

KTJ11-B

KTJ11-B DIAGNOSTIC
COKTABO

COPYRIGHT (c) 1984
AH-T876B-MC
FICHE 01 OF 01

FEB 1985

digital

Made In USA

Frame 01	Frame 02	Frame 03	Frame 04	Frame 05	Frame 06	Frame 07	Frame 08	Frame 09	Frame 10
Frame 11	Frame 12	Frame 13	Frame 14	Frame 15	Frame 16	Frame 17	Frame 18	Frame 19	Frame 20
Frame 21	Frame 22	Frame 23	Frame 24	Frame 25	Frame 26	Frame 27	Frame 28	Frame 29	Frame 30
Frame 31	Frame 32	Frame 33	Frame 34	Frame 35	Frame 36	Frame 37	Frame 38	Frame 39	Frame 40
Frame 41	Frame 42	Frame 43	Frame 44	Frame 45	Frame 46	Frame 47	Frame 48	Frame 49	Frame 50
Frame 51	Frame 52	Frame 53	Frame 54	Frame 55	Frame 56	Frame 57	Frame 58	Frame 59	Frame 60
Frame 61	Frame 62	Frame 63	Frame 64	Frame 65	Frame 66	Frame 67	Frame 68	Frame 69	Frame 70
Frame 71	Frame 72	Frame 73	Frame 74	Frame 75	Frame 76	Frame 77	Frame 78	Frame 79	Frame 80
Frame 81	Frame 82	Frame 83	Frame 84	Frame 85	Frame 86	Frame 87	Frame 88	Frame 89	Frame 90
Frame 91	Frame 92	Frame 93	Frame 94	Frame 95	Frame 96	Frame 97	Frame 98	Frame 99	Frame 100

KTJ11-B DIAGNOSTIC

MACRO M1200 05 OCT 84 09:24 PAGE 2

.REM 6

IDENTIFICATION
... ..

PRODUCT CODE: AC T875B-MC

PRODUCT NAME: COMTABO KTJ11-B DIAGNOSTIC

PRODUCT DATE: JULY, 1984

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 RESPONSIBILITY
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 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL
DECPDP
DECUSUNIBUS
DECTAPEMASSBUS
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

1. ABSTRACT

THE FOLLOWING DIAGNOSTIC TESTS THE UNIBUS ADAPTER (UBA) MODULE, KTJ11-B. THE FUNCTIONALITY OF THE MODULE IS: UNIBUS - PHI BUS ADAPTER (WHERE PHI 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 20K 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.

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 PROMT 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.

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
0	PC	0
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 PERSENT IN THE SYSTEM. IT IS ALSO RESPONSIBLE FOR CONTROLLING AND DESCRIBING THE STATUS OF THE DMA CACHE.

