN MEMORY RESPONSE IS DISABLED WHENEVER
; * THE APPROPRIATE BITS IN THE KMCR REGISTER ARE SET. IN UFD MODE ONLY BITS SET
; * IN KMCR WILL BE TESTED. IN ALL OTHER ENVIRONMENTS ALL BITS WILL BE TESTED.
;
; BGN1ST
;
; CALL MAP_PROGRAM_AREA
; LET MMRO<0> = 1 TO ENABLE MMU
; LET R1 = KMCR_LOW_BYTE
; CLEAR BITS <5,6,7> IN R1
; IF KMCR<5> EQ #1 THEN (18 BIT MODE)
; . LET KIPAR6 = #7400
; ELSE (22 BIT MODE)
; . LET KIPAR6 = #177400
; ENDF
; LET TEST_LOCATION = #140000 TO ACCESS THRU KIPAR6
; DO FOR R0 FROM #1 TO R1 ALL PAGES SPECIFIED IN KMCR
; . LET R2 = #TEST_LOCATION
; . IF NO TIMEOUT THEN
; . . ERROR KMCR<4-5> DOESNOT DISABLE MAIN MEMORY
; . ENDF
; . LET KIPAR6 = KIPAR6 - #200
; ENDDO
; IF NOT UFD MODE THEN
; . SAVE KMCR
; . CHECK KMCR<5-0> TO BE READ-WRITE
; . LET KMCR<5-0> = #0 (ALL NON-UNIBUS MEMORY)
; . LET KIPAR6 = #171600
; . IF @#140000 DOES NOT TIMEOUT THEN
; . . LET KMCR<4-0> = <1000> TO DISABLE HIGHER THAN 28K
; . . IF @#140000 DOESNOT TIMEOUT THEN
; . . . ERROR SETTING KMCR<5-0> DOESNOT DISABLE MEMORY
; . . ENDF
; . ENDF
; . RESTORE KMCR
; ENDF
;
; ENDTST

```

```

;*****
;TEST 21 MAIN MEMORY DISABLE
;*****

```

```

007136 000004
2063 007140 005737 001720
2064 007144 001002
2065 007146 000137 007616
2066 007152 052737 000400 177730 20$:
2067 007160 004737 002276
2068 007164 005237 177572
2069 007170 113701 177734
2070 007174 042701 177740
2071
2072

```

```

TST21: SCOPE
; ANY PMI MEMORY?
; IF YES, DO THE TEST
; OTHERWISE EXIT TEST
; STANDALONE MODE
; REMAP PROGRAM AREA
; ENABLE MMU
; SAVE KMCR
; R1 HAS # OF PAGES DISABLED
;
; GET THE HIGHEST NXM LOCATION

```

T21 MAIN MEMORY DISABLE

```

2073
2074 007200 032737 000040 177734 ; BIT #BIT05,KMCR ; 18 BIT MODE?
2075 007206 001404 ; BEQ 1# ; IF NOT, BRANCH
2076 007210 012737 007400 172354 ; MOV #7400,KIPAR6 ; NXM FOR 18 BITS
2077 007216 000403 ; BR 2# ; GO DO IT
2078 007220 012737 177400 172354 1# : MOV #177400,KIPAR6 ; NXM FOR 22 BITS
2079 ;
2080 ; GO THRU ALL PAGES OF MEMORY DISABLED
2081 ;
2082 007226 105701 2# : TSTB R1 ; ANY UNIBUS MEMORY?
2083 007230 001420 ; BEQ 6# ; IF NO, BRANCH
2084 007232 012737 007250 000004 ; MOV #4#,R#4 ; POINT TIMEOUT VECTOR TO PROGRAM
2085 007240 005737 140000 3# : TST #140000 ; . ACCESS THRU KIPAR6
2086 007244 104025 ; ERROR +25 ; . KMCR<4-0> DOES NOT DISABLE MAIN MEMORY
2087 007246 000402 ; BR 5# ;
2088 007250 005726 4# : TST (SP)+ ; . ADJUST STACK
2089 007252 005726 ; TST (SP)+ ;
2090 007254 162737 000200 172354 5# : SUB #200,KIPAR6 ; . ACCESS NEXT LOWER PAGE
2091 007262 077112 ; SOB R1,3# ; . DO FOR ALL PAGES IN KMCR
2092 007264 012737 002266 000004 ; MOV #TIMOUT,##4 ; RESTORE TIMEOUT VECTOR
2093 ;
2094 ; IN NON-UFD MODE, CHECK KMCR<5-0>
2095 ;
2096 007272 032737 000040 000052 6# : BIT #BIT05,#052 ; UFD MODE?
2097 007300 001146 ; BNE TST22 ;: IF YES, GO TO NEXT TEST
2098 007302 013737 177734 001162 ; MOV KMCR,#TMP1 ; SAVE KMCR
2099 007310 012701 000067 ; MOV #67,R1 ; HIGHEST VALUE (32K OF MEMORY)
2100 007314 110137 177734 7# : MOVVB R1,KMCR ; . WRITE TO KMCR
2101 007320 120137 177734 ; CMPB R1,KMCR ; . WRITTEN OK?
2102 007324 001426 ; BEQ 8# ; . IF SO, BRANCH
2103 007326 010137 001124 ; MOV R1,$GDDAT ; . NO, THEN GET THE GOOD DATA
2104 007332 052737 000200 001124 ; BIS #BIT07,$GDDAT ; . IN CASE REBOOT BIT SET
2105 007340 123737 001124 177734 ; CMPB $GDDAT,KMCR ; . WAS THAT THE ONLY WRONG?
2106 007346 001415 ; BEQ 8# ; . IF YES, BRANCH
2107 007350 010137 001124 ; MOV R1,$GDDAT ; . IF NOT, RESTORE PATTERN WRITTEN
2108 007354 013737 177734 001126 ; MOV KMCR,$BDDAT ; . GET THE BAD DATA
2109 007362 012737 177734 001122 ; MOV #KMCR,$BDADR ; . GET THE ADDRESS OF THE KMCR
2110 007370 013737 001160 177734 ; MOV $TMPO,KMCR ; . RESTORE RREGISTER
2111 007376 104011 ; ERROR +11 ; . ERROR IN KMCR<5-0>
2112 007400 000446 ; BR 12# ; . EXIT TEST ON ERROR
2113 007402 005301 8# : DEC R1 ; . LAST LOCATION (0)?
2114 007404 002343 ; BGE 7# ; . IF NOT, BRANCH
2115 ;
2116 ; IN NON-UFD MODE, GO THRU THE PAGE JUST ABOVE 32K
2117 ;
2118 ;
2119 007406 052737 000020 172516 ; BIS #BIT04,MMR3 ; ENABLE 22-BIT
2120 007414 013702 001720 ; MOV PMIS,R2 ; STORE HIGHEST PAGE
2121 007420 005037 170200 ; CLR MAPLO0 ; CLEAR MAP REG. 0
2122 007424 012737 000001 170202 ; MOV #1,MAPH00 ; POINTS TO 32K
2123 007432 052737 000040 172516 ; BIS #BIT05,MMR3 ; ENABLE MAPPING
2124 007440 005001 ; CLR R1 ; ADDRESS FOR DMA CYCLES
2125 007442 012737 007474 000004 ; MOV #10#,##4 ; TIMEOUT TO PROGRAM
2126 007450 012737 002000 172354 ; MOV #2000,KIPAR6 ; PAGE JUST ABOVE 32K
2127 007456 012737 000067 177734 ; MOV #67,KMCR ; DISABLE HIGHER THEN 32K
2128 ;
2129 ; DO A CPU CYCLE TO DISABLED MEMORY

```

T21 MAIN MEMORY DISABLE

```

2130
2131 007464 005737 140000      ;
9$:   TST      @#140000      ; . ACCESS DISABLED MEMORY
      ERROR   +25           ; . IN DISABLING MEMORY THRU KMCR<4 0>
      BR      12$          ;
2132 007470 104025
2133 007472 000411
2134 007474 005726      10$:  TST      (SP)+      ; . RESTORE STACK
2135 007476 005726      TST      (SP)+
2136
2137      ; DO A DMA DATI CYCLE TO DISABLED MEMORY
2138
2139 007500 004737 002254      ;
11$:  JSR      PC,DDIN      ; . DIAGNOSTIC DATI
      BIT      #BIT15,DCSR  ; . NXM SET?
      BNE     12$          ; . IF SET, BRANCH
2140 007504 032737 100000 177730  ERROR   +25           ; . IN DMA MEMORY NOT DISABLED
2141 007512 001001
2142 007514 104025
2143
2144      ; CHECK WHETHER ALL PAGES IN MEMORY VERIFIED
2145
2146 007516 023702 172354      ;
12$:  CMP      KIPAR6,R2    ; . IS IT IN MEMORY?
      BHIS   14$          ; . IF NOT, BRANCH
2147 007522 103020
2148 007524 062737 020000 170200  ADD     #20000,MAPLOO   ; . ACCESS NEXT 4K
2149 007532 103002          BCC     13$          ; . IF DIDN'T EXCEED 32K, BRANCH
2150 007534 005237 170202          INC     MAPH00        ; . OTHERWISE, INCREMENT MAP REGISTER
2151 007540 005337 177734      13$:  DEC     KMCR        ; . ENABLE NEXT 4K OF MEMORY
2152 007544 032737 000040 177734  BIT     #BIT05,KMCR    ; . ALL 128K DONE?
2153 007552 001404          BEQ     14$          ; . IF SO, BRANCH
2154 007554 062737 000200 172354  ADD     #200,KIPAR6    ; . ACCESS NEXT 4K
2155 007562 000740          BR      9$          ; . TRY TO DO IT
2156 007564 012737 002266 000004 14$:  MOV     $TIMOUT,@#4    ; RESTORE TIMEOUT VECTOR
2157 007572 013737 001162 177734  MOV     $TMP1,KMCR     ; RESTORE KMCR
2158 007600 005037 172516          CLR     MMR3          ; DISABLE 22-BIT MODE
2159 007604 005037 177572          CLR     MMRO         ; DISABLE MMU
2160 007610 042737 000400 177730  BIC     #BIT8,DCSR     ; EXIT DIAGNOSTIC MODE
2161 007616
100$:

```

TEST - CACHE PRESENCE

```

2163 .SBTTL TEST - CACHE PRESENCE
2164
2165 ;* THIS TEST WILL FIND OUT WHETHER THE CACHE IS PRESENT
2166 ;
2167 ; BGNTST
2168 ;
2169 ; LET KMCR<6>=#1
2170 ; IF KMCR<6> NE #1 THEN
2171 ; . CACHE NOT PRESENT OR NOT SEEN
2172 ; ENDIF
2173 ;
2174 ; ENDTST
2175 ;
2176 -----
2177
2178

```

```

;*****
;*TEST 22 CACHE PRESENCE
;*****
TST22: SCOPE
2179 007616 000004
2180 007620 005737 001720 TST PMIS ; ANY PMI MEMORY?
2181 007624 001440 BEQ TST23 ;; IF NONE, EXIT TEST
2182 007626 052737 000100 177734 BIS #BIT06,KMCR ; SET CACHE ENABLE BIT
2183 007634 032737 000100 177734 BIT #BIT06,KMCR ; DID IT GET SET?
2184 007642 001031 BNE TST23 ;; IF CACHE PRESENT, EXIT TEST
2185 007644 005737 001206 TST $PASS ; FIRST PASS?
2186 007650 001014 BNE 1$ ; IF YES, SKIP PRINTOUT
2187 007652 122737 000001 001220 CMPB #APTENV,$ENV ; IN APT MODE?
2188 007660 001410 BEQ 1$ ; IF SO, SKIP PRINTOUT
2189 007662 032737 000040 000052 BIT #BIT05,#52 ; UFD MODE ?
2190 007670 001004 BNE 1$ ; IF YES, BRANCH
2191 007672 104401 007706 TYPE ,NOCAH ; TYPE NO CACHE MESSAGE
2192 007676 104401 001175 TYPE , $CRLF ;
2193 007702 000137 013224 1$: JMP UBETST ; IF NO CACHE, SKIP ALL CACHE TSTS
2194
2195 007706 040 116 117 NOCAH: .ASCIZ / NO CACHE SEEN/
007711 040 103 101
007714 103 110 105
007717 040 123 105
007722 105 116 000
2196 .EVEN

```


TEST - CACHE DISABLED AND KMCR

```

2198 .SBTTL TEST - CACHE DISABLED AND KMCR
2199
2200 ;* THIS TEST WILL CHECK THAT WHENEVER THE DMA CACHE IS DISABLED
2201 ;* THE STATUS BITS IN KMCR REGISTER ARE INITIALISED TO POWER UP
2202 ;* CONDITIONS: THE VALID BITS ARE CLEARED, A IS THE NEXT AVAILABLE
2203 ;* SET, FOLLOWED BY B, C, AND D.
2204 ;
2205 ; BGNTST
2206 ;
2207 ; DO FOR MMR3<5>=0,1 WITH RELOCATION ENABLED AND DISABLED
2208 ; . LET KMCR<6> = #0 TO DISABLE CACHE
2209 ; . LET KMCR<8>=0
2210 ; . IF KMCR<12!11!10!9> NE #0 THEN
2211 ; . . ERROR IN VALID BITS WITH CACHE DISABLED
2212 ; . ENDF
2213 ; . LET KMCR<8>=1
2214 ; . IF KMCR<14!13!12!11!10!9!> NE #1 THEN
2215 ; . . ERROR IN LRU BITS SELECTION
2216 ; . ENDF
2217 ; ENDDO
2218 ;
2219 ; ENDTST
2220 ;
2221 ;-----
2222 ;*****
2223 ;*TEST 23 CACHE DISABLED AND KMCR
2224 ;*****

```

007726 000004

```

2224 TST23: SCOPE
2225 007730 005737 001720 TST PMIS ; ANY PMI MEMORY?
2226 007734 001471 BEQ TST24 ;; IF NONE, EXIT TEST
2227 007736 052737 000400 177730 BIS #BIT8,DCSR ; SELECT DIAGNOSTIC MODE
2228 007744 042737 000040 172516 BIC #BIT05,MMR3 ; FIRST DO WITH RELOCATION DISABLED
2229 007752 000403 BR 2$ ; GO DO IT
2230 007754 052737 000040 172516 1$: BIS #BIT05,MMR3 ; NOW DO WITH RELOCATION ENABLED
2231 007762 042737 000100 177734 2$: BIC #BIT06,KMCR ; MAKE SURE THAT CACHE IS DISABLED
2232 007770 042737 000400 177734 BIC #BIT08,KMCR ; READ VALID BITS FIRST
2233 007776 032737 017000 177734 BIT #BIT12!BIT11!BIT10!BIT9,KMCR ; ALL VALID BITS CLEAR?
2234 010004 001412 BEQ 3$ ; IF YES, BRANCH
2235 010006 013737 177734 001126 MCV KMCR,#BDDAT ; STORE KMCR
2236 010014 013737 177734 001124 MOV KMCR,$GDDAT ; STORE TO PRESERVE LOW BYTE
2237 010022 042737 017000 001124 BIC #BIT12!BIT11!BIT10!BIT9,$GDDAT ; HIGH BYTE CLEAR
2238 010030 104026 ERROR +26 ; VALID BITS NOT CLEAR WITH KMCR<6>=0
2239 010032 052737 000400 177734 3$: BIS #BIT08,KMCR ; NOW READ AVAILABILITY BITS
2240 010040 013737 177734 001160 MOV KMCR,$TMPO ; STORE KMCR
2241 010046 122737 000177 001161 CMPB #177,$TMPO+1 ; ALL SET?
2242 010054 001412 BEQ 4$ ; IF YES, BRANCH
2243 010056 013737 177734 001126 MOV KMCR,#BDDAT ; STORE KMCR
2244 010064 013737 177734 001124 MOV KMCR,$GDDAT ; STORE TO PRESERVE LOW BYTE
2245 010072 152737 000177 001124 BISB #177,$GDDAT ; HIGH BYTE SET
2246 010100 104026 ERROR +26 ; AVAIL. BITS NOT SET WITH KMCR<6>=0
2247 010102 032737 000040 172516 4$: BIT #BIT05,MMR3 ; DONE WITH RELOCATION ENABLED?
2248 010110 001721 BEQ 1$ ; IF NOT, BRANCH
2249 010112 042737 000400 177730 BIC #BIT8,DCSR ; EXIT DIAGNOSTIC MODE
2250
2251

```

TEST - AVAILABILITY OF SETS

```

2253      .SBTTL TEST - AVAILABILITY OF SETS
2254
2255      ; * THIS TEST WILL CHECK KMCR<15-8>. FIRST, EACH SET IN THE CACHE IS ALLOCATED.
2256      ; * THEN EACH SET IS MADE MOST RECENTLY USED (MRU), AND THEN EACH SET IS
2257      ; * INVALIDATED BY READING THE 8TH REGISTER OF THE SET.
2258      ;
2259      ; BGNTST
2260      ;
2261      ;     ENABLE CACHE
2262      ;     POINT R2 TO VALTBL FOR VALID BITS (KMCR<8>=0)
2263      ;     POINT R3 TO TOPTBL FOR AVAILABILITY (KMCR<8>=1)
2264      ;     ENABLE RELOCATION
2265      ; *
2266      ; * ALLOCATE SETS IN CACHE MAKING EACH OF THEM VALID AND GETTING MISSES
2267      ; *
2268      ;     DO FOR ALL 4 SETS ALLOCATING THEM IN CACHE
2269      ;     . LET KMCR<8>=0
2270      ;     . DO DMA READ ON OCTAL BOUNDARY USING DIAGNOSTIC DATA_IN
2271      ;     . IFB (R2)+ NE IN KMCR+1 THEN
2272      ;     .     ERROR IN READING VALID BITS ON MISS
2273      ;     . ENDIF
2274      ;     . LET KMCR<8>=1
2275      ;     . IFB KMCR+1 NE (R3)+ THEN
2276      ;     .     ERROR READING AVAILABILITY BITS ON MISS
2277      ;     . ENDIF
2278      ;     ENDDO
2279      ; *
2280      ; * READ A REGISTER FROM A VALID SET MAKING EACH OF THE SETS MOST RECENTLY USED
2281      ; *
2282      ;     DO FOR ALL 4 SETS MAKING THEM MRU AND THEN MOST AVAILABLE
2283      ;     . LET KMCR<8>=0
2284      ;     . DO DMA READ FROM 3RD LOCATION USING DIAGNOSTIC_DATA_IN
2285      ;     . IFB KMCR+1 NE (R2)+ THEN
2286      ;     .     ERROR IN READING VALID BITS ON HIT
2287      ;     . ENDIF
2288      ;     . LET KMCR<8>=1
2289      ;     . IFB KMCR+1 NE (R3)+ THEN
2290      ;     .     ERROR READING AVAILABILITY BITS ON HIT
2291      ;     . ENDIF
2292      ; *
2293      ; * READ THE LAST REGISTER FROM A SET MAKING EACH SET INVALID AND MOST AVAILABLE
2294      ; *
2295      ;     . LET KMCR<8>=0
2296      ;     . DO DMA READ FROM 8TH LOCATION USING DIAGNOSTIC_DATA IN
2297      ;     . IFB KMCR+1 NE (R2)+ THEN
2298      ;     .     ERROR IN READING VALID BITS INVALIDATING A SET
2299      ;     . ENDIF
2300      ;     . LET KMCR<8>=1
2301      ;     . IFB KMCR+1 NE (R3)+ THEN
2302      ;     .     ERROR READING AVAILABILITY BITS INVALIDATING A SET
2303      ;     . ENDIF
2304      ;     ENDDO
2305      ;
2306      ;     ENDTST
2307      ;
2308      ; -----
2309

```

T24 AVAILABILITY OF SETS

```

2310 ;*****
; *TEST 24 AVAILABILITY OF SETS
;*****
TST24: SCOPE

2311 010120 000004
2312 010122 005737 001720 TST PMIS ; ANY PMI MEMORY?
2313 010126 001002 BNE 20$ ; IF SOME, DO THE TEST
2314 010130 000137 010546 JMP 100$ ; OTHERWISE EXIT
2315 010134 052737 000400 177730 20$: BIS #BIT8,DCSR ; SELECT DIAGNOSTIC MODE
2316 010142 052737 000100 177734 BIS #BIT06,KMCR ; ENABLE CACHE
2317 010150 012702 010550 MOV #VALTBL,R2 ; POINT TO VALID BITS
2318 010154 012703 010564 MOV #TOPTBL,R3 ; POINT TO AVAILABILITY BITS
2319 010160 052737 000040 172516 BIS #BIT05,MMR3 ; ENABLE RELOCATION
2320 010166 012737 000200 170200 MOV #200,MAPLOO ; POINT MAPOO TO FIRST 8KB
2321 010174 012737 000000 170202 MOV #0,MAPH00 ; IN HIGH AND LOW MAP REGISTERS
2322 010202 013737 177734 001124 MOV KMCR,$GDDAT ; PRESERVE LOW BYTE
2323 010210 105037 001125 CLRB $GDDAT+1 ; CLEAR TO REPORT ERRORS, IF ANY
2324 ;
2325 ; ALLOCATE 4 SETS BY READING THEM THRU DIAGNOSTIC_DATA_IN
2326 ;
2327 010214 012701 000000 MOV #0,R1 ; START READING MEMORY FROM 0
2328 010220 004737 002254 1$: JSR PC,DDIN ; . ALLOCATE IN CACHE NEXT 8 WORDS
2329 010224 042737 000400 177734 BIC #BIT08,KMCR ; . READ VALID BITS
2330 010232 013737 177734 001126 MOV KMCR,$BDDAT ; . STORE KMCR
2331 010240 122237 001127 CMPB (R2)+,$BDDAT+1 ; . VALID BITS OK?
2332 010244 001404 BEQ 2$ ; . IF YES, BRANCH
2333 010246 105742 TSTB -(R2) ; . GET THE PATTERN JUST WRITTEN
2334 010250 112237 001125 MOVB (R2)+,$GDDAT+1 ; . STORE HIGH BYTE
2335 010254 104026 ERROR +26 ; . ERROR IN VALID BITS
2336 010256 052737 000400 177734 2$: BIS #BIT08,KMCR ; . READ AVAILABILITY BIT
2337 010264 013737 177734 001126 MOV KMCR,$BDDAT ; . STORE KMCR
2338 010272 122337 001127 CMPB (R3)+,$BDDAT+1 ; . AVAILABILITY BITS OK?
2339 010276 001404 BEQ 3$ ; . IF YES, BRANCH
2340 010300 105743 TSTB -(R3) ; . GET THE PATTERN JUST WRITTEN
2341 010302 112337 001125 MOVB (R3)+,$GDDAT+1 ; . STORE HIGH BYTE
2342 010306 104026 ERROR +26 ; . ERROR IN AVAILABILITY BITS
2343 010310 062701 000020 3$: ADD #20,R1 ; . DO FOR NEXT OCTAL BOUNDARY
2344 010314 022701 000100 CMP #100,R1 ; . ALL 4 DONE?
2345 010320 003337 BGT 1$ ; . IF NOT, BRANCH
2346 ;
2347 ; NOW READ 3RD AND THEN 8TH LOCATION FROM CACHE
2348 ;
2349 010322 012701 000004 MOV #4,R1 ; START WITH 3RD LOCATION A SET
2350 010326 042737 000400 177734 4$: BIC #BIT08,KMCR ; . READ VALID BITS
2351 010334 004737 002254 JSR PC,DDIN ; . READ FROM CACHE
2352 010340 013737 177734 001126 MOV KMCR,$BDDAT ; . STORE KMCR
2353 010346 122237 001127 CMPB (R2)+,$BDDAT+1 ; . VALID BITS OK?
2354 010352 001404 BEQ 5$ ; . IF YES, BRANCH
2355 010354 105742 TSTB -(R2) ; . GET THE PATTERN JUST WRITTEN
2356 010356 112237 001125 MOVB (R2)+,$GDDAT+1 ; . STORE HIGH BYTE
2357 010362 104026 ERROR +26 ; . ERROR IN VALID BITS
2358 010364 052737 000400 177734 5$: BIS #BIT08,KMCR ; . READ AVAILABILITY BIT
2359 010372 013737 177734 001126 MOV KMCR,$BDDAT ; . STORE KMCR
2360 010400 122337 001127 CMPB (R3)+,$BDDAT+1 ; . AVAILABILITY BITS OK?
2361 010404 001404 BEQ 6$ ; . IF YES, BRANCH
2362 010406 105743 TSTB (R3) ; . GET THE PATTERN JUST WRITTEN
2363 010410 112337 001125 MOVB (R3)+,$GDDAT+1 ; . STORE HIGH BYTE

```

T24 AVAILABILITY OF SETS

```

2364 010414 104026          ERROR +26          ; . ERROR IN AVAILABILITY BITS
2365 010416 062701 000012 6$: ADD #12,R1      ; . READ THE 8TH WORD
2366 010422 052737 000400 177734 BIS #BIT08,KMCR ; . READ AVAILABILITY BIT
2367 010430 004737 002254 JSR PC,DDIN     ; . READ FROM CACHE
2368 010434 013737 177734 001126 MOV KMCR,$BDDAT ; . STORE KMCR
2369 010442 122337 001127 CMPB (R3)+,$BDDAT+1 ; . AVAILABILITY BITS OK?
2370 010446 001404 BEQ 7$         ; . IF YES, BRANCH
2371 010450 105743 TSTB -(R3)     ; . GET THE PATTERN JUST WRITTEN
2372 010452 112337 001125 MOVB (R3)+,$GDDAT+1 ; . STORE HIGH BYTE
2373 010456 104026          ERROR +26          ; . ERROR IN AVAILABILITY BITS
2374 010460 042737 000400 177734 7$: BIC #BIT08,KMCR ; . READ VALID BITS
2375 010466 013737 177734 001126 MOV KMCR,$BDDAT ; . STORE KMCR
2376 010474 122237 001127 CMPB (R2)+,$BDDAT+1 ; . VALID BITS OK?
2377 010500 001404 BEQ 8$         ; . IF YES, BRANCH
2378 010502 105742 TSTB -(R2)     ; . GET THE PATTERN JUST WRITTEN
2379 010504 112237 001125 MOVB (R2)+,$GDDAT+1 ; . STORE HIGH BYTE
2380 010510 104026          ERROR +26          ; . ERROR IN VALID BITS
2381 010512 062701 000006 8$: ADD #6,R1         ; . DO FOR NEXT OCTAL BOUNDARY
2382 010516 022701 000100 CMP #100,R1     ; . ALL 4 DONE?
2383 010522 003301 BGT 4$         ; . IF NOT, BRANCH
2384 010524 042737 000040 172516 BIC #BIT05,MMR3 ; . DISABLE RELOCATION
2385 010532 042737 000100 177734 BIC #BIT06,KMCR ; . DISABLE CACHE
2386 010540 042737 000400 177730 BIC #BIT8,DCSR  ; . EXIT DIAGNOSTIC MODE
2387 010546          100$: BR TST25          ;; EXIT TEST
      010546 000414

2388
2389          ;* THIS TABLE HAS BITS KMCR<15-8> WHEN KMCR<8>=0
2390          ;* KMCR<15>=0 FLAGGING CACHE MISS
2391
2392 010550 020 030 034 VALTBL: .BYTE 20,30,34,36 ; ORDER FOR VALID: A, AB, ABC, ABCD
      010553 036

2393          ; NOW ALL VALID, HITS-MISS(WRITE TO I/O)
2394 010554 236 016 .BYTE 236,16 ; HIT-MISS, A: VALID - NOT VALID
2395 010556 216 006 .BYTE 216,6 ; HIT-MISS, B: VALID - NOT VALID
2396 010560 206 002 .BYTE 206,2 ; HIT-MISS, C: VALID - NOT VALID
2397 010562 202 000 .BYTE 202,0 ; HIT-MISS, D: VALID NOT VALID
2398
2399          ;* THIS TABLE HAS BITS KMCR<15-8> WHEN KMCR<8>=1
2400          ;* AT FIRST NO HITS ARE RECORDED, THE COMMENT FIELD SHOWS SETS
2401          ;* STARTING WITH LEAST AVAILABLE
2402
2403 010564 017 TOPTBL: .BYTE 17 ; ADCB - 000111
2404 010565 103 .BYTE 103 ; BADC - 100001
2405 010566 151 .BYTE 151 ; CBAD - 110100
2406 010567 177 .BYTE 177 ; DCBA - 111111
2407          ; NOW MISS-HITS
2408 010570 017 377 .BYTE 17,377 ; ->ADCB->DCBA - 000111,111111
2409 010572 163 277 .BYTE 163,277 ; ->BDCA->DCAB - 111001,011111
2410 010574 075 227 .BYTE 75,227 ; ->CDAB->DABC - 011110,001011
2411 010576 027 201 .BYTE 27,201 ; ->DABC->ABCD - 001011,000000
2412
2413          .EVEN

```

TEST DEALLOCATION OF SETS

```

2415 .SBTTL TEST - DEALLOCATION OF SETS
2416
2417 ;* THIS TEST CHECKS THAT EACH SET CAN BE MADE MOST AVAILABLE AND
2418 ;* DEALLOCATED ON THE NEXT READ ON THE OCTAL BOUNDARY.
2419 ;
2420 ;     DISABLE CACHE TO MAKE A MOST AVAILABLE
2421 ;     ENABLE CACHE
2422 ;*
2423 ;* ALLOCATE SETS IN CACHE MAKING EACH OF THEM VALID AND GETTING MISSES
2424 ;*
2425 ;     DO FOR ALL 4 SETS ALLOCATING THEM IN CACHE
2426 ;     . DO DMA READ ON OCTAL BOUNDARY USING DIAGNOSTIC_DATA_IN
2427 ;     . ENDDO - THIS LEAVES A AS THE MOST AVAILABLE SET
2428 ;*
2429 ;* TRY TO BRING A NEW LOCATIONS TO SET A
2430 ;*
2431 ;     DO DMA READ ON OCTAL BOUNDARY FROM A FIFTH LOCATION TO SET A AGAIN
2432 ;     LET KMCR<8> = #0
2433 ;     IF KMCR<15!12!11!10!9!> NE <0,1,1,1,1> THEN
2434 ;     . ERROR DEALLOCATIONING LRU SET (A)
2435 ;     ENDIF
2436 ;     LET KMCR<8> = #1
2437 ;     IF KMCR<15-9> NE <0,0,0,0,1,1,1> THEN
2438 ;     . ERROR DEALLOCATING LRU SET (A)
2439 ;     ENDIF
2440 ;*
2441 ;* TRY TO BRING A NEW LOCATIONS TO SET B
2442 ;*
2443 ;     DO DMA READ ON OCTAL BOUNDARY FROM 1ST LOCATION TO SET B
2444 ;     LET KMCR<8> = #0
2445 ;     IF KMCR<15!12!11!10!9!> NE <0,1,1,1,1> THEN
2446 ;     . ERROR DEALLOCATIONING LRU SET (B)
2447 ;     ENDIF
2448 ;     LET KMCR<8> = #1
2449 ;     IF KMCR<15-9> NE <0,1,0,0,0,0,1> THEN
2450 ;     . ERROR DEALLOCATING LRU SET (B)
2451 ;     ENDIF
2452 ;*
2453 ;* TRY TO BRING A NEW LOCATIONS TO SET C
2454 ;*
2455 ;     DO DMA READ ON OCTAL BOUNDARY FROM 1ST LOCATION TO SET C
2456 ;     LET KMCR<8> = #0
2457 ;     IF KMCR<15!12!11!10!9!> NE <0,1,1,1,1> THEN
2458 ;     . ERROR DEALLOCATIONING LRU SET (C)
2459 ;     ENDIF
2460 ;     LET KMCR<8> = #1
2461 ;     IF KMCR<15-9> NE <0,1,1,0,1,0,0> THEN
2462 ;     . ERROR DEALLOCATING LRU SET (C)
2463 ;     ENDIF
2464 ;*
2465 ;* TRY TO BRING A NEW LOCATIONS TO SET D
2466 ;*
2467 ;     DO DMA READ ON OCTAL BOUNDARY FROM 1ST LOCATION TO SET D
2468 ;     LET KMCR<8> = #0
2469 ;     IF KMCR<15!12!11!10!9!> NE <0,1,1,1,1> THEN
2470 ;     . ERROR DEALLOCATIONING LRU SET (D)
2471 ;     ENDIF

```

TEST - DEALLOCATION OF SETS

```

2472 ; LET KMCR<8> = #1
2473 ; IF KMCR<15-9> NE \0,1,1,1,1,1,1,1> THEN
2474 ; . ERROR DEALLOCATING LRU SET (D)
2475 ; ENDF
2476 ;
2477 ; ENDTST
2478 ;
2479 ;
2480 ;-----
2481 ;*****
;*TEST 25 DEALLOCATION OF SETS
;*****
TST25: SCOPE
010600 000004
2482
2483 010602 005737 001720 TST PMIS ; ANY PMI MEMORY?
2484 010606 001002 BNE 20$ ; IF SOME, DO THE TEST
2485 010610 000137 011310 JMP 100$ ; OTHERWISE, EXIT
2486 010614 052737 000400 177730 20$: BIS #BIT8,DCSR ; SELECT DIAGNOSTIC MODE
2487 010622 052737 000040 172516 BIS #BIT05,MMR3 ; ENABLE RELOCATION
2488 010630 052737 000100 177734 BIS #BIT06,KMCR ; ENABLE CACHE
2489 010636 012737 000200 170200 MOV #200,MAPLOO ; POINT MAPOO TO FIRST 8KB
2490 010644 012737 000000 170202 MOV #0,MAPH00 ; IN HIGH AND LOW MAP REGISTERS
2491 010652 013737 177734 001124 MOV KMCR,$GDDAT ; STORE LOW BYTE OF KMCR
2492 ;
2493 ; ALLOCATE ALL SETS MAKING A MOST AVAILABLE
2494 ;
2495 010660 012701 000000 MOV #0,R1 ; START WITH 0 IN SET A
2496 010664 004737 002254 1$: JSR PC,DDIN ; . ALLOCATE A SET
2497 010670 062701 000020 ADD #20,R1 ; . GET READY FOR NEXT SET
2498 010674 022701 000100 CMP #100,R1 ; . ALL 4 DONE?
2499 010700 003371 BGT 1$ ; . IF NOT, BRANCH
2500 ;
2501 ; TRY TO BRING A NEW SET TO A
2502 ;
2503 010702 012701 000100 MOV #100,R1 ; TRY TO DO DATI
2504 010706 042737 000400 177734 BIC #BIT08,KMCR ; READ VALID BITS
2505 010714 004737 002254 JSR PC,DDIN ; ALLOCATE IN CACHE
2506 010720 013737 177734 001126 MOV KMCR,$BDDAT ; STORE KMCR
2507 010726 122737 000036 001127 CMPB #36,$BDDAT+1 ; ALL SETS VALID?
2508 010734 001404 BEQ 2$ ; IF OK, BRANCH
2509 010736 112737 000036 001125 MOVB #36,$GDDAT+1 ; STORE EXPECTED PATTERN
2510 010744 104026 ERROR +26 ; DEALLOCATING A
2511 010746 052737 000400 177734 2$: BIS #BIT08,KMCR ; READ AVAILABILITY BITS
2512 010754 013737 177734 001126 MOV KMCR,$BDDAT ; STORE KMCR
2513 010762 122737 000017 001127 CMPB #17,$BDDAT+1 ; MRU:A,D,C,B
2514 010770 001404 BEQ 3$ ; IF OK, BRANCH
2515 010772 112737 000017 001125 MOVB #17,$GDDAT+1 ; STORE EXPECTED PATTERN
2516 011000 104026 ERROR +26 ; DEALLOCATING A
2517 ;
2518 ; TRY TO BRING A NEW SET TO B
2519 ;
2520 011002 012701 000120 3$: MOV #120,R1 ; PREPARE FOR NEXT READ
2521 011006 042737 000400 177734 BIC #BIT08,KMCR ; READ VALID BITS
2522 011014 004737 002254 JSR PC,DDIN ; READ INTO CACHE
2523 011020 013737 177734 001126 MOV KMCR,$BDDAT ; STORE KMCR
2524 011026 122737 000036 001127 CMPB #36,$BDDAT+1 ; ALL VALID?
2525 011034 001404 BEQ 4$ ; IF YES, BRANCH

```

T25 DEALLOCATION OF SETS

```

2526 011036 112737 000036 001125      MOVB  #36,#GDDAT+1      ; STORE EXPECTED PATTERN
2527 011044 104026                      ERROR  +26              ; DEALLOCATING B
2528 011046 052737 000400 177734 4#:  BIS  #BIT08,KMCR        ; READ AVAILABILITY BITS
2529 011054 013737 177734 001126      MOV   KMCR,#BDDAT       ; STORE KMCR
2530 011062 122737 000103 001127      CMPB  #103,#BDDAT+1     ; MRU:B,A,D,C
2531 011070 001404                      BEQ   5#                ; IF OK, BRANCH
2532 011072 112737 000103 001125      MOVB  #103,#GDDAT+1     ; STORE EXPECTED PATTERN
2533 011100 104026                      ERROR  +26              ; DEALLOCATING B
2534
2535      ; TRY TO BRING A NEW SET TO C
2536
2537 011102 012701 000140                5#:  MOV   #140,R1        ; PREPARE FOR NEXT READ
2538 011106 042737 000400 177734      BIC  #BIT08,KMCR        ; READ VALID BITS
2539 011114 004737 002254                JSR  PC,DDIN            ; READ INTO CACHE
2540 011120 013737 177734 001126      MOV   KMCR,#BDDAT       ; STORE KMCR
2541 011126 122737 000036 001127      CMPB  #36,#BDDAT+1     ; ALL VALID?
2542 011134 001404                      BEQ   6#                ; IF YES, BRANCH
2543 011136 112737 000036 001125      MOVB  #36,#GDDAT+1     ; STORE EXPECTED PATTERN
2544 011144 104026                      ERROR  +26              ; DEALLOCATING C
2545 011146 052737 000400 177734 6#:  BIS  #BIT08,KMCR        ; READ AVAILABILITY BITS
2546 011154 013737 177734 001126      MOV   KMCR,#BDDAT       ; STORE KMCR
2547 011162 122737 000151 001127      CMPB  #151,#BDDAT+1    ; MRU:C,B,A,D
2548 011170 001404                      BEQ   7#                ; IF OK, BRANCH
2549 011172 112737 000151 001125      MOVB  #151,#GDDAT+1    ; STORE EXPECTED PATTERN
2550 011200 104026                      ERROR  +26              ; DEALLOCATING C
2551
2552      ; TRY TO BRING A NEW SET TO D
2553
2554 011202 012701 000160                7#:  MOV   #160,R1        ; PREPARE FOR NEXT READ
2555 011206 042737 000400 177734      BIC  #BIT08,KMCR        ; READ VALID BITS
2556 011214 004737 002254                JSR  PC,DDIN            ; READ INTO CACHE
2557 011220 013737 177734 001126      MOV   KMCR,#BDDAT       ; STORE KMCR
2558 011226 122737 000036 001127      CMPB  #36,#BDDAT+1     ; ALL VALID?
2559 011234 001404                      BEQ   8#                ; IF YES, BRANCH
2560 011236 112737 000036 001125      MOVB  #36,#GDDAT+1     ; STORE EXPECTED PATTERN
2561 011244 104026                      ERROR  +26              ; DEALLOCATING D
2562 011246 052737 000400 177734 8#:  BIS  #BIT08,KMCR        ; READ AVAILABILITY BITS
2563 011254 013737 177734 001126      MOV   KMCR,#BDDAT       ; STORE KMCR
2564 011262 042737 000400 177730      BIC  #BIT08,DCSR        ; EXIT DIAGNOSTIC MODE
2565 011270 122737 000177 001127      CMPB  #177,#BDDAT+1    ; MRU:D,C,B,A
2566 011276 001404                      BEQ   TST26             ;
2567 011300 112737 000177 001125      MOVB  #177,#GDDAT+1    ; STORE EXPECTED PATTERN
2568 011306 104026                      ERROR  +26              ; DEALLOCATING D
2569 011310
2570      100#:

```

TEST - CACHE WITH RELOCATION DISABLED

```

2572 .SBTTL TEST - CACHE WITH RELOCATION DISABLED
2573
2574 ;* THIS TEST CHECKS THAT CACHE IS NOT OPERATIONAL WHEN RELOCATION IS
2575 ;* DISABLED AND KMCR<15-9> ARE NOT INITIALIZED.
2576 ;
2577 ; BGMTST
2578 ;
2579 ; SAVE KMCR<15-9> FOR KMCR<8>=0,1
2580 ; LET MMR3<5>=#0 TO DISABLE RELOCATION
2581 ; DO DMA READ USING DIAGNOSTIC_DATA_IN
2582 ; LET KMCR<8>=0
2583 ; IFB KMCR+1 NE SAVED THEN
2584 ; . ERROR RELOCATION DISABLED STILL AFFECTS THE CACHE
2585 ; ENDF
2586 ; LET KMCR<8>=1
2587 ; IFB KMCR+1 NE SAVED THEN
2588 ; . ERROR RELOCATION DISABLED STILL AFFECTS THE CACHE
2589 ; ENDF
2590 ;
2591 ; ENDTST
2592 ;
2593 ;-----
2594 ;*****
2595 ;*TEST 26 CACHE WITH RELOCATION DISABLED
;*****

```

011310 000004

```

2596
2597 011312 005737 001720 TST PMIS ; ANY PMI MEMORY?
2598 011316 001455 BEQ TST27 ;; IF NONE, EXIT TEST
2599 011320 052737 000100 177734 BIS #BIT06,KMCR ; MAKE SURE THAT CACHE IS ENABLED
2600 011326 052737 000400 177730 BIS #BIT08,DCSR ; SELECT DIAGNOSTIC MODE
2601 011334 042737 000040 172516 BIC #BIT05,MMR3 ; DISABLE RELOCATION
2602 011342 042737 000400 177734 BIC #BIT08,KMCR ; READ VALID BITS FIRST
2603 011350 013737 177734 001124 MOV KMCR,#GDDAT ; STORE LOW BYTE
2604 011356 105037 001125 CLRB #GDDAT+1 ; CLEAR HIGH BYTE
2605 011362 032737 017000 177734 BIT #BIT12!BIT11!BIT10!BIT9,KMCR ; ALL VALID BITS CLEAR?
2606 011370 001404 BEQ 3# ; IF YES, BRANCH
2607 011372 013737 177734 001126 MOV KMCR,#BDDAT ; STORE KMCR
2608 011400 104026 ERROR +26 ; VALID BITS NOT CLEAR WITH KMCR<6>=0
2609 011402 052737 000400 177734 3#: BIS #BIT08,KMCR ; NOW READ AVAILABILITY BITS
2610 011410 013737 177734 001126 MOV KMCR,#BDDAT ; STORE KMCR
2611 011416 122737 000177 001127 CMPB #177,#BDDAT+1 ; ALL SET?
2612 011424 001404 BEQ 4# ; IF YES, EXIT TEST
2613 011426 112737 000177 001125 MOVB #177,#GDDAT+1 ; EXPECTED PATTERN
2614 011434 104026 ERROR +26 ; AVAIL. BITS NOT SET WITH KMCR<6>=0
2615 011436 042737 000100 177734 4#: BIC #BIT06,KMCR ; DISABLE CACHE
2616 011444 042737 000400 177730 BIC #BIT08,DCSR ; EXIT DIAGNOSTIC MODE
2617

```


TEST - WRITE CYCLES AND CACHE

2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644

```

.SBTTL TEST - WRITE CYCLES AND CACHE
;* THIS TEST CHECKS THAT DMA WRITE CYCLES DON'T AFFECT THE CACHE, EXCEPT
;* WRITE HITS WHICH INVALIDATE THE SET AND MAKE THIS SET THE MOST AVAILABLE.
;
; BGNTST
;
;     SAVE KMCR<15-9> FOR KMCR<8>=0,1
;     DO DIAGNOSTIC_DATA_OUT WITH TAGS NOT CORRESPONDING TO ANY SETS
;     DO FOR KMCR<8>=0,1
;     . IF KMCR<15-9> NE SAVED VALUES THEN
;     . ERROR WRITE CYCLES AFFECT THE CACHE
;     . ENDF
; ENDDO
; DO DIAGNOSTIC_DATA_OUT THAT CAUSES A HIT
; IF NOT A HIT THEN
; . ERROR RECORDING HITS
; IF THE SET IS VALID OR NOT MOST AVAILABLE THEN
; . ERROR IN LEAST RECENTLY USED LOGIC
; ENDF
;
; ENDTST

```

```

;*****
;*TEST 27 WRITE CYCLES AND CACHE
;*****
TST27: SCOPE

```

```

011452 000004
2645
2646 011454 005737 001720
2647 011460 001002
2648 011462 000137 012134
2649 011466 052737 000400 177730
2650 011474 052737 000100 177734
2651 011502 052737 000040 172516
2652 011510 012737 000200 170200
2653 011516 012737 000000 170202
2654 011524 042737 000400 177734
2655 011532 013702 177734
2656 011536 052737 000400 177734
2657 011544 013703 177734
2658
2659
2660
2661 011550 012701 000200
2662 011554 013737 000400 001124
2663 011562 004737 002242
2664 011566 042737 000400 177734
2665 011574 020237 177734
2666 011600 001406
2667 011602 010237 001124
2668 011606 013737 177734 001126
2669 011614 104026
2670 011616 052737 000400 177734 14:
2671 011624 020337 177734
2672 011630 001406

```

```

TST PMIS ; ANY PMI MEMORY?
BNE 204 ; IF SOME, DO THE TEST
JMP 1004 ; OTHERWISE EXIT
204: BIS #BIT08,DCSR ; SELECT DIAGNOSTIC MODE
BIS #BIT06,KMCR ; CACHE STILL ENABLED
BIS #BIT05,MWR3 ; ENABLE RELOCATION
MOV #200,MAPL00 ; POINT MAP00 TO FIRST 8KB
MOV #0,MAPH00 ; IN HIGH AND LOW MAP REGISTERS
BIC #BIT08,KMCR ; SELECT VALID BITS
MOV KMCR,R2 ; SAVE VALID BITS OF KMCR
BIS #BIT08,KMCR ; SELECT AVAILABILITY BITS
MOV KMCR,R3 ; SAVE AVAILABILITY BITS