7. TESTS LIST

179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235

TEST 1 - UNIBUS MAP REGISTER TESTS
TEST 2 - UNIBUS MAP REGISTER BIT PATTERN
TEST 3 - UNIBUS MAP REGISTER ADDRESS UNIQUENESS
TEST 4 - UNIQUENESS BETWEEN LOW MAP REGISTERS AND HIGH MAP REGISTERS
TEST 5 - DCSR REGISTER RESPONSE TEST
TEST 6 - KMCR BITS TEST
TEST 7 - UNIBUS TIMEOUT TEST
TEST 8 - DATA OUT WITHOUT RELOCATION
TEST 9 - DATA IN WITHOUT RELOCATION
TEST 10 - CONTENT OF DDR
TEST 11 - INDIRECT ACCESSING OF UNIBUS MAP REGISTERS
TEST 12 - DISABLING OF THE MAPPING REGISTERS
TEST 13 - NXM MEMORY TIMEOUT
TEST 14 - CARRY PROPOGATION TEST
TEST 15 - EXTENSIVE CARRY PROPOGATION TEST
TEST 16 - ALU TEST
TEST 17 - MAIN MEMORY DISABLE
TEST 18 - CACHE PRESENCE
TEST 19 - CACHE DISABLED AND KMCR
TEST 20 - AVAILABILITY OF SETS
TEST 21 - DEALLOCATION OF SETS
TEST 22 - CACHE WITH RELOCATION DISABLED
TEST 23 - WRITE CYCLES AND CACHE
TEST 24 - DMA READ WITH INDEX NOT ZERO
TEST 25 - TAG REGISTERS
TEST 26 - CACHE RAM BIT PATTERN TEST
TEST 27 - BOOT ROMS TEST
TEST 28 - UNIBUS MEMORY TEST
TEST 29 - UBE AUTOSIZING ROUTINE
TEST 30 - NPG ARBITRATION
TEST 31 - NO BUS GRANTS WITH PROCESSOR AT HIGHER PRIORITY
TEST 32 - BR7-BR4 ARBITRATION
TEST 33 - ARBITRATION BETWEEN INTERRUPTS AND PIRQ'S
TEST 34 - ARBITRATION BETWEEN INTERRUPTS FROM 2 UBE
TEST 35 - POWER DOWN TEST
TEST 36 - WRONG PARITY TEST
TEST 37 - NO SACK TIMEOUT
TEST 38 - NO INTERRUPT TEST
TEST 39 - UNIBUS DEVICE DATO CYCLE
TEST 40 - UNIBUS DEVICE DATI CYCLE
TEST 41 - UNIBUS DEVICE DATO CYCLE WITH RELOCATION ENABLED
TEST 42 - UNIBUS DEVICE DATI CYCLE WITH RELOCATION ENABLED
TEST 43 - ALU TEST USING UBE
TEST 44 - CARRY PROPOGATION TEST USING UBE
TEST 45 - NXM TEST USING UBE
TEST 46 - RELOCATION WITH UNIBUS MEMORY
TEST 47 - MAIN MEMORY DISABLE THRU UBE
TEST 48 - UNIBUS DEVICE DATOB CYCLE
TEST 49 - UNIBUS DEVICE DATIP CYCLE
TEST 50 - UNIBUS DEVICE I/O PAGE READ CYCLE
TEST 51 - UNIBUS DEVICE I/O PAGE WRITE CYCLE
TEST 52 - MAPPING REGISTERS TEST USING UBE
TEST 53 - UNIBUS DEVICE DATI CYCLE WITH CACHE ENABLED
TEST 54 - WRONG PARITY AND CACHE

ε

237
238
259
260
261
262

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.
;*
```

263

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>
;*
```

265

001100
104000
000004

```

.SBTTL BASIC DEFINITIONS
;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
STACK= 1100
ERROR= FMT          ;;BASIC DEFINITION OF ERROR CALL
SCOPE= IOT          ;;BASIC DEFINITION OF SCOPE CALL
;*MISCELLANEOUS DEFINITIONS
MT= 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
STKLM= 177774        ;;STACK LIMIT REGISTER
PIRQ= 177772         ;;PROGRAM INTERRUPT REQUEST REGISTER
DSWR= 177570         ;;HARDWARE SWITCH REGISTER
DDISP= 177570        ;;HARDWARE DISPLAY REGISTER
;*
```

000000
000001
000002
000003
000004
000005
000006
000007
000006
000007

```

;*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
;*
```

000000
000040

```

;*PRIORITY LEVEL DEFINITIONS
PRO= 0              ;;PRIORITY LEVEL 0
PRI= 40             ;;PRIORITY LEVEL 1
;*
```

BASIC DEFINITIONS

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	
000010	BIT3=	BIT03	

BASIC DEFINITIONS

```

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
; *BASIC "CPU" TRAP VECTOR ADDRESSES
; *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
172370      KDPAR4= 172370

```

MEMORY MANAGEMENT DEFINITIONS

172372
172374
172376

KDPAR5= 172372
KDPAR6= 172374
KDPAR7= 172376

.SBTTL UNIBUS MAP REGISTER DEFINITIONS

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

267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
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

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
170336

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
MAPH27 = 170336

UNIBUS MAP REGISTER DEFINITIONS

321	170340	MAPL30 = 170340
322	170342	MAPH30 = 170342
323	170344	MAPL31 = 170344
324	170346	MAPH31 = 170346
325	170350	MAPL32 = 170350
326	170352	MAPH32 = 170352
327	170354	MAPL33 = 170354
328	170356	MAPH33 = 170356
329	170360	MAPL34 = 170360
330	170362	MAPH34 = 170362
331	170364	MAPL35 = 170364
332	170366	MAPH35 = 170366
333	170370	MAPL36 = 170370
334	170372	MAPH36 = 170372
335	170374	MAPL37 = 170374
336	170376	MAPH37 = 170376

337		
338	170200	MAPL0 = MAPL00
339	170202	MAPH0 = MAPH00
340	170204	MAPL1 = MAPL01
341	170206	MAPH1 = MAPH01
342	170210	MAPL2 = MAPL02
343	170212	MAPH2 = MAPH02
344	170214	MAPL3 = MAPL03
345	170216	MAPH3 = MAPH03
346	170220	MAPL4 = MAPL04
347	170222	MAPH4 = MAPH04
348	170224	MAPL5 = MAPL05
349	170226	MAPH5 = MAPH05
350	170230	MAPL6 = MAPL06
351	170232	MAPH6 = MAPH06
352	170234	MAPL7 = MAPL07
353	170236	MAPH7 = MAPH07

.SBTTL UBA SPECIFIC REGISTERS

354			
355			
356			
357	177572	MMR0 = 177572	; MEMORY MANAGEMENT REGISTER DEFINITIONS
358	177574	MMR1 = 177574	;
359	177576	MMR2 = 177576	;
360	172516	MMR3 = 172516	;
361	000001	UFDSET = 1	; FLAG FOR UFD MODE
362	177520	BCSR = 177520	; BOOT/DIAGNOSTIC STATUS REGISTER
363	177730	DCSR = 177730	; DIAGNOSTIC CONTROLLER STATUS REGISTER
364	177732	DDR = 177732	; DIAGNOSTIC DATA REGISTER
365	177734	KMCR = 177734	; KTJ11-B MEMORY CONFIGURATION REGISTER
366	177746	CCR = 177746	; CACHE CONTROL REGISTER FOR CPU
367	177522	PCR = 177522	; PAGE CONTROL REGISTER
368	120001	POLY = 120001	; POLYNOMIAL USED FOR CRC ROUTINES
369	170014	SIMLGO = 170014	; SIMULTANEOUS GO ADDRESS FOR MULTIPLE UBE'S

.SBTTL TRAP CATCHER

370			
371	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	
000174	000000	

DISPREG: .WORD 0

; SOFTWARE DISPLAY REGISTER

TRAP CATCHER

```

000176 000000      SWREG:  .WORD  0          ;;SOFTWARE SWITCH REGISTER
372      000200      .=200
373 000200 005037 001160      CLR    $TMP0
374 000204 000137 002500      JMP    @START
375      000220      .=220
376 000220 012737 000777 001160      MOV    @777,$TMP0
377 000226 000137 002500      JMP    @START
378
379      .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
380      .SBTTL  APT PARAMETER BLOCK
      ;;*****
      ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
      ;;*****
      .IX=.          ;;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.
      $APTHDR        ;;POINT TO APT HEADER BLOCK
      .=$IX          ;;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)
      $UNITM: .WORD          ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
      .WORD  $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
000232 000000
000232 001200
000236 000000
000240 000000
000242 000000
000244 000052
000046 023142
000052 000000
000044 000232
000024 000200
000044 000044
000044 000232
000044 000232