; ACCESS LOCATION NOT IN CACHE
;
MOV #200,R1 ; ACCESS ADDRESS
MOV #400,$GDDAT ; THE PATTERN TO BE WRITTEN
JSR PC,DDOUT ; DO DIAGNOSTIC DATA OUT
BIC #BIT08,KMCR ; SELECT VALID BITS
CMP R2,KMCR ; KMCR CHANGED?
BEQ 14 ; IF NOT, BRANCH
MOV R2,$GDDAT ; EXPECTED PATTERN
MOV KMCR,$BDDAT ; RECIEVED PATTERN
ERROR -26 ; WRITE MISSES AFFECTS CACHE
14: BIS #BIT08,KMCR ; SELECT AVAILABILITY BITS
CMP R3,KMCR ; KMCR CHANGED?
BEQ 24 ; IF OK, BRANCH

```

T27 WRITE CYCLES AND CACHE

2673	011632	010337	001124		MOV	R3,\$GDDAT		; EXPECTED PATTERN
2674	011636	013737	177734	001126	MOV	KMCR,\$BDDAT		; RECIEVED PATTERN
2675	011644	104026			ERROR	+26		; WRITE MISSES AFFETS CACHE
2676								
2677								
2678								
2679	011646	042737	000100	177734	2\$: BIC	@BIT06,KMCR		; DISABLE CACHE TO GET TO KNOW STATE
2680	011654	052737	000100	177734	BIS	@BIT06,KMCR		; ENABLE CACHE
2681	011662	012701	000000		MOV	@0,R1		; ADDRESS 0
2682	011666	004737	002254		JSR	PC,DDIN		; ALLOCATE CACHE SET A
2683	011672	012701	000020		MOV	@20,R1		; ADDRESS 20
2684	011676	004737	002254		JSR	PC,DDIN		; AND ALLOCATE SET B
2685	011702	005001			CLR	R1		; ACCESS SET A
2686	011704	013737	000200	001124	MOV	@200,\$GDDAT		; THE PATTERN TO BE WRITTEN
2687	011712	004737	002242		JSR	PC,DDOUT		; INVALIDATE CACHE
2688	011716	032737	100000	177734	BIT	@BIT15,KMCR		; HIT?
2689	011724	001012			BNE	3\$; IF YES, BRANCH
2690	011726	013737	177734	001124	MOV	KMCR,\$GDDAT		; PRESERVE LOW BYTE IF ERROR
2691	011734	112737	000363	001124	MOVB	@363,\$GDDAT		; HIT, B LEAST AVAILABLE
2692	011742	013737	177734	001126	MOV	KMCR,\$BDDAT		; RECIEVED PATTERN
2693	011750	104026			ERROR	+26		; WRITE HIT WAS NOT RECORDED
2694	011752	042737	000400	177734	3\$: BIC	@BIT08,KMCR		; READ VALID BITS
2695	011760	032737	010000	177734	BIT	@BIT12,KMCR		; A VALID?
2696	011766	001412			BEQ	4\$; IF INVALIDATED, BRANCH
2697	011770	013737	177734	001124	MOV	KMCR,\$GDDAT		; PRESERVE LOW BYTE IF ERROR
2698	011776	112737	000010	001125	MOVB	@BIT03,\$GDDAT+1		; MISS, B VALID
2699	012004	013737	177734	001126	MOV	KMCR,\$BDDAT		; RECIEVED PATTERN
2700	012012	104026			ERROR	+26		; WRITE HIT DOES NOT INVALIDATE
2701	012014	013737	177734	001124	4\$: MOV	KMCR,\$GDDAT		; PRESERVE LOW BYTE IF ERROR
2702	012022	112737	000200	001125	MOVB	@200,\$GDDAT+1		; HIT RECIEVED
2703	012030	013737	000220	000220	MOV	@220,@220		; CPU WRITE TO SET B
2704	012036	032737	100000	177734	BIT	@BIT15,KMCR		; HIT?
2705	012044	001004			BNE	5\$; IF YES, BRANCH
2706	012046	013737	177734	001126	MOV	KMCR,\$BDDAT		; STORE KMCR
2707	012054	104026			ERROR	+26		; HIT NOT RECORDED
2708	012056	032737	004000	177734	5\$: BIT	@BIT11,KMCR		; B STILL VALID?
2709	012064	001412			BEQ	6\$; IF NOT, EXIT TEST
2710	012066	013737	177734	001124	MOV	KMCR,\$GDDAT		; IN CASE HIT WENT AWAY
2711	012074	142737	000030	001125	BICB	@BIT04!BIT03,\$GDDAT+1		; CLEAR ALL EXTRA
2712	012102	013737	177734	001126	MOV	KMCR,\$BDDAT		; RECIEVED PATTERN
2713	012110	104026			ERROR	+26		; CPU WRITE DOES NOT INVALIDATE
2714	012112	042737	000100	177734	6\$: BIC	@BIT06,KMCR		; DISABLE CACHE
2715	012120	042737	000040	172516	BIC	@BIT05,MMR3		; AND MAPPING
2716	012126	042737	000400	177730	BIC	@BIT08,DCSR		; EXIT DIAGNOSTIC MODE
2717	012134				100\$:			

TEST - DMA READ WITH INDEX NOT ZERO

```

2719 .SBTTL TEST - DMA READ WITH INDEX NOT ZERO
2720
2721 ;* THIS TEST CHECKS THAT A DMA READ MISS DOES NOT AFFECT THE CACHE IF THE INDEX
2722 ;* FIELD IS NOT ZERO.
2723 ;
2724 ; BGNTST
2725 ;
2726 ; SAVE KMCR<15-9> FOR KMCR<8>=0,1
2727 ; DO DIAGNOSTIC_DATA_IN FOR A LOCATION WITH NON-ZERO INDEX
2728 ; DO FOR KMCR<8>=0,1
2729 ; . IF KMCR<15-9> NE SAVED THEN
2730 ; . ERROR IN COMPARING INDEXES
2731 ; . ENDF
2732 ; ENDDO
2733 ;
2734 ; ENDTST
2735 ;
2736 ;-----
2737 ;
2738 ;

```

```

;*****
;*TEST 30 DMA READ WITH INDEX NOT ZERO
;*****
TST30: SCOPE

```

```

2739 012134 000004
2740 012136 005737 001720 TST PMIS ; ANY PMI MEMORY?
2741 012142 001502 BEQ TST31 ;; IF NONE, EXIT TEST
2742
2743
2744 ;
2745 ; ENABLE CACHE AND ALLOCATE SET A

```

```

2746 012144 052737 000400 177730 BIS #BIT08,DCSR ; SELECT DIAGNOSTIC MODE
2747 012152 052737 000040 172516 BIS #BIT05,MMR3 ; ENABLE RELOCATION
2748 012160 052737 000100 177734 BIS #BIT06,KMCR ; ENABLE CACHE
2749 012166 012737 000200 170200 MOV #200,MAPLOO ; POINT MAPOO TO FIRST 8KB
2750 012174 012737 000000 170202 MOV #0,MAPHOO ; IN HIGH AND LOW MAP REGISTERS
2751 012202 012701 000000 MOV #0,R1 ; ALLOCATE 200-216
2752 012206 004737 002254 JSR PC,DDIN ; IN CACHE
2753 ;
2754 ; TRY TO CHECK WHETHER NON-ZERO INDEX EFFECT CACHE
2755 ;

```

```

2756 012212 042737 000400 177734 BIC #BIT08,KMCR ; SELECT VALID BITS
2757 012220 013737 177734 001124 MOV KMCR,$GDDAT ; SAVE VALID BITS OF KMCR
2758 012226 052737 000400 177734 BIS #BIT08,KMCR ; SELECT AVAILABILITY BITS
2759 012234 013703 177734 MOV KMCR,R3 ; SAVE AVAILABILITY BITS
2760 012240 012701 000102 MOV #102,R1 ; TRY CACHING 102
2761 012244 004737 002254 JSR PC,DDIN ; DO DIAGNOSTIC DATA IN
2762 012250 042737 000400 177734 BIC #BIT08,KMCR ; SELECT VALID BITS
2763 012256 023737 001124 177734 CMP $GDDAT,KMCR ; KMCR CHANGED?
2764 012264 001404 BEQ 1$ ; IF NOT, BRANCH
2765 012266 013737 177734 001126 MOV KMCR,$BDDAT ; RECIEVED PATTERN
2766 012274 104026 ERROR +26 ; NON-ZERO INDEX AFFECTS CACHE
2767 012276 052737 000400 177734 1$: BIS #BIT08,KMCR ; SELECT AVAILABILITY BITS
2768 012304 020337 177734 CMP R3,KMCR ; KMCR CHANGED?
2769 012310 001406 BEQ 2$ ; IF NOT, EXIT TEST
2770 012312 010337 001124 MOV R3,$GDDAT ; EXPECTED DATA
2771 012316 013737 177734 001126 MOV KMCR,$BDDAT ; RECIEVED DATE
2772 012324 104026 ERROR +26 ; NON-ZERO INDEX AFFECTS CACHE

```


TEST - TAG REGISTERS

```

2778          .SBTTL TEST - TAG REGISTERS
2779
2780          ; * THIS TEST WILL AUTOSIZE MEMORY IN 128K WORDS. THEN DEPENDING
2781          ; * ON THE AMOUNT OF MEMORY AVAILABLE, A PATTERN OF ALTERNATING 0'S
2782          ; * AND 1'S WILL BE CONSTRUCTED TO VERIFY THE TAG REGISTERS.
2783          ;
2784          ; BGNST
2785          ;
2786          ;     SAVE TIMEOUT VECTOR
2787          ;     ENABLE MMU
2788          ;     LET R1 = KIPAR6
2789          ;     REPEAT TO CONSTRUCT A MASK FOR NXM FOR MAPH01
2790          ;     . LET R1 = R1 SHIFT.LEFT BY #1
2791          ;     . LET R1 = R1 * KIPAR6
2792          ;     UNTIL CARRY SET
2793          ;     SWAP R1
2794          ;     LET R1 = R1 SHIFT.RIGHT BY #3 TO GET BITS<21-16> FROM <21-13>
2795          ;     LET KMCR<6> = #1 TO ENABLE CACHE
2796          ;     LET R2 = #TBLML FOR PATTERNS FOR LOW REGISTER
2797          ;     LET R3 = #TBLMH FOR PATTERNS FOR HIGH REGISTER
2798          ;     DO FOR ALL 16 PATTERNS
2799          ;     . DO FOR 4 PATTERNS
2800          ;     . . LET MAPL01 = (R2)*
2801          ;     . . LET MAPH01 = (R3)*
2802          ;     . . LET MAPH01 = MAPH01 CLEAR.BY R1 NOT TO GET NXM
2803          ;     . . DO DIAGNOSTIC_DATA_IN TO ALLOCATE TAG PATTERN
2804          ;     . ENDDO
2805          ;     . MOVE POINTERS THRU PATTERN TABLE TO PREVIOUS 4 WORDS
2806          ;     . ENABLE 22-BIT MMU
2807          ;     . DO FOR 4 PATTERNS
2808          ;     . . LET MAPL01 = (R2)*
2809          ;     . . LET MAPH01 = (R3)*
2810          ;     . . LET MAPH01 = MAPH01 CLEAR.BY R1 NOT TO GET NXM
2811          ;     . . DO DIAGNOSTIC_DATA_IN FROM IN CACHE
2812          ;     . . IF NOT A HIT THEN
2813          ;     . . . ERROR IN TAG REGISTERS
2814          ;     . ENDDO
2815          ;     . DISABLE 22-BIT MMU
2816          ;     ENDDO
2817          ;
2818          ; ENDTST
2819          ;
2820          ;-----
2821          ; *****
2822          ; *TEST 31 TAG REGISTERS
2823          ; *****

```

```

012350 000004
2823
2824 012352 005737 001720
2825 012356 001002
2826 012360 000137 012670
2827 012364 032737 000040 177734 1$:
2828 012372 001177
2829 012374 052737 000400 177730
2830 012402 004737 002276
2831 012406 005237 177572

```

```

TST31: SCOPE
          TST    PMIS          ; ANY PMI MEMORY?
          BNE    1$           ; YES
          JMP    100$         ; IF NONE, EXIT TEST
          BIT    #BIT05,KMCR  ; 18-BIT MODE?
          BNE    TST32        ;; IF YES, EXIT TEST
          BIS    #BIT08,DCSR  ; SELECT DIAGNOSTIC MODE
          JSR    PC,MAPPR     ; REMAP PROGRAM AREA
          INC    MMRO         ; ENABLE MMU

```

T31 TAG REGISTERS

```

2832 012412 052737 000020 172516      BIS      #BIT04,MMR3      ; ENABLE 22-BIT
2833 012420 022737 000040 001720      CMP      #40,PMIS     ; IF 2M PRESENT, ONLY 21ST MASKED
2834 012426 001412                BEQ      5$           ; DON'T CONSTRUCT MASK
2835
2836 ;
2837 ; IF LESS THEN 2M OF MEMORY, MAKE UP A MASK FOR MAP HIGH REGISTERS
2838 012430 000241                CLC      ; CLEAR CARRY
2839 012432 013701 001720      MOV      PMIS,R1     ; SAVE KIPAR6
2840 012436 006101                ROL      R1          ; . ROTATE LEFT R1
2841 012440 053701 001720      BIS      PMIS,R1     ; . AND ADD BIT AT FIRST POSITION
2842 012444 103374                BHIS    4$           ; . IF CARRY NOT SET, BRANCH
2843 012446 000301                SWAB    R1          ; <15-8><----><7-0>
2844 012450 006001                ROR     R1          ; NOW GET ADDRESS BITS <21-16>
2845 012452 006001                ROR     R1          ; FROM <21-14>
2846
2847 ;
2848 ; ALLOCATE DIFFERENT PATTERNS IN TAG REGISTERS
2849 012454 042737 000100 177734 5$:    BIC      #BIT06,KMCR ; DISABLE CACHE
2850 012462 052737 000100 177734      BIS      #BIT06,KMCR ; ENABLE CACHE
2851 012470 042737 000400 177734      BIC      #BIT08,KMCR ; MAKE SURE THAT VALID READ
2852 012476 052737 000040 172516      BIS      #BIT05,MMR3 ; ENABLE MAPPING
2853 012504 012702 012672      MOV      #TBLML,R2  ; POINTER FOR LOW MAP PATTERNS
2854 012510 012703 012732      MOV      #TBLMH,R3  ; POINTER FOR HIGH MAP PATTERNS
2855 012514 010137 001160      MOV      R1,$TMPO   ; SAVE THE MASK
2856 012520 012704 000004      MOV      #4,R4      ; COUNTER FOR 16 PATTERNS
2857 012524 012701 020000 6$:    MOV      #20000,R1  ; . TO ACCESS THRU MAP REG. 1
2858 012530 012705 000004      MOV      #4,R5      ; . DO EACH FOR AT A TIME
2859 012534 012237 170204 7$:    MOV      (R2)+,MAPL01 ; . . LOAD LOW MAP
2860 012540 012337 170206      MOV      (R3)+,MAPH01 ; . . LOAD HIGH MAP
2861 012544 043737 001160 170206      BIC      $TMPO,MAPH01 ; . . MASK OUT NXM
2862 012552 004737 002254      JSR     PC,DDIN     ; . . DO DIAGNOSTIC DATI
2863 012556 077512                SOB     R5,7$       ; . . DO FOR ALL 4 PATTERNS
2864 012560 013701 177734      MOV      KMCR,R1    ; . STORE SETS VALID?
2865 012564 005101                COM     R1          ; . COMPLEMENT VALID BITS
2866 012566 032701 017000      BIT     #17000,R1   ; . THAT'S ALL VALID BITS
2867 012572 001401                BEQ     8$           ; . IF ALL SET BEFORE COM, BRANCH
2868 012574 104027                ERROR   +27        ; . NOT ALL VALID BITS SET
2869
2870 ;
2871 ; NOW INVALIDATE BY WRITING TO THE SAME LOCATIONS
2872 012576 162702 000010 8$:    SUB      #10,R2     ; . MOVE POINTER TO -4
2873 012602 162703 000010      SUB      #10,R3     ; . MOVE HIGH MAP POINTER TOO
2874 012606 012705 000004      MOV      #4,R5      ; . VALIDATE ALLOCATED PATTERNS
2875 012612 012701 020000      MOV      #20000,R1  ; . POINTER TO MAP 1
2876 012616 012237 170204 9$:    MOV      (R2)+,MAPL01 ; . . LOAD LOW MAP
2877 012622 012337 170206      MOV      (R3)+,MAPH01 ; . . LOAD HIGH MAP
2878 012626 043737 001160 170206      BIC      $TMPO,MAPH01 ; . . MASK OUT NXM
2879 012634 004737 002254      JSR     PC,DDIN     ; . . DO DAIGN. DATA IN
2880 012640 032737 100000 177734      BIT     #BIT15,KMCR ; . . HIT?
2881 012646 001001                BNE     10$         ; . . IF HIT, BRANCH
2882 012650 104027                ERROR   +27        ; . . IN TAG REGISTERS
2883 012652 077517                SOB     R5,9$       ; . . DO FOR ALL 4 PATTERNS
2884 012654 077455                SOB     R4,6$       ; . REPEAT FOR ALL 16 PATTERNS
2885 012656 005037 177572      CLR     MMRO        ; DISABLE MMU
2886 012662 042737 000400 177730      BIC      #BIT08,DCSR ; EXIT DIAGNOSTIC MODE
2887 012670
2887 012670 000440 100$:    BR      TST32      ;: EXIT TEST

```

T31 TAG REGISTERS

```

2888
2889
2890
2891
2892 012672 125240 052520 000000 ; PATTERNS FOR LOW MAP REGISTER
      012700 177760 ;
      TBLML: .WORD 125240,52520,0,177760 ; FIRST PATTERN THRU 4 TAGS
2893 012702 052520 000000 177760 .WORD 52520,0,177760,125240 ; SECOND PATTERN
      012710 125240
2894 012712 000000 177760 125240 .WORD 0,177760,125240,52520 ; THIRD
      012720 052520
2895 012722 177760 125240 052520 .WORD 177760,125240,52520,0 ; FOURTH
      012730 000000

2896
2897
2898
2899 012732 000052 000025 000000 ; PATTERNS FOR HIGH MAP REGISTER
      012740 000077 ;
      TBLMH: .WORD 52,25,0,77 ; FIRST FOR HIGH MAP REGISTER
2900 012742 000025 000000 000077 .WORD 25,0,77,52 ; SECOND
      012750 000052
2901 012752 000000 000077 000052 .WORD 0,77,52,25 ; THIRD
      012760 000025
2902 012762 000077 000052 000025 .WORD 77,52,25,0 ; FOURTH
      012770 000000

```

TEST - CACHE RAM BIT PATTERN TEST

```

2904 .SBTTL TEST - CACHE RAM BIT PATTERN TEST
2905
2906 ;* THIS TEST IS DONE TO CHECK THE OPERATION OF
2907 ;* THE CACHE RAM . BIT PATTERNS 0,177777,52525
2908 ;* AND 125252 ARE WRITTEN TO ALL OF THE MEMORY
2909 ;* USED AND THEN VERIFIED
2910 ;
2911 ; BGNST
2912 ;
2913 ;*
2914 ;* GO SET UP THE CACHE IMAGE TABLE
2915 ;*
2916 ; LET RO := #CTBLE
2917 ; REPEAT
2918 ; . IF RO POINTS TO OCTAL BOUNDARY ADDRESS THEN
2919 ; . . WE HAVE FOUND THE BEGINNING OF THE TABLE
2920 ; . ELSE
2921 ; . . POINT RO TO THE NEXT ADDRESS
2922 ; . ENDF
2923 ; UNTIL WE HAVE FOUND BEGINNING OF THE TABLE
2924 ;*
2925 ;* SET UP BOARD FOR THE TRANSFERS
2926 ;*
2927 ; ENABLE UNIBUS MAPPING
2928 ; ENABLE DIAGNOSTIC MODE
2929 ; ENABLE CACHE
2930 ; DO FOR PATTERN := 0,177777,52525,125252
2931 ;*
2932 ;* INITIALIZE THE CACHE TABLE
2933 ;*
2934 ; . DO FOR ALL OF THE CACHE TABLE
2935 ; . . WRITE PATTERN TO TABLE
2936 ; . ENDDO
2937 ;*
2938 ;* ALLOCATE ALL THE CACHE
2939 ;*
2940 ; . DO UNTIL ALL CACHE ALLOCATED
2941 ; . . DO A DIAGNOSTIC DATI NPR CYCLE (ON OCTAL BOUNDARY)
2942 ; . ENDDO
2943 ;*
2944 ;* CHECK THAT THE DATA GOT CACHED CORRECTLY
2945 ;*
2946 ; . DO FOR ALL OF THE CACHE TABLE
2947 ; . . DO A DIAGNOSTIC DATI NPR CYCLE
2948 ; . . IF DDR NEQ PATTERN THEN
2949 ; . . . ERROR IN CACHE RAM
2950 ; . . ENDF
2951 ; . ENDDO
2952 ; ENDDO
2953 ;
2954 ; ENDTST
2955 ;
2956 ;-----
2957 ;*****
2958 ;*TEST 32 CACHE RAM BIT TEST
;*****

```


T32 CACHE RAM BIT TEST

```

2959 012772 000004      TST32: SCOPE
2960 012774 005737 001720    TST     PMIS      ; ANY PMI MEMORY?
2961 013000 001511      BEQ     TST33     ;; IF NONE, EXIT TEST
2962      ;
2963      ; FIND THE FIRST OCTAL BOUNDARY ADDRESS IN THE TABLE
2964 013002 052737 000400 177730    BIS     #BIT08,DCSR ; SELECT DIAGNOSTIC MODE
2965 013010 012700 002054      MOV     #CTBLE,R0  ; GET POINTER TO THE START OF THE SPACE ALLOCATED FOR THE CA
CHE TABLE
2966 013014 010001      5$:    MOV     R0,R1      ; SAVE IT IN R1 FOR WORKING
2967 013016 042701 177760    BIC     #177760,R1 ; MASK IN THE LEAST SIGNIFICANT DIGIT
2968 013022 005701      TST     R1          ; IS THE ADDRESS ON AN OCTAL BOUNDARY ?
2969 013024 001402      BEQ     10$        ; YES, THEN GO SET UP THE TABLE ADDRESSES
2970 013026 005720      TST     (R0)+      ; NO, THEN GET THE NEXT ADDRESS
2971 013030 000771      BR     5$         ; . GO CHECK IF IT FALLS ON AN OCTAL BOUNDARY
2972 013032 012702 002172    10$:   MOV     #OBADR,R2 ; GET POINTER TO THE OCTAL BOUNDARY TABLE
2973 013036 012703 000004      MOV     #4,R3      ; SET UP THE LOOP COUNTER
2974 013042 010022      12$:   MOV     R0,(R2)+   ; . GO PUT ADDRESS INTO THE TABLE
2975 013044 062700 000020      ADD     #20,R0     ; . GET NEXT OCTAL BOUNDARY
2976 013050 077304      SOB     R3,12$    ; . FILL UP THE TABLE ?
2977      ;
2978      ; INITIALIZE THE BOARD FOR THE TRANSFERS
2979      ;
2980 013052 052737 000040 172516    BIS     #BIT5,#MMR3 ; ENABLE UNIBUS MAPPING
2981 013060 005037 170202      CLR     MAPH00     ; CLEAR HIGH MAP REGISTER
2982 013064 052737 000400 177730    BIS     #BIT8,DCSR ; ENABLE DIAGNOSTIC MODE
2983 013072 052737 000100 177734    BIS     #BIT6,KMCR ; ENABLE THE CACHE
2984 013100 012700 001774      MOV     #PTRN16,R0 ; GET POINTER TO THE CACHE PATTERN TABLE
2985 013104 013701 002172    15$:   MOV     OBADR,R1  ; GET POINTER TO THE CACHE IMAGE TABLE
2986      ;
2987      ; INITIALIZE THE CACHE IMAGE TABLE TO CURRENT PATTERN
2988      ;
2989 013110 012702 000040      MOV     #40,R2     ; . SET UP LOOP COUNTER
2990 013114 011021      20$:   MOV     (R0),(R1)+ ; . . MOVE PATTERN TO TABLE
2991 013116 077202      SOB     R2,20$    ; . . HAVE WE DONE IT 40 TIMES ?
2992      ;
2993      ; ALLOCATE ALL THE CACHE MEMORY AVAILABLE
2994      ;
2995 013120 012702 002172      MOV     #OBADR,R2 ; . GET ADDRESS OF OCTAL BOUNDARIES WITHIN THE TAB:E
2996 013124 012703 000004      MOV     #4,R3      ; . SET UP LOOP COUNTER
2997 013130 005001      CLR     R1         ; . NON-MAPPED ADDRESS=0
2998 013132 012237 170200    25$:   MOV     (R2)+,MAPL00 ; . . SET UP MAP FOR DIAG NPR DATI CALL
2999 013136 004737 002254      JSR     PC,DDIN    ; . . GO DO A DIAGNOSTIC DATA IN CYCLE
3000 013142 077305      SOB     R3,25$    ; . . HAVE WE DONE IT 4 TIMES ?
3001      ;
3002      ; CHECK THE CACHE RAM IMAGE PATTERN
3003      ;
3004 013144 012703 000040      MOV     #40,R3     ; . SET UP LOOP COUNTER
3005 013150 013702 002172      MOV     OBADR,R2  ; . GET ADDRESS OF CACHE IMAGE TABLE
3006      ; (THIS WHOLE TABLE SHOULD BE CACHED)
3007 013154 005001      CLR     R1         ; . USE ADDRESS 0
3008 013156 010237 170200    30$:   MOV     R2,MAPL00 ; . . GET ADDRESS FPR DIAG. DATA IN CYCLE
3009 013162 004737 002254      JSR     PC,DDIN    ; . . GO DO A DIAGNOSTIC DATA IN
3010 013166 032737 100000 177734    BIT     #BIT15,KMCR ; . . HIT?
3011 013174 001001      BNE     35$       ; . . IF YES, BRANCH
3012 013176 104030      ERROR  +30        ; . . OTHERWISE, ERROR
3013 013200 023722 177732    35$:   CMP     DDR,(R2)+ ; . . DID IT GET STORED CORRECTLY ?
3014 013204 001401      BEQ     40$       ; . . YES, THEN GO CHECK THE NEXT LOCATION

```

T32 CACHE RAM BIT TEST

3015	013206	104030		ERROR	+30		; . . NO, THEN ERROR IN CACHE RAM
3016	013210	077316		S0B	R3,30\$; . . HAVE WE CHECKED ALL THE RAM ?
3017	013212	005720	40\$:	TST	(R0)+		; . ALL PATTERNS DONE?
3018	013214	001333		BNE	15\$; . IF NOT HAVE DONE, BRANCH
3019	013216	042737	000400 177730	BIC	#BIT08,DCSR		; EXIT DIAGNOSTIC MODE
3020							

TEST - BOOT ROMS TEST

```

3022 .SBTTL TEST - BOOT ROMS TEST
3023
3024 ;* THIS TEST CHECKS FOR THE PRESENCE OF BOOT ROMS. IF THEY ARE FOUND,
3025 ;* A CRC TEST IS PERFORMED FOR ALL THE ROM'S.
3026 ;
3027 ; BGMTST
3028 ;
3029 ; LET BCSR = BCSR SET BY #BIT05 TO ENABLE ACCESSING OF ROM'S
3030 ; LET PCR = #0 TO ACCESS FIRST PAGE
3031 ; LET R4 = #173000 FOR STARTING ADDRESS
3032 ; LET $TMPO = #0 TO CLEAR COUNT FOR EMPTY ROM'S
3033 ; DO FOR ALL 4 POSSIBLE ROMS
3034 ; . IF (R4) EQ #161777 (EMPTY SOCKET) THEN
3035 ; . . INCREMENT $TMPO TO COUNT EMPTY ROM'S
3036 ; . . IDENTIFY POSITION OF EMPTY SOCKET
3037 ; . ELSE
3038 ; . . LET R1 = #0 TO COUNT BYTES IN ROMS
3039 ; . . DO FOR EACH LOCATION IN A ROM UNTILL R1=128
3040 ; . . . IF LOCATION 24 ACCESSED THEN
3041 ; . . . . IF (R4) NE #173000 THEN
3042 ; . . . . . ERROR IN LOCATION 24
3043 ; . . . . . ENDF
3044 ; . . . . INCREMENT R1 NOT TO DO BYTE 25
3045 ; . . . . ELSE
3046 ; . . . . . CALCULATE CRC FOR THE BYTE
3047 ; . . . . . ENDF
3048 ; . . . . INCREMENT R1 FOR THE NEXT BYTE
3049 ; . . ENDDO
3050 ; ENDDO
3051 ; IF $TMPO NE #4 NOT ALL EMPTY
3052 ; . TRY TO WRITE TO @#173000 IN DIAGNOSTIC MODE AND OUT
3053 ; . IF NO TIMEOUT THEN
3054 ; . . ERROR WRITE ACCESS TO ROMS DOES NOT TIMEOUT
3055 ; . ENDF
3056 ; ENDF
3057 ;
3058 ; ENDTST
3059 ;
3060 ;

```

```

3061
3062 013224
UBETST:
;*****
;*TEST 33 BOOT ROMS TEST
;*****
TST33: SCOPE

```

```

013224 000004
3063
3064
3065 ; PREPARE TO DO CRC
3066 ;
3067 013226 042737 000410 177730 BIC #BIT08!BIT03,DCSR ; LEAVE STANDALONE MODE AND ENABLE ROMS
3068 013234 052737 000200 177520 BIS #BIT07,BCSR ; TO ENABLE BOOT ROMS
3069 013242 005037 177522 CLR PCR ; TO READ PAGE 0
3070 013246 012704 173000 MOV #173000,R4 ; R4 POINTS TO FIRST ADDRESS
3071 013252 005037 001160 CLR $TMPO ; CLEAR EMPTY SOCKET COUNT
3072 ;
3073 ; DO FOR EACH POSSIBLE ROM
3074 ;

```

T33 BOOT ROMS TEST

```

3075 013256 022704 173776      1$:   CMP      #173776,R4      ; . ALL ROM'S DONE?
3076 013262 103475              BLO      11$           ; . IF SO, EXIT
3077 013264 005001              CLR      R1            ; . CLEAR COUNTER THRU A ROM
3078 013266 005005              CLR      R5            ; . INITIALIZE PARTIAL CRC
3079
3080      ; CHECK FOR EMPTY SOCKETS AND EXIT IF SO
3081
3082 013270 022714 161777      :     CMP      #161777,(R4) ; . EMPTY SOCKET?
3083 013274 001027              BNE      3$           ; . IF NOT, BRANCH TO DO CRC
3084 013276 005737 001206      :     TST      $PASS     ; . FIRST PASS?
3085 013302 001017              BNE      2$           ; . IF NOT, BRANCH
3086 013304 122737 000001 001220 :     CMPB   #APTENV,$ENV ; . IN APT MODE?
3087 013312 001413              BEQ      2$           ; IF SO, SKIP PRINTOUT
3088 013314 032737 000040 000052 :     BIT      #BIT05,@#52 ; . IN UFD MODE?
3089 013322 001007              BNE      2$           ; . IF IN UFD, BRANCH
3090 013324 010446              MOV      R4,-(SP)     ; . PUT NUMBER TO TYPE OUT
3091 013326 104401 013612      :     TYPE     ,NROM    ; . TYPE ASCII MESSAGE
3092 013332 104403              TYPOS    ; . CALL TYPE OUT ROUTINE
3093 013334      006              .BYTE   6            ; . TYPE 6 DIGITS
3094 013335      000              .BYTE   0            ; . SUPPRESS LEADING 0'S
3095 013336 104401 001175      :     TYPE     ,$CRLF   ; . CARRIAGE RETURN
3096 013342 005237 001160      2$:   INC      $TMPO    ; . INCREMENT EMPTY SOCKET COUNT
3097 013346 062704 000200      :     ADD      #200,R4  ; . PREPARE TO DO NEXT ROM
3098 013352 000741              BR       1$           ; . BRANCH TO DO NEXT ROM
3099
3100      ; IN EACH ROM CHECK LOCATION 24 AND DON'T DO CRC ON IT
3101
3102 013354 022701 000024      3$:   CMP      #24,R1      ; . LOCATION 24 ACCESSED?
3103 013360 001007              BNE      5$           ; . IF NO, GO DO CRC
3104 013362 022724 173000      :     CMP      #173000,(R4)+ ; . PROPER DATA AT 24?
3105 013366 001401              BEQ      4$           ; . IF YES, BRANCH
3106 013370 104031              ERROR   +31          ; . WRONG DATA AT 24
3107 013372 005201              4$:   INC      R1        ; . INCREMENT FOR AN EXTRA BYTE
3108 013374 000137 013434      :     JMP      8$           ; . DO NOT DO CRC FOR 24
3109
3110      ; DO CRC FOR EACH BYTE
3111
3112 013400 112403              5$:   MOVB   (R4)+,R3    ; . STORE CORRECT DATA IN R3
3113 013402 012702 000010      :     MOV      #8.,R2    ; . NUMBER OF BITS PER BYTE
3114 013406 000241              6$:   CLC              ; . . CLEAR CARRY
3115 013410 006005              :     ROR      R5        ; . . LOW BIT PARTIAL TO CARRY
3116 013412 006003              :     ROR      R3        ; . . CARRY TO BYTE AND BYTE TO CARRY
3117 013414 102006              :     BVC     7$         ; . . XOR OF PARTIAL AND BYTE LOW BITS
3118 013416 012746 120001      :     MOV      #POLY,-(SP) ; . . XOR POLY TO PARTIAL (4 INSTRUCTIONS)
3119 013422 040516              :     BIC     R5,(SP)    ; . . NOT PARTIAL AND POLY
3120 013424 042705 120001      :     BIC     #POLY,R5   ; . . NOT POLY AND PARTIAL
3121 013430 052605              :     BIS     (SP)+,R5   ; . . POLY XOR PARTIAL
3122 013432 077213              7$:   SOB     R2,6$      ; . . DECREMENT BIT COUNT AND CONTINUE
3123 013434 005201              8$:   INC      R1        ; . COUNT BYTES
3124
3125      ; IF A ROM DONE, CHECK FOR 0 IN R5
3126
3127 013436 022701 000200      :     CMP      #128.,R1  ; . ALL 64 WORDS DONE?
3128 013442 003344              BGT      3$           ; . IF NOT, BRANCH
3129 013444 005705              TST      R5           ; . IF YES, CRC 0?
3130 013446 001401              BEQ      10$          ; . IF CRC = 0, BRANCH
3131 013450 104031              ERROR   +31          ; . CRC FOR A ROM NOT EQUAL TO 0

```

T33 BOOT ROMS TEST

```

3132 013452 000137 013256      10$:   JMP      1$           ; . DO FOR NEXT ROM
3133                               ;
3134                               ; TRY TO WRITE TO GET A TIMEOUT
3135                               ;
3136 013456 013737 000004 001160 11$:   MOV      ERRVEC,$TMP0      ; SAVE TIMEOUT VECTOR
3137 013464 012737 013502 000004      MOV      #15$,ERRVEC      ; POINT NEW TO PROGRAM
3138 013472 005037 173000      14$:   CLR      @#173000        ; TRY TO WRITE TO 1ST PAGE OF ROM
3139 013476 104031                ERROR   +31                ; WRITE ACCESS TO ROM'S DIDN'T TIMEOUT
3140 013500 000402                BR      16$                ;
3141 013502 005726                15$:   TST      (SP)+          ; RESTORE STACK
3142 013504 005726                TST      (SP)+          ;
3143 013506 032737 000400 177730 16$:   BIT      #BIT08,DCSR      ; OUT OF STANDALONE MODE?
3144 013514 001013                BNE     20$                ; IF NOT, EXIT TEST
3145 013516 013702 177734                MOV      KMCR,R2          ; MAKE SURE THAT
3146 013522 042702 177700                BIC      #177700,R2       ; AT LEAST SOME OF
3147 013526 022702 000077                CMP      #77,R2          ; MEMORY PMI
3148 013532 001404                BEQ     20$                ; IF NOT, BRANCH
3149 013534 052737 000400 177730      BIS      #BIT08,DCSR      ; DO 1 MORE TIME IN STANDALONE MODE
3150 013542 000753                BR      14$                ;
3151 013544 052737 000010 177730 20$:   BIS      #BIT03,DCSR      ; DISABLE UBA ROM RESPONSE
3152 013552 012737 013570 000004      MOV      #25$,ERRVEC      ; POINT NEW TIMEOUT VECTOR
3153 013560 005737 173000                TST      @#173000        ; READ BOOT ROM
3154 013564 104037                ERROR   +37                ; DSCR<3> DIDN'T DISABLE UBA ROM
3155 013566 000402                BR      26$                ;
3156 013570 005726                25$:   TST      (SP)+          ; RESTORE STACK
3157 013572 005726                TST      (SP)+          ;
3158 013574 042737 000410 177730 26$:   BIC      #BIT08!BIT03,DCSR ; LEAVE STANDALONE MODE
3159 013602 013737 001160 000004      MOV      $TMP0,ERRVEC      ; RESTORE STACK
3160 013610 000417                BR      TST34              ;: EXIT TEST
3161
3162 013612      101      104      104  NROM:  .ASCIZ  /ADDRESS OF EMPTY UBA SOCKET /
      013615      122      105      123
      013620      123      040      117
      013623      106      040      105
      013626      115      120      124
      013631      131      040      125
      013634      102      101      040
      013637      123      117      103
      013642      113      105      124
      013645      040      000
3163                               .EVEN

```

TEST - UNIBUS MEMORY TEST

```

3165 .SBTTL TEST UNIBUS MEMORY TEST
3166
3167 ; * THIS TEST READS KMCR<5-0> TO FIND OUT HOW MUCH UNIBUS MEMORY IS
3168 ; * AVAILABLE. THEN ALTERNATING 0'S AND 1'S WILL BE WRITTEN AND READ
3169 ; * FROM MEMORY. IF THE SYSTEM CONTAINS ALL UNIBUS MEMORY, THE FIRST
3170 ; * 32K ARE NOT GOING TO TESTED.
3171 ;
3172 ; BGNTST
3173 ; IF KMCR<5-0> = <0-0> THEN NO UNIBUS MEMORY
3174 ; . EXIT TEST
3175 ; . ENDF
3176 ; IF KMCR<5-0> = <1-1> THEN ALL UNIBUS MEMORY
3177 ; . LET R1 = #1600 LOWER BOUNDARY FOR PAR
3178 ; . LET R2 = #7600 HIGH BOUNDARY FOR PAR
3179 ; ELSE
3180 ; . LET R1 = COMPLEMENT(KMCR<4-0>)
3181 ; . LET R1 = R1 SHIFT.LEFT BY #7 TO GET LOWER BOUNDARY PAR
3182 ; . LET R2 = #7600 HIGH BOUNDARY
3183 ; . IF KMCR<5> = #0 THEN 22 BIT MODE
3184 ; . . LET R1 = R1 SET.BY #BIT15!BIT14!BIT13!BIT12 FOR 22 BITS
3185 ; . . LET R2 = R2 SET.BY #BIT15!BIT14!BIT13!BIT12
3186 ; . ENDF
3187 ; ENDF
3188 ; LET MMRO<0> = #1 TO ENABLE MMU
3189 ; REMAP PROGRAM AREA
3190 ; DO FOR KIPAR6 FROM R1 TO R2
3191 ; . DO FOR R4 FROM #140000 TO #157776 BY #2 FOR 4K THRU KIPAR6
3192 ; . . LET (R4) = #125252 TO WRITE A PATTERN
3193 ; . . IF (R4) NE #125252 THEN
3194 ; . . . ERROR IN UNIBUS MEMORY
3195 ; . . . ENDF
3196 ; . . LET (R4) = COMPLEMENT <(R4)>
3197 ; . . IF (R4) NE #52525 THEN
3198 ; . . . ERROR IN UNIBUS MEMORY
3199 ; . . . ENDF
3200 ; . ENDDO
3201 ; ENDDO
3202 ; DISABLE MMU
3203 ;
3204 ; ENDTST
3205 ;
3206 ; -----
3207 ; *****
3208 ; *TEST 34 UNIBUS MEMORY TEST
3209 ; *****

```

```

013650 000004
3209
3210
3211
3212
3213 013652 032737 000037 177734
3214 013660 001473
3215
3216
3217
3218 013662 013701 177734

```

```

; *****
; *TEST 34 UNIBUS MEMORY TEST
; *****
TST34: SCOPE
;
; CHECK IF ALL NON-UNIBUS MEMORY
;
; BIT #37,KMCR ; BITS <5-0> CLEAR?
; BEQ TST35 ;; IF YES, SKIP TEST
;
; CHECK IF ALL UNIBUS MEMORY
;
; MOV KMCR,R1 ; SAVE KMCR

```

T34 UNIBUS MEMORY TEST

```

3219 013666 042701 177700          BIC    #177700,R1          ; LEAVE ONLY <5-0>
3220 013672 022701 000077          CMP    #77,R1             ; BITS <5-0> ALL SET?
3221 013676 001005                   BNE    1#                 ; IF NOT, BRANCH
3222 013700 012701 001600          MOV    #1600,R1          ; DON'T TEST FIRST 32K
3223 013704 012702 007600          MOV    #7600,R2         ; 248KB MAXIMUM CONFIGURATION
3224 013710 000420                   BR     2#                 ; GO DO TEST
3225 013712 004737 002456          1#:   JSR    PC,UMSIZ     ; SIZE UNIBUS MEMORY
3226                                     ;
3227                                     ; ON RETURN R2 HAS PAR VALUE FOR UNIBUS MEMORY
3228                                     ;
3229 013716 010201                   MOV    R2,R1             ; STORE LOWER BOUNDARY
3230 013720 012702 007600          MOV    #7600,R2         ; HIGHER BOUNDARY
3231 013724 032737 000040 177734    BIT    #BIT05,KMCR      ; 18 BIT MODE?
3232 013732 001007                   BNE    2#                 ; IF YES, GO DO TEST
3233 013734 052701 170000          BIS    #170000,R1       ; EXTEND TO 22 BITS
3234 013740 052702 170000          BIS    #170000,R2       ; FOR HIGHER BOUNDARY TOO
3235 013744 052737 000020 172516    BIS    #BIT04,MMR3      ; ENABLE 22 BIT MAPPING
3236                                     ;
3237                                     ; NOW WRITE MEMORY WITH 01 PATTERN, VERIFY IT AND DO THE SAME FOR 10 PATTERN
3238                                     ;
3239 013752 004737 002276          2#:   JSR    PC,MAPPR     ; REMAP PROGRAM TO FIRST 32K
3240 013756 005237 177572          INC    MMRO              ; ENABLE MMU
3241 013762 010137 172354          MOV    R1,KIPAR6        ; START AT LOWER BOUNDARY
3242 013766 012704 140000          3#:   MOV    #140000,R4   ; . START 4K PAGE
3243 013772 012714 125252          4#:   MOV    #125252,(R4) ; . . WRITE FIRST PATTERN
3244 013776 022714 125252          CMP    #125252,(R4)     ; . . WRITEN OK?
3245 014002 001401                   BEQ    5#                 ; . . IF OK, BRANCH
3246 014004 104034                   ERROR  +34               ; . . IN UNIBUS MEMORY
3247 014006 005114                   CMP    (R4)              ; . . COMPLEMENT INITIAL PATTERN
3248 014010 022714 052525          5#:   CMP    #52525,(R4)  ; . . NEW PATTERN OK?
3249 014014 001401                   BEQ    6#                 ; . . IF OK, BRANCH
3250 014016 104034                   ERROR  +34               ; . . IN UNIBUS MEMORY
3251 014020 005724                   6#:   TST    (R4)         ; . . GET NEXT MEMORY LOCATION
3252 014022 022704 160000          CMP    #160000,R4       ; . . LAST ONE IN 4K PAGE?
3253 014026 101361                   BHI    4#                 ; . . IF NOT, BRANCH
3254 014030 062737 000200 172354    ADD    #200,KIPAR6      ; . GET NEXT PAGE
3255 014036 020237 172354          CMP    R2,KIPAR6        ; . LAST PAGE DONE?
3256 014042 101351                   BHI    3#                 ; . IF NOT, BRANCH
3257 014044 005037 177572          CLR    MMRO              ; DISABLE MMU
3258                                     ;
3259                                     .SBTTL UBE TESTS FOR THE UNIBUS ADAPTER BOARD
3260                                     ;
3261                                     ;*
3262                                     ;* THE FOLLOWING TESTS REQUIRE THE USE OF THE UNIBUS EXERCISER
3263                                     ;* TO TEST OUT THE UNIBUS ADAPTER BOARD.SOME TESTS REQUIRE ONE
3264                                     ;* UBE OTHER TESTS REQUIRE TWO UBE'S.
3265                                     ;*
3266                                     ;* THE PROGRAM WILL AUTOSIZE TO SEE HOW MANY UBE'S ARE IN THE
3267                                     ;* SYSTEM. DEPENDING ON THE RESULTS OF THE AUTOSIZING CERTAIN
3268                                     ;* TESTS WILL BE SELECTED OR DESELECTED.
3269                                     ;*
3270                                     ;*
3271                                     ;*

```

TEST - UBE AUTOSIZING ROUTINE

```

3273 .SBTTL TEST - UBE AUTOSIZING ROUTINE
3274
3275 ;*
3276 ;* THE FOLLOWING ROUTINE IS USED TO AUTOSIZE THE NUMBER OF
3277 ;* UBE'S PRESENT IN THE SYSTEM.
3278 ;*
3279 ;
3280 ; BGNTST
3281 ;
3282 ; LET R0 := #170000 ;1ST UBE ADDRESS
3283 ; LET R2 := #510 ;1ST UBE VECTOR
3284 ; LET R1 := #BE1DB ;TABLE FOR 1ST UBE
3285 ; LET R3 := #BE1VEC ;LOCATION FOR FIRST UBE VECTOR
3286 ; LET 4 := #TOUT ;SETUP TIMEOUT VECTOR
3287 ; DO FOR R0 := #170000 TO 170160 BY 20
3288 ; . TEST FOR (R0) ;WILL TIMEOUT IF NOT THERE
3289 ; . IF TIMEOUT OCCURS THEN
3290 ; . . LET R0 := R0 + 20 ;GET NEXT UBE ADDRESS
3291 ; . . LET R2 := R2 + 4 ;GET NEXT UBE VECTOR LOCATION
3292 ; . . LET (SP) := #CHECK
3293 ; . ELSE
3294 ;*
3295 ;* ASSIGN THE UBE ADDRESSES TO THE CURRENT UBE TABLE
3296 ;*
3297 ; . . DO FOR R4 := 1 TO 5 BY 1
3298 ; . . . LET (R1)+ := R0
3299 ; . . . LET R0 := R0+2
3300 ; . . ENDDO
3301 ; . . LET R0 := R0 + 4 ;POINT TO LAST DEVICE ADDRESS
3302 ; . . LET (R1)+ := R0
3303 ; . . LET R0 := R0 + 2 ;POINT R0 TO NEXT UBE ADDRESSES
3304 ;*
3305 ;* ASSIGN UBE VECTORS TO CURRENT UBE TABLE
3306 ;*
3307 ; . . LET (R3)+ := R2 ;GET POINTER TO VECTOR ADDRESS
3308 ; . . LET R2 := R2 + 2 ;GET VECTOR PSW LOCATION
3309 ; . . LET (R3)+ := R2 ;GET POINTER TO VECTOR PSW
3310 ; . . LET R2 := R2 + 2 ;GET NEXT VECTOR ADDRESS
3311 ; . . IF WE HAVE FOUND TWO UBE'S THEN
3312 ; . . . EXIT TEST
3313 ; . . ENDF
3314 ; . ENDF
3315 ; ENDDO
3316 ; ENDTST
3317 ;
3318 ;
3319 ;
3320 ;
3321 ;*****

```