```

COMMON TAGS

381

```

.SBTTL COMMON TAGS
;*****
;THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
;USED IN THE PROGRAM.
      . = 1100

001100 001100 $CMTAG:                ;; START OF COMMON TAGS
001100 000000      .WORD 0                ;; CONTAINS THE TEST NUMBER
001102 000      $TSTNM: .BYTE 0            ;; CONTAINS ERROR FLAG
001103 000      $ERFLG: .BYTE 0            ;; CONTAINS SUBTEST ITERATION COUNT
001104 000000      $ICNT: .WORD 0           ;; CONTAINS SCOPE LOOP ADDRESS
001106 000000      $LPADR: .WORD 0          ;; CONTAINS SCOPE RETURN FOR ERRORS
001110 000000      $LPERR: .WORD 0          ;; CONTAINS TOTAL ERRORS DETECTED
001112 000000      $ERTTL: .WORD 0          ;; CONTAINS ITEM CONTROL BYTE
001114 000      $ITEMB: .BYTE 0            ;; CONTAINS MAX. ERRORS PER TEST
001115 001      $ERMAX: .BYTE 1             ;; CONTAINS PC OF LAST ERROR INSTRUCTION
001116 000000      $ERRPC: .WORD 0          ;; CONTAINS ADDRESS OF 'GOOD' DATA
001120 000000      $GADDR: .WORD 0          ;; CONTAINS ADDRESS OF 'BAD' DATA
001122 000000      $BDADR: .WORD 0          ;; CONTAINS 'GOOD' DATA
001124 000000      $GDDAT: .WORD 0          ;; CONTAINS 'BAD' DATA
001126 000000      $BDDAT: .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)
001160 000002      .REPT 2
001162 000000      $TMP0: .WORD 0            ;; USER DEFINED
001164 000000      $TMP1: .WORD 0            ;; USER DEFINED
001166 000000      $TIMES: 0                 ;; MAX. NUMBER OF ITERATIONS
001170 207      $ESCAPE: 0                   ;; ESCAPE ON ERROR ADDRESS
001173 000      $BELL: .ASCIZ <207><377><377> ;; CODE FOR BELL
001174 077      $QUES: .ASCII /?/            ;; QUESTION MARK
001175 015      $CRLF: .ASCII <15>           ;; CARRIAGE RETURN
001176 012      $LF: .ASCIZ <12>             ;; LINE FEED
;*****
.SBTTL APT MAILBOX-ETABLE
;*****
.EVEN
001200 $MAIL:                ;; APT MAILBOX
001200 000000 $MSGTY: .WORD MSGTY           ;; 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 MSGAD           ;; 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	ENVH: .BYTE	AENVH	ENVIRONMENT MODE BITS
001222	000000	SMREG: .WORD	ASWREG	APT SWITCH REGISTER
001224	000000	USMR: .WORD	AUSMR	USER SWITCHES
001226	000000	CPUP: .WORD	ACPUOP	CPU TYPE, OPTIONS
		;		BITS 15-11-CPU TYPE
		;		11/04-01, 11/05-02, 11/20-03, 11/40-04, 11/45-05
		;		11/70-06, PD9-07, 8-10
		;		BIT 10-REAL TIME CLOCK
		;		BIT 9-FLOATING POINT PROCESSOR
		;		BIT 8-MEMORY MANAGEMENT
001230	000	HMS1: .BYTE	AHMS1	HIGH ADDRESS, M.S. BYTE
001231	000	HTYP1: .BYTE	ANTYP1	MEM. TYPE, BLK01
		;		MEM. TYPE BYTE -- (HIGH BYTE)
		;		900 NSEC CORE-001
		;		300 NSEC BIPOLAR-002
		;		500 NSEC MOS-003
001232	000000	HADR1: .WORD	AHADR1	HIGH ADDRESS, BLK01
		;		MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE
001234	000	HMS2: .BYTE	AHMS2	HIGH ADDRESS, M.S. BYTE
001235	000	HTYP2: .BYTE	ANTYP2	MEM. TYPE, BLK02
001236	000000	HADR2: .WORD	AHADR2	MEM. LAST ADDRESS, BLK02
001240	000	HMS3: .BYTE	AHMS3	HIGH ADDRESS, M.S. BYTE
001241	000	HTYP3: .BYTE	ANTYP3	MEM. TYPE, BLK03
001242	000000	HADR3: .WORD	AHADR3	MEM. LAST ADDRESS, BLK03
001244	000	HMS4: .BYTE	AHMS4	HIGH ADDRESS, M.S. BYTE
001245	000	HTYP4: .BYTE	ANTYP4	MEM. TYPE, BLK04
001246	000000	HADR4: .WORD	AHADR4	MEM. LAST ADDRESS, BLK04
001250	000000	VECT1: .WORD	AVECT1	INTERRUPT VECTOR01, BUS PRIORITY01
001252	000000	VECT2: .WORD	AVECT2	INTERRUPT VECTOR02, BUS PRIORITY02
001254	000000	BASE: .WORD	ABASE	BASE ADDRESS OF EQUIPMENT UNDER TEST