```

;*****
;*TEST 35 UNIBUS EXERCISER AUTOSIZING ROUTINE
;*****
TST35: SCOPE

```

3322	014050	000004				
3323	014052	042737	000400	177730	BIC	#BIT08,DCSR ; MAKE SURE THAT OUT OF DIAGN. MODE
3324	014060	042737	000040	172516	BIC	#BIT05,MMR3 ; MAPPING DISABLED
3325	014066	005737	001206		TST	@#PASS ; IS THIS THE FIRST PASS ?
3326	014072	001051			BNE	10\$; NO, THEN NO NEED TO SIZE AGAIN

T35 UNIBUS EXERCISER AUTOSIZING ROUTINE

```

3327 ;
3328 ; INITIALIZE POINTERS FOR AUTO-SIZING
3329 ;
3330 014074 012700 1700J0          MOV    #170000,R0      ; GET ADDRESS OF FIRST POSSIBLE UBE
3331 014100 012702 000510          MOV    #510,R2        ; GET VECTOR OF FIRST POSSIBLE UBE
3332 014104 012701 002202          MOV    #BE1DB,R1     ; TABLE HEADER FOR 1ST UBE
3333 014110 012737 014122 000004  MOV    #1$,@#4       ; SET UP TIMEOUT VECTOR
3334 014116 005710          100$: TST    (R0)      ; . IS THERE A UBE HERE ?
3335 014120 000412          BR     2$           ; . . YES, THEN BRANCH AROUND TOMEOUT ROUTINE
3336 ;
3337 ; TIMEOUT ROUTINE
3338 ;
3339 ;
3340 014122 062706 000004          1$:  ADD    #4,SP      ; . . READJUST THE STACK
3341 014126 020027 170160          CMP    RO,#170160    ; . . HAVE WE CHECKED ALL THE ADDRESSES
3342 014132 001431          BEQ    10$          ; . . YES THEN GO SEE IF WE FOUND ANY UBE'S
3343 014134 062700 000020          ADD    #20,R0       ; . . NO, THEN GET THE NEXT UBE ADDRESS
3344 014140 062702 000004          ADD    #4,R2        ; . . GET THE NEXT UBE VECTOR
3345 014144 000764          BR     100$        ; . . GC SEE IF THE NEXT UBE IS THERE
3346 ;
3347 ; ASSIGN UBE ADDRESSES TO THE CURRENT UBE TABLE
3348 ;
3349 ;
3350 014146 012704 000005          2$:  MOV    #5,R4      ; . SET UP LOOP COUNTER
3351 014152 010021          3$:  MOV    RO,(R1)+   ; . . ASSIGN AN ADDRESS TO THE POINTER
3352 014154 005720          TST    (R0)+        ; . . GET THE NEXT ADDRESS
3353 014156 077403          SOB    R4,3$       ; . . ARE WE DONE ? NO THEN GO GET NEXT ADDRESS
3354 014160 062700 000004          ADD    #4,R0       ; . POINT RO TO LAST UBE ADDRESS
3355 014164 010021          MOV    RO,(R1)+   ; . PUT THE ADDRESS INTO THE POINTER TABLE
3356 014166 005720          TST    (R0)+        ; . POINT RO TO NEXT UBE ADDRESS
3357 ;
3358 ;
3359 ; ASSIGN UBE VECTORS TO CURRENT UBE TABLE
3360 ;
3361 ;
3362 014170 010221          MOV    R2,(R1)+   ; . GET POINTER TO VECTOR ADDRESS
3363 014172 005722          TST    (R2)+        ; . GET THE VECTOR PSW LOCATION
3364 014174 010221          MOV    R2,(R1)+   ; . GET POINTER TO VECTOR PSW
3365 014176 005722          TST    (R2)+        ; . GET THE NEXT UBE'S VECTOR ADDRESS
3366 014200 005237 001722          INC    UBECT       ; . FLAG WE HAVE FOUND ANOTHER UBE
3367 ;
3368 ;
3369 ; SEE IF WE HAVE FOUND TWO UBE'S
3370 ;
3371 ;
3372 014204 022737 000002 001722  CMP    #2,UBECT     ; . HAVE WE FOUND TWO UBE'S
3373 014212 001401          BEQ    10$          ; . GO DO THE UBE TESTS
3374 014214 000740          BR     100$
3375 ;
3376 ;
3377 ; PROGRAM WILL CHECK IF ANY UBE'S WERE FOUND
3378 ;
3379 ;
3380 014216 012737 002266 000004 10$:  MOV    #TIMOUT,@#4  ; RESTORE TIMEOUT VECTOR
3381 014224 005737 001722          TST    UBECT       ; HAVE ANY UBE'S BEEN FOUND ?
3382 014230 001002          BNE    TST36       ;; GO DO THE SELECTED TESTS FOR 1 UBE
3383 014232 000137 023122          JMP    UBEM        ; SKIP ALL THE UBE TESTS

```


TEST - NO BUS GRANTS WITH PROCESSOR AT HIGHER PRIORITY

```
3424 .SBTTL TEST - NO BUS GRANTS WITH PROCESSOR AT HIGHER PRIORITY
3425 ;
3426 ; * THIS TEST CHECKS THAT NO BUS REQUESTS ARE GOING TO BE HONORED
3427 ; * WHEN THE PROCESSOR HAS HIGHER PRIORITY THAN THE REQUESTING DEVICE.
3428 ; *
3429 ;
3430 ; BGNTST
3431 ;
3432 ; SET UP @BE1VEC TO POINT TO ERROR CHECK_ROUTINE
3433 ; SET UP @BE1PSW TO #340
3434 ; *
3435 ; * TRY TO DO BR7 WITH CPU PRIORITY AT 7
3436 ; *
3437 ; LET @BE1CR1 = #21 TO DO 1 DATI
3438 ; LET PSW = #340 TO SET PRIORITY AT 7
3439 ; DO "NOP"
3440 ; IF INTERRUPT THEN
3441 ; . ERROR BG7 GRANTED WITH PROCESSOR AT HIGHER PRIORITY
3442 ; . ENDF
3443 ;
3444 ; *
3445 ; * TRY TO DO BR6 WITH CPU PRIORITY AT 7 - 6
3446 ; *
3447 ; DO FOR R3 FROM #340 DOWNT0 #300 BY #40
3448 ; . LET @BE1CR1 = #11 TO DO 1 DATI
3449 ; . LET PSW = R3 TO CHANGE PRIORITY
3450 ; . DO "NOP"
3451 ; . IF INTERRUPT THEN
3452 ; . . ERROR BG6 GRANTED WITH PROCESSOR AT HIGHER PRIORITY
3453 ; . ENDF
3454 ; ENDDO
3455 ; LET PSW = #340 FOR THE NEXT PART
3456 ; *
3457 ; * TRY TO DO BR5 WITH CPU PRIORITY AT 7 - 5
3458 ; *
3459 ; DO FOR R3 FROM #340 DOWNT0 #240 BY #40
3460 ; . LET @BE1CR1 = #5 TO DO 1 DATI
3461 ; . LET PSW = R3 TO CHANGE PRIORITY
3462 ; . DO "NOP"
3463 ; . IF INTERRUPT THEN
3464 ; . . ERROR BG5 GRANTED WITH PROCESSOR AT HIGHER PRIORITY
3465 ; . ENDF
3466 ; ENDDO
3467 ; LET PSW = #340
3468 ; *
3469 ; * TRY TO DO BR4 WITH CPU PRIORITY AT 7 - 4
3470 ; *
3471 ; DO FOR R3 FROM #340 DOWNT0 #200 BY #40
3472 ; . LET @BE1CR1 = #2003 TO DO 1 DATI
3473 ; . LET PSW = R3 TO CHANGE PRIORITY
3474 ; . DO "NOP"
3475 ; . IF INTERRUPT THEN
3476 ; . . ERROR BR4 GRANTED WITH PROCESSOR AT HIGHER PRIORITY
3477 ; . ENDF
3478 ; ENDDO
3479 ;
3480 ; ENDTST
```

TEST - NO BUS GRANTS WITH PROCESSOR AT HIGHER PRIORITY

```

3481 ;
3482 ;-----
3483 ;*****
3484 ;*TEST 37 NO BUS GRANTS WITH PROCESSOR AT HIGHER PRIORITY
;*****
014340 000004 TST37: SCOPE
3485
3486 014342 004737 002502 JSR PC,IUBE ; INITIALIZE THE UBE
3487 014346 012737 000340 177776 MOV #340,PSW ; SET CPU PRIORITY AT 7
3488 014354 012777 000340 165636 MOV #340,@BE1PSW ; AT PRIORITY 7
3489 ;
3490 ; TRY TO DO BR7 WITH CPU PRIORITY AT 7
3491 ;
3492 014362 012777 014402 165626 MOV #1,@BE1VEC ; POINT UBE VECTOR TO PROGRAM
3493 014370 012777 000021 165612 MOV #21,@BE1CR1 ; BR7 WITH FUNO
3494 014376 000240 NOP ; JUST IN CASE
3495 014400 000405 BR 2$ ; BRANCH AROUND IF NO INTERRUPT
3496 014402 005077 165602 1$: CLR @BE1CR1 ; CLEAR ANY OTHER REQUESTS
3497 014406 062706 000004 ADD #4,SP ; ADJUST STACK POINTER
3498 014412 104032 ERROR +32 ; BG7 GRANTED WITH CPU AT 7
3499 ;
3500 ; TRY TO DO BR6 WITH CPU PRIORITY AT 7-6
3501 ;
3502 014414 012777 014444 165574 2$: MOV #4,@BE1VEC ; POINT UBE VECTOR TO PROGRAM
3503 014422 012703 000340 MOV #340,R3 ; START WITH PRIORITY AT 7
3504 014426 012777 000011 165554 3$: MOV #11,@BE1CR1 ; . BR6 WITH FUNO
3505 014434 010337 177776 MOV R3,PSW ; . CHANGE PRIORITY
3506 014440 000240 NOP ; . JUST IN CASE OF INTERRUPTS
3507 014442 000405 BR 5$ ; . IF NO INTERRUPTS, BRANCH
3508 014444 005077 165540 4$: CLR @BE1CR1 ; . CLEAR ANY OTHER REQUESTS
3509 014450 062706 000004 ADD #4,SP ; . ADJUST STACK POINTER
3510 014454 104032 ERROR +32 ; . BG6 GRANTED WITH CPU AT 6-7
3511 014456 162703 000040 5$: SUB #40,R3 ; . LOWER PRIORITY
3512 014462 020327 000300 CMP R3,#300 ; . LAST ONE TO CHECK?
3513 014466 002357 BGE 3$ ; . IF NOT, BRANCH
3514 ;
3515 ; TRY TO DO BR5 WITH CPU PRIORITY AT 7-5
3516 ;
3517 014470 012777 014520 165520 MOV #7,@BE1VEC ; POINT UBE VECTOR TO PROGRAM
3518 014476 012703 000340 MOV #340,R3 ; START WITH PRIORITY AT 7
3519 014502 012777 000005 165500 6$: MOV #5,@BE1CR1 ; . BR5 WITH FUNO
3520 014510 010337 177776 MOV R3,PSW ; . CHANGE PRIORITY
3521 014514 000240 NOP ; . JUST IN CASE OF INTERRUPTS
3522 014516 000405 BR 8$ ; . IF NO INTERRUPTS, BRANCH
3523 014520 005077 165464 7$: CLR @BE1CR1 ; . CLEAR ANY OTHER REQUESTS
3524 014524 062706 000004 ADD #4,SP ; . ADJUST STACK POINTER
3525 014530 104032 ERROR +32 ; . BG5 GRANTED WITH CPU AT 5-7
3526 014532 162703 000040 8$: SUB #40,R3 ; . LOWER PRIORITY
3527 014536 020327 000240 CMP R3,#240 ; . LAST ONE TO CHECK?
3528 014542 002357 BGE 6$ ; . IF NOT, BRANCH
3529 ;
3530 ; TRY TO DO BR4 WITH CPU PRIORITY AT 7-4
3531 ;
3532 014544 012777 014574 165444 MOV #10,@BE1VEC ; POINT UBE VECTOR TO PROGRAM
3533 014552 012703 000340 MOV #340,R3 ; START WITH PRIORITY AT 7
3534 014556 012777 000003 165424 9$: MOV #3,@BE1CR1 ; . BR6 WITH FUNO

```

T37 NO BUS GRANTS WITH PROCESSOR AT HIGHER PRIORITY

3535	014564	010337	177776		MOV	R3,PSW	:	. CHANGE PRIORITY
3536	014570	000240			NOP		:	. JUST IN CASE OF INTERRUPTS
3537	014572	000405			BR	11‡	:	. IF NO INTERRUPTS, BRANCH
3538	014574	005077	165410	10‡:	CLR	@BE1CR1	:	. CLEAR ANY OTHER REQUESTS
3539	014600	062706	000004		ADD	#4,SP	:	. ADJUST STACK POINTER
3540	014604	104032			ERROR	+32	:	. BG4 GRANTED WITH CPU AT 4-7
3541	014606	162703	000040	11‡:	SUB	#40,R3	:	. LOWER PRIORITY
3542	014612	020327	000200		CMP	R3,#200	:	. LAST ONE TO CHECK?
3543	014616	002357			BGE	9‡	:	. IF NOT, BRANCH
3544								
3545								

TEST - BR7-BR4 ARBITRATION

```

3547      .SBTTL TEST - BR7-BR4 ARBITRATION
3548
3549      ;* THIS TEST CHECKS THAT BR7 BR4 INTERRUPTS CAN OCCUR WITH A PROCESSOR
3550      ;* PRIORITY AT A LOWER LEVEL
3551      ;*
3552      ;
3553      ;   BGNTST
3554      ;
3555      ;       SET UP @BE1PSW TO #340
3556      ;*
3557      ;* TRY TO DO BR7 WITH CPU PRIORITY AT 6 - 0
3558      ;*
3559      ;       DO FOR R3 FROM #300 DOWNT0 #0 BY #40
3560      ;       .   LET PSW = #340
3561      ;       .   LET @BE1CR1 = #21 TO DO 1 DATI
3562      ;       .   LET PSW = R3 TO CHANGE PRIORITY
3563      ;       .   DO "NOP"
3564      ;       .   IF NO INTERRUPT THEN
3565      ;       .       ERROR BG7 DOESN'T OCCUR
3566      ;       .   ENDF
3567      ;       ENDDO
3568      ;*
3569      ;* TRY TO DO BR6 WITH CPU PRIORITY AT 5 - 0
3570      ;*
3571      ;       DO FOR R3 FROM #240 DOWNT0 #0 BY #40
3572      ;       .   LET PSW = #340
3573      ;       .   LET @BE1CR1 = #11 TO DO 1 DATI
3574      ;       .   LET PSW = R3 TO CHANGE PRIORITY
3575      ;       .   DO "NOP"
3576      ;       .   IF NO INTERUPT THEN
3577      ;       .       ERROR BG6 DOESN'T OCCUR
3578      ;       .   ENDF
3579      ;       ENDDO
3580      ;*
3581      ;* TRY TO DO BR5 WITH CPU PRIORITY AT 4 - 0
3582      ;*
3583      ;       DO FOR R3 FROM #200 DOWNT0 #0 BY #40
3584      ;       .   LET PSW = #340
3585      ;       .   LET @BE1CR1 = #5 TO DO 1 DATI
3586      ;       .   LET PSW = R3 TO CHANGE PRIORITY
3587      ;       .   DO "NOP"
3588      ;       .   IF NO INTERRUPT THEN
3589      ;       .       ERROR BG5 DOESN'T OCCUR
3590      ;       .   ENDF
3591      ;       ENDDO
3592      ;*
3593      ;* TRY TO DO BR4 WITH CPU PRIORITY AT 3 - 0
3594      ;*
3595      ;       DO FOR R3 FROM #140 DOWNT0 #0 BY #40
3596      ;       .   LET PSW = #340
3597      ;       .   LET @BE1CR1 = #3 TO DO 1 DATI
3598      ;       .   LET PSW = R3 TO CHANGE PRIORITY
3599      ;       .   DO "NOP"
3600      ;       .   IF NO INTERRUPT THEN
3601      ;       .       ERROR BG4 DOESN'T OCCUR
3602      ;       .   ENDF
3603      ;       ENDDO

```

TEST - BR7-BR4 ARBITRATION

```

3604 ;
3605 ;   ENDTST
3606 ;
3607 ;-----
3608 ;
3609 ;*****
;*TEST 40   BR7-BR4 ARBITRATION
;*****
TST40: SCOPE
      014620  000004
3610
3611 014622  004737  002502          JSR   PC,IUBE          ; INITIALIZE THE UBE
3612 014626  012777  000340  165364  MOV   #340,@BE1PSW      ; AT PRIORITY 7
3613 ;
3614 ; TRY TO DO BR7 WITH CPU PRIORITY AT 6-0
3615 ;
3616 014634  012777  014700  165354  MOV   #2$,@BE1VEC      ; POINT UBE VECTOR TO PROGRAM
3617 014642  012703  000300          MOV   #300,R3          ; START WITH PRIORITY AT 6
3618 014646  012737  000340  177776  1$:  MOV   #340,PSW        ; . RAISE PRIORITY TO 7
3619 014654  012777  000021  165326  MOV   #21,@BE1CR1     ; . BR7 WITH FUNO
3620 014662  010337  177776          MOV   R3,PSW          ; . LOWER PRIORITY
3621 014666  000240          NOP                    ; . JUST IN CASE OF INTERRUPTS
3622 014670  005077  165314  CLR   @BE1CR1          ; . CLEAR ANY OTHER REQUESTS
3623 014674  104032  ERROR  +32             ; . BR7 NOT GRANTED WITH CPU AT 6 0
3624 014676  000402          BR    3$              ; . DON'T ADJUST STACK IF NO INTERRUPT
3625 014700  062706  000004  2$:  ADD   #4,SP          ; . ADJUST STACK POINTER
3626 014704  162703  000040  3$:  SUB   #40,R3         ; . LOWER PRIORITY
3627 014710  022703  000000          CMP   #0,R3           ; . LAST ONE TO CHECK?
3628 014714  002354          BGE   1$              ; . IF NOT, BRANCH
3629 ;
3630 ; TRY TO DO BR6 WITH CPU PRIORITY AT 5-0
3631 ;
3632 014716  004737  002502          JSR   PC,IUBE          ; INITIALIZE THE UBE
3633 014722  012777  014766  165266  MOV   #5$,@BE1VEC     ; POINT UBE VECTOR TO PROGRAM
3634 014730  012703  000240          MOV   #240,R3         ; START WITH PRIORITY AT 5
3635 014734  012737  000340  177776  4$:  MOV   #340,PSW        ; . RAISE PRIORITY TO 7
3636 014742  012777  000011  165240  MOV   #11,@BE1CR1     ; . BR6 WITH FUNO
3637 014750  010337  177776          MOV   R3,PSW          ; . LOWER PRIORITY
3638 014754  000240          NOP                    ; . JUST IN CASE OF INTERRUPTS
3639 014756  005077  165226  CLR   @BE1CR1          ; . CLEAR ANY OTHER REQUESTS
3640 014762  104032  ERROR  +32             ; . BR6 NOT GRANTED WITH CPU AT 5-0
3641 014764  000402          BR    6$              ; . DON'T ADJUST STACK
3642 014766  062706  000004  5$:  ADD   #4,SP          ; . ADJUST STACK POINTER
3643 014772  162703  000040  6$:  SUB   #40,R3         ; . LOWER PRIORITY
3644 014776  022703  000000          CMP   #0,R3           ; . LAST ONE TO CHECK?
3645 015002  002354          BGE   4$              ; . IF NOT, BRANCH
3646 ;
3647 ; TRY TO DO BR5 WITH CPU PRIORITY AT 4-0
3648 ;
3649 015004  004737  002502          JSR   PC,IUBE          ; INITIALIZE THE UBE
3650 015010  012777  015054  165200  MOV   #8$,@BE1VEC     ; POINT UBE VECTOR TO PROGRAM
3651 015016  012703  000200          MOV   #200,R3         ; START WITH PRIORITY AT 4
3652 015022  012737  000340  177776  7$:  MOV   #40,PSW        ; . RAISE PRIORITY TO 7
3653 015030  012777  000005  165152  MOV   #5,@BE1CR1     ; . BR5 WITH FUNO
3654 015036  010337  177776          MOV   R3,PSW          ; . LOWER PRIORITY
3655 015042  000240          NOP                    ; . JUST IN CASE OF INTERRUPTS
3656 015044  005077  165140  CLR   @BE1CR1          ; . CLEAR ANY OTHER REQUESTS
3657 015050  104032  ERROR  +32             ; . BR5 NOT GRANTED WITH CPU AT 4-0

```

T40 BR7-BR4 ARBITRATION

```

3658 C15052 000402          BR      9#
3659 015054 062706 000004    8#:    ADD    #4,SP          ; . DON'T TOUCH STACK
3660 015060 162703 000040    9#:    SUB    #40,R3         ; . ADJUST STACK POINTER
3661 015064 022703 000000          CMP    #0,R3          ; . LOWER PRIORITY
3662 015070 002354          BGE    7#              ; . LAST ONE TO CHECK?
3663                                     ; . IF NOT, BRANCH
3664                                     ;
3665                                     ; TRY TO DO BR4 WITH CPU PRIORITY AT 3-0
3666 015072 004737 002502          JSR    PC,IUBE        ; INITIALIZE THE UBE
3667 015076 012777 015142 165112    MOV    #11#,@BE1VEC  ; POINT UBE VECTOR TO PROGRAM
3668 015104 012703 000140          MOV    #140,R3       ; START WITH PRIORITY AT 3
3669 015110 012737 000340 177776 10#:  MOV    #340,PSW     ; . RAISE PRIORITY TO 7
3670 015116 012777 000003 165064    MOV    #3,@BE1CR1   ; . BR4 WITH FUNO
3671 015124 010337 177776          MOV    R3,PSW       ; . LOWER PRIORITY
3672 015130 000240          NOP                  ; . JUST IN CASE OF INTERRUPTS
3673 015132 005077 165052    CLR    @BE1CR1      ; . CLEAR ANY OTHER REQUESTS
3674 015136 104032          ERROR +32           ; . BR4 NOT GRANTED WITH CPU AT 3 0
3675 015140 000402          BR     12#          ; . DON'T ADJUST STACK
3676 015142 062706 000004    11#:  ADD    #4,SP          ; . ADJUST STACK POINTER
3677 015146 162703 000040    12#:  SUB    #40,R3         ; . LOWER PRIORITY
3678 015152 022703 000000          CMP    #0,R3          ; . LAST ONE TO CHECK?
3679 015156 002354          BGE    10#          ; . IF NOT, BRANCH
3680
3681

```


TEST - ARBITRATION BETWEEN INTERRUPTS AND PIRQ'S

```

3683 .SBTTL TEST ARBITRATION BETWEEN INTERRUPTS AND PIRQ'S
3684
3685 ;* THIS TEST CHECKS THAT PIRQ REQUESTS OVERRIDE INTERRUPT REQUESTS
3686 ;* OF THE SAME LEVEL.
3687 ;*
3688 ; SET UP @BE1VEC AND PIRQVEC AND PRIORITY FOR BOTH
3689 ; LET R1 = #20 FOR BG7 FROM UBE
3690 ; LET R2 = #100000 FOR PIRQ7
3691 ; DO FOR R3 FROM #300 DOWNT0 #140 BY #40
3692 ; . LET PSW = #340
3693 ; . LET @BE1CR1 = R1 FOR INTERRUPT LEVEL
3694 ; . LET @BE1CR1 = @BE1CR1 SET.BY #1 TO DO 1 DATI
3695 ; . LET PIRQ = PIRQ SET.BY R2 FOR PIRQ LEVEL
3696 ; . LET PSW = R3 TO CHANGE PRIORITY
3697 ; . WAIT FOR INTERRUPT
3698 ; . IF PIRQ INTERRUPT DIDN'T HAPPEN OR @BE1CC NE 1 THEN
3699 ; . . ERROR IN ARBITRATION BETWEEN PIRQ'S AND INTERRUPTS
3700 ; . ENDF
3701 ; . LET R1 = R1 SHIFT RIGHT 1
3702 ; . LET R2 = R2 SHIFT RIGHT 1
3703 ; ENDDO
3704 ;
3705 ; ENDTST
3706 ;
3707 ;-----
3708 ;
3709 ;*****

```

;TEST 41 ARBITRATION BETWEEN INTERRUPTS AND PIRQ'S
;*****

```

TST41: SCOPE
3710 015160 000004 JSR PC,IUBE ; INITIALIZE THE UBE
3711 015162 004737 002502 MOV #2,@BE1CR1 ; POINT UBE VECTOR TO PROGRAM
3712 015174 012777 015256 165014 MOV #340,@BE1PSW ; AT PRIORITY 7
3713 015202 012737 015266 000240 MOV #3@,PIRQVEC ; POINT PIRQ VECTOR TO PROGRAM
3714 015210 012737 000340 000242 MOV #340,@#242 ; AT PRIORITY 7
3715 015216 012701 000020 MOV #20,R1 ; STORE FOR BR7
3716 015222 012702 100000 MOV #100000,R2 ; STORE FOR PIRQ 7
3717 015226 012703 000300 MOV #300,R3 ; START WITH PRIORITY 6
3718 ;
3719 ; DO ARBITRATION FOR LEVEL 7-4
3720 ;
3721 015232 012737 000340 177776 1$: MOV #340,PSW ; . START WITH PRIORITY 7
3722 015240 010177 164744 MOV R1,@BE1CR1 ; . CHANGE UBE PRIORITY
3723 015244 010237 177772 MOV R2,PIRQ ; . CHANGE PIRQ PRIORITY
3724 015250 010337 177776 MOV R3,PSW ; . LOWER CPU PRIORITY
3725 015254 000001 WAIT ; . WAIT FOR INTERRUPT
3726 015256 062706 000004 2$: ADD #4,SP ; . CLEAN UP STACK
3727 015262 104032 ERROR +32 ; . PIRQ'S DON'T TAKE PRIORITY
3728 015264 000402 BR 4$ ; . DON'T CLEAN UP STACK TWICE
3729 015266 062706 000004 3$: ADD #4,SP ; . CLEAN UP STACK
3730 015272 006001 ROR R1 ; . ADJUST BR FOR UBE
3731 015274 006002 ROR R2 ; . ADJUST PIRQ'S
3732 015276 162703 000040 SUB #40,R3 ; . LOWER FOR CPU PRIORITY
3733 015302 022703 000140 CMP #140,R3 ; . PRIORITY 3?
3734 015306 001351 BNE 1$ ; . BRANCH IF NOT YET
3735 015310 005037 177772 CLR PIRQ ; CLEAR ANY REQUESTS

```

TEST - ARBITRATION BETWEEN INTERRUPTS FROM 2 UBE

3737
3738
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769

```

.SBTTL TEST - ARBITRATION BETWEEN INTERRUPTS FROM 2 UBE
;* THIS TEST CHECKS THAT WHEN TWO INTERRUPTS FROM DIFFERENT UBE'S
;* COME IN AT THE SAME TIME, THE BUS IS GRANTED TO THE REQUEST WITH HIGHER
;* PRIORITY.
;*
;
;   BGNTST
;
;       IF UBECT NE #2 THEN THERE'S NO 2ND UBE
;       . EXIT TEST
;   ENDF
;   SET UP VECTORS AND PSW FOR BOTH UBE'S
;   LET R1 = #20 FOR BR7
;   DO 3 TIMES FOR BG7-BG6, BG6-BG5, BG5-BG4
;       LET PSW = #340 FOR PRIORITY 7
;       LET @BE1CR1 = R1
;       LET R1 = R1 SHIFT RIGHT 1
;       LET @BE2CR1 = R1 TO SET PRIORITY OF 2ND UBE
;       LET @SIMLGO = #1 TO DO SIMULTANEOUS "GO"
;       LET PSW = #140 TO LOWER PRIORITY FOR INTERRUPTS
;       WAIT FOR INTERRUPTS
;       IF INTERRUPTS FROM 1ST HAPPENED AFTER 2ND THEN
;       . ERROR IN BR ARBITRATION
;   ENDF
;   ENDDO
;
;   ENDTST

```

```

;*****
;*TEST 42      ARBITRATION BETWEEN INTERRUPTS FROM 2 UBE
;*****
TST42: SCOPE

```

```

015314 000004
3770
3771 015316 022737 000002 001722
3772 015324 001112
3773 015326 004737 002502
3774
3775
3776
3777 015332 012703 000240
3778 015336 012701 00002J
3779 015342 012777 015512 164646
3780 015350 012777 000340 164642
3781 015356 012777 015532 164652
3782 015364 012777 000340 164646
3783
3784
3785
3786 015372 012737 000340 177776
3787 015400 010177 164604
3788 015404 006001
3789 015406 010177 164616
3790 015412 052777 000001 164570

```

```

;*****
; 2 UBE'S?
; INITIALIZE THE UBE
;
; INITIAL SETUP
;
;   MOV      #240,R3          ; START AT PRIORITY 6
;   MOV      #20,R1          ; UBE AT LEVEL 7
;   MOV      @BE1SV,@BE1VEC  ; POINT UBE1 VECTOR TO PROGRAM
;   MOV      #340,@BE1PSW    ; AT PRIORITY 7
;   MOV      @BE2SV,@BE2VEC  ; POINT UBE2 VECTOR TO PROGRAM
;   MOV      #340,@BE2PSW    ; AT PRIORITY 7
;
; ARBITRATE BETWEEN 2 INTERRUPTS OF DIFFERENT LEVEL
;
;   MOV      #340,PSW        ; . RAISE PRIORITY TO 7
;   MOV      R1,@BE1CR1     ; . HIGHER PRIORITY TO UBE1
;   ROR      R1              ; . GET LOWER PRIORITY
;   MOV      R1,@BE2CR1     ; . LOWER PRIORITY TO UBE2
;   BIS      @BIT00,@BE1CR1 ; . SET GO BIT OF UBE #1

```

T42 ARBITRATION BETWEEN INTERRUPTS FROM 2 UBE

```

3791 015420 052777 000001 164602      BIS      #BIT00,@BE2CR1      ; . SET GO BIT OF UBE #2
3792 015426 010337 177776              MOV      R3,PSW          ; . LOWER CPU PRIORITY
3793 015432 000240              NOP                      ; . GIVE UBE TIME TO INTERRUPT
3794 015434 000240              NOP                      ;
3795 015436 000240              NOP                      ;
3796 015440 005737 001724      TST      BE1INT          ; . DID UBE1 INTERRUPT
3797 015444 001002              BNE      5$              ; . YES, THEN GO MAKE SURE UBE2 DIDN'T
3798 015446 104032              ERROR   +32             ; . LOWER ORDER INTERRUPTS HAPPNE BEFORE
3799 015450 000404              BR       10$            ; . GO SEE IF WE ARE DONE
3800 015452 005737 001726      5$:    TST      BE2INT          ; . DID UBE2 INTERRUPT
3801 015456 001401              BEQ      10$            ; . NO, THEN GO SEE IF WE ARE DONE
3802 015460 104032              ERROR   +32             ; . YES, THEN ERROR IN ARBITRATION
3803 015462 005037 001724      10$:   CLR      BE1INT          ; . INITIALIZE INTERRUPT FLAGS
3804 015466 005037 001726      CLR      BE2INT          ;
3805 015472 162703 000040      SUB      #40,R3          ; . DO THE NEXT LEVEL
3806 015476 022703 000100      CMP      #100,R3        ; . CPU AT 2 (LAST ONE)?
3807 015502 001333              BNE      1$              ; . BRANCH IF NOT YET
3808 015504 005077 164522      CLR      @BE2CLR         ; CLEAR ERRORS ON 2ND UBE
3809 015510 000420              BR       TST43          ;
3810                                     ;;      GO TO NEXT TEST
3811 015512 012777 000000 164510  BE1SV:  MOV      #0,@BE2CR1      ; CLEAR PENDING UBE #2 INTERRUPTS
3812 015520 005237 001724              INC      BE1INT          ; SET BE1 INTERRUPT FLAG
3813 015524 005037 001726              CLR      BE2INT          ; CLEAR BE2 INTERRUPT FLAG
3814 015530 000002              RTI
3815
3816 015532 012777 000000 164450  BE2SV:  MOV      #0,@BE1CR1      ; CLEAR PENDING UBE #1 INTERRUPTS
3817 015540 005237 001726              INC      BE2INT          ; SET BE2 INTERRUPT FLAG
3818 015544 005037 001724              CLR      BE1INT          ; CLEAR BE1 INTERRUPT FLAG
3819 015550 000002              RTI
3820

```

TEST - POWER DOWN TEST

3822
3823
3824
3825
3826
3827
3828
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848

```

.SBTTL TEST - POWER DOWN TEST
;* THIS TEST INSURES THAT POWER DOWN CYCLE CAN BE INVOKED FROM THE
;* UNIBUS SIDE OF UBA. IT WILL RUN ONLY IF POWER UP OPTION 11
;* IS SELECTED (TRAP THRU 24/26) IN THE EAROM ON CPU BOARD.
;*
;
;   BGNTST
;
;       LET BCSR<6,5>=<0,1> TO ACCESS EAROM
;       IF @#165002<7,6> NE <1,1> FOR POWER UP CODE 00 THEN
;           . EXIT TEST
;       ENDIF
;       SAVE PWRVEC
;       POINT PWRVEC TO PROGRAM AREA
;       LET @BE1CR2<4> = #1
;       IF NO TRAP TO 24 THEN
;           . ERROR EXECUTING POWER DOWN THRU UNIBUS
;       ENDIF
;       LET @BE1CR2<4> = #0 FOR POWER UP
;       RESTORE PWRVEC
;
;   ENDTST

```

```

;*****
;*TEST 43      POWER DOWN TEST
;*****
TST43: SCOPE

```

015552 000004

3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875

```

; RESTRICTIONS ON RUNNING THIS TEST
;
;   TST      $PASS      ; FIRST PASS
;   BNE      TST44      ;; IF NOT 1ST PASS, DON'T DO IT
;   BIT      @BIT05,@#52 ; UFD MODE?
;   BNE      TST44      ;; EXIT TEST, IF SO
;   CMPB    @APTENV,@ENV ; APT?
;   BEQ      TST44      ;; EXIT TEST IF APT
;   BIT      @BIT03,@#177524 ; FORCED CONSOLE MODE?
;   BNE      TST44      ;; IF YES, EXIT TEST
;   BIC      @BIT06,BCSR ; ENABLE ACCESS THRU 165000
;   BIS      @BIT05,BCSR ; READ EAROM
;   BIT      @BIT07,@#165002 ; POWER UP CODE 00?
;   BEQ      TST44      ;; EXIT TEST, IF NOT
;   BIT      @BIT06,@#165002 ; POWER UP CODE 00?
;   BEQ      TST44      ;; EXIT TEST, IF NOT
;   JSR     PC,IUBE     ; INITIALIZE THE UBE
;
; DO POWER DOWN SEQUENCE
;
;   MOV     PWRVEC,@TMP0 ; STORE POWER DOWN VECTOR
;   MOV     #1,PWRVEC    ; POINT NEW ONE TO PROGRAM
;   MOV     PWRVEC+2,@#26 ; AT PRIORITY 7
;   BIS     @BIT04,@BE1CR2 ; START POWER DOWN
;   NOP
;   NOP                ; WAIT WHAT HAPPENED

```

015554 005737 001206
015560 001076
015562 032737 000040 000052
015570 001072
015572 122737 000001 001220
015600 001466
015602 032737 000010 177524
015610 001062
015612 042737 000100 177520
015620 052737 000040 177520
015626 032737 000200 165002
015634 001450
015636 032737 000100 165002
015644 001444
015646 C04737 002502

TEST - NO SACK TIMEOUT

```

3928 .SBTTL TEST - NO SACK TIMEOUT
3929
3930 ;* THIS TEST INSURES THAT THE CPU TIMES OUT AND DROPS A GRANT IF NO
3931 ;* SACK SIGNAL IS RECEIVED. IF THE CPU DOES NOT TIMEOUT, THE UBE
3932 ;* WILL TIME OUT AND SEND SACK TO PREVENT THE BUS FROM HANGING AND WILL SET
3933 ;* THE ERROR BIT IN CR2.
3934 ;*
3935 ;
3936 ; BGNTST
3937 ;
3938 ; LET @BE1CC = -1 TO DO 1 TRANSFER
3939 ; LET @BE1CR2 = @BE1CR2 SET.BY #BIT03 TO INHIBIT SACK
3940 ; LET @BE1CR1 = #6003 TO DO FUN 3
3941 ; WAIT FOR TIMEOUT OR @BE1CC NE -1
3942 ; IF NO TIMEOUT AND @BE1CR2 SET.BY #BIT07 THEN
3943 ; . ERROR CPU FAILED TO DO NO SACK TIMEOUT
3944 ; ENDIF
3945 ;
3946 ; ENDTST
3947 ;
3948 ;
3949 ;
3950 ;-----

```

```

;*****
;*TEST 45 NO SACK TIMEOUT
;*****
TST45: SCOPE

```

```

016050 000004
3951
3952 016052 004737 002502 JSR PC,IUBE ; INITIALIZE THE UBE
3953 016056 012777 000010 164130 MOV #BIT03,@BE1CR2 ; INHIBIT SACK BIT
3954 016064 005037 177776 CLR PSW ; LOWER PRIORITY
3955 016070 012777 006003 164112 MOV #6003,@BE1CR1 ; GIVE UP BUS AFTER BECOMING MASTER
3956 016076 012701 007777 MOV #7777,R1 ; DELAY CONSTANT
3957 016102 077101 1$: SOB R1,1$ ; WAIT IN A LOOP
3958 016104 105777 164104 TSTB @BE1CR2 ; NO NO SACK TIMEOUT?
3959 016110 100001 BPL 2$ ; IF NO, BRANCH
3960 016112 104032 ERROR +32 ; NO NO SACK TIMEOUT
3961 016114 012777 000000 164072 2$: MOV #0,@BE1CR2 ; CLEAR INHIBIT SACK BIT
3962 016122 012737 000340 177776 MOV #340,PSW ; RESTORE PRIORITY

```

TEST - NO INTERRUPT TEST

```

3964 .SBTTL TEST - NO INTERRUPT TEST
3965
3966 ; * THIS TEST CHECKS THAT NO INTERRUPT CONDITION DOES NOT HANG THE BUS.
3967 ; *
3968 ;
3969 ; BGNTST
3970 ;
3971 ; PROGRAM UBE TO DO DMA ON BR7
3972 ; IF CYCLE NOT DONE OR INTERRUPT THEN
3973 ; . ERROR IN NO INTERRUPT LOGIC
3974 ; ENDF
3975 ;
3976 ; ENDTST
3977 ;
3978 ;-----
3979 ;
3980 ;*****
; *TEST 46 PASSIVE RELEASE
;*****
TST46: SCOPE
016130 000004
3981
3982 016132 004737 002502 JSR PC,JUBE ; INITIALISE UBE
3983 016136 012777 177777 164040 MOV #-1,@BE1CC ; DO 1 CYCLE
3984 016144 012777 001160 164034 MOV #$TMPO,@BE1BA ; AT ADDRESS $TMPO
3985 016152 012777 016224 164036 MOV #10,@BE1VEC ; POINT VECTOR
3986 016160 012737 000003 177776 MOV #3,PSW ; LOWER PRIORITY TO 3
3987 016166 012777 002021 164014 MOV #2021,@BE1CR1 ; BR7, DATI
3988 016174 105777 164010 1$: TSTB @BE1CR1 ; DONE?
3989 016200 100375 BPL 1$ ; IF NOT, WAIT
3990 016202 012737 000340 177776 MOV #340,PSW ; RESTORE PRIORITY
3991 016210 023777 001160 163764 CMP $TMPO,@BE1DB ; DATA OK?
3992 016216 001404 BEQ TST47 ;; IF OK, EXIT TEST
3993 016220 104032 ERROR +32 ; DATI ON BR7 IS WRONG
3994 016222 000402 BR TST47 ;; GO TO NEXT TEST
3995
3996 016224 104032 10$: ERROR +32 ; NO INTERRUPT LOGIC IS WRONG
3997 016226 000002 RTI
3998

```


TEST - UNIBUS DEVICE DATO CYCLE

```

4000      .SBTTL TEST - UNIBUS DEVICE DATO CYCLE
4001
4002      ;* THE FOLLOWING TEST WILL SEE IF A UNIBUS DEVICE DATA OUT
4003      ;* CAN OCCUR ON THE KTJ11-B MODULE.THE UNIBUS DATA PATH
4004      ;* ON THE BOARD IS ALSO TESTED OUT.
4005      ;*
4006      ;
4007      ;   BGNTST
4008      ;
4009      ;       DISABLE UNIBUS MAPPING
4010      ;       LET RO := POINTER TO WRITE BUFFER
4011      ;       LET R1 := POINTER TO PATTERN TABLE
4012      ;*
4013      ;*   GO DO THE TRANSFERS
4014      ;*
4015      ;       DO FOR PATTERNS 377,7417,31463,52525,125252
4016      ;       . LET BE1DB := (R1)+           ;DATA IS CURRENT PATTERN
4017      ;       . LET BE1BA := (RO)+         ;ADDRESS IS CURRENT ADDRESS OF WRITE BUFFER
4018      ;       . LET BE1CC := -1           ;SET UBE FOR 1 DATA XFER
4019      ;       . SET UBE FOR NPR-DATO-1 XFER
4020      ;       . SET OFF THE TRANSFER
4021      ;       ENDDO
4022      ;       LET RO := POINTER TO WRITE BUFFER
4023      ;       LET R1 := POINTER TO PATTERN TABLE
4024      ;*
4025      ;*   CHECK IF THE TRANSFERS WERE CORRECT
4026      ;*
4027      ;       DO UNTIL TABLE IS COMPLETE
4028      ;       . IF (RO)+ NEQ (R1)+ THEN
4029      ;       .   ERROR DATO DID NOT OCLUR CORRECTLY
4030      ;       .   ENDIF
4031      ;       ENDDO
4032      ;
4033      ;   ENDTST
4034      ;
4035      ;-----
4036      ;*****
4037      ;*TEST 47      UNIBUS DEVICE DATO CYCLE TEST
4038      ;*****
4039      ;*****
4040      016230  000004      TST47:  SCOPE
4041
4042      016232  004737  002502      JSR      PC,IUBE      ; INITIALIZE THE UBE
4043      ;
4044      ;   INITIALIZE POINTERS FOR THE TRANSFERS
4045      ;
4046      016236  042737  000040  172516      BIC      #BIT5,MMR3    ; MAKE SURE UNIBUS MAPPING IS DISABLED
4047      016244  012700  001734      MOV      #WRTBUF,RO    ; POINTER TO WRITE BUFFER
4048      016250  012701  001774      MOV      #PTRN16,R1    ; POINTER TO PATTERN TABLE
4049      ;
4050      ;   EXECUTE THE LOOP TO DO THE TRANSFERS
4051      ;
4052      ;
4053      016254  012737  000002  177730      MOV      #BIT01,DCSR   ; SELECT UNIBUS LINES

```

T47 UNIBUS DEVICE DATO CYCLE TEST

```

4054 016262 012704 000005          MOV    #5,R4          ; SET UP LOOP COUNT
4055 016266 012177 163710          1$:   MOV    (R1)+,@BE1DB ; . GET CURRENT DATA
4056 016272 010077 163710          MOV    R0,@BE1BA     ; . GET ADDRESS IN THE WRITE BUFFER
4057 016276 005077 163712          CLR    @BE1CR2       ; . CLEAR ADDRESS BITS 16,17
4058 016302 012777 177777 163674   MOV    #1,@BE1CC     ; . SET UBE FOR 1 DATA TRANSFER
4059 016310 012777 003041 163672   MOV    #3041,@BE1CR1 ; . SET UBE FOR NPR DATO-1 XFER
4060
4061                               ;
4062                               ; WAIT FOR THE TRANSFER TO BE COMPLETE
4063                               ;
4064
4065 016316 032777 000200 163664   5$:   BIT    @BIT7,@BE1CR1 ; . . IS THE TRANSFER COMPLETE ?
4066 016324 001774                   BEQ    5$             ; . . NO, THEN GO WAIT FOR IT
4067 016326 005737 177732          TST    DDR           ; . . UNIBUS LINES 0?
4068 016332 001415                   BEQ    7$             ; . . IF SO, BRANCH
4069 016334 022704 000003          CMP    #3,R4         ; . . SELECTING CONTROL LINES
4070 016340 001004                   BNE    6$             ; . . IF NOT, BRANCH
4071 016342 032737 000373 177732   BIT    #373,DDR      ; . . ONLY USED LINES 0'S?
4072 016350 001406                   BEQ    7$             ; . . IF SO, BRANCH
4073 016352 013737 177732 001126   6$:   MOV    DDR,$BDDAT   ; . . RECIEVED DATA
4074 016360 005037 001124          CLR    $GDDAT        ; . . EXPECTED DATA
4075 016364 104016                   ERROR  +16           ; . . ERROR IN UNIBUS LINES
4076 016366 062737 000002 177730   7$:   ADD    @BIT01,DCSR   ; . . THRU ALL COMBINATIONS
4077 016374 032737 000006 177730   BIT    #6,DCSR       ; . . ALL DONE?
4078 016402 001003                   BNE    8$             ; . . IF NOT, BRANCH
4079 016404 012737 000002 177730   MOV    @BIT01,DCSR   ; . . OTHERWISE, START OVER
4080 016412 062700 000002          8$:   ADD    #2,R0        ; . . GET THE NEXT ADDRESS TO TRANSFER TO
4081 016416 077455                   SOB    R4,1$         ; . . HAVE WE GONE THROUGH THE TABLE ?
4082
4083                               ;
4084                               ; CHECK IF THE TRANSFERS WERE COMPLETED CORRECTLY
4085                               ;
4086
4087 016420 012700 001734          MOV    #WRTBUF,R0    ; POINTER TO WRITE BUFFER
4088 016424 012701 001774          MOV    #PTRN16,R1   ; POINTER TO PATTERN TABLE
4089 016430 012704 000005          MOV    #5,R4         ; SET UP LOOP COUNT
4090 016434 022021          10$:  CMP    (R0)+,(R1)+   ; . WAS THE TRANSFER CORRECT ?
4091 016436 001401          BEQ    15$           ; . YES, THEN SKIP THE ERROR MESSAGE
4092 016440 104032          ERROR  +32          ; . NO, THEN ERROR IN THE TRANSFER
4093 016442 077404          15$:  SOB    R4,10$     ; . HAVE WE CHECKED ALL 5 XFERS ?
4094
4095
4096

```

TEST - UNIBUS DEVICE DATI CYCLE

```

4098 .SBTTL TEST - UNIBUS DEVICE DATI CYCLE
4099
4100 ;* THIS TEST WILL SEE IF THE KTJ11-B MODULE CAN EXECUTE A
4101 ;* UNIBUS DEVICE DATI CYCLE.THE UNIBUS ADDRESS PATH ON THE
4102 ;* MODULE IS ALSO TESTED OUT.
4103 ;*
4104 ;
4105 ; BGNST
4106 ;
4107 ;*
4108 ;* INITIALIZE THE POINTERS FOR THE TRANSFERS
4109 ;*
4110 ; DISABLE UNIBUS MAPPING
4111 ; LET R0 := POINTER TO 16 BIT PATTERNS
4112 ; LET R1 := (R0) ;CURRENT PATTERN
4113 ;*
4114 ;* GO DO THE TRANSFER THEN CHECK IF IT OCCURED CORRECTLY
4115 ;*
4116 ; LET BE1DB := 0 ;CLEAR DATA BUFFER
4117 ; LET BE1BA := R0 ;READ FROM DATA PATTERN
4118 ; LET BE1CC := -1 ;1 DATA XFER
4119 ; SETUP UBE TO DO NPR-DATI-1 DATA XFER
4120 ; SET OFF THE TRANSFER
4121 ; IF BE1DB NEQ #377 THEN
4122 ; . ERROR DATA READ IN WAS INCORRECT
4123 ; ENDF
4124 ;
4125 ; ENDTST
4126 ;
4127 ;-----
4128 ;*****
4129 ;*TEST 50 UNIBUS DEVICE DATI CYCLE TEST
;*****

```

```

016444 000004
4130
4131 016446 004737 002502 JSR PC,IUBE ; INITIALIZE THE UBE
4132 016452 042737 000040 172516 BIC #BITS,MMR3 ; DISABLE UNIBUS MAPPING
4133
4134 ;
4135 ; INITIALIZE POINTERS FOR THE TRANSFERS
4136 ;
4137
4138 016460 012700 001774 MOV #PTRN16,R0 ; GET POINTER TO PATTERN TABLE
4139 016464 011001 MOV (R0),R1 ; GET CURRENT PATTERN
4140
4141 ;
4142 ; GO DO THE TRANSFER
4143 ;
4144
4145 016466 005077 163510 5$: CLR @BE1DB ; . INITIALIZE THE DATA BUFFER
4146 016472 010077 163510 MOV R0,@BE1BA ; . GET ADDRESS WE WANT TO READ
4147 016476 005077 163512 CLR @BE1CR2 ; . CLEAR ADDRESS BITS 16,17
4148 016502 012777 177777 163474 MOV #1,@BE1CC ; . SET UBE FOR 1 TRANSFER
4149 016510 012777 002041 163472 MOV #2041,@BE1CR1 ; . SET UBE FOR NPR-DATI-1 XFER
4150
4151 ;

```

T50 UNIBUS DEVICE DATA CYCLE TEST

```

4152 ; WAIT FOR THE TRANSFER TO COMPLETE
4153 ;
4154 ;
4155 016516 032777 000200 163464 10$: BIT @BIT7,@BE1CR1 ; . . IS THE TRANSFER COMPLETE ?
4156 016524 001774 ; BEQ 10$ ; . . WAIT FOR IT TO BE COMPLETE
4157 ;
4158 ;
4159 ; CHECK THE TRANSFER
4160 ;
4161 ;
4162 016526 027710 163450 CMP @BE1DB,(R0) ; . WAS THE TRANSFER CORRECT ?
4163 016532 001401 BEQ 18$ ; . YES, THEN GO TO THE NEXT TEST
4164 016534 104032 ERROR +32 ; . NO, THEN ERROR IN THE TRANSFER
4165 016536 005720 18$: TST (R0)+ ; . DONE WITH PATTERNS?
4166 016540 001352 BNE 5$ ; . LOOP TILL DONE
4167 ;
4168 ;
4169 ;

```

TEST - UNIBUS DEVICE DATO CYCLE WITH RELOCATION ENABLED

```

4171 .SBTTL TEST - UNIBUS DEVICE DATO CYCLE WITH RELOCATION ENABLED
4172
4173 ; * THIS TEST WILL SEE IF THE KTJ11 B MODULE CAN EXECUTE
4174 ; * A UNIBUS DEVICE DATO CYCLE THROUGH THE UNIBUS MAP.
4175 ; *
4176 ;
4177 ; BGNTST
4178 ; IF ALL UNIBUS MEMORY THEN GO TO END OF PASS
4179 ; *
4180 ; * SET UP UBE AND UNIBUS MAP REGISTERS FOR TRANSFER
4181 ; *
4182 ; LET BE1BA := 0 ;POINT TO UMRO
4183 ; LET BE1DB := 125252 ;DATA PATTERN IS 125252
4184 ; LET BE1CC := -8. ;DO 8 XFERS
4185 ; SET UBE TO DO NPR-DATO-2 DATA XFERS
4186 ; LET MAPLOO := ADDRESS OF WRITE BUFFER
4187 ; LET MAPHOO := 0
4188 ; *
4189 ; * GO DO THE TRANSFER
4190 ; *
4191 ; SET OFF TRANSFER
4192 ; WAIT TILL XFER IS DONE
4193 ; *
4194 ; * CHECK IF THE TRANSFER OCCURED CORRECTLY
4195 ; *
4196 ; LET RO := POINTER TO WRITE BUFFER
4197 ; DO FOR R1 := 1 TO 8
4198 ; . IF (PO)+ NEQ #125252 THEN
4199 ; . ERROR DATO DID NOT EXECUTE CORRECTLY
4200 ; . ENDF
4201 ; ENDDO
4202 ;
4203 ; ENDTST
4204 ;
4205 ;
4206 ;
4207 ;

```

```

;*****
;*TEST 51 UNIBUS DEVICE DATO CYCLE WITH RELOCATION ENABLED
;*****
TST51: SCOPE

```

```

016542 000004
4208
4209 016544 013701 177734 MOV KMCR,R1 ; STORE KMCR
4210 016550 042701 177700 BIC #177700,R1 ; LEAVE ONLY BITS <5-0>
4211 016554 022701 000077 CMP #77,R1 ; ALL UNIBUS MEMORY?
4212 016560 001002 BNE 1$ ; IF NOT, BRANCH
4213 016562 000137 023122 JMP UBEM ; IF ALL UNIBUS, SKIP TILL END OF PASS
4214 016566 004737 002502 1$: JSR PC,IUBE ; INITIALIZE THE UBE
4215 016572 012702 001774 MOV #PTRN16,R2 ; TEST PATTERN ADDRESS
4216
4217 ;
4218 ; SET UP UBE AND UNIBUS MAP REGISTERS FOR TRANSFER
4219 ;
4220 ;
4221 016576 005077 163404 2$: CLR @BE1BA ; POINT UBE ADDRESS TO UNIBUS MAP
4222 016602 005077 163406 CLR @BE1CR2 ; REGISTER #0
4223 016606 011277 163370 MOV (R2),@BE1DB ; DATA PATTERN
4224 016612 012777 177770 163364 MOV #-8.,@BE1CC ; DO 8. TRANSFERS

```

T51 UNIBUS DEVICE DATO CYCLE WITH RELOCATION ENABLED

```

4225 016620 012737 001734 170200      MOV    #WRTBUF,MAPLOO ; SET UP LOW MAP REGISTER
4226 016626 005037 170202      CLR    MAPH00        ; SET UP HIGH MAP REGISTER
4227 016632 052737 000040 172516      BIS    #BITS,MMR3     ; ENABLE UNIBUS MAPPING
4228
4229                                    ;
4230                                    ; GO DO THE TRANSFER
4231                                    ;
4232
4233 016640 012777 003041 163342      MOV    #3041,@BE1CR1 ; DO A NPR-DATO-1 XFER
4234 016646 032777 000200 163334 10$: BIT    #BIT7,@BE1CR1 ; IS THE TRANSFER COMPLETE ?
4235 016654 001774                    BEQ    10$            ; WAIT FOR IT TO COMPLETE
4236
4237                                    ;
4238                                    ; CHECK IF THE TRANSFER OCCURED CORRECTLY
4239                                    ;
4240
4241 016656 042737 000040 172516      BIC    #BITS,MMR3     ; DISABLE UNIBUS MAPPING
4242 016664 012700 001734            MOV    #WRTBUF,RO    ; GET POINTER TO WRITE BUFFER
4243 016670 012701 000010            MOV    #8.,R1        ; SET UP LOOP COUNTER
4244 016674 022012                    15$: CMP    (R0)+,(R2)   ; . SEE IF TRANSFER OCCURED CORRECTLY ?
4245 016676 001401                    BEQ    20$            ; . YES, THEN SKIP ERROR MESSAGE
4246 016700 104032                    ERROR +32            ; . NO, THEN ERROR IN TRANSFER
4247 016702 077104                    20$: SOB    R1,15$     ; . HAVE WE CHECKED ALL THE TRANSFERS
4248 016704 005722                    TST    (R2)+        ; GET NEXT PATTERN
4249 016706 001333                    BNE    2$            ; LOOP TILL DONE
4250

```

TEST - UNIBUS DEVICE DATI CYCLE WITH RELOCATION ENABLED

```

4252 .SBTTL TEST - UNIBUS DEVICE DATI CYCLE WITH RELOCATION ENABLED
4253
4254 ;* THIS TEST WILL SEE IF THE KTJ11-B MODULE CAN EXECUTE
4255 ;* A UNIBUS DEVICE DATI CYCLE THROUGH THE UNIBUS MAP WITH
4256 ;* CACHE DISABLED.
4257 ;*
4258 ;
4259 ; BGNTST
4260 ;
4261 ;*
4262 ;* SET UP THE UBE FOR A DATI CYCLE
4263 ;*
4264 ; LET BE1BA := 0 ;POINT ADDRESS TO MAPX00
4265 ; LET BE1CR2 := 0 ;
4266 ; LET BE1DB := 0 ;INITIALIZE DATA BUFFER
4267 ; LET BE1CC := -1 ;SET FOR 1 XFER
4268 ; SET UBE FOR A NPR-DATI-1 XFER
4269 ;*
4270 ;* SET UP THE MAP REGISTER FOR THE TRANSFER
4271 ;*
4272 ; LET MAPL00 := R0 ;POINT UMR #0 TO DATA PATTERN
4273 ; LET MAPH00 := 0
4274 ; ENABLE UNIBUS MAPPING
4275 ;*
4276 ;* GO DO THE TRANSFER
4277 ;*
4278 ; SET OFF THE TRANSFER
4279 ; WAIT FOR TRANSFER TO COMPLETE
4280 ; DISABLE UNIBUS MAPPING
4281 ;*
4282 ;* CHECK IF THE TRANSFER OCCURED CORRECTLY
4283 ;*
4284 ; IF BE1DB NEQ (R0)+ THEN
4285 ; . ERROR DATI THROUGH UNIBUS MAP DID NOT EXECUTE CORRECTLY
4286 ; ENDIF
4287 ;
4288 ; ENDTST
4289 ;
4290 ;-----
4291 ;
4292 ;*****
4293 ;*TEST 52 UNIBUS DEVICE DATI CYCLE WITH RELOCATION ENABLED
;*****
TST52: SCOPE

```

```

016710 000004
4294
4295 016712 004737 002502 JSR PC,IUBE ; INITIALIZE THE UBE
4296
4297 ;
4298 ; SET UP UBE AND UNIBUS MAP REGISTERS FOR A DATI CYCLE
4299 ;
4300
4301 016716 012700 001774 MOV #PTRN16,R0 ; PATTERN AREA
4302 016722 005077 163260 1$: CLR @BE1BA ; POINT UBE TO THE FIRST UNIBUS MAP
4303 016726 005077 163262 CLR @BE1CR2 ; REGISTER
4304 016732 005077 163244 CLR @BE1DB ; INITIALIZE THE DATA BUFFER
4305 016736 012777 177777 163240 MOV #-1,@BE1CC ; SET FOR 1 TRANSFER

```

T52 UNIBUS DEVICE DATI CYCLE WITH RELOCATION ENABLED

```

4306 016744 010037 170200          MOV    R0,MAPL00      ; POINT MAP REGISTERS TO PATTERN TABLE
4307 016750 005037 170202          CLR    MAPH00        ;
4308 016754 052737 000040 172516    BIS    @BITS,MMR3    ; ENABLE UNIBUS MAPPING
4309
4310                                ;
4311                                ; GO DO THE TRANSFER
4312                                ;
4313
4314 016762 012777 002041 163220    MOV    @2041,@@BE1CR1 ; SET UBE FOR NPR-DATI-1 XFER
4315 016770 032777 000200 163212    S# :   BIT    @BIT7,@BE1CR1 ; IS THE TRANSFER COMPLETE
4316 016776 001774          BEQ    S#           ; NO, THEN WAIT FOR IT TO BE COMPLETE
4317 017000 042737 000040 172516    BIC    @BITS,@MMR3   ; YES, THEN DISABLE UNIBUS MAPPING
4318
4319                                ;
4320                                ; CHECK THE TRANSFER
4321                                ;
4322
4323 017006 027710 163170          CMP    @@BE1DB,(R0)  ; DID THE TRANSFER HAPPEN CORRECTLY ?
4324 017012 001401          BEQ    10#         ; YES
4325 017014 104032          ERROR +32        ; TRANSFER DID NOT OCCUR CORRECTLY
4326 017016 005720          TST   (R0)+        ; ALL DONE?
4327 017020 001340          BNE   1#          ; LOOP TILL DONE
4328
4329
4330

```


TEST - ALU TEST USING UBE

```

4332 .SBTTL TEST - ALU TEST USING UBE
4333
4334 ;* THIS TEST WILL CHECK THE FIRST 3 ALU'S EACH ONE AT A TIME. AT FIRST THE
4335 ;* PATTERN THAT DOES NOT RESULT IN OVERFLOW WILL BE USED, THE SECOND PATTERN
4336 ;* GENERATES OVERFLOW INTO THE NEXT ALU.
4337 ;*
4338 ;
4339 ; BGNTST
4340 ;
4341 ; IN 18-BIT MODE DON'T RUN THIS TEST
4342 ; LET R1 := 1001 (PATTERN WITHOUT OVERFLOW FROM ALU)
4343 ; LET R2 := 0101 (MAP REGISTER PATTERN)
4344 ; LET R3 := #MAPLOO
4345 ; LET R4 := 0110 (OVERFLOW)
4346 ; LET R5 := 1011
4347 ; DO FOR 3 PATTERNS
4348 ; . LET BE1BA := R1 ;POINT ADDRESS TO MAPX00
4349 ; . LET BE1CR2 := 0 ;
4350 ; . LET BE1DB := 0 ;INITIALIZE DATA BUFFER
4351 ; . LET BE1CC := -1 ;SET FOR 1 XFER
4352 ; . SET UBE FOR A NPR-DATI-1 XFER
4353 ;*
4354 ;* SET UP THE MAP REGISTER FOR THE TRANSFER WITHOUT OVERFLOW FROM ALU
4355 ;*
4356 ; . LET (R3) := R2 ;POINT UMR #0 TO DATA PATTERN
4357 ; . LET 2(R3) := 0
4358 ; . ENABLE UNIBUS MAPPING
4359 ;*
4360 ;* GO DO THE TRANSFER
4361 ;*
4362 ; . SET OFF THE TRANSFER
4363 ; . WAIT FOR TRANSFER TO COMPLETE
4364 ; . DISABLE UNIBUS MAPPING
4365 ;*
4366 ;* CHECK IF THE TRANSFER OCCURED CORRECTLY
4367 ;*
4368 ; . IF BE1DB NEQ (R1+R2) THEN
4369 ; . . ERROR DATI THROUGH UNIBUS MAP DID NOT EXECUTE CORRECTLY
4370 ; . ENDF
4371 ;*
4372 ;* NOW CHECK CARRY OUT FROM ALU
4373 ;*
4374 ; . LET BE1BA := R4 ;POINT ADDRESS TO MAPX00
4375 ; . LET BE1CR2 := 0 ;
4376 ; . LET BE1DB := 0 ;INITIALIZE DATA BUFFER
4377 ; . LET BE1CC := -1 ;SET FOR 1 XFER
4378 ; . SET UBE FOR A NPR-DATI-1 XFER
4379 ;*
4380 ;* SET UP THE MAP REGISTER FOR THE TRANSFER
4381 ;*
4382 ; . LET (R3) := R5 ;POINT UMR #0 TO DATA PATTERN
4383 ; . LET 2(R3) := 0
4384 ; . ENABLE UNIBUS MAPPING
4385 ;*
4386 ;* GO DO THE TRANSFER
4387 ;*
4388 ; . SET OFF THE TRANSFER

```

TEST - ALU TEST USING UBE

```

4389 ; . WAIT FOR TRANSFER TO COMPLETE
4390 ; . DISABLE UNIBUS MAPPING
4391 ;*
4392 ;* CHECK IF THE TRANSFER OCCURED CORRECTLY
4393 ;*
4394 ; . IF BE1DB NEQ (R4,R5) THEN
4395 ; . ERROR DATI THROUGH UNIBUS MAP DID NOT EXECUTE CORRECTLY
4396 ; . ENDIF
4397 ;*
4398 ;* CHANGE THE PATTERN
4399 ;*
4400 ; . SHIFT R1,R2,R4,R5 4 TIMES AND POINT TO MAPLO3 AND MAPLO6
4401 ; . POINT R3 TO MAPLO3 AND THEN TO MAPLO6
4402 ; . ENDDO
4403 ;
4404 ; ENDTST
4405 ;
4406 ;
4407 ;
4408 ;
4409 ;

```

```

;*****
;*TEST 53 ALU TEST USING UBE
;*****
TST53: SCOPE

```

```

017022 000004
4410
4411 017024 032737 000040 177734 BIT #BIT05,KMCR ; 18 BIT MODE?
4412 017032 001402 BEQ 10$ ; IF NOT, CONTINUE
4413 017034 000137 017452 JMP 20$ ; OTHERWISE, EXIT
4414 017040 005037 172354 10$: CLR KIPAR6 ; FOR ERROR REPORTING
4415 017044 004737 002502 JSR PC,IUBE ; INITIALIZE THE UBE
4416 017050 012700 000022 MOV #22,R0 ; INITIAL PATTERN FOR ADDRESS LINES
4417 017054 012701 000012 MOV #12,R1 ; INITIAL PATTERN FOR MAP REGISTER
4418 017060 012703 170200 MOV #MAPLO0,R3 ; FIRST REGISTER PAIR TO BE USED
4419 017064 012704 000014 MOV #14,R4 ; PATTERN FOR THE OVERFLOW FOR A. LINES
4420 017070 012705 000026 MOV #26,R5 ; PATTERN FOR MAP PAIR
4421 017074 005037 001162 CLR $TMP1 ; LOCATION TO SELECT REGISTER PAIR
4422 ;
4423 ; SET UP UBE AND UNIBUS MAP REGISTERS FOR A DATI CYCLE
4424 ;
4425
4426 017100 010077 163102 1$: MOV R0,@BE1BA ; POINT UBE TO THE PATTERN
4427 017104 005077 163104 CLR @BE1CR2 ; REGISTER
4428 017110 005077 163066 CLR @BE1DB ; INITIALIZE THE DATA BUFFER
4429 017114 012777 177777 163062 MOV #-1,@BE1CC ; SET FOR 1 TRANSFER
4430 017122 010113 MOV R1,(R3) ; POINT MAP REGISTERS TO PATTERN
4431 017124 005063 000002 CLR 2(R3) ; CLEAR HIGH MAP REGISTER
4432 017130 052737 000040 172516 BIS #BIT5,MMR3 ; ENABLE UNIBUS MAPPING
4433 ;
4434 ;
4435 ; GO DO THE TRANSFER
4436 ;
4437 ;
4438 017136 012777 002041 163044 MOV #2041,@BE1CR1 ; SET UBE FOR NPR-DATI-1 XFER
4439 017144 032777 000200 163036 2$: BIT #BIT7,@BE1CR1 ; IS THE TRANSFER COMPLETE
4440 017152 001774 BEQ 2$ ; NO, THEN WAIT FOR IT TO BE COMPLETE
4441 017154 042737 000040 172516 BIC #BIT5,@MMR3 ; YES, THEN DISABLE UNIBUS MAPPING
4442

```

T53 ALU TEST USING UBE

```

4443
4444 ; CHECK THE TRANSFER
4445 ;
4446
4447 017162 010037 001160          MOV    R0,$TMP0      ; FIND ADDRESS ACCESSED
4448 017166 042737 160000 001160  BIC    #160000,$TMP0 ; MAKE SURE THAT USING ALL 16 BITS FOR ADDRESS
4449 017174 060137 001160          ADD    R1,$TMP0      ;
4450 017200 027777 162776 161752  CMP    @BE1DB,@$TMP0 ; DID THE TRANSFER HAPPEN CORRECTLY ?
4451 017206 001412                BEQ    3$            ; IF SO, BRANCH
4452 017210 017737 162766 001126  MOV    @BE1DB,$BDDAT ; WRONG DATA
4453 017216 017737 161736 001124  MOV    @$TMP0,$GDDAT ; EXPECTED DATA
4454 017224 013737 001160 001122  MOV    $TMP0,$BDADR  ; ADDRESS USED
4455 017232 104023                ERROR  +23          ; TRANSFER DID NOT OCCUR CORRECTLY
4456
4457 ; NOW CHECK THE SAME ALU FOR OVERFLOW
4458 ;
4459 017234 010477 162746          3$:   UV    R4,@BE1BA    ; POINT UBE TO THE PATTERN
4460 017240 005077 162750          CLR    @BE1CR2      ; REGISTER
4461 017244 005077 162732          CLR    @BE1DB       ; INITIALIZE THE DATA BUFFER
4462 017250 012777 177777 162726  MOV    #-1,@BE1CC   ; SET FOR 1 TRANSFER
4463 017256 010513                MOV    R5,(R3)      ; POINT MAP REGISTERS TO PATTERN
4464 017260 005063 000002          CLR    2(R3)        ; CLEAR HIGH MAP REGISTER
4465 017264 052737 000040 172516  BIS    @BITS,MMR3   ; ENABLE UNIBUS MAPPING
4466
4467 ; GO DO THE TRANSFER
4468 ;
4469 ;
4470
4471 017272 012777 002041 162710  MOV    #2041,@BE1CR1 ; SET UBE FOR NPR-DATI-1 XFER
4472 017300 032777 000200 162702  4$:   BIT    #BIT7,@BE1CR1 ; IS THE TRANSFER COMPLETE
4473 017306 001774                BEQ    4$            ; NO, THEN WAIT FOR IT TO BE COMPLETE
4474 017310 042737 000040 172516  BIC    @BITS,@MMR3  ; YES, THEN DISABLE UNIBUS MAPPING
4475
4476 ; CHECK THE TRANSFER
4477 ;
4478 ;
4479
4480 017316 010437 001160          MOV    R4,$TMP0      ; FIND ADDRESS ACCESSED
4481 017322 042737 160000 001160  BIC    #160000,$TMP0 ; MAKE SURE THAT USING ALL 16 BITS FOR ADDRESS
4482 017330 060537 001160          ADD    R5,$TMP0      ;
4483 017334 027777 162642 161616  CMP    @BE1DB,@$TMP0 ; DID THE TRANSFER HAPPEN CORRECTLY ?
4484 017342 001412                BEQ    5$            ; IF SO, BRANCH
4485 017344 017737 162632 001126  MOV    @BE1DB,$BDDAT ; WRONG DATA
4486 017352 017737 161602 001124  MOV    @$TMP0,$GDDAT ; EXPECTED DATA
4487 017360 013737 001160 001122  MOV    $TMP0,$BDADR  ; ADDRESS USED
4488 017366 104023                ERROR  +23          ; TRANSFER DID NOT OCCUR CORRECTLY
4489
4490 ; CHANGE THE PATTERN
4491 ;
4492 017370 022737 002000 001720  5$:   CMP    #2000,PMIS  ; MORE THAN 32K OF PMI MEMORY?
4493 017376 103025                BHIS   TST54        ; IF NOT, EXIT TEST
4494 017400 072027 000004          ASH    #4,R0         ; TO GET TO NEXT ALU
4495 017404 072127 000004          ASH    #4,R1         ; BY SHIFYING 4 TIMES
4496 017410 072427 000004          ASH    #4,R4         ;
4497 017414 072527 000004          ASH    #4,R5         ;
4498 017420 062703 000014          ADD    #12.,R3      ; GET NEXT REGISTER PAIR
4499 017424 062737 060000 001162  ADD    #60000,$TMP1 ;

```

T53 ALU TEST USING UBE

```

4500 017432 053700 001162            BIS    $TMP1,R0            ; SET TO SELECT NEXT REGISTER PAIR
4501 017436 053704 001162            BIS    $TMP1,R4            ;
4502 017442 022737 020000 001162    CMP    #20000,$TMP1       ; ALL 3 ALU'S DONE?
4503 017450 001213                    BNE    1$                ; IF NOT, BRANCH
4504 017452                            20$:

```

TEST - CARRY PROPOGATION TEST USING UBE

```

4506 .SBTTL TEST - CARRY PROPOGATION TEST USING UBE
4507
4508 ;* THIS TEST VALIDATES THAT CARRY CAN BE PROPOGATED THRU ALL ALU'S CORRECTLY.
4509 ;* THE RESULT IS OBTAINED FROM LOCATION 0.
4510 ;
4511 ; BGNTST
4512 ;
4513 ;*
4514 ;* SET UP THE UBE FOR A DATI CYCLE
4515 ;*
4516 ; LET @#0 := #52525
4517 ; LET BE1BA := #2
4518 ; LET BE1CR2 := 0 ;
4519 ; LET BE1DB := 0 ;INITIALIZE DATA BUFFER
4520 ; LET BE1CC := -1 ;SET FOR 1 XFER
4521 ; SET UBE FOR A NPR-DATI-1 XFER
4522 ;*
4523 ;* SET UP THE MAP REGISTER FOR THE TRANSFER
4524 ;*
4525 ; LET MAPLOO := #17777 ;POINT UMR #0 TO DATA PATTERN
4526 ; LET MAPHOO := #77
4527 ; ENABLE UNIBUS MAPPING
4528 ;*
4529 ;* GO DO THE TRANSFER
4530 ;*
4531 ; SET OFF THE TRANSFER
4532 ; WAIT FOR TRANSFER TO COMPLETE
4533 ; DISABLE UNIBUS MAPPING
4534 ;*
4535 ;* CHECK IF THE TRANSFER OCCURED CORRECTLY
4536 ;*
4537 ; IF BE1DB NEQ @#0 (52525) THEN
4538 ; . ERROR DATI THROUGH UNIBUS MAP DID NOT EXECUTE CORRECTLY
4539 ; . ENDIF
4540 ;
4541 ; ENDTST
4542 ;
4543 ;
4544 ;-----

```

```

;*****
;*TEST 54 CARRY PROPOGATION TEST USING UBE
;*****
TST54: SCOPE

```

```

017452 000004
4545
4546 017454 005037 172354 CLR KIPAR6 ; CLEAR FOR ERROR REPORTS
4547 017460 004737 002502 JSR PC,IUBE ; INITIALIZE THE UBE
4548
4549 ;
4550 ; SET UP UBE AND UNIBUS MAP REGISTERS FOR A DATI CYCLE
4551 ;
4552
4553 017464 012737 052525 000000 MOV #52525,@#0 ; TEST LOCATION
4554 017472 012777 000002 162506 MOV #2,@BE1BA ; POINT UBE TO THE FIRST UNIBUS MAP
4555 017500 005077 162510 CLR @BE1CR2 ; REGISTER
4556 017504 005077 162472 CLR @BE1DB ; INITIALIZE THE DATA BUFFER
4557 017510 012777 177777 162466 MOV #-1,@BE1CC ; SET FOR 1 TRANSFER
4558 017516 012737 177777 170200 MOV #177777,MAPLOO ; POINT MAP REGISTERS TO OVERFLOW
4559 017524 012737 000077 170202 MOV #77,MAPHOO ;

```

T54 CARRY PROPOGATION TEST USING UBE

```

4560 017532 052737 000040 172516      BIS      #BIT5,MMR3      ; ENABLE UNIBUS MAPPING
4561
4562
4563
4564
4565
4566 017540 012777 002041 162442      MOV      #2041,@BE1CR1  ; SET UBE FOR NPR DATI-1 XFER
4567 017546 032777 000200 162434 5$:  BIT      #BIT7,@BE1CR1  ; IS THE TRANSFER COMPLETE
4568 017554 001774
4569 017556 042737 000040 172516      BEQ      5$
4570
4571
4572
4573
4574
4575 017564 027727 162412 052525      CMP      @BE1DB,#52525  ; DID THE TRANSFER HAPPEN CORRECTLY ?
4576 017572 001411
4577 017574 017737 162402 001126      BEQ      TST55          ; ; GO TO THE NEXT TEST
4578 017602 012737 052525 001124      MOV      @BE1DB,$BDDAT  ; WRONG DATA
4579 017610 005037 001122
4580 017614 104023
4581
4582
4583
4584
4585
4586
4587
4588
4589
4590
4591
4592
4593
4594
4595
4596
4597
4598
4599
4600
4601
4602
4603
4604
4605
4606
4607
4608
4609
4610
4611
4612
4613
4614
4615
4616
4617
4618
4619
4620
4621
4622
4623
4624
4625
4626
4627
4628
4629
4630
4631
4632
4633
4634
4635
4636
4637
4638
4639
4640
4641
4642
4643
4644
4645
4646
4647
4648
4649
4650
4651
4652
4653
4654
4655
4656
4657
4658
4659
4660
4661
4662
4663
4664
4665
4666
4667
4668
4669
4670
4671
4672
4673
4674
4675
4676
4677
4678
4679
4680
4681
4682
4683
4684
4685
4686
4687
4688
4689
4690
4691
4692
4693
4694
4695
4696
4697
4698
4699
4700
4701
4702
4703
4704
4705
4706
4707
4708
4709
4710
4711
4712
4713
4714
4715
4716
4717
4718
4719
4720
4721
4722
4723
4724
4725
4726
4727
4728
4729
4730
4731
4732
4733
4734
4735
4736
4737
4738
4739
4740
4741
4742
4743
4744
4745
4746
4747
4748
4749
4750
4751
4752
4753
4754
4755
4756
4757
4758
4759
4760
4761
4762
4763
4764
4765
4766
4767
4768
4769
4770
4771
4772
4773
4774
4775
4776
4777
4778
4779
4780
4781
4782
4783
4784
4785
4786
4787
4788
4789
4790
4791
4792
4793
4794
4795
4796
4797
4798
4799
4800
4801
4802
4803
4804
4805
4806
4807
4808
4809
4810
4811
4812
4813
4814
4815
4816
4817
4818
4819
4820
4821
4822
4823
4824
4825
4826
4827
4828
4829
4830
4831
4832
4833
4834
4835
4836
4837
4838
4839
4840
4841
4842
4843
4844
4845
4846
4847
4848
4849
4850
4851
4852
4853
4854
4855
4856
4857
4858
4859
4860
4861
4862
4863
4864
4865
4866
4867
4868
4869
4870
4871
4872
4873
4874
4875
4876
4877
4878
4879
4880
4881
4882
4883
4884
4885
4886
4887
4888
4889
4890
4891
4892
4893
4894
4895
4896
4897
4898
4899
4900
4901
4902
4903
4904
4905
4906
4907
4908
4909
4910
4911
4912
4913
4914
4915
4916
4917
4918
4919
4920
4921
4922
4923
4924
4925
4926
4927
4928
4929
4930
4931
4932
4933
4934
4935
4936
4937
4938
4939
4940
4941
4942
4943
4944
4945
4946
4947
4948
4949
4950
4951
4952
4953
4954
4955
4956
4957
4958
4959
4960
4961
4962
4963
4964
4965
4966
4967
4968
4969
4970
4971
4972
4973
4974
4975
4976
4977
4978
4979
4980
4981
4982
4983
4984
4985
4986
4987
4988
4989
4990
4991
4992
4993
4994
4995
4996
4997
4998
4999
5000

```

TEST - NXM TEST USING UBE

```

4583 .SBTTL TEST - NXM TEST USING UBE
4584 ;* THIS TEST CHECKS THAT ACCESSING NXM MEMORY CAN BE DONE CORRECTLY THRU
4585 ;* UBE. TEST LOCATION USED IS 17760000.
4586 ;
4587 ; BGNTST
4588 ;
4589 ;*
4590 ;* SET UP THE UBE FOR A DATI CYCLE
4591 ;*
4592 ;       LET BE1BA := 0           ;POINT ADDRESS TO MAPX00
4593 ;       LET BE1CR2 := 0         ;
4594 ;       LET BE1DB := 0         ;INITIALIZE DATA BUFFER
4595 ;       LET BE1CC := -1        ;SET FOR 1 XFER
4596 ;       SET UBE FOR A NPR-DATI-1 XFER
4597 ;*
4598 ;* SET UP THE MAP REGISTER FOR THE TRANSFER
4599 ;*
4600 ;       LET MAPL00 := #160000   ;POINT UMR #0 TO DATA PATTERN
4601 ;       LET MAPH00 := #77
4602 ;       ENABLE UNIBUS MAPPING
4603 ;*
4604 ;* GO DO THE TRANSFER
4605 ;*
4606 ;       SET OFF THE TRANSFER
4607 ;*
4608 ;* CHECK IF THE TRANSFER OCCURED CORRECTLY
4609 ;*
4610 ;       IF #BIT08 NOTSET IN @BE1CR2 THEN
4611 ;           ERROR DATI TO NXM IS WRONG
4612 ;       ENDIF
4613 ;
4614 ; ENDTST
4615 ;
4616 ;-----
4617 ;
4618 ;*****
;*TEST 55      NXM TEST USING UBE WITH RELOCATION ENABLED
;*****
TST55:  SCOPE
4619 017616 000004
017620 004737 002502
4620 JSR      PC,IUBE      ; INITIALIZE THE UBE
4621 ;
4622 ; SET UP UBE AND UNIBUS MAP REGISTERS FOR A DATI CYCLE
4623 ;
4624 017624 012777 017754 162364      MOV      #104,@BE1VEC      ; POINT INTERRUPT VECTOR
4625 017632 012777 000340 162360      MOV      #340,@BE1PSW
4626 017640 005077 162342              CLR      @BE1BA           ; POINT UBE TO THE FIRST UNIBUS MAP
4627 017644 005077 162344              CLR      @BE1CR2         ; REGISTER
4628 017650 005077 162326              CLR      @BE1DB         ; INITIALIZE THE DATA BUFFER
4629 017654 012777 177777 162322      MOV      #-1,@BE1CC      ; SET FOR 1 TRANSFER
4630 017662 012737 160000 170200      MOV      #160000,MAPL00  ; POINT MAP REGISTERS TO NXM 17760000
4631 017670 012737 000077 170202      MOV      #77,MAPH00
4632 017676 052737 000040 172516      BIS      #BIT5,MMR3     ; ENABLE UNIBUS MAPPING
4633 ;
4634 ; GO DO THE TRANSFER
4635 ;
4636 017704 012777 002041 162276      MOV      #2041,@BE1CR1  ; SET UBE FOR NPR-DATI-1 XFER

```

T55 NXM TEST USING UBE WITH RELOCATION ENABLED

```

4637 017712 032777 000200 162270 5$:    BIT    #BIT7,@BE1CR1    ; IS THE TRANSFER COMPLETE
4638 017720 001774                        BEQ    5$                ; NO, THEN WAIT FOR IT TO BE COMPLETE
4639 017722 032777 000400 162264                        BIT    #BIT08,@BE1CR2    ; NXM ?
4640 017730 001001                        BNE    6$                ; IF SO, BRANCH
4641 017732 104022                        ERROR +22                ; NXM NOT SET
4642 017734 106427 000140                        MTPS  #140                ; ALLOW INTERRUPTS
4643 017740 000240                        NOP                        ;
4644 017742 000240                        NOP                        ;
4645 017744 104022                        ERROR +22                ; NXM WITHOUT INTERRUPT
4646 017746 000404                        BR     12$                ;
4647 017750 106427 000340                        MTPS  #340                ; RESTORE PRIORITY
4648                                        ;
4649                                        ; CHECK THE TRANSFER
4650                                        ;
4651 017754 062706 000004                        10$:    ADD    #4,SP                ; ADJUST STACK
4652 017760 042737 000040 172516 12$:    BIC    #BITS,@#MMR3        ; YES, THEN DISABLE UNIBUS MAPPING
4653

```


TEST - RELOCATION WITH UNIBUS MEMORY

```

4655 .SBTTL TEST - RELOCATION WITH UNIBUS MEMORY
4656
4657 ;* THIS TEST WILL CHECK THAT MAPPING REGISTER PAIRS THAT ARE
4658 ;* ASSOCIATED WITH UNIBUS MEMORY DO NOT PERFORM RELOCATION.
4659 ;* IF ANY UNIBUS MEMORY PRESENT, DATI CYCLES TO THAT MEMORY
4660 ;* WILL BE CHECKED.
4661 ;*
4662 ;
4663 ; BGNST
4664 ;
4665 ; DO FOR KMCR = #67,27 (18-BIT AND 22 BIT MODES, 32K OF PMI MEMORY)
4666 ; DO WITH MAPPING PAIRS 10 TO 36
4667 ; . DO DATI TO 0 USING SELCTED MAPPING PAIR
4668 ; . IF NO TIMEOUT THEN
4669 ; . . ERROR DISABLED REGISTER PAIRS DO RELOCATION
4670 ; . . ENDF
4671 ; . ENDDO
4672 ; IF ANY UNIBUS MEMORY PRESENT THEN
4673 ; . DO DATI WITH RELOCATION ENABLED
4674 ; . IF RELOCATED THEN
4675 ; . . ERROR
4676 ; . . ENDF
4677 ; . ENDF
4678 ;
4679 ; ENDTST
4680 ;
4681 ;-----
4682 ;*****
4683 ;*TEST 56 RELOCATION WITH UNIBUS MEMORY
;*****

```

```

017766 000004
4684
4685 017770 013737 177734 001162 MOV KMCR,$TMP1 ; STORE KMCR
4686 017776 013701 177734 MOV KMCR,R1 ; STORE KMCR
4687 020002 042701 177740 BIC #177740,R1 ; LEAVE ONLY <4-0>
4688 020006 012737 170374 001160 MOV #MAPL37,$TMPO ; START WITH HIGHEST ADDRESS
4689 020014 032701 000037 BIT #37,R1 ; ANY UNIBUS MEMRORY?
4690 020020 001404 BEQ 101$ ; IF NOT, BRANCH
4691 020022 162737 000074 001160 100$: SUB #4,$TMPO ; GET LOWER REGISTER PAIR
4692 020030 077104 SOB R1,100$ ; LEAVE IN $TMPO HIGHEST AVAILABLE PAIR
4693 020032 052737 000400 177730 101$: BIS #BIT08,DCSR ; ENABLE DIAGN. MODE
4694 020040 012737 000067 177734 MOV #67,KMCR ; 18-BIT, 32K PMI
4695 020046 042737 000400 177730 BIC #BIT08,DCSR ; DISABLE DIAGN. MODE
4696
4697 ; SETUP MAPPING REGISTERS FOR INITIAL CONDITIONS
4698 ;
4699 020054 052737 000040 172516 BIS #BIT5,MMR3 ; ENABLE UNIBUS MAPPING
4700 020062 012701 170240 1$: MOV #MAPL10,R1 ; . POINT TO MAPL10
4701 020066 005002 CLR R2 ; . ADDRESS BITS <15-0> REG. 10
4702 020070 012703 000001 MOV #1,R3 ; . ADDRESS FOR CR2<17-16>
4703 020074 004737 002502 2$: JSR PC,IUBE ; . . INITIALIZE THE UBE
4704
4705 ;
4706 ; SET UP UBE AND UNIBUS MAP REGISTERS FOR A DATI CYCLE
4707 ;
4708 ;

```

T56 RELOCATION WITH UNIBUS MEMORY

```

4709 020100 010277 162102      MOV    R2,@BE1BA      ; . . POINT UBE TO FIRST MAP REGISTER
4710 020104 005077 162104      CLR    @BE1CR2       ; . . REGISTER
4711 020110 010377 162100      MOV    R3,@BE1CR2   ; . . LOAD ADDRESS <17-16>
4712 020114 005077 162062      CLR    @BE1DB       ; . . INITIALIZE THE DATA BUFFER
4713 020120 012777 177777 162056  MOV    #-1,@BE1CC   ; . . SET FOR 1 TRANSFER
4714 020126 005021              CLR    (R1)+        ; . . CLEAR LOW MAP REGISTER
4715 020130 005021              CLR    (R1)+        ; . . AND HIGH MAP TOO
4716 020132 012777 020220 162056  MOV    #10,@BE1VEC  ; . . LOAD INTERRUPT VECTOR
4717 020140 012777 000340 162052  MOV    #340,@BE1PSW ;
4718                                ;
4719                                ; GO DO THE TRANSFER
4720                                ;
4721                                ;
4722 020146 012777 002041 162034      MOV    #2041,@BE1CR1 ; . . SET UBE FOR NPR-DATI-1 XFER
4723 020154 032777 000200 162026 5$:  BIT    #BIT7,@BE1CR1 ; . . IS THE TRANSFER COMPLETE
4724 020162 001774              BEQ    5$           ; . . NO, THEN WAIT FOR IT TO BE COMPLETE
4725 020164 032777 000400 162022      BIT    #BIT08,@BE1CR2 ; . . NXM SET?
4726 020172 001002              BNE    6$           ; . . IF SO, BRANCH
4727 020174 104022              ERROR  +22         ; . . NO NXM
4728 020176 000412              BR     11$         ;
4729 020200 106427 000140 6$:      MTPS   #140        ; . . LOWER PRIORITY
4730 020204 000240              NOP                    ; . . WAIT FOR INTERRUPT
4731 020206 000240              NOP                    ;
4732 020210 104022              ERROR  +22         ; . . NO INTERRUPT ON NXM
4733 020212 106427 000340      MTPS   #340        ; . . RAISE PRIORITY
4734 020216 000402              BR     11$         ;
4735 020220 062706 000004 10$:    ADD    #4,SP        ; . . ADJUST STACK
4736 020224 062702 020000 11$:    ADD    #20000,R2    ; . . SELECT NEXT MAP PAIR REGISTER
4737 020230 103001              BCC    12$         ; . . IF NO CARRY, BRANCH
4738 020232 005203              INC    R3           ; . . OTHERWISE INCREMENT ADDRESS <17-16>
4739 020234 023701 001160 12$:    CMP    $TMP0,R1    ; . . ALL DONE?
4740 020240 101315              BHI    2$           ; . . IF LESS THAN, BRANCH
4741                                ;
4742                                ; CHECK IF DONE WITH 22-BIT MODE AND 18-BIT
4743                                ;
4744 020242 032737 000040 177734      BIT    #BIT05,KMCR  ; . . 22-BIT MODE DONE?
4745 020250 001412              BEQ    13$         ; . . IF YES, BRANCH TO EXIT
4746 020252 052737 000400 177730      BIS    #BIT08,DCSR ; . . ENABLE DIAGN. MODE
4747 020260 012737 000027 177734      MOV    #27,KMCR    ; . . 22-BIT, 32K PMI
4748 020266 042737 000400 177730      BIC    #BIT08,DCSR ; . . DISABLE DIAGN. MODE
4749 020274 000672              BR     1$           ; . . DO AGAIN IN 22 BIT MODE
4750                                ;
4751                                ; IF ANY UNIBUS MEMORY PRESENT, DO RELOCATION THRU IT
4752                                ;
4753 020276 022737 170374 001160 13$:  CMP    #MAPL37,$TMP0 ; ANY UNIBUS MEMORY?
4754 020304 001432              BEQ    200$        ; IF NO, EXIT
4755 020306 012777 140000 161672      MOV    #140000,@BE1BA ; TRY TO DO DATI THRU
4756 020314 012777 000003 161672      MOV    #3,@BE1CR2   ; THRU MAP 36 REGISTER
4757 020322 005037 170370              CLR    MAPL36       ; CLEAR MAPPING REGISTER
4758 020326 005037 170372              CLR    MAPH36       ; PAIR
4759 020332 012737 123456 000000      MOV    #123456,@#0  ; SET UP PATTERN AT 0
4760 020340 005077 161636              CLR    @BE1DB       ; CLEAR DATA REGISTER
4761 020344 012777 002071 161636      MOV    #2071,@BE1CR1 ; START DATI
4762 020352 105777 161632 20$:    TSTB   @BE1CR1     ; WAIT TILL
4763 020356 100375              BPL    20$         ; DONE
4764 020360 022777 123456 161614      CMP    #123456,@BE1DB ; DATA CAME FROM #0?
4765 020366 001001              BNE    200$        ; IF NOT, BRANCH

```

T56 RELOCATION WITH UNIBUS MEMORY

4766	020370	104033				ERROR	+33		; RELOCATED UNIBUS MEMORY
4767	020372	052737	000400	177730	2004:	BIS	#BIT08,DCSR		; ENABLE DIAGN. MODE
4768	020400	013737	001162	177734		MOV	\$TMP1,KMCR		; RESTORE KMCR
4769	020406	042737	000400	177730		BIC	#BIT08,DCSR		; DISABLE DIAGN. MODE

MAIN MEMORY DISABLE THRU UBE

```

4771 .SBTTL MAIN MEMORY DISABLE THRU UBE
4772
4773 ; * THIS TEST CHECKS THAT A MAIN MEMORY RESPONSE IS DISABLED WHENEVER
4774 ; * THE APPROPRIATE BITS IN KMCR ARE SET. THE FIRST 32K ARE NOT GOING
4775 ; * TO BE CHECKED. THE DMA DATI CYCLES WILL BE DONE THAT SHOULD CAUSE
4776 ; * NXM BIT TO BE SET IN THE UBE REGISTER.
4777 ;
4778 ; BGNTST
4779 ;
4780 ; SIZE THE MEMORY AVAILABLE
4781 ; DO FOR ALL MEMORY PAGES
4782 ; . LET KMCR DISABLE THAT PAGE IN MEMORY
4783 ; . DO DATI TO THAT LOCATION
4784 ; . IF NO TIMEOUT THEN
4785 ; . . ERROR DISABLED MEMORY DOES NOT TIMEOUT
4786 ; . ENDIF
4787 ; ENDDO
4788 ;
4789 ; ENDTST
4790 ;
4791 ;-----
4792 ; *****
4793 ; *TEST 57 MAIN MEMORY DISABLE THRU UBE
4794 ; *****

```