001256	000000	DEVH: .WORD	ADEVH	DEVICE MAP
001260	000000	CDW1: .WORD	ACDW1	CONTROLLER DESCRIPTION WORD01
001262	000000	CDW2: .WORD	ACDW2	CONTROLLER DESCRIPTION WORD02
001264	000000	DDW0: .WORD	ADDW0	DEVICE DESCRIPTOR WORD00
001266	000000	DDW1: .WORD	ADDW1	DEVICE DESCRIPTOR WORD01
001270	000000	DDW2: .WORD	ADDW2	DEVICE DESCRIPTOR WORD02
001272	000000	DDW3: .WORD	ADDW3	DEVICE DESCRIPTOR WORD03
001274	000000	DDW4: .WORD	ADDW4	DEVICE DESCRIPTOR WORD04
001276	000000	DDW5: .WORD	ADDW5	DEVICE DESCRIPTOR WORD05
001300	000000	DDW6: .WORD	ADDW6	DEVICE DESCRIPTOR WORD06
001302	000000	DDW7: .WORD	ADDW7	DEVICE DESCRIPTOR WORD07
001304	000000	DDW8: .WORD	ADDW8	DEVICE DESCRIPTOR WORD08
001306	000000	DDW9: .WORD	ADDW9	DEVICE DESCRIPTOR WORD09
001310	000000	DDW10: .WORD	ADDW10	DEVICE DESCRIPTOR WORD10
001312	000000	DDW11: .WORD	ADDW11	DEVICE DESCRIPTOR WORD11
001314	000000	DDW12: .WORD	ADDW12	DEVICE DESCRIPTOR WORD12
001316	000000	DDW13: .WORD	ADDW13	DEVICE DESCRIPTOR WORD13
001320	000000	DDW14: .WORD	ADDW14	DEVICE DESCRIPTOR WORD14
001322	000000	DDW15: .WORD	ADDW15	DEVICE DESCRIPTOR WORD15
001324		ETEND:		

.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
; * DM ;POINTS TO THE DATA HEADER
; * DT ;POINTS TO THE DATA
; * DF ;POINTS TO THE DATA FORMAT
\$ERRTP.

	001324			
382				
383				
384				
385				
386	001324	027006	EM1	; TIMEOUT ON ACCESSING A MAP REGISTER
387	001326	031174	DM1	; TEST * ERROR PC ERROR * ADDRESS
388	001330	032166	DT1	; TEST, \$ERRPC, ERRNUM, \$BDADR
389	001332	000000	0	
390				
391				
392	001334	027052	EM2	; MAP REGISTER COULD NOT BE CLEARED
393	001336	031243	DM2	; GOOD BAD
394				
395	001340	032200	DT2	; TEST * ERROR PC ERROR * DATA DATA ADDRESS
396	001342	000000	0	; TEST, \$ERRPC, ERRNUM, \$GDOAT, \$BDOAT, \$BDADR
397				
398				
399	001344	027114	EM3	; MAP REGISTER COULD NOT HOLD PATTERN
400	001346	031243	DM2	; GOOD BAD
401				
402	001350	032200	DT2	; TEST * ERROR PC ERROR * DATA DATA ADDRESS
403	001352	000000	0	; TEST, \$ERRPC, ERRNUM, \$GDOAT, \$BDOAT, \$BDADR
404				
405				
406	001354	027160	EM4	; MAP REGISTER HAS NOT BEEN ADDRESSED CORRECTLY
407	001356	031335	DM3	; GOOD BAD
408				
409	001360	032216	DT3	; TEST * ERROR PC ERROR * ADDRESS ADDRESS
410	001362	000000	0	; TEST, \$ERRPC, ERRNUM, \$GDADR, \$BDADR
411				
412				
413	001364	027236	EM5	; THERE WAS NO DIFFERENCE FOUND BETWEEN HI AND LO MAP REGIST
414	001366	031425	DM4	; HI LOW
415				
416	001370	032232	DT4	; TEST * ERROR PC ERROR * MAP MAP
417	001372	000000	0	; TEST, \$ERRPC, ERRNUM, \$GDOAT, \$BDOAT
418				
419				
420	001374	027334	EM6	; ERROR IN BITS 3-6, 9-14 IN THE DCSR
421	001376	031503	DM5	; GOOD BAD
422				
423	001400	032232	DT4	; TEST * ERROR PC ERROR * DATA DATA
424	001402	000000	0	; TEST, \$ERRPC, ERRNUM, \$GDOAT, \$BDOAT
425				
426				
427	001404	027377	EM7	; DCSR DID RESPOND PROPERLY ON RESET

ERROR DEFINITIONS

428	001406	031503	DM5	:	GOOD BAD
429				:	TEST # ERROR PC ERROR # DATA DATA
430	001410	032232	DT4	:	TEST, \$ERRPC, ERRNUM, \$GDOAT, \$BDOAT
431	001412	000000	0		
432			ERROR 10	:	
433					
434	001414	027446	EM10	:	TIMEOUT HAS OCCURED ON ACCESS TO THE DCSR
435	001416	031174	DM1	:	TEST # ERROR PC ERROR # ADDRESS
436	001420	032166	DT1	:	TEST, \$ERRPC, ERRNUM, \$BDOAT
437	001422	000000	0		
438			ERROR 11	:	
439					
440	001424	027520	EM11	:	KMCR BITS 0-5,8 DID NOT GET SET CORRECTLY
441	001426	031243	DM2	:	GOOD BAD
442				:	TEST # ERROR PC ERROR # DATA DATA ADDRESS
443	001430	032200	DT2	:	TEST, \$ERRPC, ERRNUM, \$GDOAT, \$BDOAT, \$BDOAT