```

020414 000004
4794
4795 020416 005737 001720          TST      PMIS          ; ANY PMI MEMORY?
4796 020422 001561          BEQ      TST60         ;; IF NONE, EXIT TEST
4797
4798 ; ENABLE MMU AND SIZE MEMORY
4799 ;
4800 020424 004737 002276          JSR      PC,MAPPR     ; REMAP PROGRAM AREA
4801 020430 052737 000001 177572    BIS      #BIT00,MMR0  ; ENABLE MMU
4802 020436 052737 000020 172516    BIS      #BIT04,MMR3  ; ENABLE 22-BIT MODE
4803 020444 013702 001720          MOV      PMIS,R2     ; STORE HIGHEST PAGE
4804 020450 042737 000001 177572    BIC      #BIT00,MMR0  ; DISABLE MMU
4805
4806 ; SETUP MAPPING REGISTERS AND KMCR
4807 ;
4808 020456 052737 000040 172516    BIS      #BIT5,@MMR3  ; ENABLE UNIBUS MAPPING
4809 020464 013737 177734 001160    MOV      KMCR,@TMPO  ; SAVE KMCR
4810 020472 052737 000400 177730    BIS      #BIT08,DCSR  ; ENABLE DIAGNOSTIC MODE
4811 020500 012737 000067 177734    MOV      #67,KMCR    ; 18-BIT, >32K DISABLED
4812 020506 042737 000400 177730    BIC      #BIT08,DCSR  ; DISABLE DIAGNOSTIC MODE
4813 020514 005037 170200          CLR      MAPLOO      ; CLEAR LOW MAP REGISTER
4814 020520 012737 000001 170202    MOV      #1,MAPH00   ; POINT TO 32K
4815 020526 012777 020640 161462    MOV      #20,@BE1VEC ; . LOAD INTERRUPT VECTOR
4816 020534 012777 000340 161456    MOV      #340,@BE1PSW ;
4817 020542 012737 002000 172354    MOV      #2000,KIPAR6 ; INITIAL POINTER TO ABOVE 32K
4818
4819 ; DO DATI TO DISABLED MEMORY
4820 ;
4821 020550 004737 002502          104: JSR      PC,IUBE  ; INITIALISE UBE
4822 020554 005077 161426          CLR      @BE1BA      ; . ADDRESS TO ACCESS THRU MAP 0
4823 020560 012777 177777 161416    MOV      #-1,@BE1CC  ; . ONE CYCLE
4824 020566 012777 002041 161414    MOV      #2041,@BE1CR1 ; . SET UBE FOR NPR-DATI-1 XFER

```

T57 MAIN MEMORY DISABLE THRU UBE

```

4825 020574 032777 000200 161406 14#: BIT #BIT7,#BE1CR1 ; . IS THE TRANSFER COMPLETE
4826 020602 001774 BEQ 14# ; . NO, THEN WAIT FOR IT TO BE COMPLETE
4827 020604 032777 000400 161402 BIT #BIT08,#BE1CR2 ; . NXM SET?
4828 020612 001002 BNE 15# ; . IF SO, BRANCH
4829 020614 104022 ERROR +22 ; . NO NXM
4830 020616 000412 BR 25# ;
4831 020620 106427 000140 15#: MTPS #140 ; . LOWER PRIORITY
4832 020624 000240 NOP ; . WAIT FOR INTERRUPT
4833 020626 000240 NOP ;
4834 020630 104022 ERROR +22 ; . NO INTERRUPT ON NXM
4835 020632 106427 000340 MTPS #340 ; . RAISE PRIORITY
4836 020636 000402 BR 25# ; . BRANCH
4837 020640 062706 000004 20#: ADD #4,SP ; . ADJUST STACK
4838 ;
4839 ; CHANGE PAGE ACCESSED AND CHECK END CONDITIONS
4840 ;
4841 020644 023702 172354 25#: CMP KIPAR6,R2 ; . ALL PAGES DONE?
4842 020650 103032 BHIS 30# ; . IF YES, EXIT
4843 020652 062737 004000 170200 ADD #4000,MAPL00 ; . INCREMENT IN 1K
4844 020660 103002 BCC 26# ; . IF NO CARRY, BRANCH
4845 020662 005237 170202 INC MAPH00 ; . OTHERWISE INCR. HIGH MAP
4846 020666 032737 017777 170200 26#: BIT #17777,MAPL00 ; . IS IT ON 4K BOUNDARY?
4847 020674 001017 BNE 27# ; . IF NOT, BRANCH
4848 020676 062737 000200 17235- ADD #200,KIPAR6 ; . INCREMENT COUNTER
4849 020704 052737 000400 177730 BIS #BIT08,DCSR ; . ENABLE DIAG. MODE
4850 020712 005337 177734 DEC KMCR ; . ACCESS NEXT HIGHER LOCATION
4851 020716 032737 0C9040 177734 BIT #BIT05,KMCR ; . ALL 128K DONE?
4852 020724 001404 BEQ 30# ; . IF SO, BRANCH
4853 020726 042737 000400 177730 BIC #BIT08,DCSR ; . DISABLE DIAG. MODE
4854 020734 000705 BR 10# ; . DO FOR THE NEXT PAGE
4855 020736 042737 000040 172516 27#: BIC #BIT05,MMR3 ; . DISABLE MAPPING
4856 020744 052737 000400 177730 30#: BIS #BIT08,DCSR ; . ENABLE DIAG. MODE
4857 020752 013737 001160 177734 MOV #TMP0,KMCR ; . RESTORE KMCR
4858 020760 042737 000400 177730 BIC #BIT08,DCSR ; . DISABLE DIAG. MODE
4859

```

TEST UNIBUS DEVICE DATOB CYCLE

```

4861 .SBTTL TEST - UNIBUS DEVICE DATOB CYCLE
4862
4863 ; * THIS TEST WILL SEE IF THE KTJ11-B MODULE CAN EXECUTE
4864 ; * A UNIBUS DEVICE DATOB .
4865 ; *
4866 ;
4867 ; BGNTST
4868 ;
4869 ; *
4870 ; * SET UP UBE FOR A DATOB CYCLE
4871 ; *
4872 ; LET R1 := ADDRESS OF THE WRITE BUFFER
4873 ; LET (R1) := 0 ; CLEAR OUT THE LOCATION
4874 ; LET BE1DB := 77777 ; DATA PATTERN IS 77777
4875 ; LET BE1CC := -1 ; SET FOR 1 DATA XFER
4876 ; LET BE1BA := R1 ; ADDRESS IS FIRST LOCATION IN WRITE BUFFER
4877 ; SET UBE FOR NPR-DATOB-1 XFER
4878 ; SET OFF THE TRANSFER
4879 ; WAIT FOR THE TRANSFER TO COMPLETE
4880 ; *
4881 ; * CHECK IF THE TRANSFER OCCURED CORRECTLY
4882 ; *
4883 ; IF (R1) NEQ #377 THEN
4884 ; . ERROR TRANSFER DID NOT EXECUTE CORRECTLY
4885 ; . ENDIF
4886 ;
4887 ; ENDTST
4888 ;
4889 ; -----
4890 ; *****
4891 ; *TEST 60 UNIBUS DEVICE DATOB CYCLE TEST
; *****

```

020766	000004		
4892			
4893	020770	004737	002502
4894	020774	012701	001734
4895	021000	005011	
4896	021002	012777	077777 161172
4897	021010	012777	177777 161166
4898	021016	010177	161164
4899			
4900			
4901			
4902	021022	012777	003441 161160
4903	021030	032777	000200 161152
4904	021036	001774	
4905			
4906			
4907			
4908	021040	021127	000377
4909	021044	001401	
4910	021046	104033	
4911			
4912			

```

; *****
; *TEST 60 UNIBUS DEVICE DATOB CYCLE TEST
; *****
TST60: SCOPE
; JSR PC,IUBE ; INITIALIZE THE UBE
; MOV @WRTBUF,R1 ; GET POINTER TO WRITE BUFFER
; CLR (R1) ; CLEAR OUT THE LOCATION
; MOV #77777,@BE1DB ; SET DATA BUFFER UP
; MOV #-1,@BE1CC ; SET FOR 1 DATA TRANSFER
; MOV R1,@BE1BA ; POINT UBE XFER TO WRITE BUFFER
;
; DO THE TRANSFER
;
; MOV #3441,@BE1CR1 ; SET UP UBE FOR NPR-DATOB-1 XFER
; BIT @BIT7,@BE1CR1 ; IS THE TRANSFER COMPLETE ?
; BEQ 5$ ; NO, THEN WAIT FOR IT TO COMPLETE
;
; CHECK THE TRANSFER
;
; CMP (R1),#377 ; WAS THE TRANSFER CORRECT ?
; BEQ TST61 ; YES GO TO THE NEXT TEST
; ERROR +33 ; ERROR TRANSFER WAS INCORRECT

```

TEST - UNIBUS DEVICE DATIP CYCLE

```

4914          .SBTTL TEST - UNIBUS DEVICE DATIP CYCLE
4915
4916          ; * THIS TEST WILL SEE IF THE KTJ11-B MODULE CAN EXECUTE
4917          ; * A UNIBUS DEVICE DATIP CYCLE.
4918          ; *
4919          ;
4920          ; BGNST
4921          ; *
4922          ; * INITIALIZE WRITE BUFFER FOR DATIP CYCLE
4923          ; *
4924          ;       LET R1 := POINTER TO WRITE BUFFER
4925          ;       DO FOR R2 := 1 TO 8
4926          ;           . LET (R1)+ := #52525
4927          ;       ENDDO
4928          ; *
4929          ; * SET UP UBE TO DO 8 DATIP'S TO WRITE BUFFER
4930          ; *
4931          ;       LET BE1BA := ADDRESS OF WRITE BUFFER
4932          ;       LET BE1CC := -8.           ; SET FOR 8 TRANSFERS
4933          ;       SET UBE FOR NPR-DATIP-1 XFER
4934          ;       SET OFF THE TRANSFER
4935          ;       WAIT FOR TRANSFER TO BE COMPLETE
4936          ; *
4937          ; * CHECK IF DATIP WAS EXECUTED CORRECTLY
4938          ; *
4939          ;       LET R1 := ADDRESS OF WRITE BUFFER
4940          ;       DO FOR R2 := 1 TO 8
4941          ;           . IF (R1)+ NEQ 125252 THEN
4942          ;               . ERROR TRANSFER DID NOT EXECUTE CORRECTLY
4943          ;           . ENDF
4944          ;       ENDDO
4945          ;
4946          ; ENDTST
4947          ;
4948          ; -----
4949          ; ;*****
4950          ; ;*TEST 61      UNIBUS DEVICE DATIP CYCLE TEST
4951          ; ;*****
4952          ; TST61:  SCOPE
4953          ;
4954          ;
4955          ; INITIALIZE WRITE BUFFER FOR DATIP CYCLE
4956          ;
4957          ;
4958          ; MOV     #WRTBUF,R1      ; GET POINTER TO WRITE BUFFER
4959          ; MOV     #8.,R2         ; SET UP LOOP COUNTER
4960          ; 5$:   MOV     #52525,(R1)+ ; . INITIALIZE WRITE BUFFER LOCATION
4961          ;       SOB     R2,5$     ; . HAVE WE DONE IT 8 TIMES ?
4962          ;
4963          ;
4964          ; SET UP UBE TO 8 DATIP'S TO WRITE BUFFER
4965          ;
4966          ;
4967          ; MOV     #WRTBUF,#BE1BA ; SET UP UBE TO WRITE BUFFER

```

021050 000004

4951
4952 021052 004737 002502

4953

4954

4955

4956

4957

4958 021056 012701 001734

4959 021062 012702 000010

4960 021066 012721 052525

4961 021072 077203

4962

4963

4964

4965

4966

4967 021074 012777 001734 161104

T61 UNIBUS DEVICE DATIP CYCLE TEST

```

4968 021102 005077 161106          CLR    @BE1CR2          ;
4969 021106 012777 177770 161070    MOV    #-8,@BE1CC      ; SET UP FOR 8 TRANSFERS
4970 021114 012777 002441 161066    MOV    #2441,@BE1CR1   ; SET UP UBE FOR NPR DATIP 1 XFER
4971 021122 032777 000200 161060 10$: BIT    @BIT7,@BE1CR1   ; IS THE TRANSFER COMPLETE ?
4972 021130 001774          BEQ    10$              ; NO, THEN WAIT FOR IT TO BE COMPLETE
4973
4974
4975          ; CHECK IF DATIP WAS EXECUTED CORRECTLY
4976          ;
4977
4978 021132 012701 001734          MOV    #WRTBUF,R1      ; GET POINTER TO WRITE BUFFER
4979 021136 012702 000010          MOV    #8,R2          ; SET UP LOOP COUNTER
4980 021142 022127 125252 15$:    CMP    (R1)+,@125252   ; . WAS THE TRANSFER CORRECT ?
4981 021146 001401          BEQ    20$              ; . YES, THEN GO SEE IF WE ARE DONE CHECKING
4982 021150 104033          ERROR  +33           ; . NO, THEN ERROR IN THE TRANSFER
4983 021152 077205 20$:    SOB    R2,15$              ; . HAVE WE CHECKED ALL THE XFERS
4984
4985

```


TEST UNIBUS DEVICE I/O PAGE READ CYCLE

```

4987      .SBTTL TEST - UNIBUS DEVICE I/O PAGE READ CYCLE
4988
4989      ;* THIS TEST WILL SEE IF THE KTJ11-B MODULE CAN EXECUTE
4990      ;* A UNIBUS DEVICE I/O PAGE READ CYCLE. THIS CONSISTS OF
4991      ;* FOUR DIFFERENT TYPES OF READS :
4992      ;*
4993      ;*      1. UNIBUS DEVICE PMI I/O PAGE READ CYCLE
4994      ;*      2. UNIBUS DEVICE TO UNIBUS DEVICE I/O PAGE READ CYCLE
4995      ;*      3. UNIBUS DEVICE BOOT ROM READ CYCLE
4996      ;*      4. UNIBUS DEVICE MAP REGISTER READ CYCLE.
4997      ;*
4998      ;* IT ALSO WILL CHECK THAT A UNIBUS DEVICE READ CYCLE NOT
4999      ;* OF THE ABOVE THREE TYPES SHOULD CAUSE A TIMEOUT.
5000      ;*
5001      ;
5002      ; BGNTST
5003      ;*
5004      ;* DO A UNIBUS DEVICE PMI I/O PAGE READ CYCLE
5005      ;*
5006      ;       LET BE1BA := ADDRESS OF CSR OF MEMORY BOARD
5007      ;       LET BE1CC := -1                               ;SET FOR 1 TRANSFER
5008      ;       SET UBE FOR A NPR-DATI-1 DATA XFER
5009      ;       SET OFF THE TRANSFER
5010      ;       WAIT FOR THE TRANSFER TO BE COMPLETE
5011      ;       IF BE1DB NEQ CONTENTS OF MEMORY CSR THEN
5012      ;       . ERROR UNIBUS DEVICE PMI I/O PAGE READ CYCLE DID NOT EXECUTE CORRECTLY
5013      ;       ENDIF
5014      ;*
5015      ;* DO A UNIBUS DEVICE TO UNIBUS DEVICE I/O PAGE READ CYCLE
5016      ;*
5017      ;       LET BE1BA := ADDRESS OF BE1BA
5018      ;       LET BE1CC := -1                               ;SET FOR 1 TRANSFER
5019      ;       SET UBE FOR A NPR-DATI-1 DATA XFER
5020      ;       SET OFF THE TRANSFER
5021      ;       WAIT FOR THE TRANSFER TO BE COMPLETE
5022      ;       IF BE1DB NEQ ADDRESS OF BE1BA
5023      ;       . ERROR UNIBUS DEVICE TO UNIBUS DEVICE I/O PAGE READ CYCLE DID NOT EXECUTE CORRECTLY
5024      ;       ENDIF
5025      ;*
5026      ;* DO A UNIBUS DEVICE BOOT ROM READ CYCLE
5027      ;*
5028      ;       LET BE1BA := 773000
5029      ;       LET BE1CC := -1                               ;SET FOR 1 TRANSFER
5030      ;       SET UBE FOR A NPR-DATI-1 DATA XFER
5031      ;       SET OFF THE TRANSFER
5032      ;       WAIT FOR THE TRANSFER TO BE COMPLETE
5033      ;       IF BE1DB NEQ @0173000 THEN
5034      ;       . ERROR UNIBUS DEVICE BOOT ROM READ CYCLE DID NOT EXECUTE CORRECTLY
5035      ;       ENDIF
5036      ;*
5037      ;* DO A UNIBUS DEVICE MAP REGISTER READ CYCLE
5038      ;*
5039      ;       LET BE1BA := 770200                               ;UNIBUS MAP REGISTER #0
5040      ;       LET BE1CC := -1                               ;SET FOR 1 TRANSFER
5041      ;       SET UBF FOR A NPR-DATI-1 DATA XFER
5042      ;       SET OFF THE TRANSFER
5043      ;       WAIT FOR THE TRANSFER TO BE COMPLETE

```

TEST - UNIBUS DEVICE I/O PAGE READ CYCLE

```

5044 ; IF BE1DB NEQ @#770200 THEN
5045 ; . ERROR UNIBUS DEVICE MAP REGISTER READ CYCLE DID NOT EXECUTE CORRECTLY
5046 ; ENDF
5047 ;*
5048 ;* DO AN ILLEGAL UNIBUS DEVICE I/O PAGE READ CYCLE
5049 ;*
5050 ; LET BE1BA := 777730,777732,777734 ;DCSR,DDR,KMCR ADDRESS
5051 ; LET BE1CC := -1 ;1 DATA TRANSFER
5052 ; SET UBE FOR NPR-DATI-1 DATA XFER
5053 ; SET OFF THE TRANSFER
5054 ; WAIT FOR THE TRANSFER TO BE COMPLETE
5055 ; IF TRANSFER DID NOT TIMEOUT THEN
5056 ; . ERROR AN ILLEGAL UNIBUS DEVICE CYCLE HAS OCCURED
5057 ; ENDF
5058 ;
5059 ; ENDTST
5060 ;
5061 ;
5062 ;
5063 ;

```

```

*****
;*TEST 62 UNIBUS DEVICE I/O PAGE READ CYCLE
*****
TST62: SCOPE

```

```

021154 000004
5064
5065 021156 004737 002502 JSR PC,IUBE ; INITIALIZE THE UBE
5066
5067
5068 ; DO A UNIBUS DEVICE PMI I/O PAGE CYCLE
5069 ;
5070 021162 013777 001732 161016 MOV MCSR,@BE1BA ; GET ADDRESS OF MEMORY CSR ADDRESS
5071 021170 012777 000003 161016 MOV #3,@BE1CR2 ; SET THE UPPER TWO ADDRESS BITS
5072 021176 012777 177777 161000 MOV #-1,@BE1CC ; SET FOR 1 TRANSFER
5073 021204 012777 002041 160776 MOV #2041,@BE1CR1 ; SET UBE FOR NPR-DATI-1 XFER
5074 021212 032777 000200 160770 5$: BIT #BIT7,@BE1CR1 ; ARE WE DONE THE TRANSFER ?
5075 021220 001774 BEQ 5$ ; NO,THEN FOR WAIT FOR IT TO
5076 ;
5077 ; CHECK OUT THE TRANSFER
5078 ;
5079 021222 027777 160504 160752 CMP @MCSR,@BE1DB ; WAS THE TRANSFER CORRECT
5080 021230 001401 BEQ 6$ ; YES,THEN GO DO BOOT ROM READ
5081 021232 104033 ERROR +33 ; NO,THEN ERROR IN READ CYCLE
5082 ;
5083 ; DO A UNIBUS DEVICE UNIBUS DEVICE I/O PAGE CYCLE
5084 ;
5085 021234 013777 002206 160744 6$: MOV BE1BA,@BE1BA ; GET AN ADDRESS IN I/O PAGE
5086 021242 012777 000003 160744 MOV #3,@BE1CR2 ; SET THE UPPER TWO ADDRESS BITS
5087 021250 012777 177777 160726 MOV #-1,@BE1CC ; SET FOR 1 TRANSFER
5088 021256 012777 002041 160724 MOV #2041,@BE1CR1 ; SET UBE FOR NPR-DATI-1 XFER
5089 021264 032777 000200 160716 8$: BIT #BIT7,@BE1CR1 ; ARE WE DONE THE TRANSFER ?
5090 021272 001774 BEQ 8$ ; NO,THEN FOR WAIT FOR IT TO
5091 ;
5092 ; CHECK OUT THE TRANSFER
5093 ;
5094 021274 023777 002206 160700 CMP BE1BA,@BE1DB ; WAS THE TRANSFER CORRECT
5095 021302 001401 BEQ 10$ ; YES,THEN GO DO BOOT ROM READ
5096 021304 104033 ERROR +33 ; NO,THEN ERROR IN READ CYCLE
5097 ;

```

T62 UNIBUS DEVICE I/O PAGE READ CYCLE

```

5098 ; DO A UNIBUS DEVICE BOOT ROM READ CYCLE
5099 ;
5100 021306 012777 173000 160672 10$: MOV #173000,@BE1BA ; GET ADDRESS OF OF THE BOOT ROM
5101 021314 012777 000003 160672 MOV #3,@BE1CR2 ; SET THE UPPER TWO ADDRESS BITS
5102 021322 012777 177777 160654 MOV #-1,@BE1CC ; SET FOR 1 TRANSFER
5103 021330 012777 002041 160652 MOV #2041,@BE1CR1 ; SET UBE FOR NPR-DATI-1 XFER
5104 021336 032777 000200 160644 15$: BIT #BIT7,@BE1CR1 ; ARE WE DONE THE TRANSFER ?
5105 021344 001774 BEQ 15$ ; NO,THEN FOR WAIT FOR IT TO
5106 ;
5107 ; CHECK OUT THE TRANSFER
5108 ;
5109 021346 052737 000200 177520 BIS #BIT07,BCSR ; SELECT RIGHT ROM
5110 021354 023777 173000 160620 CMP @#173000,@BE1DB ; WAS THE TRANSFER CORRECT
5111 021362 001401 BEQ 20$ ; YES,THEN GO DO UNIBUS DEVICE MAP REGISTER READ
5112 021364 104033 ERROR +33 ; NO,THEN ERROR IN READ CYCLE
5113 ;
5114 ; DO A UNIBUS DEVICE MAP REGISTER READ
5115 ;
5116 021366 012777 170200 160612 20$: MOV #170200,@BE1BA ; GET ADDRESS OF THE MAP REGISTER
5117 021374 012777 000003 160612 MOV #3,@BE1CR2 ; SET THE UPPER TWO ADDRESS BITS
5118 021402 012777 177777 160574 MOV #-1,@BE1CC ; SET FOR 1 TRANSFER
5119 021410 012777 002041 160572 MOV #2041,@BE1CR1 ; SET UBE FOR NPR-DATI-1 XFER
5120 021416 032777 000200 160564 25$: BIT #BIT7,@BE1CR1 ; ARE WE DONE THE TRANSFER ?
5121 021424 001774 BEQ 25$ ; NO,THEN FOR WAIT FOR IT TO
5122 ;
5123 ; CHECK OUT THE TRANSFER
5124 ;
5125 021426 023777 170200 160546 CMP @#170200,@BE1DB ; WAS THE TRANSFER CORRECT
5126 021434 001401 BEQ 30$ ; YES,THEN GO DO ILLEGAL UNIBUS DEVICE I/O READ
5127 021436 104033 ERROR +33 ; NO,THEN ERROR IN READ CYCLE
5128 ;
5129 ; DO AN ILLEGAL UNIBUS DEVICE I/O PAGE READ
5130 ;
5131 021440 012777 021542 160550 30$: MOV #40$,@BE1VEC ; SET UP INTERRUPT VECTOR
5132 021446 012777 000340 160544 MOV #340,@BE1PSW ;
5133 021454 012701 000003 MOV #3,R1 ; DO FOR 3 REGISTERS
5134 021460 012702 177730 MOV #177730,R2 ; STARTING WITH DCSR (THEN DDR, KMCR)
5135 021464 010277 160516 31$: MOV R2,@BE1BA ; GET ADDRESS OF THE DIAGNOSTIC REGISTER
5136 021470 012777 000003 160516 MOV #3,@BE1CR2 ; SET THE UPPER TWO ADDRESS BITS
5137 021476 012777 177777 160500 MOV #-1,@BE1CC ; SET FOR 1 TRANSFER
5138 021504 012777 002041 160476 MOV #2041,@BE1CR1 ; SET UBE FOR NPR-DATI-1 XFER
5139 021512 106427 000140 MTPS #140 ; LOWER PRIORITY
5140 021516 032777 000200 160464 35$: BIT #BIT7,@BE1CR1 ; ARE WE DONE THE TRANSFER ?
5141 021524 001774 BEQ 35$ ; NO,THEN FOR WAIT FOR IT TO
5142 021526 000240 NOP ; WAIT FOR INTERRUPT
5143 021530 000240 NOP ;
5144 021532 104033 ERROR +33 ; ERROR KTJ11 RESPONDED TO AN ILLEGAL ADDRESS
5145 021534 106427 000340 MTPS #340 ; RAISE PRIORITY BACK
5146 021540 000407 BR 45$ ;
5147 021542 062706 000004 40$: ADD #4,SP ; CLEAN UP THE STACK AND THEN GO DO NEXT TEST
5148 021546 032777 000400 160440 BIT #BIT08,@BE1CR2 ; NXM SET?
5149 021554 001001 BNE 45$ ; IF SET, BRANCH
5150 021556 104033 ERROR +33 ; NXM NOT SET ON ILLEGAL REFERENCE
5151 021560 005722 45$: TST (R2)+ ; GET ADDRESS OF NEXT REGISTER
5152 021562 004737 002502 JSR PC,IUBE ; CLEAN UP UBE
5153 021566 077142 SOB R1,31$ ; DO FOR ALL 3 REGISTERS
5154

```

TEST - UNIBUS DEVICE I/O PAGE WRITE CYCLE

```

5156                   .SBTTL TEST - UNIBUS DEVICE I/O PAGE WRITE CYCLE
5157
5158                   ;* THIS TEST WILL SEE IF THE KTJ11-B MODULE CAN EXECUTE
5159                   ;* A UNIBUS DEVICE I/O PAGE WRITE CYCLE. THIS CONSISTS OF
5160                   ;* TWO DIFFERENT TYPES OF WRITES :
5161                   ;*
5162                   ;*       1. UNIBUS DEVICE PMI I/O PAGE WRITE CYCLE
5163                   ;*       2. UNIBUS DEVICE UNIBUS DEVICE I/O PAGE WRITE CYCLE
5164                   ;*       3. UNIBUS DEVICE MAP REGISTER WRITE CYCLE.
5165                   ;*
5166                   ;* IT ALSO WILL CHECK THAT A UNIBUS DEVICE WRITE CYCLE NOT
5167                   ;* OF THE ABOVE THREE TYPES SHOULD CAUSE A TIMEOUT.
5168                   ;*
5169                   ;
5170                   ;   BGNTST
5171                   ;
5172                   ;*
5173                   ;* DO A UNIBUS DEVICE PMI I/O PAGE WRITE CYCLE
5174                   ;*
5175                   ;       LET BE1DB := 52525
5176                   ;       LET BE1BA := ADDRESS OF CSR OF MEMORY BOARD
5177                   ;       LET BE1CC := -1                               ;SET FOR 1 TRANSFER
5178                   ;       SET UBE FOR A NPR-DATO-1 DATA XFER
5179                   ;       SET OFF THE TRANSFER
5180                   ;       WAIT FOR THE TRANSFER TO BE COMPLETE
5181                   ;       IF CONTENTS OF MEMORY CSR NEQ 52525 THEN
5182                   ;       . ERROR UNIBUS DEVICE PMI I/O PAGE WRITE CYCLE DID NOT EXECUTE CORRECTLY
5183                   ;       ENDIF
5184                   ;*
5185                   ;* DO A UNIBUS DEVICE TO UNIBUS DEVICE WRITE CYCLE
5186                   ;*
5187                   ;       IF 2 UBE'S THEN
5188                   ;       LET BE2DB := 125252                           ;DATA PATTERN
5189                   ;       LET BE2BA := ADDR. OF BE2DB                 ;UNIBUS I/O PAGE ADDRESS
5190                   ;       LET BE2CC := -1                               ;SET FOR 1 TRANSFER
5191                   ;       SET UBE FOR A NPR-DATO-1 DATA XFER
5192                   ;       SET OFF THE TRANSFER
5193                   ;       WAIT FOR THE TRANSFER TO BE COMPLETE
5194                   ;       IF BE1DB NEQ #125252 THEN
5195                   ;       . ERROR UNIBUS DEVICE TO UNIBUS DEVICE WRITE CYCLE DID NOT EXECUTE CORRECTLY
5196                   ;       ENDIF
5197                   ;       ENDIF
5198                   ;*
5199                   ;* DO A UNIBUS DEVICE MAP REGISTER WRITE CYCLE
5200                   ;*
5201                   ;       LET BE1DB := 52525                           ;DATA PATTERN
5202                   ;       LET BE1BA := 770200                         ;UNIBUS MAP REGISTER #0
5203                   ;       LET BE1CC := -1                               ;SET FOR 1 TRANSFER
5204                   ;       SET UBE FOR A NPR-DATO-1 DATA XFER
5205                   ;       SET OFF THE TRANSFER
5206                   ;       WAIT FOR THE TRANSFER TO BE COMPLETE
5207                   ;       IF MAP REGISTER 0 NEQ #52525 THEN
5208                   ;       . ERROR UNIBUS DEVICE MAP REGISTER WRITE CYCLE DID NOT EXECUTE CORRECTLY
5209                   ;       ENDIF
5210                   ;*
5211                   ;* DO A UNIBUS DEVICE TO BOOT ROM WRITE CYCLE
5212                   ;*

```

TEST - UNIBUS DEVICE I/O PAGE WRITE CYCLE

```

5213 ; LET BE1BA := 773000
5214 ; LET BE1CC := -1 ;SET FOR 1 TRANSFER
5215 ; SET UBE FOR A NPR-DATO 1 DATA XFER
5216 ; SET OFF THE TRANSFER
5217 ; WAIT FOR THE TRANSFER TO BE COMPLETE
5218 ; IF NO TIMEOUT THEN
5219 ; . ERROR UNIBUS DEVICE BOOT ROM WRITE CYCLE DID NOT EXECUTE CORRECTLY
5220 ; ENDF
5221 ;*
5222 ;* DO AN ILLEGAL UNIBUS DEVICE I/O PAGE WRITE CYCLE
5223 ;*
5224 ; LET BE1DB := 52525 ;DATA PATTERN
5225 ; LET BE1BA := 777730,777732,777734 ;DCSR, DDR, KMCR ADDRESSES
5226 ; LET BE1CC := -1 ;1 DATA TRANSFER
5227 ; SET UBE FOR NPR-DATO-1 DATA XFER
5228 ; SET OFF THE TRANSFER
5229 ; WAIT FOR THE TRANSFER TO BE COMPLETE
5230 ; IF TRANSFER DID NOT TIMEOUT THEN
5231 ; . ERROR AN ILLEGAL UNIBUS DEVICE CYCLE HAS OCCURED
5232 ; ENDF
5233 ; DISABLE UNIBUS MAPPING
5234 ;
5235 ; ENDTST

```

```

-----
;*****
;*TEST 63 UNIBUS DEVICE I/O PAGE WRITE CYCLE
;*****
TST63: SCOPE

```

```

021570 000004
5240
5241 021572 004737 002502 JSR PC,IUBE ; INITIALIZE THE UBE
5242 ;
5243 ; DO A UNIBUS DEVICE PMI I/O PAGE CYCLE
5244 ;
5245 021576 017737 160130 001160 MOV @MCSR,$TMPO ; STORE MEMORY CSR
5246 021604 013777 001732 160374 MOV MCSR,@BE1BA ; GET ADDRESS OF MEMORY CSR
5247 021612 012777 000003 160374 MOV #3,@BE1CR2 ; SET UPPER TWO ADDRESS BITS
5248 021620 012777 040000 160354 MOV @BIT14,@BE1DB ; GET WRITE PATTERN
5249 021626 012777 177777 160350 MOV #-1,@BE1CC ; SET FOR 1 XFER
5250 021634 012777 003041 160346 MOV #3041,@BE1CR1 ; SET UBE FOR NPR-DATO-1 XFER
5251 021642 032777 000200 160340 5$: BIT @BIT7,@BE1CR1 ; IS THE TRANSFER COMPLETE ?
5252 021650 001774 BEQ 5$ ; NO, THEN WAIT FOR IT TO COMPLETE
5253 ;
5254 ; CHECK THE TRANSFER
5255 ;
5256 021652 032777 040000 160052 BIT @BIT14,@MCSR ; WAS THE TRANSFER CORRECT ?
5257 021660 001001 BNE 6$ ; YES, THEN GO DO A MAP REGISTER WRITE
5258 021662 104033 ERROR +33 ; NO, THEN ERROR IN THE TRANSFER
5259 ;
5260 ; DO A UNIBUS DEVICE TO UNIBUS DEVICE WRITE CYCLE
5261 ;
5262 021664 013777 001160 160040 6$: MOV $TMPO,@MCSR ; RESTORE MEMORY CSR
5263 021672 022737 000002 001722 CMP #2,UBECT ; 2 UBE'S?
5264 021700 001030 BNE 10$ ; NO, SKIP SUBTEST
5265 021702 013777 002202 160316 MOV BE1DB,@BE2BA ; GET ADDRESS OF TARGET UBE REGISTER
5266 021710 012777 000003 160316 MOV #3,@BE2CR2 ; SET UPPER TWO ADDRESS BITS

```

T63 UNIBUS DEVICE I/O PAGE WRITE CYCLE

```

5267 021716 012777 125252 160276      MOV    #125252,@BE2DB ; GET WRITE PATTERN
5268 021724 012777 177777 160272      MOV    #-1,@BE2CC    ; SET FOR 1 XFER
5269 021732 012777 003041 160270      MOV    #3041,@BE2CR1 ; SET UBE FOR NPR-DATO 1 XFER
5270 021740 032777 000200 160262 8$:   BIT    #BIT7,@BE2CR1 ; IS THE TRANSFER COMPLETE ?
5271 021746 001774                BEQ    8$            ; NO, THEN WAIT FOR IT TO COMPLETE
5272                ;
5273                ; CHECK THE TRANSFER
5274                ;
5275 021750 027727 160226 125252      CMP    @BE1DB,#125252 ; WAS THE TRANSFER CORRECT ?
5276 021756 001401                BEQ    10$          ; YES, THEN GO DO A MAP REGISTER WRITE
5277 021760 104033                ERROR  +33         ; NO, THEN ERROR IN THE TRANSFER
5278                ;
5279                ; DO A UNIBUS DEVICE MAP REGISTER WRITE CYCLE
5280                ;
5281 021762 005037 170200      10$:   CLR    MAPL00      ; CLEAR MAP REGISTER PAIR
5282 021766 005037 170200                CLR    MAPH00      ;
5283 021772 012777 170200      MOV    #MAPL00,@BE1BA ; GET ADDRESS OF MAP REGISTER
5284 022000 012777 000003 160206      MOV    #3,@BE1CR2   ; SET UPPER TWO ADDRESS BITS
5285 022006 012777 052525 160166      MOV    #52525,@BE1DB ; GET WRITE PATTERN
5286 022014 012777 177776 160162      MOV    #-2,@BE1CC   ; SET FOR 2 XFER
5287 022022 012777 003041 160160      MOV    #3041,@BE1CR1 ; SET UBE FOR NPR-DATO-1 XFER
5288 022030 032777 000200 160152 15$:   BIT    #BIT7,@BE1CR1 ; IS THE TRANSFER COMPLETE ?
5289 022036 001774                BEQ    15$         ; NO, THEN WAIT FOR IT TO COMPLETE
5290                ;
5291                ; CHECK THE TRANSFER
5292                ;
5293 022040 023727 170200 052524      CMP    MAPL00,#52524 ; WAS THE TRANSFER CORRECT ?
5294 022046 001401                BEQ    16$          ; YES, THEN GO DO A MAP REGISTER WRITE
5295 022050 104033                ERROR  +33         ; NO, THEN ERROR IN THE TRANSFER
5296 022052 022737 000025 170202 16$:   CMP    #25,MAPH00   ; SECOND WORD OK TOO?
5297 022060 001401                BEQ    17$          ; IF SO, BRANCH
5298 022062 104033                ERROR  +33         ; NO, THEN ERROR IN THE TRANSFER
5299                ;
5300                ; DO A WRITE CYCLE TO BOOT ROM THAT SHOULD CAUSE A TIMEOUT
5301                ;
5302 022064 012777 173000 160114 17$:   MOV    #173000,@BE1BA ; ADDRESS TO WRITE TO
5303 022072 012777 177777 160104      MOV    #-1,@BE1CC   ; 1 CYCLE
5304 022100 012777 022146 160110      MOV    #19$,@BE1VEC ; POINT INTERRUPT VECTOR TO PROGRAM
5305 022106 012777 000340 160104      MOV    #340,@BE1PSW ; AT PRIORITY 7
5306 022114 012777 003041 160066      MOV    #3041,@BE1CR1 ; 1 DATO-NPR
5307 022122 106427 000140                MTPS   #140        ; LOWER PRIORITY
5308 022126 105777 160056      18$:   TSTB   @BE1CR1    ; DONE?
5309 022132 100375                BPL    18$         ; WAIT TILL DONE
5310 022134 000240                NOP
5311 022136 104033                ERROR  +33         ; DIDN'T TIMEOUT ON BOOT ROM WRITE
5312 022140 106427 000340                MTPS   #340        ; RAISE PRIORITY
5313 022144 000407                BR     20$         ;
5314 022146 062706 000004      19$:   ADD    #4,SP        ; ADJUST STACK
5315 022152 032777 000400 160034      BIT    #BIT08,@BE1CR2 ; NXM SET?
5316 022160 001001                BNE    20$         ; IF SET, BRANCH
5317 022162 104033                ERROR  +33         ; NXM NOT SET ON BOOT ROM WRITE
5318                ;
5319                ; DO AN ILLEGAL UNIBUS DEVICE I/O PAGE WRITE CYCLE
5320                ;
5321 022164 004737 002502      20$:   JSR    PC,IUBE     ; CLEAN UP UBE
5322 022170 012701 000003                MOV    #3,R1       ; DO FOR DCSR,DDR,KMCR
5323 022174 012702 177730                MOV    #177730,R2  ; START WITH DCSR

```

T63 UNIBUS DEVICE I/O PAGE WRITE CYCLE

5324	022200	012777	022306	160010		MOV	#30\$,@BE1VEC	:	SET UP TIMEOUT VECTOR
5325	022206	012777	000340	160004		MOV	#340,@BE1PSW	:	
5326	022214	010277	157766		21\$:	MOV	R2,@BE1BA	:	GET ADDRESS OF THE DIAGN. REGISTER
5327	022220	012777	000003	157766		MOV	#3,@BE1CR2	:	SET UPPER TWO ADDRESS BITS
5328	022226	012777	052525	157746		MOV	#52525,@BE1DB	:	GET WRITE PATTERN
5329	022234	012777	177777	157742		MOV	#1,@BE1CC	:	SET FOR 1 XFER
5330	022242	052737	000040	172516		BIS	#BIT5,MMR3	:	ENABLE UNIBUS MAPPING
5331	022250	012777	003041	157732		MOV	#3041,@BE1CR1	:	SET UBE FOR NPR-DATO-1 XFER
5332	022256	005037	177776			CLR	PSW	:	LOWER PRIORITY
5333	022262	032777	000200	157720	25\$:	BIT	#BIT7,@BE1CR1	:	IS THE TRANSFER COMPLETE ?
5334	022270	001774				BEQ	25\$:	NO, THEN WAIT FOR IT TO COMPLETE
5335	022272	000240				NOP		:	WAIT FOR INTERRUPT
5336	022274	000240				NOP		:	
5337	022276	104033				ERROR	+33	:	ERROR - AN ILLEGAL CYCLE DID NOT TIMEOUT
5338	022278	106427	000340			MTPS	#340	:	RAISE PRIORITY BACK
5339	022280	000407				BR	35\$:	
5340	022306	062706	000004		30\$:	ADD	#4,SP	:	ADJUST THE STACK AND GO TO NEXT TEST
5341	022312	032777	000400	157674		BIT	#BIT08,@BE1CR2	:	NXM SET?
5342	022320	001001				BNE	35\$:	IF SET, BRANCH
5343	022322	104033				ERROR	+33	:	NXM NOT SET ON ILLEGAL REFERENCE
5344	022324	005722			35\$:	TST	(R2)+	:	GET NEXT DIAGNOSTIC REGISTER
5345	022326	004737	002502			JSR	PC,IUBE	:	INITIALISE UBE
5346	022332	077150				SOB	R1,21\$:	DO FOR ALL OF THEM

TEST - MAPPING REGISTERS TEST USING UBE

```

5348 .SBTTL TEST - MAPPING REGISTERS TEST USING UBE
5349
5350 ;* THIS TEST WRITES DIFFERENT PATTERNS TO MAPPING REGISTERS USING
5351 ;* DATI AND DATO CYCLES THRU UBE.
5352 ;
5353 ; BGMTST
5354 ;
5355 ; ENABLE UNIBUS MAPPING
5356 ; WRITE ALL MAPPING REGISTERS WITH "10" PATTERN
5357 ; READ ALL REGISTERS BACK CLEARING THE ONE JUST READ
5358 ; IF ANY REGISTER DOES NOT HAVE THE PATTERN
5359 ; . ERROR IN WRITING TO MAP REGISTERS
5360 ; ENDF
5361 ;
5362 ; ENDTST
5363 ;
5364 ;-----
5365 ;*****
5366 ;*TEST 64 MAPPING REGISTERS TEST USING UBE
;*****
TST64: SCOPE
5367 022334 000004 JSR PC,IUBE ; CLEAN UP UBE
5368 022336 004737 002502
5369 ;
5370 ; DO DATO TO ALL 100 REGISTERS WITH 125252 AS PATTERN
5371 ;
5371 022342 012777 170200 157636 MOV #MAPLOO,@BE18A ; THE STARTING ADDRESS
5372 022350 012777 177700 157626 MOV #-100,@BE1CC ; 64 WORDS TO WRITE TO
5373 022356 012777 125252 157616 MOV #125252,@BE10B ; THE PATTERN TO BE WRITTEN
5374 022364 012777 000003 157622 MOV #3,@BE1CR2 ; ADDRESS <17-16>
5375 022372 012777 003041 157610 MOV #3041,@BE1CR1 ; GO DO DATO'S
5376 022400 105777 157604 1$: TSTB @BE1CR1 ; DONE BIT SET?
5377 022404 100375 BPL 1$ ; WAIT
5378 ;
5379 ; CHECK THAT ALL REGISTERS WRITTEN OK
5380 ;
5381 022406 012701 170200 MOV #MAPLOO,R1 ; START WITH MAP REGISTER 1
5382 022412 010177 157570 MOV R1,@BE18A ; STARTING ADDRESS
5383 022416 012777 177777 157560 2$: MOV #-1,@BE1CC ; . DO JUST 1 CYCLE
5384 022424 012777 002041 157556 MOV #2041,@BE1CR1 ; . DATI FROM LOW MAP REGISTER
5385 022432 105777 157552 3$: TSTB @BE1CR1 ; . DONE BIT SET?
5386 022436 100375 BPL 3$ ; . IF NOT, WAIT
5387 022440 022777 125252 157534 CMP #125252,@BE10B ; . READ OK?
5388 022446 001401 BEQ 4$ ; . IF SO, BRANCH
5389 022450 104033 ERROR +33 ; . ERROR IN WRITING AND READING MAP REGISTERS
5390 022452 012777 177777 157524 4$: MOV #-1,@BE1CC ; . NOW DO HIGH MAP REGISTER
5391 022460 012777 002041 157522 MOV #2041,@BE1CR1 ; . DATI FROM HIGH REGISTER
5392 022466 105777 157516 5$: TSTB @BE1CR1 ; . DONE BIT SET?
5393 022472 100375 BPL 5$ ; . IF NOT, WAIT
5394 022474 022777 000052 157500 CMP #52,@BE10B ; . READ OK?
5395 022502 001401 BEQ 6$ ; . IF SO, BRANCH
5396 022504 104033 ERROR +33 ; . ERROR IN WRITING AND READING MAP REGISTERS
5397 022506 005021 6$: CLR (R1)+ ; . CLEAR REGISTER JUST WRITTEN
5398 022510 005021 CLR (R1)+ ;
5399 022512 022701 170376 CMP #MAPH37,R1 ; . ALL DONE?
5400 022516 101337 BHI 2$ ; . IF NOT, BRANCH

```


TEST UNIBUS DEVICE DATI CYCLE WITH CACHE ENABLED

```

5402 .SBTTL TEST UNIBUS DEVICE DATI CYCLE WITH CACHE ENABLED
5403
5404 ;* THIS TEST WILL SEE IF THE CACHE OPERATES CORRECTLY
5405 ;* WHEN A UNIBUS DEVICE READ CYCLE IS DONE
5406 ;*
5407 ;
5408 ; BGNTST
5409 ;
5410 ; ENABLE THE CACHE
5411 ; ENABLE UNIBUS MAPPING
5412 ;*
5413 ;* CHECK INITIAL SETTING OF THE CACHE VALID BITS
5414 ;*
5415 ; SELECT THE CACHE VALID BITS
5416 ; IF KMCR <15-9> NEQ †B0000000 THEN
5417 ; . ERROR IN THE CACHE
5418 ; ENDF
5419 ;*
5420 ;* CHECK INITIAL SETTING OF THE CACHE AVAILABILITY BITS
5421 ;*
5422 ; SELECT THE CACHE AVAILABLE SET BITS
5423 ; IF KMCR <14-9> NEQ †B1111111 THEN
5424 ; . ERROR IN THE CACHE
5425 ; ENDF
5426 ;*
5427 ;* GO DO TWO CONSECUTIVE READS
5428 ;* THE FIRST WILL ALLOCATE SET A
5429 ;* THE SECOND WILL CAUSE A CACHE HIT IN SET A
5430 ;*
5431 ; LET MAP REGISTER 1 POINT TO LOCATION 2000
5432 ; LET BE1BA := 20000 ;OCTAL BOUNDARY READ
5433 ; LET BE1CR2 := 0 ;CLEAR UPPER 2 ADDRESS BITS
5434 ; LET BE1CC := -2 ;SET FOR TWO TRANSFERS
5435 ; SET UBE FOR NPR-DATI-1 XFER
5436 ; SET OFF THE TRANSFER
5437 ; WAIT FOR THE TRANSFER TO COMPLETE
5438 ; DISABLE UNIBUS MAPPING
5439 ; TURN OFF THE CACHE
5440 ; SELECT THE CACHE VALID BITS
5441 ;*
5442 ;* CHECK IF THE CACHE OPERATED CORRECTLY
5443 ;*
5444 ; IF KMCR <15-9> NEQ †B1001000 THEN
5445 ; . ERROR CACHE DID NOT GET UPDATED CORRECTLY
5446 ; ENDF
5447 ; SELECT THE CACHE AVAILABILITY BITS
5448 ; IF KMCR <15-9> NEQ †B1000111 THEN
5449 ; . ERROR CACHE DID NOT OPERATE CORRECTLY
5450 ; ENDF
5451 ;
5452 ; ENDTST
5453 ;
5454 ;-----
5455 ;*****
5456 ;*TEST 65 UNIBUS DEVICE CYCLE WITH CACHE ENABLED
;*****

```

T65 UNIBUS DEVICE CYCLF WITH CACHE ENABLED

```

022520 000004          TST65: SCOPE
5457
5458 022522 004737 002502      JSR    PC,IUBE      ; INITIALIZE THE UBE
5459 022526 052737 000100 177734  BIS    #BIT6,KMCR   ; ENABLE THE DMA CACHE
5460 022534 032737 000100 177734  BIT    #BIT6,KMCR   ; IS THE CACHE PRESENT ?
5461 022542 001002          BNE    1#          ; IF YES, CONTINUE WITH TEST
5462 022544 000137 023122      JMP    UDEM        ; OTHERWISE, DO END OF PASS
5463 022550 012737 160000 170374 1# :  MOV    #160000,MAPL37 ; INITIALIZE MAP REGISTERS
5464 022556 012737 000077 170376      MOV    #77,MAPH37  ;
5465 022564 005037 170200      CLR    MAPL00      ;
5466 022570 005037 170202      CLR    MAPH00      ;
5467 022574 052737 000040 172516  BIS    #BIT5,M#R3   ; ENABLE UNIBUS MAPPING
5468 022602 052737 000400 177730  BIS    #BIT08,DCSR  ; ENABLE DIAGNOSTIC MODE
5469 022610 042737 000400 177734  BIC    #BIT08,KMCR  ; MAKE SURE THAT VALID BITS SET
5470 022616 042737 000400 177730  BIC    #BIT08,DCSR  ; DISABLE DIAGNOSTIC MODE
5471 022624          5# :
5472          ;
5473          ; GO DO TWO CONSECUTIVE READS STARTING AT A OCTAL BOUNDARY.
5474          ; THE FIRST READ WILL ALLOCATE CACHE SET A. THE SECOND READ
5475          ; WILL CAUSE A CACHE HIT IN SET A
5476          ;
5477 022624 012737 000200 170204 10# :  MOV    #200,MAPL01   ; POINT MAP REGISTER TO LOCATION 200
5478 022632 005037 170206      CLR    MAPH01      ;
5479 022636 012777 020000 157342      MOV    #20000,#BE1BA ; SET TRANSFER ADDRESS TO OCTAL BOUNDARY
5480 022644 012777 000000 157342      MOV    #0,#BE1CR2  ; CLEAR UPPER 2 ADDRESS BITS
5481 022652 012777 177776 157324      MOV    #-2,#BE1CC  ; SET UBE FOR TWO TRANSFERS
5482          ;
5483          ; GO DO THE TRANSFER
5484          ;
5485 022660 012777 002041 157322      MOV    #2041,#BE1CR1 ; SET UBE FOR NPR-DATI-1 XFER
5486 022666 032777 000200 157314 15# :  BIT    #BIT17,#BE1CR1 ; IS THE TRANSFER COMPLETE ?
5487 022674 001774          BEQ    15#          ; NO, THEN WAIT FOR IT TO BE COMPLETE
5488          ;
5489          ; CHECK THE VALID BITS IN THE KMCR
5490          ;
5491 022676 013703 177734          MOV    KMCR,R3      ; PUT KMCR INTO R3 FOR MASKING
5492 022702 042703 000777          BIC    #777,R3      ; MASK OUT BITS 0-8
5493 022706 022703 110000          CMP    #110000,R3   ; ARE THE BITS SET CORRECTLY ?
5494 022712 001401          BEQ    20#          ; YES, THEN GO CHECK THE AVAILABILITY BITS
5495 022714 104033          ERROR  +33        ; NO, THEN ERROR IN DMA CACHE
5496 022716 005037 172516 20# :  CLR    M#R3        ; DISABLE MAPPING
5497 022722 042737 000100 177734  BIC    #BIT06,KMCR  ; AND CACHE
5498 022730 023777 000202 157244  CMP    #202,#BE1DB  ; DATA READ OK?
5499 022736 001401          BEQ    TST66        ; IF OK, EXIT TEST
5500 022740 104033          ERROR  +33        ; OTHERWISE ERROR IN CACHE
5501

```

TEST - WRONG PARITY AND CACHE

5503
5504
5505
5506
5507
5508
5509
5510
5511
5512
5513
5514
5515
5516
5517
5518
5519

```
.SBTTL TEST - WRONG PARITY AND CACHE
; * THIS TEST VERIFIES THAT IF DATA RECIEVED HAS BAD PARITY IT DOES NOT
; * GET ALLOCATED IN CACHE.
;
;   BGNTST
;
;   WRITE LOCATION WITH BAD PARITY
;   USING UBE DO DATI FROM THAT LOCATION
;   IF KMCR BITS ARE NOT IN RESET CONDITION THEN
;   . ERROR
;   ENOIF
;
;   ENDTST
```

```
-----
;*****
; *TEST 66      WRONG PARITY AND CACHE
;*****
TST66: SCOPE
```

022742 000004

```
5520
5521 022744 004737 002502      JSR    PC,IUBE      ; CLEAR UBE
5522 022750 052737 000005 172100  BIS    #BIT02,BIT00,#172100 ; SET WRONG PARITY IN MEMORY CSR
5523 022756 042737 003740 172100  BIC    #3740,#172100   ; CLEAR CHECK BITS
5524 022764 005037 000000      CLR    #0             ; WRITE LOCATION 0 WITH WPONG PARITY
5525 022770 042737 000004 172100  BIC    #BIT02,#172100   ; CLEAR WRONG PARITY
5526 022776 052737 000040 172516  BIS    #BIT05,MMR3     ; ENABLE MEMORY MAPPING
5527 023004 042737 000100 177734  BIC    #BIT06,KMCR     ; DISABLE JUST TO MAKE SURE
5528 023012 052737 000100 177734  BIS    #BIT06,KMCR     ; ENABLE CACHE
5529 023020 012737 000000 170204  MOV    #0,MAPLO1      ; POINT MAP REGISTER TO LOCATION 200
5530 023026 005037 170206      CLR    MAPH01
5531 023032 012777 020000 157146  MOV    #20000,#BE18A  ; SET TRANSFER ADDRESS TO OCTAL BOUNDARY
5532 023040 012777 000000 157146  MOV    #0,#BE1CR2    ; CLEAR UPPER 2 ADDRESS BITS
5533 023046 012777 177777 157130  MOV    #-1,#BE1CC    ; SET UBE FOR TWO TRANSFERS
5534
5535 ; GO DO THE TRANSFER
5536 ;
5537 023054 012777 002041 157126  MOV    #2041,#BE1CR1  ; SET UBE FOR NPR-DATI-1 XFER
5538 023062 032777 000200 157120 15#: BIT    #BIT7,#BE1CR1  ; IS THE TRANSFER COMPLETE ?
5539 023070 001774      BEQ    15#           ; NO, THEN WAIT FOR IT TO BE COMPLETE
5540 023072 032737 177000 177734  BIT    #177000,KMCR   ; WAS ALLOCATED?
5541 023100 001401      BEQ    16#           ; IF NOT, BRANCH
5542 023102 104032      ERROR  +32          ; IF WAS, REPORT ERROR
5543 023104 005037 000000 16#: CLR    #0             ; REWRITE 0 WITH RIGHT PARITY
5544 023110 005037 172516      CLR    MMR3          ; CLEAR MAPPING ENABLED
5545 023114 042737 000001 172100  BIC    #BIT00,#172100
5546
5547
5548 023122
```

```
UBEM:
;*****
; *TEST 67      DUMMY TEST
;*****
TST67: SCOPE
```

```
5549 023122 000004      TST    $PASS        ; FIRST PASS?
5550 023124 005737 001206      BNE    $EOP         ; IF NOT, GOTO $EOP
5551 023130 001015      TST    PMIS         ; IF NO PMI MEMORY
5552 023132 005737 001720      BEQ    $EOP         ; PRINT MESSAGE EVERY PASS
```

T67 DUMMY TEST

```

5553 023140 012737 000024 023220      MOV    #24,$ENDCT      ; PRINT EVERY 20 PASSES
5554 023146 032737 000017 177734      BIT    #17,@#KMCR
5555 023154 001403                BEQ    $EOP            ; ONLY IF NO UNIBUS MEMORY
5556 023156 012737 000012 023220      MOV    #12,$ENDCT      ; PRINT EVERY 10TH
5557
5558 ; END OF PASS ROUTINE
5559 ;
5560
5561

```

```

.SBTTL END OF PASS ROUTINE
;*****
;*INCREMENT THE PASS NUMBER ($PASS)
;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
;*IF THERES A MONITOR GO TO IT
;*IF THERE ISN'T JUMP TO RESTART
$EOP:

```

```

023164
023164 000004      SCOPE
023166 005037 001102      CLR    $TSTNM          ;;ZERO THE TEST NUMBER
023172 005037 001164      CLR    $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
023176 005237 001206      INC    $PASS           ;;INCREMENT THE PASS NUMBER
023202 042737 100000 001206      BIC    #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
023210 005327                DEC    (PC)+           ;;LOOP?
023212 000001      $EOPCT: .WORD 1
023214 003055      BGT    $DOAGN          ;;YES
023216 012737      MOV    (PC)+,@(PC)+   ;;RESTORE COUNTER
023220 000001      $ENDCT: .WORD 1
023222 023212      $EOPCT
023224 005737 002624      TST    UQUIET         ;NO MESSAGE IF RUNNING UFD QUIET MODE
023230 001037      BNE    1$
023232 104401 023240      TYPE   ,30009$        ;;TYPE ASCIZ STRING
023236 000407      BR    30008$         ;;GET OVER THE ASCIZ
;;30009$:
30008$:
023256 013746 001206      MOV    $PASS,-(SP)    ;;SAVE $PASS FOR TYPEOUT
;;TYPE PASS NUMBER
;;GO TYPE--DECIMAL ASCII WITH SIGN
023262 104405      TYPDS
023264 104401 023272      TYPE   ,30011$        ;;TYPE ASCIZ STRING
023270 000412      BR    30010$         ;;GET OVER THE ASCIZ
;;30011$:
30010$:
023316 013746 001112      MOV    $ERTTL,-(SP)   ;;SAVE $ERTTL FOR TYPEOUT
;;TOTAL NUMBER OF ERRORS
;;GO TYPE--DECIMAL ASCII WITH SIGN
023322 104405      TYPDS
023324 104401 001175      TYPE   , $CRLF        ;;TYPE CARRIAGE RETURN, LINE FEED
023330
1$:
023330 013700 000042      $GET42: MOV    @#42,RO  ;;GET MONITOR ADDRESS
023334 001405      BEQ    $DOAGN         ;;BRANCH IF NO MONITOR
023336 000005      RESET                ;;CLEAR THE WORLD
023340 004710      $ENDAD: JSR    PC,(RO) ;;GO TO MONITOR
023342 000240      NOP                   ;;SAVE ROOM
023344 000240      NOP                   ;;FOR
023346 000240      NOP                   ;;ACT11
023350
023350 000137      $DOAGN: JMP    @(PC)+         ;;RETURN
023352 003320      $RTNAD: .WORD  RESTART
023354 377 377 000 $ENULL: .BYTE  -1,-1,0    ;;NULL CHARACTER STRING
.EVEN

```

5562

END OF PASS ROUTINE

```

5563 ; SYSMAC ROUTINES
5564 ;
5565 ;
5566 ;SCOPE ENABLE HALT-ON-BREAK BETWEEN TESTS, SO APT CAN CATCH IT. DISABLE HOB
5567 ;AT END SO TESTS WILL NOT BE INTERRUPTED BY APT BREAKS WHICH CAUSES CERTAIN
5568 ;TESTS TO REPORT ERRORS WHEN THERE ARE REALLY NONE.
5569
5570 .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<5:0>
;CALL
;* SCOPE ;:SCOPE=IOT
$SCOPE:
023360 104407 CKSWR ;:TEST FOR CHANGE IN SOFT-SWR
023360 052737 001000 177520 BIS #BIT09,BCSR ;ENABLE HOB
023370 032777 040000 155542 1$: BIT #BIT14,@SWR ;:LOOP ON PRESENT TEST?
023376 001117 BNE $OVER ;:YES IF SW14=1
;####START OF CODE FOR THE XOR TESTER####
023400 000416 $XTSTR: BR 6$ ;:IF RUNNING ON THE "XOR" TESTER CHANGE
;THIS INSTRUCTION TO A "NOP" (NOP=240)
023402 013746 000004 MOV @ERRVEC,-(SP) ;:SAVE THE CONTENTS OF THE ERROR VECTOR
023406 012737 023426 000004 MOV #5,@ERRVEC ;:SET FOR TIMEOUT
023414 005737 177060 TST @177060 ;:TIME OUT ON XOR?
023420 012637 000004 MOV (SP)+,@ERRVEC ;:RESTORE THE ERROR VECTOR
023424 000466 BR $SVLAD ;:GO TO THE NEXT TEST
023426 022626 5$: CMP (SP)+,(SP)+ ;:CLEAR THE STACK AFTER A TIME OUT
023430 012637 000004 MOV (SP)+,@ERRVEC ;:RESTORE THE ERROR VECTOR
023434 000426 BR 7$ ;:LOOP ON THE PRESENT TEST
023436 6$:;####END OF CODE FOR THE XOR TESTER####
023436 032777 000400 155474 BIT #BIT08,@SWR ;:LOOP ON SPEC. TEST?
023444 001407 BEQ 2$ ;:BR IF NO
023446 017746 155466 MOV @SWR,-(SP) ;:SET DESIRED TEST NUM. FROM SWR
023452 042716 000300 BIC #SWRMK,(SP) ;:STRIP AWAY UNDESIRED BITS
023456 122637 001102 CMPB (SP)+,$TSTNM ;:ON THE RIGHT TEST?
023462 001465 BEQ $OVER ;:BR IF YES
023464 105737 001103 2$: TSTB $ERFLG ;:HAS AN ERROR OCCURRED?
023470 001421 BEQ 3$ ;:BR IF NO
023472 123737 001115 001103 CMPB $ERMAX,$ERFLG ;:MAX. ERRORS FOR THIS TEST OCCURRED?
023500 101015 BHI 3$ ;:BR IF NO
023502 032777 001000 155430 BIT #BIT09,@SWR ;:LOOP ON ERROR?
023510 001404 BEQ 4$ ;:BR IF NO
023512 013737 001110 001106 7$: MOV $LPERR,$LPADR ;:SET LOOP ADDRESS TO LAST SCOPE
023520 000446 BR $OVER
023522 105037 001103 4$: CLRB $ERFLG ;:ZERO THE ERROR FLAG
023526 005037 001164 CLR $TIMES ;:CLEAR THE NUMBER OF ITERATIONS TO MAKE
023532 000415 BR 1$ ;:ESCAPE TO THE NEXT TEST
023534 032777 004000 155376 3$: BIT #BIT11,@SWR ;:INHIBIT ITERATIONS?
023542 001011 BNE 1$ ;:BR IF YES
023544 005737 001206 TST $PASS ;:IF FIRST PASS OF PROGRAM
023550 001406 BEQ 1$ ;: INHIBIT ITERATIONS

```

SCOPE HANDLER ROUTINE

```

023552 005237 001104      INC      $ICNT      ;; INCREMENT ITERATION COUNT
023556 023737 001164 001104  CMP      $TIMES,$ICNT ;; CHECK THE NUMBER OF ITERATIONS MADE
023564 002024      BGE      $OVER      ;; BR IF MORE ITERATION REQUIRED
023566 012737 000001 001104 1$: MOV      @1,$ICNT   ;; REINITIALIZE THE ITERATION COUNTER
023574 013737 023710 001164  MOV      $MXCNT,$TIMES ;; SET NUMBER OF ITERATIONS TO DO
023602 105237 001102      $SVLAD: INCB     $TSTNM      ;; COUNT TEST NUMBERS
023606 113737 001102 001204  MOVB     $TSTNM,$TESTN ;; SET TEST NUMBER IN APT MAILBOX
023614 011637 001106      MOV      (SP),$LPADR  ;; SAVE SCOPE LOOP ADDRESS
023620 011637 001110      MOV      (SP),$LPERR  ;; SAVE ERROR LOOP ADDRESS
023624 005037 001166      CLR      $ESCAPE     ;; CLEAR THE ESCAPE FROM ERROR ADDRESS
023630 112737 000001 001115  MOVB     @1,$ERMAX    ;; ONLY ALLOW ONE(1) ERROR ON NEXT TEST
023636 013777 001102 155276 $OVER: MOV      $TSTNM,@DISPLAY ;; DISPLAY TEST NUMBER
023644 013716 001106      MOV      $LPADR,(SP) ;; FUDGE RETURN ADDRESS
023650 123727 001102 000022  CMPB     $TSTNM,@18.
023656 103413      BLO     1$
023660 123727 001102 000033  CMPB     $TSTNM,@27.
023666 103404      BLO     2$
023670 123727 001102 000065  CMPB     $TSTNM,@53.
023676 103403      BLO     1$
023700 042737 001000 177520 2$: BIC      @1000,@#BCSR ;; TESTS 18-26 AND 53-54 REQUIRE HALT-ON BREAK TO
;; BE DISABLED TO PREVENT ERRONEOUS ERROR REPORTS

023706 000002      1$: RTI
023710 000001      $MXCNT: 1      ;; MAX. NUMBER OF ITERATIONS

```

5571

.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
;*

```

```

023712 105737 001157      $TYPE: TSTB     $TPFLG      ;; IS THERE A TERMINAL?
023716 100002      BPL      1$          ;; BR IF YES
023720 000000      HALT     ;; HALT HERE IF NO TERMINAL
023722 000430      BR      3$          ;; LEAVE
023724 010046      1$: MOV      RO,-(SP)   ;; SAVE RO
023726 017600 000002      MOV      @2(SP),RO    ;; GET ADDRESS OF ASCIZ STRING
023732 122737 000001 001220  CMPB     @APTENV,$ENV  ;; RUNNING IN APT MODE
023740 001011      BNE     62$         ;; NO,GO CHECK FOR APT CONSOLE
023742 132737 000100 001221  BITB     @APTPOOL,$ENVM ;; SPOOL MESSAGE TO APT
023750 001405      BEQ     62$         ;; NO,GO CHECK FOR CONSOLE
023752 010037 023762      MOV      RO,61$      ;; SETUP MESSAGE ADDRESS FOR APT
023756 004737 026376      JSR     PC,$ATY3    ;; SPOOL MESSAGE TO APT
023762 000000      61$: .WORD     0      ;; MESSAGE ADDRESS
023764 132737 000040 001221  62$: BITB     @APTCSUP,$ENVM ;; APT CONSOLE SUPPRESSED
023772 001003      BNE     60$         ;; YES,SKIP TYPE OUT
023774 112046      2$: MOVB     (RO)+,-(SP) ;; PUSH CHARACTER TO BE TYPED ONTO STACK
023776 001005      BNE     4$          ;; BR IF IT ISN'T THE TERMINATOR
024000 005726      TST     (SP)+       ;; IF TERMINATOR POP IT OFF THE STACK
024002 012600      60$: MOV      (SP)+,RO  ;; RESTORE RO

```

TYPE ROUTINE

```

024004 062716 000002      3$:  ADD      #2,(SP)      ;;ADJUST RETURN PC
024010 000002              RTI              ;;RETURN
024012 122716 000011      4$:  CMPB     #HT,(SP)      ;;BRANCH IF <HT>
024016 001430              BEQ      8$
024020 122716 000200      CMPB     #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
024024 001006              BNE      5$
024026 005726              TST      (SP)+          ;;POP <CR><LF> EQUIV
024030 104401              TYPE              ;;TYPE A CR AND LF
024032 001175              $CRLF
024034 105037 024242      CLRB     $CHARCNT      ;;CLEAR CHARACTER COUNT
024040 000755              BR       2$            ;;GET NEXT CHARACTER
024042 004737 024124      5$:  JSR      PC,$TYPEPC   ;;GO TYPE THIS CHARACTER
024046 123726 001156      6$:  CMPB     $FILLC,(SP)+  ;;IS IT TIME FOR FILLER CHARS.?
024052 001350              BNE      2$            ;;IF NO GO GET NEXT CHAR.
024054 013746 001154      MOV      $NULL,-(SP)    ;;GET # OF FILLER CHARS. NEEDED
                                ;;AND THE NULL CHAR.
024060 105366 000001      7$:  DECB     1(SP)        ;;DOES A NULL NEED TO BE TYPED?
024064 002770              BLT      6$            ;;BR IF NO--GO POP THE NULL OFF OF STACK
024066 004737 024124      JSR      PC,$TYPEPC   ;;GO TYPE A NULL
024072 105337 024242      DECB     $CHARCNT      ;;DO NOT COUNT AS A COUNT
024076 000770              BR       7$            ;;LOOP
                                ;HORIZONTAL TAB PROCESSOR
024100 112716 000040      8$:  MOVB     #' ,(SP)     ;;REPLACE TAB WITH SPACE
024104 004737 024124      9$:  JSR      PC,$TYPEPC   ;;TYPE A SPACE
024110 132737 000007 024242  BITB     #7,$CHARCNT    ;;BRANCH IF NOT AT
024116 001372              BNE      9$            ;;TAB STOP
024120 005726              TST      (SP)+          ;;POP SPACE OFF STACK
024122 000724              BR       2$            ;;GET NEXT CHARACTER
024124              $TYPEPC:
024124 105777 155014              TSTB     @#TKS          ;;CHAR IN KYBD BUFFER?
024130 100022              BPL      10$           ;;BR IF NOT
024132 017746 155010              MOV      @#TKB,-(SP)   ;;GET CHAR
024136 042716 177600              BIC      #177600,(SP)  ;;STRIP EXTRANEIOUS BITS
024142 122716 000023              CMPB     #XOFF,(SP)    ;;WAS CHAR XOFF
024146 001012              BNE      102$          ;;BR IF NOT
024150              101$:
024150 105777 154770              TSTB     @#TKS          ;;WAIT FOR CHAR
024154 100375              BPL      101$          ;;
024156 117716 154764              MOVB     @#TKB,(SP)    ;;GET CHAR
024162 042716 177600              BIC      #177600,(SP)  ;;STRIP IT
024166 122716 000021              CMPB     #XON,(SP)     ;;WAS IT XON?
024172 001366              BNE      101$          ;;BR IF NOT
024174              102$:
024174 005726              TST      (SP)+          ;;FIX STACK
024176              10$:
024176 105777 154746              TSTB     @#TPS          ;;WAIT UNTIL PRINTER IS READY
024202 100375              BPL      10$           ;;
024204 116677 000002 154740  MOVB     2(SP),@#TPB    ;;LOAD CHAR TO BE TYPED INTO DATA REG.
024212 122766 000015 000002  CMPB     #CR,2(SP)     ;;IS CHARACTER A CARRIAGE RETURN?
024220 001003              BNE      1$            ;;BRANCH IF NO
024222 105037 024242      CLRB     $CHARCNT      ;;YES--CLEAR CHARACTER COUNT
024226 000406              BR       $TYPEPC       ;;EXIT
024230 122766 000012 000002  1$:  CMPB     #LF,2(SP)     ;;IS CHARACTER A LINE FEED?
024236 001402              BEQ      $TYPEPC       ;;BRANCH IF YES
024240 105227              INCB     (PC)+          ;;COUNT THE CHARACTER
024242 000000              $CHARCNT: .WORD      0  ;;CHARACTER COUNT STORAGE
024244 000207              $TYPEPC: RTS          PC

```

BINARY TO OCTAL (ASCII) AND TYPE

5572

```

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

```

```

024246 017646 000000
024252 116637 000001 024471
024260 112637 024473
024264 062716 000002
024270 000406
024272 112737 000001 024471
024300 112737 000006 024473
024306 112737 000005 024470
024314 010346
024316 010446
024320 010546
024322 113704 024473
024326 005404
024330 062704 000006
024334 110437 024472
024340 113704 024471
024344 016605 000012
024350 005003
024352 006105 1$:
024354 000404 BR 3$
024356 006105 2$:
024360 006105 ROL R5
024362 006105 ROL R5
024364 010503 MOV R5, R3
024366 006103 3$:
024370 105337 024472 DECB $OMODE
024374 100016 BPL 7$
024376 042703 177770 BIC #177770, R3
024402 001002 BNE 4$
024404 005704 TST R4
024406 001403 BEQ 5$
024410 005204 4$:
024412 052703 000060 BIS #'0, R3
024416 052703 000040 5$:

```


BINARY TO OCTAL (ASCII) AND TYPE

```

024422 110337 024466          MOV      R3,R4          ;;SAVE FOR TYPING
024426 104401 024466          TYPE      .R4          ;;GO TYPE THIS DIGIT
024432 105337 024470          7$:    DEC      $OCNT          ;;COUNT BY 1
024436 003347                BGT      2$          ;;BR IF MORE TO DO
024440 002402                BLT      6$          ;;BR IF DONE
024442 005204                INC      R4          ;;INSURE LAST DIGIT ISN'T A BLANK
024444 000744                BR       2$          ;;GO DO THE LAST DIGIT
024446 012605                6$:    MOV      (SP)+,R5          ;;RESTORE R5
024450 012604                MOV      (SP)+,R4          ;;RESTORE R4
024452 012603                MOV      (SP)+,R3          ;;RESTORE R3
024454 016666 000002 000004    MOV      2(SP),4(SP)      ;;SET THE STACK FOR RETURNING
024462 012616                MOV      (SP)+,(SP)
024464 000002                RTI                      ;;RETURN
024466 000                8$:    .BYTE    0          ;;STORAGE FOR ASCII DIGIT
024467 000                .BYTE    0          ;;TERMINATOR FOR TYPE ROUTINE
024470 000                $OCNT:  .BYTE    0          ;;OCTAL DIGIT COUNTER
024471 000                $OFILL: .BYTE    0          ;;ZERO FILL SWITCH
024472 000000                $OMODE: .WORD    0          ;;NUMBER OF DIGITS TO TYPE
5573                .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
;*****
;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
;*REPLACED WITH SPACES.
;*CALL:
;*      MOV      NUM,-(SP)          ;;PUT THE BINARY NUMBER ON THE STACK
;*      TYPDS          ;;GO TO THE ROUTINE
$TYPDS:
024474                MOV      R0,-(SP)          ;;PUSH R0 ON STACK
024474 010046                MOV      R1,-(SP)          ;;PUSH R1 ON STACK
024476 010146                MOV      R2,-(SP)          ;;PUSH R2 ON STACK
024500 010246                MOV      R3,-(SP)          ;;PUSH R3 ON STACK
024502 010346                MOV      R5,-(SP)          ;;PUSH R5 ON STACK
024504 010546                MOV      $20200,-(SP)      ;;SET BLANK SWITCH AND SIGN
024506 012746 020200        MOV      20(SP),R5          ;;GET THE INPUT NUMBER
024512 016605 000020        BPL      1$          ;;BR IF INPUT IS POS.
024516 100004                NEG      R5          ;;MAKE THE BINARY NUMBER POS.
024520 005405                MOV      #'-,1(SP)        ;;MAKE THE ASCII NUMBER NEG.
024522 112766 000055 000001 1$:    CLR      R0          ;;ZERO THE CONSTANTS INDEX
024530 005000                MOV      $#DBLK,R3        ;;SETUP THE OUTPUT POINTER
024532 012703 024710        MOV      #' ,(R3)+        ;;SET THE FIRST CHARACTER TO A BLANK
024536 112723 000040        2$:    CLR      R2          ;;CLEAR THE BCD NUMBER
024542 005002                MOV      $DTBL(R0),R1     ;;GET THE CONSTANT
024544 016001 024700        3$:    SUB      R1,R5          ;;FORM THIS BCD DIGIT
024550 160105                BLT      4$          ;;BR IF DONE
024552 002402                INC      R2          ;;INCREASE THE BCD DIGIT BY 1
024554 005202                BR       3$
024556 000774                4$:    ADD      R1,R5          ;;ADD BACK THE CONSTANT
024560 060105                TST      R2          ;;CHECK IF BCD DIGIT=0
024562 005702                BNE      5$          ;;FALL THROUGH IF 0
024564 001002                TSTB     (SP)          ;;STILL DOING LEADING 0'S?
024566 105716                BMI      7$          ;;BR IF YES
024570 100407                5$:    ASLB     (SP)          ;;MSD?
024572 106316                BCC      6$          ;;BR IF NO
024574 103003                MOV      1(SP),-1(R3)     ;;YES--SET THE SIGN
024576 116663 000001 177777 6$:    BIS      #'0,R2          ;;MAKE THE BCD DIGIT ASCII
024604 052702 000060

```

CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```

024610 052702 000040      7$:  BIS      #' ,R2      ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
024614 110223          MOVB     P2,(R3)+   ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
024616 005720          TST      (R0)+     ;;JUST INCREMENTING
024620 020027 000010          CMP      R0,#10    ;;CHECK THE TABLE INDEX
024624 002746          BLT      2$      ;;GO DO THE NEXT DIGIT
024626 003002          BGT      8$      ;;GO TO EXIT
024630 010502          MOV      R5,R2     ;;GET THE LSD
024632 000764          BR       6$      ;;GO CHANGE TO ASCII
024634 105726          8$:  TSTB     (SP)+     ;;WAS THE LSD THE FIRST NON-ZERO?
024636 100003          BPL      9$      ;;BR IF NO
024640 116663 177777 177776  MOVB     -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
024646 105013          9$:  CLRB     (R3)      ;;SET THE TERMINATOR
024650 012605          MOV      (SP)+,R5   ;;POP STACK INTO R5
024652 012603          MOV      (SP)+,R3   ;;POP STACK INTO R3
024654 012602          MOV      (SP)+,R2   ;;POP STACK INTO R2
024656 012601          MOV      (SP)+,R1   ;;POP STACK INTO R1
024660 012600          MOV      (SP)+,R0   ;;POP STACK INTO R0
024662 104401 024710          TYPE     , $DBLK   ;;NOW TYPE THE NUMBER
024666 016666 000002 000004  MOV      2(SP),4(SP)  ;;ADJUST THE STACK
024674 012616          MOV      (SP)+,(SP)
024676 000002          RTI      ;;RETURN TO USER
024700 023420          $DTBL: 10000.
024702 001750          1000.
024704 000144          100.
024706 000012          10.
024710          $DBLK: .BLKW 4
5574          .SBTTL  TTY INPUT ROUTINE
          ;;*****
          .ENABL  LSB
          ;;*****
          ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
          ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
          ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
          ;*WHEN OPERATING IN TTY FLAG MODE.
024720 022737 000176 001140 $CKSWR: CMP      $SWREG,SWR   ;;IS THE SOFT-SWR SELECTED?
024726 001074          BNE      15$      ;;BRANCH IF NO
024730 105777 154210          TSTB     @TKS      ;;CHAR THERE?
024734 100071          BPL      15$      ;;IF NO, DON'T WAIT AROUND
024736 117746 154204          MOVB     @TKB,-(SP)  ;;SAVE THE CHAR
024742 042716 177600          BIC      @C177,(SP) ;;STRIP-OFF THE ASCII
024746 022726 000007          CMP      @7,(SP)+   ;;IS IT A CONTROL G?
024752 001062          BNE      15$      ;;NO, RETURN TO USER
024754 123727 001134 000001  CMPB     $AUTOB,#1   ;;ARE WE RUNNING IN AUTO-MODE?
024762 001456          BEQ      15$      ;;BRANCH IF YES
024764 104401 025455          TYPE     , $CNTLG  ;;ECHO THE CONTROL-G (+G)
024770 104401 025462          $GTSWR: TYPE     , $MSWR  ;;TYPE CURRENT CONTENTS
024774 013746 000176          MOV      SWREG,-(SP) ;;SAVE SWREG FOR TYPEOUT
025000 104402          TYPOC   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
025002 104401 025473          TYPE     , $MNEW   ;;PROMPT FOR NEW SWR
025006 005046          19$:  CLR      -(SP)   ;;CLEAR COUNTER
025010 005046          CLR      -(SP)   ;;THE NEW SWR
025012 105777 154126          7$:  TSTB     @TKS      ;;CHAR THERE?
025016 100375          BPL      7$      ;;IF NOT TRY AGAIN
025020 117746 154122          MOVB     @TKB,-(SP)  ;;PICK UP CHAR
025024 042716 177600          BIC      @C177,(SP)  ;;MAKE IT 7-BIT ASCII
025030 021627 000025          9$:  CMP      (SP),#25  ;;IS IT A CONTROL-U?
025034 001005          BNE      10$      ;;BRANCH IF NOT

```

TTY INPUT ROUTINE

```

025036 104401 025450          TYPE      , $CNTLU          ;; YES, ECHO CONTROL-U (+U)
025042 062706 000006          20$: ADD      #6,SP          ;; IGNORE PREVIOUS INPUT
025046 000757          BR          19$          ;; LET'S TRY IT AGAIN
025050 021627 000015          10$: CMP      (SP),#15          ;; IS IT A <CR>?
025054 001022          BNE      16$          ;; BRANCH IF NO
025056 005766 000004          TST      4(SP)          ;; YES, IS IT THE FIRST CHAR?
025062 001403          BEQ      11$          ;; BRANCH IF YES
025061 016677 000002 154046          MOV      2(SP),@SWR          ;; SAVE NEW SWR
025062 062706 000006          11$: ADD      #6,SP          ;; CLEAR UP STACK
025065 104401 001175          14$: TYPE      , $CRLF          ;; ECHO <CR> AND <LF>
025066 123727 001135 000001          CMPB     $INTAG,#1          ;; RE-ENABLE TTY KBD INTERRUPTS?
025110 001003          BNE      15$          ;; BRANCH IF NOT
025112 012777 000100 154024          MOV      #100,@TKS          ;; RE-ENABLE TTY KBD INTERRUPTS
025120 000002          15$: RTI          ;; RETURN
025122 004737 024124          16$: JSR      PC,$TYPEC          ;; ECHO CHAR
025126 021627 000060          CMP      (SP),#60          ;; CHAR < 0?
025132 002420          BLT      18$          ;; BRANCH IF YES
025134 021627 000067          CMP      (SP),#67          ;; CHAR > 7?
025140 003015          BGT      18$          ;; BRANCH IF YES
025142 042726 000060          BIC      #60,(SP)          ;; STRIP-OFF ASCII
025146 005766 000002          TST      2(SP)          ;; IS THIS THE FIRST CHAR
025152 001403          BEQ      17$          ;; BRANCH IF YES
025154 006316          ASL      (SP)          ;; NO, SHIFT PRESENT
025156 006316          ASL      (SP)          ;; CHAR OVER TO MAKE
025160 006316          ASL      (SP)          ;; ROOM FOR NEW ONE.
025162 005266 000002          17$: INC      2(SP)          ;; KEEP COUNT OF CHAR
025166 056616 177776          BIS      -2(SP),(SP)          ;; SET IN NEW CHAR
025172 000707          BR          7$          ;; GET THE NEXT ONE
025174 104401 001174          18$: TYPE      , $QUES          ;; TYPE ?<CR><LF>
025200 000720          BR          20$          ;; SIMULATE CONTROL-U
.DSABL  LSB
;*****
;THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
;*CALL:
;* RDCHR          ;; INPUT A SINGLE CHARACTER FROM THE TTY
;* RETURN HERE   ;; CHARACTER IS ON THE STACK
;*              ;; WITH PARITY BIT STRIPPED OFF
;
$RDCHR: MOV      (SP),-(SP)          ;; PUSH DOWN THE PC
025204 016666 000004 000002          MOV      4(SP),2(SP)          ;; SAVE THE PS
025212 105777 153726          1$: TSTB     @TKS          ;; WAIT FOR
025216 100375          BPL      1$          ;; A CHARACTER
025220 117766 153722 000004          MOVB     @TKB,4(SP)          ;; READ THE TTY
025226 042766 177600 000004          BIC      #C<177>,4(SP)          ;; GET RID OF JUNK IF ANY
025234 026627 000004 000023          CMP      4(SP),#23          ;; IS IT A CONTROL-S?
025242 001013          BNE      3$          ;; BRANCH IF NO
025244 105777 153674          2$: TSTB     @TKS          ;; WAIT FOR A CHARACTER
025250 100375          BPL      2$          ;; LOOP UNTIL ITS THERE
025252 117746 153670          MOVB     @TKB,-(SP)          ;; GET CHARACTER
025256 042716 177600          BIC      #C177,(SP)          ;; MAKE IT 7-BIT ASCII
025262 022627 000021          CMP      (SP),#21          ;; IS IT A CONTROL-Q?
025266 001366          BNE      2$          ;; IF NOT DISCARD IT
025270 000750          BR          1$          ;; YES, RESUME
025272 026627 000004 000021 3$: CMP      4(SP),#$XON          ;; IS IT A RANDOM XON?          ;:RAN001
025300 001744          BEQ      1$          ;; BRANCH IF YES          ;:RAN001
025302 026627 000004 000140          CMP      4(SP),#140          ;; IS IT UPPER CASE?
025310 002407          BLT      4$          ;; BRANCH IF YES

```

TTY INPUT ROUTINE

```

025312 026627 000004 000175      CMP      4(SP),#175      ;;IS IT A SPECIAL CHAR?
025320 003003                    BGT      4$              ;;BRANCH IF YES
025322 042766 000040 000004      BIC      #40,4(SP)      ;;MAKE IT UPPER CASE
025330 000002                    4$:      RTI              ;;GO BACK TO USER
;*****
;THIS ROUTINE WILL INPUT A STRING FROM THE TTY
;CALL:
;*      RDLIN              ;;INPUT A STRING FROM THE TTY
;*      RETURN HERE       ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
;*                          ;;TERMINATOR WILL BE A BYTE OF ALL O'S
025332 010346                    $RDLIN: MOV      R3,-(SP)    ;;SAVE R3
025334 012703 025440              1$:      MOV      #TTYIN,R3    ;;GET ADDRESS
025340 022703 025450              2$:      CMP      #TTYIN+8.,R3  ;;BUFFER FULL?
025344 101405                    BLOS     4$              ;;BR IF YES
025346 104410                    RDCHR     ;;GO READ ONE CHARACTER FROM THE TTY
025350 112613                    MOVB     (SP)+,(R3)      ;;GET CHARACTER
025352 122713 000177              10$:    CMPB     #177,(R3)     ;;IS IT A RUBOUT
025356 001003                    BNE     3$              ;;SKIP IF NOT
025360 104401 001174              4$:      TYPE     , $QUES    ;;TYPE A '?'
025364 000763                    BR       1$              ;;CLEAR THE BUFFER AND LOOP
025366 111337 025436              3$:      MOVB     (R3),9$      ;;ECHO THE CHARACTER
025372 104401 025436              TYPE     ,9$
025376 122723 000015              CMPB     #15,(R3)+     ;;CHECK FOR RETURN
025402 001356                    BNE     2$              ;;LOOP IF NOT RETURN
025404 105063 177777              CLRB     -1(R3)        ;;CLEAR RETURN (THE 15)
025410 104401 001176              TYPE     , $LF         ;;TYPE A LINE FEED
025414 012603                    MOV      (SP)+,R3      ;;RESTORE R3
025416 011646                    MOV      (SP)-,(SP)    ;;ADJUST THE STACK AND PUT ADDRESS OF THE
025420 016666 000004 000002      MOV      4(SP),2(SP)  ;; FIRST ASCII CHARACTER ON IT
025426 012766 025440 000004      MOV      #TTYIN,4(SP)
025434 000002                    RTI              ;;RETURN
025436 000                      9$:      .BYTE     0              ;;STORAGE FOR ASCII CHAR. TO TYPE
025437 000                      .BYTE     0              ;;TERMINATOR
025440                    $TTYIN: .BLKB     8.      ;;RESERVE 8 BYTES FOR TTY INPUT
025450 136 125 015              $CNTLU: .ASCIZ /↑U/<15><12> ;;CONTROL "U"
025453 012 000                    $CNTLG: .ASCIZ /↑G/<15><12> ;;CONTROL "G"
025455 136 107 015
025460 012 000                    $MSWR: .ASCIZ <15><12>/SWR = /
025462 015 012 123
025465 127 122 040
025470 075 040 000
025473 040 040 116              $MNEW: .ASCIZ / NEW = /
025476 105 127 040
025501 075 040 000

```

5575

```

.SBTTL READ AN OCTAL NUMBER FROM THE TTY
;*****
;THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
;CHANGE IT TO BINARY.
;CALL:

```

```

;*      RDOCT              ;;READ AN OCTAL NUMBER
;*      RETURN HERE       ;;LOW ORDER BITS ARE ON TOP OF THE STACK
;*                          ;;HIGH ORDER BITS ARE IN $HIOCT
025504 011646                    $RDOCT: MOV      (SP)-,(SP)  ;;PROVIDE SPACE FOR THE
025506 016666 000004 000002      MOV      4(SP),2(SP)  ;;INPUT NUMBER
025514 010046                    MOV      R0,(SP)      ;;PUSH R0 ON STACK
025516 010146                    MOV      R1,-(SP)     ;;PUSH R1 ON STACK
025520 010246                    MOV      R2,(SP)     ;;PUSH R2 ON STACK

```

READ AN OCTAL NUMBER FROM THE TTY

```

025522 104411          1$:  RDLIN          ;;READ AN ASCIZ LINE
025524 012600          MOV      (SP)+,RO      ;;GET ADDRESS OF 1ST CHARACTER
025526 005001          CLR      R1              ;;CLEAR DATA WORD
025530 005002          CLR      R2
025532 112046          2$:  MOVB      (RO)+,-(SP)      ;;PICKUP THIS CHARACTER
025534 001412          BEQ      3$              ;;IF ZERO GET OUT
025536 006301          ASL      R1              ;;*2
025540 006102          ROL      R2
025542 006301          ASL      R1              ;;*4
025544 006102          ROL      R2
025546 006301          ASL      R1              ;;*8
025550 006102          ROL      R2
025552 042716 177770   BIC      #C7,(SP)      ;;STRIP THE ASCII JUNK
025556 062601          ADD      (SP)+,R1      ;;ADD IN THIS DIGIT
025560 000764          BR      2$              ;;LOOP
025562 005726          3$:  TST      (SP)+      ;;CLEAN TERMINATOR FROM STACK
025564 010166 000012   MOV      R1,12(SP)     ;;SAVE THE RESULT
025570 010237 025604   MOV      R2,#HIOCT
025574 012602          MOV      (SP)+,R2      ;;POP STACK INTO R2
025576 012601          MOV      (SP)+,R1      ;;POP STACK INTO R1
025600 012600          MOV      (SP)+,R0      ;;POP STACK INTO R0
025602 000002          RTI              ;;RETURN
025604 000000          $HIOCT: .WORD 0      ;;HIGH ORDER BITS GO HERE
5576 .SBTTL READ A DECIMAL NUMBER FROM THE TTY
;*****
;*THIS ROUTINE WILL READ A DECIMAL (ASCII) NUMBER FROM THE TTY AND
;*CHANGE IT TO BINARY. IF TOO MANY CHARACTERS OR ANY ILLEGAL CHARACTERS
;*ARE READ A "?" FOLLOWED BY A CARRIAGE RETURN-LINE FEED WILL BE TYPED.
;*THE COMPLETE NUMBER MUST BE RETYPED. THE INPUT IS TERMINATED BY THE
;*USER TYPING A CARRIAGE RETURN. THE RANGE OF THE INPUT NUMBER IS
;*POSITIVE 32767 TO NEGATIVE 32768.
;*CALL:
;*
;* RDDEC          ;;READ A DECIMAL NUMBER
;* RETURN HERE    ;;NUMBER IS ON TOP OF THE STACK
;
025606 011646          $RDDEC: MOV      (SP),-(SP)      ;;PROVIDE SPACE FOR
025610 016666 000004 000002 MOV      4(SP),2(SP)     ;;THE INPUT NUMBER
025616 010046          MOV      R0,-(SP)      ;;PUSH R0 ON STACK
025620 010146          MOV      R1,-(SP)      ;;PUSH R1 ON STACK
025622 010246          MOV      R2,-(SP)      ;;PUSH R2 ON STACK
025624 104411          1$:  RDLIN          ;;READ AN ASCIZ LINE
025626 012600          MOV      (SP)+,RO      ;;ADDRESS OF 1ST CHAR.
025630 010037 025754   MOV      R0,6$         ;;SAVE INCASE OF BAD INPUT
025634 005046          CLR      -(SP)         ;;CLEAR DATA WORD
025636 005002          CLR      R2           ;;SIGN SET POSITIVE
025640 122710 000055   CMPB    #'-(,R0)       ;;SEE IF A MINUS SIGN WAS TYPED
025644 001001          BNE     2$            ;;BR IF NO MINUS SIGN
025646 112002          MOVB    (RO)+,R2      ;;SAVE FOR LATER USE
025650 112001          2$:  MOVB    (RO)+,R1      ;;PICKUP THIS CHARACTER
025652 001424          BEQ     3$            ;;GET OUT IF ZERO
025654 122701 000060   CMPB    #'0,R1        ;;MAKE SURE THIS CHARACTER
025660 003032          BGT     5$            ;;IS A DIGIT BETWEEN 0 & 9
025662 122701 000071   CMPB    #'9,R1
025666 002427          BLT     5$
025670 032716 170000   BIT     #C7777,(SP)   ;;DON'T LET NUMBER GET TO BIG
025674 001024          BNE     5$            ;;BR IF NUMBER WOULD OVERFLOW
025676 006316          ASL     (SP)          ;;*2

```

READ A DECIMAL NUMBER FROM THE TTY