444	001432	000000	0		
445			ERROR 12	:	
446					
447	001434	027572	EM12	:	TIMEOUT HAS OCCURED ON ACCESS TO THE KMCR
448	001436	031174	DM1	:	TEST # ERROR PC ERROR # ADDRESS
449	001440	032166	DT1	:	TEST, \$ERRPC, ERRNUM, \$BDOAT
450	001442	000000	0		
451			ERROR 13	:	
452					
453	001444	027644	EM13	:	ERROR IN DATA PATH
454	001446	031565	DM13	:	TEST # ERROR PC ERROR # PATTERN DOR
455	001450	032232	DT4	:	TEST, \$ERRPC, ERRNUM, \$GDOAT, \$BDOAT
456	001452	000000	0		
457			ERROR 14	:	
458					
459	001454	027713	EM14	:	ERROR IN DATA OUT
460	001456	031503	DM5	:	GOOD BAD
461				:	TEST # ERROR PC ERROR # DATA DATA
462	001460	032232	DT4	:	TEST, \$ERRPC, ERRNUM, \$GDOAT, \$BDOAT
463	001462	000000	0		
464			ERROR 15	:	
465					
466	001464	027743	EM15	:	ERROR IN DATA IN
467	001466	031565	DM13	:	TEST # ERROR PC ERROR # PATTERN DOR
468	001470	032232	DT4	:	TEST, \$ERRPC, ERRNUM, \$GDOAT, \$BDOAT
469	001472	000000	0		
470			ERROR 16	:	
471					
472	001474	027772	EM16	:	DOR NOT ZERO WHEN DCSR SELECTS UNIBUS LINES
473	001476	031503	DM5	:	GOOD BAD
474				:	TEST # ERROR PC ERROR # DATA DATA
475	001500	032232	DT4	:	TEST, \$ERRPC, ERRNUM, \$GDOAT, \$BDOAT
476	001502	000000	0		
477			ERROR 17	:	
478					
479	001504	030054	EM17	:	ERROR IN SETTING DCSR<7>
480	001506	031640	DM16	:	TEST # ERROR PC ERROR #
481	001510	032246	DT16	:	TEST, \$ERRPC, ERRNUM
482	001512	000000	0		
483			ERROR 20	:	
484					

ERROR DEFINITIONS

485	001514	030107	EM20	; ERROR IN UNIQUE ADDRESSING OF REGISTERS
486	001516	031677	DM20	; TEST # ERROR PC ERROR # KMCR PAIR FAILED
487	001520	032256	DT20	; TEST, \$ERRPC, ERRNUM, KMCR, \$BDADR
488	001522	000000	0	
489			ERROR 21	
490				
491	001524	030165	EM21	; REG. PAIR 31. PERFORMS RELOCATION
492	001526	031640	DM16	; TEST # ERROR PC ERROR #
493	001530	032246	DT16	; TEST, \$ERRPC, ERRNUM
494	001532	000000	0	
495			ERROR 22	
496				
497	001534	030232	EM22	; NXM CONDITION COULDN'T BE CREATED THRU ALU
498	001536	031640	DM16	; TEST # ERROR PC ERROR #
499	001540	032246	DT16	; TEST, \$ERRPC, ERRNUM
500	001542	000000	0	
501			ERROR 23	
502				
503	001544	030306	EM23	; ALU ERROR
504	001546	031757	DM23	; TEST # ERROR PC ERROR # DATA DATA ADDRESS
505				; TEST, \$ERRPC, ERRNUM, \$GDDAT, \$BDDAT, KIPAR6, \$BDADR
506	001550	032302	DT23	
507	001552	000000	0	
508			ERROR 24	
509				
510	001554	030320	EM24	; CPU CACHE ERROR
511	001556	031640	DM16	; TEST # ERROR PC ERROR #
512	001560	032246	DT16	; TEST, \$ERRPC, ERRNUM
513	001562	000000	0	
514			ERROR 25	
515				
516	001564	030340	EM25	; KMCR<4-0> DOESN'T DISABLE MAIN MEMORY
517	001566	031640	DM16	; TEST # ERROR PC ERROR #
518	001570	032246	DT16	; TEST, \$ERRPC, ERRNUM
519	001572	000000	0	
520			ERROR 26	
521				
522	001574	030410	EM26	; KMCR DOES NOT REFLECT EXPECTED STATUS OF THE CACHE/
523	001576	031503	DM5	; TEST # ERROR PC ERROR # DATA DATA
524				; TEST, \$ERRPC, ERRNUM, \$GDDAT, \$BDDAT
525	001600	032232	DT4	
526	001602	000000	0	
527			ERROR 27	
528				
529	001604	030473	EM27	; ERROR IN CACHE TAG REGISTERS
530	001606	032070	DM27	; TEST # ERROR PC ERROR # ADDRESS
531	001610	032322	DT27	; TEST, \$ERRPC, ERRNUM, MAPH01, MAPL01
532	001612	000000	0	
533			ERROR 30	
534				
535	001614	030530	EM30	; ERROR IN THE DMA CACHE DATA RAMS
536	001616	031174	DM1	; TEST # ERROR PC ERROR # ADDRESS
537	001620	032336	DT30	; TEST, \$ERRPC, ERRNUM, MAPL00
538	001622	000000	0	
539			ERROR 31	
540				
541	001624	030571	EM31	; ERROR IN THE M9312 BOOT ROM SECTION

ERROR DEFINITIONS