```

025700 011646      MOV      (SP),-(SP)      ;;SAVE FOR LATER
025702 006316      ASL      (SP)          ;;*4
025704 006316      ASL      (SP)          ;;*8.
025706 062616      ADD      (SP)+,(SP)      ;;*10.
025710 102416      BVS      5$             ;;OVERFLOW ISN'T ALLOWED
025712 162701 000060 SUB      #'0,R1         ;;STRIP AWAY THE ASCII JUNK
025716 060116      ADD      R1,(SP)       ;;ADD IN THIS DIGIT
025720 102412      BVS      5$             ;;OVERFLOW ISN'T ALLOWED
025722 000752      BR       2$             ;;LOOP
025724 005702      3$:   TST      R2          ;;CHECK IF NUMBER IS NEG
025726 001401      BEQ      4$             ;;BR IF NO
025730 005416      NEG      (SP)          ;;YES--NEGATE THE NUMBER
025732 012666 000012 4$:   MOV      (SP)+,12(SP)  ;;SAVE THE RESULT
025736 012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
025740 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
025742 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
025744 000002      RTI                     ;;RETURN
025746 005726      5$:   TST      (SP)+      ;;CLEAN PARTIAL NUMBER FROM STACK
025750 105010      CLRB    (R0)          ;;SET A TERMINATOR
025752 104401      TYPE                     ;;TYPE THE INPUT UP TO BAD CHAR.
025754 000000      6$:   .WORD    0          ;;POINTER GOES HERE
025756 104401 001174 TYPE      , $QUES      ;; "?" "CR" & "LF"
025762 000720      BR       1$             ;;TRY AGAIN

```

5577
5578
5579
5580

;;ENABLE HOB IN ERROR ROUTINE SINCE MOST LIKELY IT IS CURRENTLY DISABLED.

.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
;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

```

```

025764 005737 002624 $ERROR: TST      UQUIET          ;;TEST FOR USER-QUIET MODE
025770 001403      BEQ      9$             ;;BRANCH IF FIELD-SERVICE MODE
025772 005000      CLR      R0            ;;IN CASE R0 HAS A #3 IN IT (+C)
025774 004737 026206 JSR      PC,ABORT        ;;TEST FOR ABORT CONDITION
026000 026000      9$:   CKSWR                     ;;TEST FOR CHANGE IN SOFT-SWR
026002 104407      BIS      #BIT09,BCSR   ;;ENABLE
026010 105237 001103 7$:   INCB    $ERFLG          ;;SET THE ERROR FLAG
026014 001775      BEQ      7$             ;;DON'T LET THE FLAG GO TO ZERO
026016 013777 001102 153116 MOV      $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
026024 032777 002000 153106 BIT      #BIT10,@SWR    ;;BELL ON ERROR?
026032 001402      BEQ      1$             ;;NO - SKIP
026034 104401 001170      TYPE      , $BELL      ;;RING BELL
026040 005237 001112      1$:   INC      $ERTTL          ;;COUNT THE NUMBER OF ERRORS
026044 011637 001116      MOV      (SP), $ERRPC  ;;GET ADDRESS OF ERROR INSTRUCTION
026050 162737 000002 001116 SUB      #2,$ERRPC
026056 117737 153034 001114 MOVVB   @ $ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
026064 032777 020000 153046 BIT      #BIT13,@SWR    ;;SKIP TYPEOUT IF SET

```

ERROR HANDLER ROUTINE

```

026072 001004      BNE      20$      ;;SKIP TYPEOUTS
026074 004737 026346 JSR      PC,ERTYPE ;;GO TO USER ERROR ROUTINE
026100 104401 001175     TYPE      ,#CRLF
026104      122737 000001 001220 20$:  CMPB    #APTENV,#ENV  ;;RUNNING IN APT MODE
026112 001007      BNE      2$      ;;NO,SKIP APT ERROR REPORT
026114 113737 001114 026126     MOVB    #ITEMB,21$  ;;SET ITEM NUMBER AS ERROR NUMBER
026122 004737 026406     JSR      PC,#ATY4   ;;REPORT FATAL ERROR TO APT
026126      000      21$:  .BYTE   0
026127      000      .BYTE   0
026130 000777      BR       22$      ;;APT ERROR LOCP
026132 005777 153002 2$:  TST     @SWR      ;;HALT ON ERROR
026136 100002      BPL     3$      ;;SKIP IF CONTINUE
026140 000000      HALT                    ;;HALT ON ERROR!
026142 104407      CKSWR                    ;;TEST FOR CHANGE IN SOFT-SWR
026144 032777 001000 152766 3$:  BIT     @BIT09,@SWR  ;;LOOP ON ERROR SWITCH SET?
026152 001402      BEQ     4$      ;;BR IF NO
026154 013716 001110     MOV     #LPERR,(SP)  ;;FUDGE RETURN FOR LOOPING
026160 005737 001166     4$:  TST     #ESCAPE     ;;CHECK FOR AN ESCAPE ADDRESS
026164 001402      BEQ     5$      ;;BR IF NONE
026166 013716 001166     MOV     #ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
026172      022737 023340 000042 5$:  CMP     #ENDAD,@#42  ;;ACT-11 AUTO-ACCEPT?
026200 001001      RNE     6$      ;;BRANCH IF NO
026202 000000      HALT                    ;;YES
026204      000002 6$:  RTI      ;;RETURN
026206 005737 002622     .SBTTL  ABORT ROUTINE FOR LCP/ORION UFD MODE
026212 001454     ABORT:  TST     UFDFLG    ;;TEST FOR USER FRIENDLY MODE
026214 020027 000032     BEQ     NOABRT      ;;IF NOT UFD THEN CONTINUE NORMAL OPERATION
026220 001443     CMP     R0,#32     ;;IS IT A 1Z ?
026222 020027 000003     BEQ     ABORTZ     ;;JUST GO BACK TO CHAIN IF IT IS (NO ERROR)
026226 001404     CMP     R0,#3     ;;IS IS A 1C ?
026230 005737 002624     BEQ     ABORTC     ;;BR TO LOAD 1C ON XXDP+ STACK (NO ERROR)
026234 001443     TST     UQUIET    ;;TEST FOR USER-QUIET MODE
026236 000422     BEQ     NOABRT    ;;IF FIELD-SERVICE MODE, CONTINUE NORMAL OPERATION
026240 013737 002616 000030  BR       ABORTE     ; BECAUSE FIELD-SERVICE MODE DOES NOT QUIT ON ERROR
026246 013737 002620 000032  MOV     SAV30,30   ; SET DRSERR THEN LEAVE
026254 104043     MOV     SAV32,32  ; RESTORE EMT LOCATION (30)
026256 005720     EMT     +43      ; RESTORE EMT PRIORITY LOCATION (32)
026260 001376     1$:  TST     (R0)+     ; GET XXDP STACK LOC. INTO RC FROM MONITOR
026262 112760 000057 177777  BNE     1$      ; FIND END OF STACK
026270 112720 000136     MOVB    #'./,-1(R0) ; LOAD SLASH OVER ZERO
026274 112720 000103     MOVB    #'1,(R0)+  ; LOAD UPARROW
026300 105010     MOVB    #'C,(R0)+ ; LOAD C
026302 000412     CLRB   (R0)     ; MAKE NEW END TO STACK
026304 013737 002616 000030  BR       ABORTZ   ; NOW LEAVE
026312 013737 002620 000032  MOV     SAV30,30  ; RESTORE EMT LOCATION (30)
026320 104042     MOV     SAV32,32 ; RESTORE EMT PRIORITY LOCATION (32)
026322 012760 177777 000042  EMT     +42      ; GET DCA LOCATION INTO R0 FROM MONITOR
026330 013700 000042  ABORTZ:  MOV     #-1,42(R0) ; SET A -1 INTO LOCATION DRSERR IN MONITOR
026334 005037 000042     MOV     @#42,R0  ; AND PUT THE MONITOR RETURN ADDRESS IN R0
026340 000137 023340     CLR     @#42     ; CLEAR MONITOR RETURN FLAG
026344 000207     JMP     #ENDAD   ; RETURN TO MONITOR-DO NOT PUSH STACK HERE
NOABRT: RTS      PC ; IF NOTUFD RETURN TO MAINLINE
; * THIS ROUTINE LOADS TEST NUMBER AND ERROR NUMBER AND THEN CALLS $ERRTYP

```


APT COMMUNICATIONS ROUTINE

5587

000001
000100
000040

026636 104401 001175
026642 010046
026644 005000
026646 153700 001114
026652 001004

026654 013746 001116

026660 104402
026662 000426
026664 005300
026666 006300
026670 006300
026672 006300
026674 062700 001324
026700 012037 026710
026704 001404
026706 104401
026710 000000
026712 104401 001175
026716 012037 026726
026722 001404
026724 104401
026726 000000
026730 104401 001175
026734 011000
026736 001004
026740 012600
026742 104401 001175
026746 000207
026750
026750 013046
026752 104402
026754 005710
026756 001770
026760 104401 026766
026764 000771
026766 040 040 000

APTENV=001
APTSPool=100
APTCsUP=040
.SBTTL ERROR MESSAGE TYPEOUT ROUTINE
;*****
;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" (\$ITEMB) TO DETERMINE WHICH
;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" (\$ERRTB),
;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
\$ERRTYP:

TYPE , \$CRLF ;;"CARRIAGE RETURN" & "LINE FEED"
MOV RO,-(SP) ;;SAVE RO
CLR RO ;;PICKUP THE ITEM INDEX
BISB @#\$ITEMB,RO
BNE 1\$;;IF ITEM NUMBER IS ZERO, JUST
; ;TYPE THE PC OF THE ERROR
MOV \$ERRPC,-(SP) ;;SAVE \$ERRPC FOR TYPEOUT
; ;ERROR ADDRESS
; ;GO TYPE--OCTAL ASCII(ALL DIGITS)
BR 6\$;;GET OUT
1\$: DEC RO ;;ADJUST THE INDEX SO THAT IT WILL
ASL RO ;; WORK FOR THE ERROR TABLE
ASL RO
ASL RO
ADD @#\$ERRTB,RO ;;FORM TABLE POINTER
MOV (RO)+,2\$;;PICKUP "ERROR MESSAGE" POINTER
BEQ 3\$;;SKIP TYPEOUT IF NO POINTER
TYPE ;;TYPE THE "ERROR MESSAGE"
2\$: .WORD 0 ;;"ERROR MESSAGE" POINTER GOES HERE
TYPE , \$CRLF ;;"CARRIAGE RETURN" & "LINE FEED"
3\$: MOV (RO)+,4\$;;PICKUP "DATA HEADER" POINTER
BEQ 5\$;;SKIP TYPEOUT IF 0
TYPE ;;TYPE THE "DATA HEADER"
4\$: .WORD 0 ;;"DATA HEADER" POINTER GOES HERE
TYPE , \$CRLF ;;"CARRIAGE RETURN" & "LINE FEED"
5\$: MOV (RO),RO ;;PICKUP "DATA TABLE" POINTER
BNE 7\$;;GO TYPE THE DATA
6\$: MOV (SP)+,RO ;;RESTORE RO
TYPE , \$CRLF ;;"CARRIAGE RETURN" & "LINE FEED"
RTS PC ;;RETURN
7\$: MOV @ (RO)+,-(SP) ;;SAVE @ (RO)+ FOR TYPEOUT
TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
TST (RO) ;;IS THERE ANOTHER NUMBER?
BEQ 6\$;;BR IF NO
TYPE ,8\$;;TYPE TWO(2) SPACES
BR 7\$;;LOOP
8\$: .ASCIZ / / ;;TWO(2) SPACES
.EVEN

5588

026772 010046
026774 016600 000002
027000 005740

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

TRAP DECODER

```

027002 111000          MOV      (RO),RO          ;;GET RIGHT BYTF OF TRAP
027004 006300          ASL      RO              ;;POSITION FOR INDEXING
027006 016000 027026   MOV      $TRPAD(RO),RO    ;;INDEX TO TABLE
027012 000200          RTS      RO              ;;GO TO ROUTINE
;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
027014 011646          $TRAP2: MOV     (SP),-(SP)    ;;MOVE THE PC DOWN
027016 016666 000004 000002 MOV     4(SP),2(SP)      ;;MOVE THE PSW DOWN
027024 000002          RTI      RTI      ;;RESTORE THE PSW

.SBTTL TRAP TABLE
;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
;*BY THE "TRAP" INSTRUCTION.
; ROUTINE
; ---
027026 027014          $TRPAD: .WORD  $TRAP2
027030 023712          $TYPE  ;;CALL=TYPE   TRAP+1(104401) TTY TYPEOUT ROUTINE
027032 024272          $TYPOC ;;CALL=TYPOC  TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
027034 024246          $TYPOS ;;CALL=TYPOS  TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
027036 024306          $TYPON ;;CALL=TYPON  TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
027040 024474          $TYPDS ;;CALL=TYPDS  TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
027042 024770          $GTSWR ;;CALL=GTSWR   TRAP+6(104406) GET SOFT-SWR SETTING
027044 024720          $CKSWR ;;CALL=CKSWR   TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
027046 025202          $RDCHR ;;CALL=RDCHR  TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
027050 025332          $RDLIN ;;CALL=RDLIN   TRAP+11(104411) TTY TYPEIN STRING ROUTINE
027052 025504          $RDOCT ;;CALL=RDOCT   TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
027054 025606          $RDDEC ;;CALL=RDDEC  TRAP+13(104413) READ A DECIMAL NUMBER FROM TTY

.SBTTL POWER DOWN AND UP ROUTINES
;*****
;POWER DOWN ROUTINE
027056 012737 027216 000024 $PWRDN: MOV     $ILLUP,$PWRVEC ;;SET FOR FAST UP
027064 012737 000340 000026 MOV     $340,$PWRVEC+2 ;;PRIO:7
027072 010046          MOV     RO,-(SP)        ;;PUSH RO ON STACK
027074 010146          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
027076 010246          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
027100 010346          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
027102 010446          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
027104 010546          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
027106 017746 152026          MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
027112 010637 027222          MOV     SP,$SAVR6      ;;SAVE SP
027116 012737 027130 000024 MOV     $PWRUP,$PWRVEC ;;SET UP VECTOR
027124 000000          HALT
027126 000776          BR      -2            ;;HANG UP
;*****
;POWER UP ROUTINE
027130 012737 027216 000024 $PWRUP: MOV     $ILLUP,$PWRVEC ;;SET FOR FAST DOWN
027136 013706 027222          MOV     $SAVR6,SP      ;;GET SP
027142 005037 027222          CLR     $SAVR6        ;;WAIT LOOP FOR THE TTY
027146 005237 027222          1$: INC     $SAVR6      ;;WAIT FOR THE INC
027152 001375          BNE    1$            ;;OF WORD
027154 012677 151760          MOV     (SP)+,$SWR    ;;POP STACK INTO @SWR
027160 012605          MOV     (SP)+,R5     ;;POP STACK INTO R5
027162 012604          MOV     (SP)+,R4     ;;POP STACK INTO R4
027164 012603          MOV     (SP)+,R3     ;;POP STACK INTO R3
027166 012602          MOV     (SP)+,R2     ;;POP STACK INTO R2
027170 012601          MOV     (SP)+,R1     ;;POP STACK INTO R1
027172 012600          MOV     (SP)+,R0     ;;POP STACK INTO R0
027174 012737 027056 000024 MOV     $PWRDN,$PWRVEC ;;SET UP THE POWER DOWN VECTOR
027202 012737 000340 000026 MOV     $340,$PWRVEC+2 ;;PRIO:7

```

5589

POWER DOWN AND UP ROUTINES

```

027210 104401          TYPE          ;REPORT THE POWER FAILURE
027212 027224          $PWRMG: .WORD $POWER      ;;POWER FAIL MESSAGE POINTER
027214 000002          RTI
027216 000000          $ILLUP: HALT           ;;THE POWER UP SEQUENCE WAS STARTED
027220 000776          BR          . 2          ;; BEFORE THE POWER DOWN WAS COMPLETE
027222 000000          $SAVR6: 0              ;;PUT THE SP HERE
027224          015      012      120  $POWER: .ASCIZ <15><12>"POWER"
027227          117      127      105
027232          122      000

          .EVEN

5590
5591          ;
5592          ; ERROR MESSAGES
5593          ;
5594
5595 027234          124      111      115  EM1:  .ASCIZ /TIMEOUT ON ACCESSING A MAP REGISTER/
      027237          105      117      125
      027242          124      040      117
      027245          116      040      101
      027250          103      103      105
      027253          123      123      111
      027256          116      107      040
      027261          101      040      115
      027264          101      120      040
      027267          122      105      107
      027272          111      123      124
      027275          105      122      000
5596 027300          115      101      120  EM2:  .ASCIZ /MAP REGISTER COULD NOT BE CLEARED/
      027303          040      122      105
      027306          107      111      123
      027311          124      105      122
      027314          040      103      117
      027317          125      114      104
      027322          040      116      117
      027325          124      040      102
      027330          105      040      103
      027333          114      105      101
      027336          122      105      104
      027341          000
5597 027342          115      101      120  EM3:  .ASCIZ /MAP REGISTER COULD NOT HOLD PATTERN/
      027345          040      122      105
      027350          107      111      123
      027353          124      105      122
      027356          040      103      117
      027361          125      114      104
      027364          040      116      117
      027367          124      040      110
      027372          117      114      104
      027375          040      120      101
      027400          124      124      105
      027403          122      116      000
5598 027406          115      101      120  EM4:  .ASCIZ /MAP REGISTER HAS NOT BEEN ADDRESSED CORRECTLY/
      027411          040      122      105
      027414          107      111      123
      027417          124      105      122
      027422          040      110      101
      027425          123      040      116

```

POWER DOWN AND UP ROUTINES

	027430	117	124	040	
	027433	102	105	105	
	027436	116	040	101	
	027441	104	104	122	
	027444	105	123	123	
	027447	105	104	040	
	027452	103	117	122	
	027455	122	105	103	
	027460	124	114	131	
	027463	000			
5599	027464	124	110	105	EM5: .ASCIZ /THERE WAS NO DIFFERENCE FOUND BETWEEN HI AND LO MAP REGISTERS/
	027467	122	105	040	
	027472	127	101	123	
	027475	040	116	117	
	027500	040	104	111	
	027503	106	106	105	
	027506	122	105	116	
	027511	103	105	040	
	027514	106	117	125	
	027517	116	104	040	
	027522	102	105	124	
	027525	127	105	105	
	027530	116	040	110	
	027533	111	040	101	
	027536	116	104	040	
	027541	114	117	040	
	027544	115	101	120	
	027547	040	122	105	
	027552	107	111	123	
	027555	124	105	122	
	027560	123	000		
5600	027562	105	122	122	EM6: .ASCIZ /ERROR IN BITS 3 6,9-14 IN THE DCSR/
	027565	117	122	040	
	027570	111	116	040	
	027573	102	111	124	
	027576	123	040	063	
	027601	055	066	054	
	027604	071	055	061	
	027607	064	040	111	
	027612	116	040	124	
	027615	110	105	040	
	027620	104	103	123	
	027623	122	000		
5601	027625	104	103	123	EM7: .ASCIZ /DCSR DID NOT RESPOND PROPORLY ON RESET/
	027630	122	040	104	
	027633	111	104	040	
	027636	116	117	124	
	027641	040	122	105	
	027644	123	120	117	
	027647	116	104	040	
	027652	120	122	117	
	027655	120	117	122	
	027660	114	131	040	
	027663	117	116	040	
	027666	122	105	123	
	027671	105	124	000	
5602	027674	124	111	115	EM10: .ASCIZ /TIMEOUT HAS OCCURED ON ACCESS TO THE DCSR/

POWER DOWN AND UP ROUTINES

	027677	105	117	125	
	027702	124	040	110	
	027705	101	123	040	
	027710	117	103	103	
	027713	125	122	105	
	027716	104	040	117	
	027721	116	040	101	
	027724	103	103	105	
	027727	123	123	040	
	027732	124	117	040	
	027735	124	110	105	
	027740	040	104	103	
	027743	123	122	000	
5603	027746	113	115	103	EM11: .ASCIZ /KMCR BITS 0-5,8 DID NOT GET SET CORRECTLY/
	027751	122	040	102	
	027754	111	124	123	
	027757	040	060	055	
	027762	065	054	070	
	027765	040	104	111	
	027770	104	040	116	
	027773	117	124	040	
	027776	107	105	124	
	030001	040	123	105	
	030004	124	040	103	
	030007	117	122	122	
	030012	105	103	124	
	030015	114	131	000	
5604	030020	124	111	115	EM12: .ASCIZ /TIMEOUT HAS OCCURED ON ACCESS TO THE KMCR/
	030023	105	117	125	
	030026	124	040	110	
	030031	101	123	040	
	030034	117	103	103	
	030037	125	122	105	
	030042	104	040	117	
	030045	116	040	101	
	030050	103	103	105	
	030053	123	123	040	
	030056	124	117	040	
	030061	124	110	105	
	030064	040	113	115	
	030067	103	122	000	
5605	030072	105	122	122	EM13: .ASCIZ /ERROR IN DATA PATH PERFORMING DATA OUT/
	030075	117	122	040	
	030100	111	116	040	
	030103	104	101	124	
	030106	101	040	120	
	030111	101	124	110	
	030114	040	120	105	
	030117	122	106	117	
	030122	122	115	111	
	030125	116	107	040	
	030130	104	101	124	
	030133	101	040	117	
	030136	125	124	000	
5606	030141	105	122	122	EM14: .ASCIZ /ERROR IN DATA OUT CYCLE/
	030144	117	122	040	
	030147	111	116	040	

POWER DOWN AND UP ROUTINES

	030152	104	101	124	
	030155	101	040	117	
	030160	125	124	040	
	030163	103	131	103	
	030166	114	105	000	
5607	030171	105	122	122	EM15: .ASCIZ /ERROR IN DATA IN CYCLE/
	030174	117	122	040	
	030177	111	116	040	
	030202	104	101	124	
	030205	101	040	111	
	030210	116	040	103	
	030213	131	103	114	
	030216	105	000		
5608	030220	104	104	122	EM16: .ASCIZ /DDR IS NOT 0, WHEN DCRS<2-1> SELECTS UNIBUS LINES/
	030223	040	111	123	
	030226	040	116	117	
	030231	124	040	060	
	030234	054	040	127	
	030237	110	105	116	
	030242	040	104	103	
	030245	122	123	074	
	030250	062	055	061	
	030253	076	040	123	
	030256	105	114	105	
	030261	103	124	123	
	030264	040	125	116	
	030267	111	102	125	
	030272	123	040	114	
	030275	111	116	105	
	030300	123	000		
5609	030302	104	103	123	EM17: .ASCIZ /DCSR<07> DOES NOT BECOME 1/
	030305	122	074	060	
	030310	067	076	040	
	030313	104	117	105	
	030316	123	040	116	
	030321	117	124	040	
	030324	102	105	103	
	030327	117	115	105	
	030332	040	061	000	
5610	030335	111	116	104	EM20: .ASCIZ /INDIRECT ADDRESSING OF MAPPING REGISTER PAIRS/
	030340	111	122	105	
	030343	103	124	040	
	030346	101	104	104	
	030351	122	105	123	
	030354	123	111	116	
	030357	107	040	117	
	030362	106	040	115	
	030365	101	120	120	
	030370	111	116	107	
	030373	040	122	105	
	030376	107	111	123	
	030401	124	105	122	
	030404	040	120	101	
	030407	111	122	123	
	030412	000			
5611	030413	122	105	107	EM21: .ASCIZ /REGISTER PAIR 40 PERFORMS RELOCATION/
	030416	111	123	124	

POWER DOWN AND UP ROUTINES

	030421	105	122	040	
	030424	120	101	111	
	030427	122	040	064	
	030432	060	040	120	
	030435	105	122	106	
	030440	117	122	115	
	030443	123	040	122	
	030446	105	114	117	
	030451	103	101	124	
	030454	111	117	116	
	030457	000			
5612	030460	116	130	115	EM22: .ASCIZ /NXM CONDITION COULD NOT BE CREATED THRU ALU/
	030463	040	103	117	
	030466	116	104	111	
	030471	124	111	117	
	030474	116	040	103	
	030477	117	125	114	
	030502	104	040	116	
	030505	117	124	040	
	030510	102	105	040	
	030513	103	122	105	
	030516	101	124	105	
	030521	104	040	124	
	030524	110	122	125	
	030527	040	101	114	
	030532	125	000		
5613	030534	101	114	125	EM23: .ASCIZ /ALU ERROR/
	030537	040	105	122	
	030542	122	117	122	
	030545	000			
5614	030546	103	120	125	EM24: .ASCIZ /CPU CACHE ERROR/
	030551	040	103	101	
	030554	103	110	105	
	030557	040	105	122	
	030562	122	117	122	
	030565	000			
5615	030566	113	115	103	EM25: .ASCIZ /KMCR<4-0> DON'T DISABLE MEMORY RESPONSE/
	030571	122	074	064	
	030574	055	060	076	
	030577	040	104	117	
	030602	116	047	124	
	030605	040	104	111	
	030610	123	101	102	
	030613	114	105	040	
	030616	115	105	115	
	030621	117	122	131	
	030624	040	122	105	
	030627	123	120	117	
	030632	116	123	105	
	030635	000			
5616	030636	113	115	103	EM26: .ASCIZ /KMCR DOES NOT REFLECT EXPECTED STATUS OF THE CACHE/
	030641	122	040	104	
	030644	117	105	123	
	030647	040	116	117	
	030652	124	040	122	
	030655	105	106	114	
	030660	105	103	124	

POWER DOWN AND UP ROUTINES

	030663	040	105	130	
	030666	120	105	103	
	030671	124	105	104	
	030674	040	123	124	
	030677	101	124	125	
	030702	123	040	117	
	030705	106	040	124	
	030710	110	105	040	
	030713	103	101	103	
	030716	110	105	000	
5617	030721	105	122	122	EM27: .ASCIZ /ERROR IN CACHE TAG REGISTERS/
	030724	117	122	040	
	030727	111	116	040	
	030732	103	101	103	
	030735	110	105	040	
	030740	124	101	107	
	030743	040	122	105	
	030746	107	111	123	
	030751	124	105	122	
	030754	123	000		
5618	030756	105	122	122	EM30: .ASCIZ /ERROR IN THE DMA CACHE DATA RAMS/
	030761	117	122	040	
	030764	111	116	040	
	030767	124	110	105	
	030772	040	104	115	
	030775	101	040	103	
	031000	101	103	110	
	031003	105	040	104	
	031006	101	124	101	
	031011	040	122	101	
	031014	115	123	000	
5619	031017	105	122	122	EM31: .ASCIZ /ERROR IN THE M9312 BOOT ROM SECTION/
	031022	117	122	040	
	031025	111	116	040	
	031030	124	110	105	
	031033	040	115	071	
	031036	063	061	062	
	031041	040	102	117	
	031044	117	124	040	
	031047	122	117	115	
	031052	040	123	105	
	031055	103	124	111	
	031060	117	116	000	
5620	031063	105	122	122	EM32: .ASCIZ /ERROR IN ARBITRATION LOGIC USING UBE/
	031066	117	122	040	
	031071	111	116	040	
	031074	101	122	102	
	031077	111	124	122	
	031102	101	124	111	
	031105	117	116	040	
	031110	114	117	107	
	031113	111	103	040	
	031116	125	123	111	
	031121	116	107	040	
	031124	125	102	105	
	031127	000			
5621	031130	105	122	122	EM33: .ASCIZ /ERROR TRYING TO EXECUTE DMA CYCLES THRU UBE/

POWER DOWN AND UP ROUTINES

	031133	117	122	040	
	031136	124	122	131	
	031141	111	116	107	
	031144	040	124	117	
	031147	040	105	130	
	031152	105	103	125	
	031155	124	105	040	
	031160	104	115	101	
	031163	040	103	131	
	031166	103	114	105	
	031171	123	040	124	
	031174	110	122	125	
	031177	040	125	102	
	031202	105	000		
5622	031204	105	122	122	EM34: .ASCIZ /ERROR IN THE UNIBUS MEMORY TEST/
	031207	117	122	040	
	031212	111	116	040	
	031215	124	110	105	
	031220	040	125	116	
	031223	111	102	125	
	031226	123	040	115	
	031231	105	115	117	
	031234	122	131	040	
	031237	124	105	123	
	031242	124	000		
5623	031244	125	116	105	EM35: .ASCIZ /UNEXPECTED TIMEOUT HAS OCCURED/
	031247	130	120	105	
	031252	103	124	105	
	031255	104	040	124	
	031260	111	115	105	
	031263	117	125	124	
	031266	040	110	101	
	031271	123	040	117	
	031274	103	103	125	
	031277	122	105	104	
	031302	000			
5624	031303	125	116	111	EM36: .ASCIZ /UNIBUS TIMEOUT DID NOT WORK CORRECTLY/
	031306	102	125	123	
	031311	040	124	111	
	031314	115	105	117	
	031317	125	124	040	
	031322	104	111	104	
	031325	040	116	117	
	031330	124	040	127	
	031333	117	122	113	
	031336	040	103	117	
	031341	122	122	105	
	031344	103	124	114	
	031347	131	000		
5625	031351	104	103	123	EM37: .ASCIZ /DCSR<3> DID NOT DISABLE UBA ROM RESPONSE/
	031354	122	074	063	
	031357	076	040	104	
	031362	111	104	040	
	031365	116	117	124	
	031370	040	104	111	
	031373	123	101	102	
	031376	114	105	040	

POWER DOWN AND UP ROUTINES

	031643	101	104	104							
	031646	122	105	123							
	031651	123	000								
5633	031653	124	105	123	DH4:	.ASCII	/TEST	ERROR	ERROR	HI	LOW/<12><15>
	031656	124	011	105							
	031661	122	122	117							
	031664	122	011	105							
	031667	122	122	117							
	031672	122	011	110							
	031675	111	011	114							
	031700	117	127	012							
	031703	015									
5634	031704	040	040	043		.ASCIZ	/ #	PC	#	MAP	MAP/
	031707	011	040	040							
	031712	120	103	011							
	031715	040	040	043							
	031720	011	115	101							
	031723	120	011	115							
5635	031726	101	120	000	DH5:	.ASCII	/TEST	ERROR	ERROR	GOOD	BAD/<12><15>
	031731	124	105	123							
	031734	124	011	105							
	031737	122	122	117							
	031742	122	011	105							
	031745	122	122	117							
	031750	122	011	107							
	031753	117	117	104							
	031756	011	102	101							
5636	031761	104	012	015		.ASCIZ	/ #	PC	#	DATA	DATA/
	031764	040	040	043							
	031767	011	040	040							
	031772	120	103	011							
	031775	040	040	043							
	032000	011	104	101							
	032003	124	101	011							
	032006	104	101	124							
	032011	101	000								
5637	032013	124	105	123	DH13:	.ASCII	/TEST	ERROR	ERROR	DDR	PATTERN/<12><15>
	032016	124	011	105							
	032021	122	122	117							
	032024	122	011	105							
	032027	122	122	117							
	032032	122	011	104							
	032035	104	122	011							
	032040	120	101	124							
	032043	124	105	122							
	032046	116	012	015							
5638	032051	040	040	043		.ASCIZ	/ #	PC	#/		
	032054	011	040	040							
	032057	120	103	011							
	032062	040	040	043							
	032065	000									
5639	032066	124	105	123	DH16:	.ASCII	/TEST	ERROR	ERROR		<12><15>
	032071	124	011	105							
	032074	122	122	117							
	032077	122	011	105							
	032102	122	122	117							
	032105	122	012	015							

POWER DOWN AND UP ROUTINES

	032354	123	12	012										
	032357	015												
5646	032360	040	040	043		.ASCIZ	/	*	PC	*	<22-16>	<15-0>/		
	032363	011	040	040										
	032366	120	103	011										
	032371	040	040	043										
	032374	011	074	062										
	032377	062	055	061										
	032402	066	076	011										
	032405	074	061	065										
	032410	055	060	076										
	032413	000												
5647						.EVEN								
5648	032414	001716	001116	001714	DT1:	.WORD	TEST,	\$ERRPC,	ERRNUM,	\$BDADR,	0			
	032422	001122	000000											
5649	032426	001716	001116	001714	DT2:	.WORD	TEST,	\$ERRPC,	ERRNUM,	\$GDDAT,	\$BDDAT,	\$BDADR,	0	
	032434	001124	001126	001122										
	032442	000000												
5650	032444	001716	001116	001714	DT3:	.WORD	TEST,	\$ERRPC,	ERRNUM,	\$GDADR,	\$BDADR,	0		
	032452	001120	001122	000000										
5651	032460	001716	001116	001714	DT4:	.WORD	TEST,	\$ERRPC,	ERRNUM,	\$GDDAT,	\$BDDAT,	0		
	032466	001124	001126	000000										
5652	032474	001716	001116	001714	DT16:	.WORD	TEST,	\$ERRPC,	ERRNUM,	0				
	032502	000000												
5653	032504	001716	001116	001714	DT20:	.WORD	TEST,	\$ERRPC,	ERRNUM,	KMCR,	\$BDADR,	0		
	032512	177734	001122	000000										
5654	032520	001716	001122	001714	DT21:	.WORD	TEST,	\$BDADR,	ERRNUM,	0				
	032526	000000												
5655	032530	001716	001116	001714	DT23:	.WORD	TEST,	\$ERRPC,	ERRNUM,	\$GDDAT,	\$BDDAT,	KIPAR6,	\$BDADR,	0
	032536	001124	001126	172354										
	032544	001122	000000											
5656	032550	001716	001116	001714	DT27:	.WORD	TEST,	\$ERRPC,	ERRNUM,	MAPH01,	MAPL01,	0		
	032556	170206	170204	000000										
5657	032564	001716	001116	001714	DT30:	.WORD	TEST,	\$ERRPC,	ERRNUM,	MAPL00,	0			
	032572	170200	000000											
5658		000001												

.END

Symbol table

ABASE = 000000	BE1BA 002206	DH1 031422	ERROR = 104000	MAPH16= 170272
ABORT 026206	BE1CC 002204	DH13 032013	ERRVEC= 000004	MAPH17= 170276
ABORTC 026240	BE1CLR 002212	DH16 032066	ERTYPE 026346	MAPH2 = 170212
ABORTE 026304	BE1CR1 002210	DH2 031471	GTSWR = 104406	MAPH20= 170302
ABORTZ 026330	BE1CR2 002214	DH20 032125	HT = 000011	MAPH21= 170306
ACDW1 = 000000	BE1DB 002202	DH23 032205	IOTVEC= 000020	MAPH22= 170312
ACDW2 = 000000	BE1INT 001724	DH27 032316	IUBE 002502	MAPH23= 170316
ACPUOP= 000000	BE1PSW 002220	DH3 031563	KDPARO= 172360	MAPH24= 170322
ADDW0 = 000000	BE1SV 015512	DH4 031653	KDPAR1= 172362	MAPH25= 170326
ADDW1 = 000000	BE1VEC 002216	DH5 031731	KDPAR2= 172364	MAPH26= 170332
ADDW10= 000000	BE2BA 002226	DISPLA 001142	KDPAR3= 172366	MAPH27= 170336
ADDW11= 000000	BE2CC 002224	DISPRE 000174	KDPAR4= 172370	MAPH3 = 170216
ADDW12= 000000	BE2CLR 002232	DSWR = 177570	KDPAR5= 172372	MAPH30= 170342
ADDW13= 000000	BE2CR1 002230	DT1 032414	KDPAR6= 172374	MAPH31= 170346
ADDW14= 000000	BE2CR2 002234	DT16 032474	KDPAR7= 172376	MAPH32= 170352
ADDW15= 000000	BE2DB 002222	DT2 032425	KDPDR0= 172320	MAPH33= 170356
ADDW2 = 000000	BE2INT 001726	DT20 032504	KDPDR1= 172322	MAPH34= 170362
ADDW3 = 000000	BE2PSW 002240	DT21 032520	KDPDR2= 172324	MAPH35= 170366
ADDW4 = 000000	BE2SV 015532	DT23 032530	KDPDR3= 172326	MAPH36= 170372
ADDW5 = 000000	BE2VEC 002236	DT27 032550	KDPDR4= 172330	MAPH37= 170376
ADDW6 = 000000	BIT0 = 000001	DT3 032444	KDPDR5= 172332	MAPH4 = 170222
ADDW7 = 000000	BIT00 = 000001	DT30 032564	KDPDR6= 172334	MAPH5 = 170226
ADDW8 = 000000	BIT01 = 000002	DT4 032460	KDPDR7= 172336	MAPH6 = 170232
ADDW9 = 000000	BIT02 = 000004	EMTSAV 002530	KIPAR0= 172340	MAPH7 = 170236
ADEVCT= 000000	BIT03 = 000010	EMTVEC= 000030	KIPAR1= 172342	MAPL0 = 170200
ADEVN = 000000	BIT04 = 000020	EM1 027234	KIPAR2= 172344	MAPL00= 170200
AENV = 000000	BIT05 = 000040	EM10 027674	KIPAR3= 172346	MAPL01= 170204
AENVN = 000000	BIT06 = 000100	EM11 027746	KIPAR4= 172350	MAPL02= 170210
AFATAL= 000000	BIT07 = 000200	EM12 030020	KIPAR5= 172352	MAPL03= 170214
AMADR1= 000000	BIT08 = 000400	EM13 030072	KIPAR6= 172354	MAPL04= 170220
AMADR2= 000000	BIT09 = 001000	EM14 030141	KIPAR7= 172356	MAPL05= 170224
AMADR3= 000000	BIT1 = 000002	EM15 030171	KIPDR0= 172300	MAPL06= 170230
AMADR4= 000000	BIT10 = 002000	EM16 030220	KIPDR1= 172302	MAPL07= 170234
AMAMS1= 000000	BIT11 = 004000	EM17 030302	KIPDR2= 172304	MAPL1 = 170204
AMAMS2= 000000	BIT12 = 010000	EM2 027300	KIPDR3= 172306	MAPL10= 170240
AMAMS3= 000000	BIT13 = 020000	EM20 030335	KIPDR4= 172310	MAPL11= 170244
AMAMS4= 000000	BIT14 = 040000	EM21 030413	KIPDR5= 172312	MAPL12= 170250
AMSGAD= 000000	BIT15 = 100000	EM22 030460	KIPDR6= 172314	MAPL13= 170254
AMSGLG= 000000	BIT2 = 000004	EM23 030534	KIPDR7= 172316	MAPL14= 170260
AMSGTY= 000000	BIT3 = 000010	EM24 030546	KMCR = 177734	MAPL15= 170264
AMTYP1= 000000	BIT4 = 000020	EM25 030566	LF = 000012	MAPL16= 170270
AMTYP2= 000000	BIT5 = 000040	EM26 030636	MAPH0 = 170202	MAPL17= 170274
AMTYP3= 000000	BIT6 = 000100	EM27 030721	MAPH00= 170202	MAPL2 = 170210
AMTYP4= 000000	BIT7 = 000200	EM3 027342	MAPH01= 170206	MAPL20= 170300
APASS = 000000	BIT8 = 000400	EM30 030756	MAPH02= 170212	MAPL21= 170304
APRIOR= 000000	BIT9 = 001000	EM31 031017	MAPH03= 170216	MAPL22= 170310
APTCSU= 000040	BPTVEC= 000014	EM32 031063	MAPH04= 170222	MAPL23= 170314
APTENV= 000001	CCR = 177746	EM33 031130	MAPH05= 170226	MAPL24= 170320
APTSIZ= 000200	CKSWR = 104407	EM34 031204	MAPH06= 170232	MAPL25= 170324
APTSPO= 000100	CR = 000015	EM35 031244	MAPH07= 170236	MAPL26= 170330
ASWREG= 000000	CRLF = 000200	EM36 031303	MAPH1 = 170206	MAPL27= 170334
ATESTN= 000000	CTBLE 002054	EM37 031351	MAPH10= 170242	MAPL3 = 170214
AUNIT = 000000	DCSR = 177730	EM4 027406	MAPH11= 170246	MAPL30= 170340
AUSWR = 000000	DDIN 002254	EM5 027464	MAPH12= 170252	MAPL31= 170344
AVECT1= 000000	DDISP = 177570	EM6 027562	MAPH13= 170256	MAPL32= 170350
AVECT2= 000000	DDOUT 002242	EM7 027625	MAPH14= 170262	MAPL33= 170354
BCSR = 177520	DDR = 177732	ERRNUM 001714	MAPH15= 170266	MAPL34= 170360

Symbol table

MAPL35=	170364	SWREG	000176	TST30	012134	\$BASE	001254	\$GDADR	001120
MAPL36=	170370	SW0	= 000001	TST31	012350	\$BDADR	001122	\$GDDAT	001124
MAPL37=	170374	SW00	= 000001	TST32	012772	\$BDAT	001126	\$GET42	023330
MAPL4	= 170220	SW01	= 000002	TST33	013224	\$BELL	001170	\$GTSWR	024770
MAPL5	= 170224	SW02	= 000004	TST34	013650	\$CDW1	001260	\$HD	= 000001
MAPL6	= 170230	SW03	= 000010	TST35	014050	\$CDW2	001262	\$HIBTS	000232
MAPL7	= 170234	SW04	= 000020	TST36	014236	\$CHARC	024242	\$HIOCT	025604
MAPPR	002276	SW05	= 000040	TST37	014340	\$CKSWR	024720	\$ICNT	001104
MCSR	001732	SW06	= 000100	TST4	004006	\$CMTAG	001100	\$ILLUP	027216
MEMSIZ	002364	SW07	= 000200	TST40	014620	\$CM3	= 000000	\$INTAG	001135
MMRO	= 177572	SW08	= 000400	TST41	015160	\$CM4	= 000002	\$ITEMB	001114
MMR1	= 177574	SW09	= 001000	TST42	015314	\$CNTLG	025455	\$LF	001176
MMR2	= 177576	SW1	= 000002	TST43	015552	\$CNTLU	025450	\$FLG	026633
MMR3	= 172516	SW10	= 002000	TST44	015756	\$CPUOP	001226	\$LPADR	001106
MMVEC	= 000250	SW11	= 004000	TST45	016050	\$CRLF	001175	\$LPERR	001110
NOABRT	026344	SW12	= 010000	TST46	016130	\$DBLK	024710	\$MADR1	001232
NOCAH	007706	SW13	= 020000	TST47	016230	\$DDW0	001264	\$MADR2	001236
NROM	013612	SW14	= 040000	TST5	004064	\$DDW1	001266	\$MADR3	001242
OBADR	002172	SW15	= 100000	TST50	016444	\$DDW10	001310	\$MADR4	001246
PCR	= 177522	SW2	= 000004	TST51	016542	\$DDW11	001312	\$MAIL	001200
PIRQ	= 177772	SW3	= 000010	TST52	016710	\$DDW12	001314	\$MAMS1	001230
PIRQVE	= 000240	SW4	= 000020	TST53	017022	\$DDW13	001316	\$MAMS2	001234
PMIS	001720	SW5	= 000040	TST54	017452	\$DDW14	001320	\$MAMS3	001240
POLY	= 120001	SW6	= 000100	TST55	017616	\$DDW15	001322	\$MAMS4	001244
PRO	= 000000	SW7	= 000200	TST56	017766	\$DDW2	001270	\$MBADR	000234
PR1	= 000040	SW8	= 000400	TST57	020414	\$DDW3	001272	\$MFLG	026632
PR2	= 000100	SW9	= 001000	TST6	004250	\$DDW4	001274	\$MNEW	025473
PR3	= 000140	TBITVE	= 000014	TST60	020766	\$DDW5	001276	\$MSGAD	001214
PR4	= 000200	TBLMH	012732	TST61	021050	\$DDW6	001300	\$MSGLG	001216
PR5	= 000240	TBLML	012672	TST62	021154	\$DDW7	001302	\$MSGTY	001200
PR6	= 000300	TEST	001716	TST63	021570	\$DDW8	001304	\$MSWR	025462
PR7	= 000340	TIMOUT	002266	TST64	022334	\$DDW9	001306	\$MTYP1	001231
PS	= 177776	TKVEC	= 000060	TST65	022520	\$DEVCT	001210	\$MTYP2	001235
PSW	= 177776	TOPTBL	010564	TST66	022742	\$DEVH	001256	\$MTYP3	001241
PTRN16	001774	TOUT	001730	TST67	023122	\$DOAGN	023350	\$MTYP4	001245
PTRN6	002024	TPVEC	= 000064	TST7	004426	\$DTBL	024700	\$MXCNT	023710
PWRVEC	= 000024	TRAPVE	= 000034	TYPDS	= 104405	\$ENDAD	023340	\$NULL	001154
RDCHR	= 104410	TRTVEC	= 000014	TYPF	= 104401	\$ENDCT	023220	\$NWTST	= 000001
RDDEC	= 104413	TST1	003324	TYPOC	= 104402	\$ENULL	023354	\$OCNT	024470
RDLIN	= 104411	TST10	004510	TYPON	= 104404	\$ENV	001220	\$OMODE	024472
RDOCT	= 104412	TST11	004642	TYPOS	= 104403	\$ENVM	001221	\$OVER	023636
RESTAR	003320	TST12	004744	UBECT	001722	\$EOP	023164	\$PASS	001206
RESVEC	= 000010	TST13	005066	UBEM	023122	\$EOPCT	023212	\$PASTM	000240
R6	= 0000006	TST14	005472	UBETST	013224	\$ERFLG	001103	\$POWER	027224
R7	= 0000007	TST15	005670	UFDLFG	002622	\$ERMAX	001115	\$PWRDN	027056
SAV30	002616	TST16	006074	UFDSET	= 000001	\$ERROR	025764	\$PWRMG	027212
SAV32	002620	TST17	006264	UMSIZ	002456	\$ERRPC	001116	\$PWRUP	027130
SCOPE	= 000004	TST2	003450	UQUIET	002624	\$ERRTB	001324	\$QUES	001174
SIMLGO	= 170014	TST20	006536	VALTBL	010550	\$ERRTY	026636	\$RDCHR	025202
SRO	= 177572	TST21	007136	VMKOR	002626	\$ERTTL	001112	\$RDECT	025606
SR1	= 177574	TST22	007616	WRBUF	001734	\$ESCAP	001166	\$RDLIN	025332
SR2	= 177576	TST23	007726	\$APTHD	000232	\$ETABL	001220	\$RDOCT	025504
SR3	= 172516	TST24	010120	\$ATYC	026414	\$ETEND	001324	\$RDSZ	= 000010
STACK	= 001100	TST25	010600	\$ATY1	026370	\$FATAL	001202	\$RTNAD	023352
START	002530	TST26	011310	\$ATY3	026376	\$FFLG	026634	\$SAVR6	027222
STKLMT	= 177774	TST27	011452	\$ATY4	026406	\$FILLC	001156	\$SCOPE	023360
SWR	001140	TST3	003664	\$AUTOB	001134	\$FILLS	001155	\$SETUP	= 000137

Symbol table

\$STUP = 177777	\$TKS 001144	\$TRAP2 027014	\$TYPEC 024124	\$VECT1 001250
\$SVLAD 023602	\$TMP0 001160	\$TRP = 000014	\$TYPEX 024244	\$VECT2 001252
\$SVPC = 000232	\$TMP1 001162	\$TRPAD 027026	\$TYPOC 024272	\$XOFF = 000023
\$SWR = 167400	\$TN = 000070	\$TSTM 000236	\$TYPON 024306	\$XON = 000021
\$SWREG 001222	\$TPB 001152	\$TSTNM 001102	\$TYPOS 024246	\$XTSTR 023400
\$SWRMK= 000300	\$TPFLG 001157	\$TTYIN 025440	\$UNIT 001212	\$GET4= 000000
\$TESTN 001204	\$TPS 001150	\$TYPDS 024474	\$UNITM 000242	\$OFILL 024471
\$TIMES 001164	\$TRAP 026772	\$TYPE 023712	\$USWR 001224	.\$X = 000232
\$TKB 001146				

. ABS. 032576 000 (RW,I,GBL,ABS,OVR)
 000000 001 (RW,I,LCL,REL,CON)

Errors detected: 0

*** Assembler statistics

Work file reads: 428
 Work file writes: 330
 Size of work file: 56656 Words (222 Pages)
 Size of core pool: 19402 Words (74 Pages)
 Operating system: RSX-11M/PLUS (Under VAX/VMS)

Elapsed time: 00:05:59.63

OKTACO.BIC,COKTACO/CR/-SP/NL:TOC=ORION.MLB/ML,COKTACO/DS:GBL

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES						
ABASE	= 000000	9-393	9-393					
ABORT	026206	64-5580	#64-5580					
ABORTC	026240	64-5580	#64-5580					
ABORTE	026304	64-5580	#64-5580					
ABORTZ	026330	64-5580	64-5580	#64-5580				
ACDW1	= 000000	9-393	9-393					
ACDW2	= 000000	9-393	9-393					
ACPUOP	= 000000	9-393	9-393					
ADDW0	= 000000	9-393	9-393					
ADDW1	= 000000	9-393	9-393					
ADDW10	= 000000	9-393	9-393					
ADDW11	= 000000	9-393	9-393					
ADDW12	= 000000	9-393	9-393					
ADDW13	= 000000	9-393	9-393					
ADDW14	= 000000	9-393	9-393					
ADDW15	= 000000	9-393	9-393					
ADDW2	= 000000	9-393	9-393					
ADDW3	= 000000	9-393	9-393					
ADDW4	= 000000	9-393	9-393					
ADDW5	= 000000	9-393	9-393					
ADDW6	= 000000	9-393	9-393					
ADDW7	= 000000	9-393	9-393					
ADDW8	= 000000	9-393	9-393					
ADDW9	= 000000	9-393	9-393					
ADEVCT	= 000000	9-393	9-393					
ADEVN	= 000000	9-393	9-393					
AENV	= 000000	9-393	9-393					
AENVN	= 000000	9-393	9-393					
AFATAL	= 000000	9-393	9-393					
AMADR1	= 000000	9-393	9-393					
AMADR2	= 000000	9-393	9-393					
AMADR3	= 000000	9-393	9-393					
AMADR4	= 000000	9-393	9-393					
AMAMS1	= 000000	9-393	9-393					
AMAMS2	= 000000	9-393	9-393					
AMAMS3	= 000000	9-393	9-393					
AMAMS4	= 000000	9-393	9-393					
AMSGAD	= 000000	9-393	9-393					
AMSGLG	= 000000	9-393	9-393					
AMSGTY	= 000000	9-393	9-393					
AMTYP1	= 000000	9-393	9-393					
AMTYP2	= 000000	9-393	9-393					
AMTYP3	= 000000	9-393	9-393					
AMTYP4	= 000000	9-393	9-393					
APASS	= 000000	9-393	9-393					
APRIOR	= 000000	9-393						
APTCU	= 000040	64-5571	#64-5586					
APTENV	= 000001	10-834	28-2187	37-3086	45-3856	64-5571	64-5580	64-5586 #64-5586
APTSIZ	= 000200	10-831	#64-5586					
APTSPO	= 000100	64-5571	64-5586	#64-5586				
ASWREG	= 000000	9-393	9-393					
ATESTN	= 000000	9-393	9-393					

SYMBOL CROSS REFERENCE		CREF V02								
SYMBOL	VALUE	REFERENCES								
AUNIT	= 000000	9-393	9-393							
AUSWR	= 000000	9-393	9-393							
AVECT1	= 000000	9-393	9-393							
AVECT2	= 000000	9-393	9-393							
BCSR	= 177520	#8-374	*37-3068	*45-3860	*45-3861	*60-5109	*64-5570	*64-5570	*64-5580	
BE1BA	002206	#10-633	40-3411	46-3919	48-3984	49-4056	50-4146	51-4221	52-4302	53-4426
		53-4459	54-4554	55-4626	56-4709	56-4755	57-4822	58-4898	59-4967	60-5070
		60-5085	60-5085	60-5094	60-5100	60-5116	60-5135	61-5246	61-5283	61-5302
		61-5326	62-5371	62-5382	63-5479	64-5531				
BE1CC	002204	#10-632	40-3410	40-3415	46-3918	48-3983	49-4058	50-4148	51-4224	52-4305
		53-4429	53-4462	54-4557	55-4629	56-4713	57-4823	58-4897	59-4969	60-5072
		60-5087	60-5102	60-5118	60-5137	61-5249	61-5286	61-5303	61-5329	62-5372
		62-5383	62-5390	63-5481	64-5533					
BE1CLR	002212	#10-635								
BE1CR1	002210	#10-634	40-3412	40-3413	41-3493	41-3496	41-3504	41-3508	41-3519	41-3523
		41-3534	41-3538	42-3619	42-3622	42-3636	42-3639	42-3653	42-3656	42-3670
		42-3673	43-3711	43-3722	44-3787	44-3790	44-3816	44-3921	47-3955	48-3987
		48-3988	49-4059	49-4065	50-4149	50-4155	51-4233	51-4234	52-4314	52-4315
		53-4438	53-4439	53-4471	53-4472	54-4566	54-4567	55-4636	55-4637	56-4722
		56-4723	56-4761	56-4762	57-4824	57-4825	58-4902	58-4903	59-4970	59-4971
		60-5073	60-5074	60-5088	60-5089	60-5103	60-5104	60-5119	60-5120	60-5138
		60-5140	61-5250	61-5251	61-5287	61-5288	61-5306	61-5308	61-5331	61-5333
		62-5375	62-5376	62-5384	62-5385	62-5391	62-5392	63-5485	63-5486	64-5537
		64-5538								
BE1CR2	002214	#10-636	45-3873	45-3878	46-3920	46-3924	47-3953	47-3958	47-3961	49-4057
		50-4147	51-4222	52-4303	53-4427	53-4460	54-4555	55-4627	55-4639	56-4710
		56-4711	56-4725	56-4756	57-4827	59-4968	60-5071	60-5086	60-5101	60-5117
		60-5136	60-5148	61-5247	61-5284	61-5315	61-5327	61-5341	62-5374	63-5480
		64-5532								
BE1DB	002202	#10-631	10-818	39-3332	48-3991	49-4055	50-4145	50-4162	51-4223	52-4304
		52-4323	53-4428	53-4450	53-4452	53-4461	53-4483	53-4485	54-4556	54-4575
		54-4577	55-4628	56-4712	56-4760	56-4764	58-4896	60-5079	60-5094	60-5110
		60-5125	61-5248	61-5265	61-5275	61-5285	61-5328	62-5373	62-5387	62-5394
		63-5498								
BE1INT	001724	#10-599	44-3796	*44-3803	*44-3812	*44-3818				
BE1PSW	002220	#10-638	41-3488	42-3612	43-3712	44-3780	55-4625	56-4717	57-4816	60-5132
		61-5305	61-5325							
BE1SV	015512	44-3779	#44-3811							
BE1VEC	002216	#10-637	41-3492	41-3502	41-3517	41-3532	42-3616	42-3633	42-3650	42-3667
		44-3779	48-3985	55-4624	56-4716	57-4815	60-5131	61-5304	61-5324	
BE2BA	002226	#10-646	61-5265							
BE2CC	002224	#10-645	61-5268							
BE2CLR	002232	#10-648	44-3808							
BE2CR1	002230	#10-647	44-3789	44-3791	44-3811	61-5269	61-5270			
BE2CR2	002234	#10-649	61-5266							
BE2DB	002222	#10-644	61-5267							
BE2INT	001726	#10-600	44-3800	*44-3804	*44-3813	*44-3817				
BE2PSW	002240	#10-651	44-3782							
BE2SV	015532	44-3781	#44-3816							
BE2VEC	002236	#10-650	44-3781							
BITO	= 000001	#8-277	12-989							
BIT00	= 000001	#8-277	8-277	10-697	19-1428	44-3790	44-3791	57-4801	57-4804	64-5522

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES
BPTVEC	= 000014	#8-277
CCR	= 177746	#8-378
CKSWR	= 104407	64-5570 64-5580 64-5580 #64-5588
CR	= 000015	#8-277 64-5571 64-5571
CRLF	= 000200	#8-277 10-836 10-836 64-5571 64-5571
CTBLE	002054	#10-618 36-2965
DCSR	= 177730	#8-375 *10-697 15-1201 *15-1206 15-1207 15-1209 15-1218 *15-1223 15-1225
		15-1228 15-1230 15-1235 *16-1281 *17-1327 *17-1344 *18-1378 *18-1379 18-1383
		*18-1398 *19-1424 *19-1425 *19-1428 *19-1439 *20-1462 *20-1463 *20-1477 *20-1479
		*21-1550 *21-1551 *21-1617 *22-1663 *22-1687 *23-1729 23-1753 *23-1761
		*24-1807 *24-1808 *24-1824 *25-1873 *25-1874 *25-1916 *26-1958 *26-1959 *26-2017
		*27-2066 27-2140 *27-2160 *29-2227 *29-2249 *30-2315 *30-2386 *31-2486 *31-2564
		*32-2600 *32-2616 *33-2649 *33-2716 *34-2746 *34-2775 *35-2829 *35-2886 *36-2964
		*36-2982 *36-3019 *37-3067 37-3143 *37-3149 *37-3151 *39-3323 *49-4053
		*49-4076 49-4077 *49-4079 *56-4693 *56-4695 *56-4746 *56-4748 *56-4767 *56-4769
		*57-4810 *57-4812 *57-4849 *57-4853 *57-4856 *57-4858 *63-5468 *63-5470
DDIN	002254	#10-697 20-1467 25-1893 26-1978 27-2139 30-2328 30-2351 30-2367 31-2496
		31-2505 31-2522 31-2539 31-2556 33-2682 33-2684 34-2752 34-2761 35-2862
		35-2879 36-2999 36-3009
DDISP	= 177570	#8-277 9-393 10-831
DDOUT	002242	#10-672 18-1387 21-1592 24-1821 26-1988 33-2663 33-2687
DDR	= 177732	#8-376 *10-672 18-1388 18-1390 19-1433 19-1435 20-1470 20-1473 20-1475
		*22-1678 *23-1751 25-1896 25-1899 26-1980 26-1982 36-3013 49-4067 49-4071
		49-4073
DH1	031422	10-399 10-447 10-460 10-548 #64-5627
DH13	032013	10-466 10-479 #64-5637
DH16	032066	10-492 10-504 10-510 10-523 10-529 10-554 10-560 10 566 10-572
		10-578 10-584 10-590 #64-5639
DH2	031471	10-405 10-412 10-453 #64-5629
DH20	032125	10-498 #64-5641
DH23	032205	10-516 #64-5643
DH27	032316	10-542 #64-5645
DH3	031563	10-419 #64-5631
DH4	031653	10-426 #64-5633
DH5	031731	10-433 10-440 10-472 10-485 10-535 #64-5635
DISPLA	001142	#9-393 *10-831 *10-831 64-5570 64-5580
DISPRE	000174	#8-383 10-831
DSWR	= 177570	#8-277 9-393 10-831
DT1	032414	10-400 10-448 10-461 #64-5648
DT16	032474	10-493 10-505 10-511 10-524 10-530 10-555 10-561 10-567 10-573
		10-585 10-591 #64-5652
DT2	032426	10-407 10-414 10-455 #64-5649
DT20	032504	10-499 #64-5653
DT21	032520	10-579 #64-5654
DT23	032530	10-518 #64-5655
DT27	032550	10-543 #64-5656
DT3	032444	10-421 #64-5650
DT30	032564	10-549 #64-5657
DT4	032460	10-428 10-435 10-442 10-467 10-474 10-480 10-487 10-537 #64-5651
EMTSV	002530	#10-830
EMTVEC	= 000030	#8-277 *10-831 *10-831
EM1	027234	10-398 #64-5595

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES
GTSWR	104406	10-836 #64-5588
HT	000011	#8-277 64-5571 64-5571
IOTVEC	000020	#8-277 *10-831 *10-831
IUBE	002502	#10-817 40-3408 41-3486 42-3611 42-3632 42-3649 42-3666 43-3710 44-3773
		45-3866 46-3914 47-3952 48-3982 49-4040 50-4131 51-4214 52-4295 53-4415
		54-4547 55-4619 56-4703 57-4821 58-4893 59-4952 60-5065 60-5152 61-5241
		61-5321 62-5367 63-5458 64-5521
KDPARO	172360	#8-278
KDPAR1	172362	#8-278
KDPAR2	172364	#8-278
KDPAR3	172366	#8-278
KDPAR4	172370	#8-278
KDPAR5	172372	#8-278
KDPAR6	172374	#8-278
KDPAR7	172376	#8-278
KDPDRO	172320	#8-278
KDPDR1	172322	#8-278
KDPDR2	172324	#8-278
KDPDR3	172326	#8-278
KDPDR4	172330	#8-278
KDPDR5	172332	#8-278
KDPDR6	172334	#8-278
KDPDR7	172336	#8-278
KIPARO	172340	#8-278
KIPAR1	172342	#8-278
KIPAR2	172344	#8-278
KIPAR3	172346	#8-278
KIPAR4	172350	#8-278
KIPAR5	172352	#8-278
KIPAR6	172354	#8-278 *10-773 *10-775 10-777 10-847 *21-1568 21-1595 *21-1608 *22-1673
		*23-1740 *23-1744 *24-1815 *25-1889 25-1905 *25-1910 *25-1912 25-1913 *26-1972
		*26-1979 *26-1989 *26-1998 *27-2076 *27-2078 *27-2090 *27-2126 27-2146 *27-2154
		*38-3241 *38-3254 38-3255 *53-4414 *54-4546 *57-4817 57-4841 *57-4848 64-5655
KIPAR7	172356	#8-278
KIPDRO	172300	#8-278
KIPDR1	172302	#8-278
KIPDR2	172304	#8-278
KIPDR3	172306	#8-278
KIPDR4	172310	#8-278
KIPDR5	172312	#8-278
KIPDR6	172314	#8-278
KIPDR7	172316	#8-278
KMCR	177734	#8-377 10-791 16-1268 16-1278 *16-1283 16-1284 16-1286 16-1288 *16-1289
		16-1295 21-1547 21-1577 *21-1578 21-1581 *21-1614 22-1658 23-1738 23-1742
		*23-1743 *23-1760 24-1802 25-1868 26-1953 27-2069 27-2074 27-2098 *27-2100
		27-2101 27-2105 27-2108 27-2109 *27-2110 *27-2127 *27-2151 27-2152 *27-2157
		*28-2182 28-2183 *29-2231 *29-2232 29-2233 29-2235 29-2236 *29-2239 29-2240
		29-2243 29-2244 *30-2316 30-2322 *30-2329 30-2330 *30-2336 30-2337 *30-2350
		30-2352 *30-2358 30-2359 *30-2366 30-2368 *30-2374 30-2375 *30-2385 *31-2488
		31-2491 *31-2504 31-2506 *31-2511 31-2512 *31-2521 31-2523 *31-2528 31-2529
		*31-2538 31-2540 *31-2545 31-2546 *31-2555 31-2557 *31-2562 31-2563 *32-2599
		*32-2602 32-2603 32-2605 32-2607 *32-2609 32-2610 *32-2615 *33-2650 *33-2654

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES
		33-2655 *33-2656 33-2657 *33-2664 33-2665 33-2668 *33-2670 33-2671 33-2674
		*33-2679 *33-2680 33-2688 33-2690 33-2692 *33-2694 33-2695 33-2697 33-2699
		33-2701 33-2704 33-2706 33-2708 33-2710 33-2712 *33-2714 *34-2748 *34-2756
		34-2757 *34-2758 34-2759 *34-2762 34-2763 34-2765 *34-2767 34-2768 34-2771
		*34-2773 35-2827 *35-2849 *35-2850 *35-2851 35-2864 35-2880 *36-2983 36-3010
		37-3145 38-3213 38-3218 38-3231 51-4209 53-4411 56-4685 56-4686 *56-4694
		56-4744 *56-4747 *56-4768 57-4809 *57-4811 *57-4850 57-4851 *57-4857 *63-5459
		63-5460 *63-5469 63-5491 *63-5497 *64-5527 *64-5528 64-5540 64-5554 64-5653
LF	= 000012	*8-277 64-5571 64-5571
MAPHO	= 170202	*8-351
MAPH00	= 170202	*8-286 8-351 *23-1746 *24-1818 *26-1974 *26-2001 *26-2002 *27-2122 *27-2150
		*30-2321 *31-2490 *33-2653 *34-2750 *36-2981 *51-4226 *52-4307 *54-4559 *55-4631
		*57-4814 *57-4845 *61-5282 61-5296 *63-5466
MAPH01	= 170206	*8-288 8-353 *35-2860 *35-2861 *35-2877 *35-2878 *63-5478 *64-5530 64-5656
MAPH02	= 170212	*8-290 8-355
MAPH03	= 170216	*8-292 8-357
MAPH04	= 170222	*8-294 8-359
MAPH05	= 170226	*8-296 8-361
MAPH06	= 170232	*8-298 8-363
MAPH07	= 170236	*8-300 8-365
MAPH1	= 170206	*8-353
MAPH10	= 170242	*8-302
MAPH11	= 170246	*8-304
MAPH12	= 170252	*8-306
MAPH13	= 170256	*8-308
MAPH14	= 170262	*8-310
MAPH15	= 170266	*8-312
MAPH16	= 170272	*8-314
MAPH17	= 170276	*8-316
MAPH2	= 170212	*8-355
MAPH20	= 170302	*8-318
MAPH21	= 170306	*8-320
MAPH22	= 170312	*8-322
MAPH23	= 170316	*8-324
MAPH24	= 170322	*8-326
MAPH25	= 170326	*8-328
MAPH26	= 170332	*8-330
MAPH27	= 170336	*8-332
MAPH3	= 170216	*8-357
MAPH30	= 170342	*8-334
MAPH31	= 170346	*8-336
MAPH32	= 170352	*8-338
MAPH33	= 170356	*8-340
MAPH34	= 170362	*8-342
MAPH35	= 170366	*8-344
MAPH36	= 170372	*8-346 *56-4758
MAPH37	= 170376	*8-348 *22-1677 62-5399 *63-5464
MAPH4	= 170222	*8-359
MAPH5	= 170226	*8-361
MAPH6	= 170232	*8-363
MAPH7	= 170236	*8-365
MAPLO	= 170200	*8-350

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES
MMR3	= 172516	*8-372	*10-842	*10-846	*21-1566	*21-1616	*22-1667	*22-1685	*23-1747	*23-1757
		*24-1811	*24-1823	*25-1876	*26-1961	*27-2119	*27-2123	*27-2158	*29-2228	*29-2230
		29-2247	*30-2319	*30-2384	*31-2487	*32-2601	*33-2651	*33-2715	*34-2747	*34-2774
		*35-2832	*35-2852	*36-2980	*38-3235	*39-3324	*49-4045	*50-4132	*51-4227	*51-4241
		*52-4308	*52-4317	*53-4432	*53-4441	*53-4465	*53-4474	*54-4560	*54-4569	*55-4632
		*55-4652	*56-4699	*57-4802	*57-4808	*57-4855	*61-5330	*63-5467	*63-5496	*64-5526
		*64-5544								
MMVEC	= 000250	*8-278								
NOABRT	026344	64-5580	64-5580	*64-5580						
NOCAM	007706	28-2191	*28-2195							
NROM	013612	37-3091	*37-3162							
OBADR	002172	*10-620	36-2972	36-2985	36-2995	36-3005				
PCR	= 177522	*8-379	*37-3069							
PIRQ	= 177772	*8-277	*43-3723	*43-3735						
PIRQVE	= 000240	*8-277	*43-3713							
PMIS	001720	*10-597	*10-847	*16-1273	17-1323	18-1376	19-1422	20-1460	21-1543	22-1654
		23-1727	24-1798	25-1864	26-1951	26-1963	27-2063	27-2120	28-2180	29-2225
		30-2312	31-2483	32-2597	33-2646	34-2740	35-2824	35-2833	35-2839	35-2841
		36-2959	53-4492	57-4795	57-4803	64-5551				
POLY	= 120001	*8-380								
PRO	= 000000	*8-277								
PR1	= 000040	*8-277								
PR2	= 000100	*8-277								
PR3	= 000140	*8-277								
PP4	= 000200	*8-277								
Prj	= 000240	*8-277								
PR6	= 000300	*8-277								
PR7	= 000340	*8-277								
PS	= 177776	*8-277	8-277							
PSW	= 177776	*8-277	*40-3409	*40-3418	40-3419	*40-3421	*41-3487	*41-3505	*41-3520	*41-3535
		*42-3618	*42-3620	*42-3635	*42-3637	*42-3652	*42-3654	*42-3669	*42-3671	*43-3721
		*43-3724	*44-3786	*44-3792	*47-3954	*47-3962	*48-3986	*48-3990	*61-5332	
PTRN16	001774	*10-604	12-986	18-1380	19-1426	36-2984	49-4047	49-4088	50-4138	51-4215
		52-4301								
PTRN6	002024	*10-607	12-1009							
PWRVEC	= 000024	*8-277	*10-831	*10-831	45-3870	*45-3871	45-3872	*45-3880	*45-3886	*64-5589
		*64-5589	*64-5589	*64-5589	*64-5589	*64-5589				
		64-5574	*64-5588							
RDCHR	= 104410	*64-5588								
RDDEC	= 104413	*64-5588								
RDLIN	= 104411	64-5575	64-5576	*64-5588						
RDOCT	= 104412	*64-5588								
RESTAR	003320	*10-848	64-5561							
RESVEC	= 000010	*8-277								
R6	=#000006	*8-277	*10-831	*10-831	10-831					
R7	=#000007	*8-277								
SAV30	002616	10-830	*10-830	*10-830	64-5580	64-5580				
SAV32	002620	*10-830	*10-830	64-5580	64-5580					
SCOPE	= 000004	*8-277	11-881	12-966	13-1052	14-1118	15-1191	16-1260	17-1318	18-1374
		19-1420	20-1459	21-1541	22-1652	23-1725	24-1796	25-1862	26-1949	27-2062
		28-2178	29-2223	30-2310	31-2481	32-2595	33-2644	34-2738	35-2822	36-2958
		37-3062	38-3208	39-3321	40-3406	41-3484	42-3609	43-3709	44-3769	45-3848
		46-3912	47-3950	48-3980	49-4037	50-4129	51-4207	52-4293	53-4409	54-4544

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES								
		55-4618	56-4683	57-4793	58-4891	59-4950	60-5063	61-5239	62-5366	63-5456
		64-5519	64-5548	64-5561						
SIMLGO	= 170014	#8-381								
SRO	= 177572	#8-278								
SR1	= 177574	#8-278								
SR2	= 177576	#8-278								
SR3	= 172516	#8-278	*11-892	*11-903	*11-907					
STACK	= 001100	#8-277	10-831							
START	002530	8-386	8-389	#10-830						
STKLMT	= 177774	#8-277								
SWR	001140	#9-393	10-831	*10-831	10-831	*10-831	*10-831	10-836	64-5570	64-5570
		64-5570	64-5570	64-5570	64-5574	64-5574	64-5580	64-5580	64-5580	64-5580
		64-5589	64-5589							
SWREG	000176	#8-383	10-831	10-836	64-5574	64-5574				
SW0	= 000001	#8-277								
SW00	= 000001	#8-277	8-277							
SW01	= 000002	#8-277	8-277							
SW02	= 000004	#8-277	8-277							
SW03	= 000010	#8-277	8-277							
SW04	= 000020	#8-277	8-277							
SW05	= 000040	#8-277	8-277							
SW06	= 000100	#8-277	8-277							
SW07	= 000200	#8-277	8-277							
SW08	= 000400	#8-277	8-277							
SW09	= 001000	#8-277	8-277							
SW1	= 000002	#8-277								
SW10	= 002000	#8-277								
SW11	= 004000	#8-277								
SW12	= 010000	#8-277								
SW13	= 020000	#8-277								
SW14	= 040000	#8-277								
SW15	= 100000	#8-277								
SW2	= 000004	#8-277								
SW3	= 000010	#8-277								
SW4	= 000020	#8-277								
SW5	= 000040	#8-277								
SW6	= 000100	#8-277								
SW7	= 000200	#8-277								
SW8	= 000400	#8-277								
SW9	= 001000	#8-277								
TBITVE	= 000014	#8-277								
TBLMH	012732	35-2854	#35-2899							
TBLML	012672	35-2853	#35-2892							
TEST	001716	#10-596	*10-839	*64-5582	64-5648	64-5649	64-5650	64-5651	64-5652	64-5653
		64-5654	64-5655	64-5656	64-5657					
TIMOUT	002266	#10-715	10-837	15-1202	16-1269	17-1343	21-1563	22-1686	27-2092	27-2156
		39-3380								
TKVEC	= 000060	#8-277								
TOPTBL	010564	30-2318	#30-2403							
TOUT	001730	#10-601	*21-1590	21-1593						
TPVEC	= 000064	#8-277								
TRAPVE	= 000034	#8-277	*10-831	*10-831						

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES	CREF	V02
TRTVEC	000014	#8-277		
TST1	003324	#11-881		
TST10	004510	17-1324 #18-1374		
TST11	004642	18-1377 #19-1420		
TST12	004744	19-1423 #20-1459		
TST13	005066	20-1461 #21-1541		
TST14	005472	21-1544 #22-1652		
TST15	005670	22-1655 22-1659 #23-1725		
TST16	006074	23-1728 23-1759 #24-1796		
TST17	006264	24-1799 24-1803 24-1826 #25-1862		
TST2	003450	11-905 #12-966		
TST20	006536	25-1865 25-1869 #26-1949		
TST21	007136	26-1952 26-1954 #27-2062		
TST22	007616	27-2097 #28-2178		
TST23	007726	28-2181 28-2184 #29-2223		
TST24	010120	29-2226 #30-2310		
TST25	010600	30-2387 #31-2481		
TST26	011310	31-2566 #32-2595		
TST27	011452	32-2598 #33-2644		
TST3	003664	#13-1052		
TST30	012134	#34-2738		
TST31	012350	34-2741 #35-2822		
TST32	012772	35-2628 35-2887 #36-2958		
TST33	013224	36-2960 #37-3062		
TST34	013650	37-3160 #38-3208		
TST35	014050	38-3214 #39-3321		
TST36	014236	39-3382 #40-3406		
TST37	014340	#41-3484		
TST4	004006	#14-1118		
TST40	014620	#42-3609		
TST41	015160	#43-3709		
TST42	015314	#44-3769		
TST43	015552	44-3772 44-3809 #45-3848		
TST44	015756	45-3853 45-3855 45-3857 45-3859 45-3863 45-3865 #46-3912		
TST45	016050	#47-3950		
TST46	016130	#48-3980		
TST47	016230	48-3992 48-3994 #49-4037		
TST5	004064	#15-1191		
TST50	016444	#50-4129		
TST51	016542	#51-4207		
TST52	016710	#52-4293		
TST53	017022	#53-4409		
TST54	017452	53-4493 #54-4544		
TST55	017616	54-4576 #55-4618		
TST56	017766	#56-4683		
TST57	020414	#57-4793		
TST6	004250	15-1229 #16-1260		
TST60	020766	57-4796 #58-4891		
TST61	021050	58-4909 #59-4950		
TST62	021154	#60-5063		
TST63	021570	#61-5239		
TST64	022334	#62-5366		

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES
TST65	022520	#63-5456
TST66	022742	63-5499 #64-5519
TST67	023122	#64-5548
TST7	004426	16-1274 16-1291 #17-1318
TYPDS	= 104405	64-5561 64-5561 #64-5588
TYPE	= 104401	10-836 28-2191 28-2192 37-3091 37-3095 64-5561 64-5561 64-5561 64-5571
		64-5572 64-5573 64-5574 64-5574 64-5574 64-5574 64-5574 64-5574 64-5574 64-5574
		64-5574 64-5574 64-5576 64-5576 64-5580 64-5580 64-5587 64-5587 64-5587 64-5587
		64-5587 64-5587 64-5587 64-5587 #64-5588 64-5589 64-5589
TYPOC	= 104402	64-5574 64-5587 #64-5588
TYPON	= 104404	#64-5588
TYPOS	= 104403	37-3092 #64-5588
UBECT	001722	#10-598 #39-3366 39-3372 39-3381 44-3771 61-5263
UBEM	023122	39-3383 51-4213 63-5462 #64-5548
UBETST	013224	28-2193 #37-3062
UFDLFG	002622	*10-830 #10-830 64-5580
UFDSET	= 000001	#8-373 10-830 64-5580 64-5580
UMSIZ	002456	#10-791 38-3225
UQUIET	002624	*10-830 #10-830 10-832 64-5561 64-5580 64-5580
VALTBL	010550	30-2317 #30-2392
VMKOR	002626	10-830 10-830 10-830 #10-830
WRTBUF	001734	#10-603 49-4046 49-4087 51-4225 51-4242 58-4894 59-4958 59-4967 59-4978
#APTHD	000232	8-392 #8-392
#ASTAT	= *****	64-5586 64-5586
#ATYC	026414	64-5586 #64-5586
#ATY1	026370	#64-5586
#ATY3	026376	64-5571 #64-5586
#ATY4	026406	64-5580 #64-5586
#AUTOB	001134	#9-393 *10-836 64-5574 64-5574 64-5574
#BASE	001254	#9-393
#BDADR	001122	#9-393 *10-715 *11-915 *12-984 *12-994 *12-1007 *12-1017 *13-1071 *13-1082
		*15-1235 *16-1288 *16-1295 *21-1595 *21-1597 *24-1828 *25-1894 *25-1895 *25-1896
		25-1898 *25-1900 *26-1976 26-1980 26-1983 *26-1984 26-1990 26-1992 *26-1993
		*26-2003 *26-2004 *26-2005 *26-2006 *27-2109 *53-4454 *53-4487 *54-4579 64-5648
#BDDAT	001126	64-5649 64-5650 64-5653 64-5654 64-5655
		#9-393 *12-982 *12-992 *12-1005 *12-1015 *14-1137 *15-1211 *15-1230 *16-1287
		*18-1390 *18-1394 *19-1435 *20-1475 *21-1602 *24-1827 *25-1899 *26-1982 *26-1992
		*27-2108 *29-2235 *29-2243 *30-2330 30-2331 *30-2337 30-2338 *30-2352 30-2353
		*30-2359 30-2360 *30-2368 30-2369 *30-2375 30-2376 *31-2506 31-2507 *31-2512
		31-2513 *31-2523 31-2524 *31-2529 31-2530 *31-2540 *31-2541 *31-2546 31-2547
		*31-2557 31-2558 *31-2563 31-2565 *32-2607 *32-2610 32-2611 *33-2668 *33-2674
		*33-2692 *33-2699 *33-2706 *33-2712 *34-2765 *34-2771 *49-4073 *53-4452 *53-4485
		*54-4577 64-5649 64-5651 64-5655 64-5580
#BELL	001170	#9-393 64-5580 64-5580 64-5580
#CDW1	001260	#9-393
#CDW2	001262	#9-393
#CHARC	024242	*64-5571 *64-5571 64-5571 *64-5571 #64-5571
#CKSWR	024720	#64-5574 64-5588 64-5588
#CMTAG	001100	#9-393 10-831 10-831 10-831 10-831 10-831 10-831
#CM3	= 000000	#9-393 9-393
#CM4	= 000002	#9-393 9-393 9-393 #9-393 9-393 9-393 #9-393
#CNTLG	025455	64-5574 #64-5574

SYMBOL CROSS REFERENCE

CREF VO2

SYMBOL	VALUE	REFERENCES
#CNTLU	025450	64-5574 #64-5574
#CPUOP	001226	#9-393
#CRLF	001175	#9-393 28-2192 37-3095 64-5561 64-5571 64-5571 64-5571 64-5574 64-5574 64-5574 64-5576 64-5576 64-5580 64-5580 64 5580 64-5587 64-5587 64-5587
#DBLK	024710	64-5573 64-5573 #64-5573
#DDW0	001264	#9-393
#DDW1	001266	#9-393
#DDW10	001310	#9-393
#DDW11	001312	#9-393
#DDW12	001314	#9-393
#DDW13	001316	#9-393
#DDW14	001320	#9-393
#DDW15	001322	#9-393
#DDW2	001270	#9-393
#DDW3	001272	#9-393
#DDW4	001274	#9-393
#DDW5	001276	#9-393
#DDW6	001300	#9-393
#DDW7	001302	#9-393
#DDW8	001304	#9-393
#DDW9	001306	#9-393
#DEVCT	001210	#9-393
#DEVM	001256	#9-393
#DOAGN	023350	64-5561 64-5561 #64-5561
#DTBL	024700	64-5573 #64-5573
#ENDAD	023340	8-391 10-836 #64-5561 64-5580 64-5580
#ENDCT	023220	10-831 #64-5553 #64-5556 #64-5561
#ENULL	023354	#64-5561
#ENV	001220	#9-393 10-834 10-836 28-2187 37-3086 45-3856 64-5571 64-5580 64-5586 64-5586
#ENVH	001221	#9-393 10-831 64-5571 64-5571 64-5586
#EOP	023164	64-5550 64-5552 64-5555 #64-5561
#EOPCT	023212	*10-831 #64-5561 64-5561
#ERFLG	001103	#9-393 64-5570 64-5570 64-5570 #64-5570 64-5570 64 5570 #64 5580 64-5580 64-5580
#ERMAX	001115	#9-393 *10-831 64-5570 #64-5570 64-5570 64-5570
#ERROR	025764	10-831 #64-5580
#ERRPC	001116	#9-393 #64-5580 #64-5580 64-5580 64-5580 64-5580 64-5587 64-5648 64-5649 64-5650 64-5651 64-5652 64-5653 64-5655 64-5656 64-5657
#ERRTB	001324	#10-393 64-5587
#ERRTY	026636	64-5584 #64-5587
#ERTTL	001112	#9-393 64-5561 #64-5580 64-5580 64-5580
#ESCAP	001166	#9-393 *10-831 #64-5570 64-5580 64-5580 64-5580 64-5580 64-5580
#ETABL	001220	#9-393
#ETEND	001324	8-392 #9-393
#FATAL	001202	#9-393 #64-5586
#FFLG	026634	*64-5586 #64-5586 64-5586 #64-5586 #64-5586
#FILLC	001156	#9-393 64-5571 64-5571
#FILLS	001155	#9-393 64-5571 64-5571
#GADR	001120	#9-393 *13-1070 *13-1081 64-5650
#GDDAT	001124	#9-393 10-672 *12-983 *12-993 *12-1006 *12-1016 *14-1136 *15-1212 *15-1217

SYMBOL CROSS REFERENCE

CREF V02

SEQ 0174

SYMBOL	VALUE	REFERENCES
		*15-1218 *16-1286 *18-1386 18-1388 18-1392 *19-1427 19-1429 19-1433 *20-1465
		*20-1472 *21-1591 21-1600 *22-1674 22-1678 *24-1819 *25-1898 *26-1977 *26-1983
		*26-1986 26-1990 *26-2013 *27-2103 *27-2104 27-2105 *27-2107 *29-2236 *29-2237
		*29-2244 *29-2245 *30-2322 *30-2323 *30-2334 *30-2341 *30-2356 *30-2363 *30-2372
		*30-2379 *31-2491 *31-2509 *31-2515 *31-2526 *31-2532 *31-2543 *31-2549 *31-2560
		*31-2567 *32-2603 *32-2604 *32-2613 *33-2662 *33-2667 *33-2673 *33-2686 *33-2690
		*33-2691 *33-2697 *33-2698 *33-2701 *33-2702 *33-2710 *33-2711 *34-2757 *34-2763
		*34-2770 *49-4074 *53-4453 *53-4486 *54-4578 64-5649 64-5651 64-5655
#GET42	023330	*64-5561
#GTSWR	024770	*64-5574 64-5588 64-5588
#HD	000001	8-274 8-274 8-274
#HIBTS	000232	*8-392
#HIOCT	025604	*64-5575 *64-5575
#ICNT	001104	*9-393 *64-5570 64-5570 *64-5570 64-5570 64-5570
#ILLUP	027216	64-5589 64-5589 *64-5589
#INTAG	001135	*9-393 64-5574 64-5574 64-5574
#ITEMB	001114	*9-393 *64-5580 64-5580 64-5580 64-5580 64-5583 64-5587
#LF	001176	*9-393 64-5571 64-5571 64-5574 64-5574 64-5574 64-5576 64-5576 64-5580
		64-5580
#LFLG	026633	*64-5586 *64-5586
#LPADR	001106	*9-393 *10-831 *64-5570 *64-5570 64-5570 64-5570 64-5570
#LPERR	001110	*9-393 *10-831 64-5570 *64-5570 64-5570 64-5570 64-5580
#MADR1	001232	*9-393
#MADR2	001236	*9-393
#MADR3	001242	*9-393
#MADR4	001246	*9-393
#MAIL	001200	8-392 8-392 *9-393 10-831 10-836 64-5570 64-5571 64-5580
#MAMS1	001230	*9-393
#MAMS2	001234	*9-393
#MAMS3	001240	*9-393
#MAMS4	001244	*9-393
#MBADR	000234	*8-392
#MFLG	026632	*64-5586 64-5586 *64-5586 *64-5586
#MNEW	025473	64-5574 *64-5574
#MSGAD	001214	*9-393 *64-5586 64-5586
#MSGLG	001216	*9-393 *64-5586
#MSGTY	001200	*9-393 64-5586 *64-5586 64-5586 *64-5586
#MSWR	025462	64-5574 *64-5574
#MTYP1	001231	*9-393
#MTYP2	001235	*9-393
#MTYP3	001241	*9-393
#MTYP4	001245	*9-393
#MXCNT	023710	64-5570 64-5570 64-5570 *64-5570
#NULL	001154	*9-393 64-5571 64-5571 64-5571
#NWTST	000001	*11-881 11-881 *11-881 *12-966 *12-966 *13-1052 13-1052 *13-1052
		*14-1118 14-1118 *14-1118 *15-1191 15-1191 *15-1191 *16-1260 16-1260 *16-1260
		*17-1318 17-1318 *17-1318 *18-1374 18-1374 *18-1374 *19-1420 19-1420 *19-1420
		*20-1459 20-1459 *20-1459 *21-1541 21-1541 *21-1541 *22-1652 22-1652 *22-1652
		*23-1725 23-1725 *23-1725 *24-1796 24-1796 *24-1796 *25-1862 25-1862 *25-1862
		*26-1949 26-1949 *26-1949 *27-2062 27-2062 *27-2062 *28-2178 28-2178 *28-2178
		*29-2223 29-2223 *29-2223 *30-2310 30-2310 *30-2310 *31-2481 31-2481 *31-2481
		*32-2595 32-2595 *32-2595 *33-2644 33-2644 *33-2644 *34-2738 34-2738 *34-2738

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES
		64-5570	64-5570	64-5570	64-5570	64-5570	64-5570	64-5570	64-5570	64-5580
		64-5580	64-5580	64-5580	64-5580	64-5580	64-5580	64-5580	64-5580	64-5580
		64-5580	64-5589							
#SWREG	001222	#9-393	10-831							
#SWRMK	000300	#8-273	8-275	8-275	8-275	8-275	8-275	8-275	8-275	8-275
		8-275	64-5570	64-5570	64-5570	64-5570	64-5570	64-5570	64-5570	64-5570
		64-5570	64-5570	64-5570						
#TESTN	001204	#9-393	*64-5570							
#TIMES	001164	#9-393	*10-831	*64-5561	*64-5570	64-5570	*64-5570	64-5570	64-5570	
#TKB	001146	#9-393	64-5571	64-5571	64-5571	64-5571	64-5574	64-5574	64-5574	64-5574
		64-5574	64-5574							
#TKS	001144	#9-393	64-5571	64-5571	64-5571	64-5571	64-5574	64-5574	64-5574	64-5574
		64-5574	64-5574							
#TMPO	001160	*8-385	*8-388	#9-393	*16-1268	*16-1270	16-1271	16-1289	18-1381	18-1392
		18-1394	*19-1429	*26-1964	26-2014	27-2110	*29-2240	29-2241	*35-2855	35-2861
		35-2878	*37-3071	*37-3096	*37-3136	37-3159	40-3411	*45-3870	45-3886	46-3919
		48-3984	48-3991	*53-4447	*53-4448	*53-4449	53-4450	53-4453	53-4454	*53-4480
		*53-4481	*53-4482	53-4483	53-4486	53-4487	*56-4688	*56-4691	56-4739	56-4753
		*57-4809	57-4857	*61-5245	61-5262					
#TMP1	001162	#9-393	*21-1577	21-1614	*23-1742	23-1760	*26-1975	26-1979	26-1989	*26-2009
		*26-2010	26-2011	26-2014	*27-2098	27-2157	*53-4421	*53-4499	53-4500	53-4501
		53-4502	*56-4685	56-4768						
#TN	000070	8-274	#8-274	11-881	11-881	#11-881	11-905	12-966	12-966	#12-966
		13-1052	13-1052	#13-1052	14-1118	14-1118	#14-1118	15-1191	15-1191	#15-1191
		15-1229	16-1260	16-1260	#16-1260	16-1274	16-1291	17-1318	17-1318	#17-1318
		17-1324	18-1374	18-1374	#18-1374	18-1377	19-1420	19-1420	#19-1420	19-1423
		20-1459	20-1459	#20-1459	20-1461	21-1541	21-1541	#21-1541	21-1544	22-1652
		22-1652	#22-1652	22-1655	22-1659	23-1725	23-1725	#23-1725	23-1728	23-1759
		24-1796	24-1796	#24-1796	24-1799	24-1803	24-1826	25-1862	25-1862	#25-1862
		25-1865	25-1869	26-1949	26-1949	#26-1949	26-1952	26-1954	27-2062	27-2062
		#27-2062	27-2097	28-2178	28-2178	#28-2178	28-2181	28-2184	29-2223	29-2223
		#29-2223	29-2226	30-2310	30-2310	#30-2310	30-2387	31-2481	31-2481	#31-2481
		31-2566	32-2595	32-2595	#32-2595	32-2598	33-2644	33-2644	#33-2644	34-2738
		34-2738	#34-2738	34-2741	35-2822	35-2822	#35-2822	35-2828	35-2887	36-2958
		36-2958	#36-2958	36-2960	37-3062	37-3062	#37-3062	37-3160	38-3208	38-3208
		#38-3208	38-3214	39-3321	39-3321	#39-3321	39-3382	40-3406	40-3406	#40-3406
		41-3484	41-3484	#41-3484	42-3609	42-3609	#42-3609	43-3709	43-3709	#43-3709
		44-3769	44-3769	#44-3769	44-3772	44-3809	45-3848	45-3848	#45-3848	45-3853
		45-3855	45-3857	45-3859	45-3863	45-3865	46-3912	46-3912	#46-3912	47-3950
		47-3950	#47-3950	48-3980	48-3980	#48-3980	48-3992	48-3994	49-4037	49-4037
		#49-4037	50-4129	50-4129	#50-4129	51-4207	51-4207	#51-4207	52-4293	52-4293
		#52-4293	53-4409	53-4409	#53-4409	53-4493	54-4544	54-4544	#54-4544	54-4576
		55-4618	55-4618	#55-4618	56-4683	56-4683	#56-4683	57-4793	57-4793	#57-4793
		57-4796	58-4891	58-4891	#58-4891	58-4909	59-4950	59-4950	#59-4950	60-5063
		60-5063	#60-5063	61-5239	61-5239	#61-5239	62-5366	62-5366	#62-5366	63-5456
		63-5456	#63-5456	63-5499	64-5519	64-5519	#64-5519	64-5548	64-5548	#64-5548
#TPB	001152	#9-393	64-5571	64-5571	64-5571	64-5571				
#TPFLG	001157	#9-393	64-5571	64-5571	64-5571	64-5571				
#TPS	001150	#9-393	64-5571	64-5571	64-5571	64-5571				
#TRAP	026772	10-831	#64-5588							
#TRAP2	027014	#64-5588	64-5588							
#TRP	000014	#64-5588	64-5588	64-5588	64-5588	64-5588	#64-5588	64-5588	64-5588	64-5588

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES
		64-5588 #64-5588 64-5588 64 5588 64-5588 64-5588 #64-5588 64-5588 64-5588
		64-5588 64-5588 #64-5588 64-5588 64-5588 64-5588 #64-5588 64-5588 #64-5588 64-5588
		64-5588 64-5588 64-5588 #64-5588 64-5588 64-5588 #64-5588 64-5588 #64-5588 64-5588
		64-5588 #64-5588 64-5588 64-5588 64-5588 #64-5588 64-5588 #64-5588 64-5588
#TRPAD	027026	64-5588 #64-5588
#TSTM	000236	#8-392
#TSTNM	001102	#9-393 *64-5561 64-5570 64-5570 *64-5570 64-5570 64-5570 64-5570
		64-5570 64-5570 64-5570 64-5570 64-5570 64-5570 64-5570 64-5570
#TTYIN	025440	64-5574 64-5574 64-5574 #64-5574
#TYPBN	= *****	64-5588
#TYPDS	024474	#64-5573 64-5588 64-5588
#TYPE	023712	#64-5571 64-5586 64-5588 64-5588
#TYPEC	024124	64-5571 64-5571 64-5571 #64-5571 64-5574
#TYPEX	024244	64-5571 64-5571 #64-5571
#TYPOC	024272	#64-5572 64-5588 64-5588
#TYPON	024306	64-5572 #64-5572 64-5588
#TYPOS	024246	#64-5572 64-5588
#UNIT	001212	#9-393
#UNITM	000242	#8-392
#USWR	001224	#9-393
#VECT1	001250	#9-393
#VECT2	001252	#9-393
#XOFF	= 000023	64-5571 64-5571
#XON	= 000021	64-5571 64-5571 64-5574
#XTSTR	023400	#64-5570
##GET4	= 000000	#64-5561 64-5561
#OFILL	024471	*64-5572 #64-5572 64-5572 #64-5572
#4OCAT	= *****	64-5570 64-5580
.#ASTA	= *****	64-5586 64-5586
.#X	= 000232	#8-392 8-392