```

542 001626 031640          DM16          ; TEST # ERROR PC ERROR #
543 001630 032246          DT16          ; TEST,ERRPC,ERRNUM
544 001632 000000          0
545          ; ERROR 32
546
547 001634 030635          EM32          ; ERROR IN ARBITRATION LOGIC THRU UBE
548 001636 031640          DM16          ; TEST # ERROR PC ERROR #
549 001640 032246          DT16          ; TEST,ERRPC,ERRNUM
550 001642 000000          0
551          ; ERROR 33
552
553 001644 030702          EM33          ; ERROR TRYING TO DO DMA CYCLES THRU UBE
554 001646 031640          DM16          ; TEST # ERROR PC ERROR #
555 001650 032246          DT16          ; TEST,ERRPC,ERRNUM
556 001652 000000          0
557          ; ERROR 34
558
559 001654 030756          EM34          ; ERROR IN THE UNIBUS MEMORY TEST
560 001656 031640          DM16          ; TEST # ERROR PC ERROR #
561 001660 032246          DT16          ; TEST,ERRPC,ERRNUM
562 001662 000000          0
563          ; ERROR 35
564
565 001664 031016          EM35          ; UNEXPECTED TIMEOUT HAS OCCURED
566 001666 031640          DM16          ; TEST # ERROR PC ERROR #
567 001670 032272          DT21          ; TEST,ERRPC,ERRNUM
568 001672 000000          0
569          ; ERROR 36
570
571 001674 031055          EM36          ; UNIBUS TIMEOUT DID NOT OCCUR
572 001676 031640          DM16          ; TEST # ERROR PC ERROR #
573 001700 032246          DT16          ; TEST,ERRPC,ERRNUM
574 001702 000000          0
575          ; ERROR 37
576
577 001704 031123          EM37          ; DCSR<3> DIDN'T DISABLE UBA ROM'S
578 001706 031640          DM16          ; TEST # ERROR PC ERROR #
579 001710 032246          DT16          ; TEST,ERRPC,ERRNUM
580 001712 000000          0

```

```

581
582          .SBTTL GLOBAL VARIABLES
583 001714 000000          ERRNUM: .WORD 0          ; ERROR NUMBER FOR REPORT
584 001716 000000          TEST: .WORD 0          ; TEST NUMBER FOR REPORT
585 001720 000000          PHIS: .WORD 0          ; PMI MEMORY SIZE
586 001722 000000          UBECT: .WORD 0          ; NUMBER OF UBE'S
587 001724 000000          BE1INT: .WORD 0          ; UBE #1 INTERRUPT FLAG
588 001726 000000          BE2INT: .WORD 0          ; UBE #2 INTERRUPT FLAG
589 001730 000000          TOUT: .WORD 0          ; TIMEOUT FLAG
590 001732 172100          MCSR: .WORD 172100          ; POINTER TO MEMORY CSR
591 001734          WRTBUF: .BLKW 20          ; WRITE BUFFER
592 001774 000377 007417 031463          PTRN16: .WORD 377,7417,31463,52525,125252,0 ; 16-BIT BINARY DIVIDE
593 002002 052525 125252 000000          PTRN6: .WORD 17,14,25,52,0          ; 6-BIT BINARY DIVIDE
594 002010 000017 000014 000025
595 002016 000052 000000

```

```

596          .SBTTL CACHE RAM IMAGE TABLE

```

CACHE RAM IMAGE TABLE

```

597      ;
598      ; THE FOLLOWING BLOCK OF DATA WILL BE USED IN THE CACHE RAM BIT TEST
599      ; THE EXACT ADDRESSES WILL BE FIGURED OUT BY THE PROGRAM DEPENDING
600      ; ON THE LOCATION OF THIS TABLE.
601      ;
602
603 002022      CTBLE: .BLKW    47      ; THE LARGEST # OF LOCATIONS NEEDED
604
605 002140 000000      OBADR: .WORD    0      ; FIRST ADDRESS OF A FIELD IN TABLE
606 002142 000000      .WORD    0      ; FIRST ADDRESS OF B FIELD IN TABLE
607 002144 000000      .WORD    0      ; FIRST ADDRESS OF C FIELD IN TABLE
608 002146 000000      .WORD    0      ; FIRST ADDRESS OF D FIELD IN TABLE
609
610      .SBTTL  UNIBUS EXERCISER REGISTER TABLES
611
612      ;
613      ;      UBE #1
614      ;
615
616 002150 000000      BE1DB:  0      ; UBE #1 DATA REGISTER
617 002152 000000      BE1CC:  0      ; UBE #1 CYCLE COUNT REGISTER
618 002154 000000      BE1BA:  0      ; UBE #1 ADDRESS REGISTER
619 002156 000000      BE1CR1: 0      ; UBE #1 CONTROL REGISTER 1
620 002160 000000      BE1CLR: 0      ; UBE #1 CLEAR ERROR REGISTER ADDRESS
621 002162 000000      BE1CR2: 0      ; UBE #1 CONTROL REGISTER 2
622 002164 000000      BE1VEC: 0      ; UBE #1 VECTOR PC
623 002166 000000      BE1PSW: 0      ; UBE #1 VECTOR PSW
624
625      ;
626      ;      UBE #2
627      ;
628
629 002170 000000      BE2DB:  0      ; UBE #2 DATA REGISTER
630 002172 000000      BE2CC:  0      ; UBE #2 CYCLE COUNT REGISTER
631 002174 000000      BE2BA:  0      ; UBE #2 ADDRESS REGISTER
632 002176 000000      BE2CR1: 0      ; UBE #2 CONTROL REGISTER 1
633 002200 000000      BE2CLR: 0      ; UBE #2 CLEAR ERROR REGISTER ADDRESS
634 002202 000000      BE2CR2: 0      ; UBE #2 CONTROL REGISTER 2
635 002204 000000      BE2VEC: 0      ; UBE #2 VECTOR PC
636 002206 000000      BE2PSW: 0      ; UBE #2 VECTOR PSW
637
638      .SBTTL  SUBROUTINE - DIAGNOSTIC_DATA_OUT SUBROUTINE
639
640      ;* INPUTS: PATTERN TO BE STORED IN DDR AND THEN WRITTEN TO MEMORY %GDDAT
641      ;*      TEST_LOCATION TO BE WRITTEN TO (16 OR 22 BITS) (R1)
642      ;
643      ;* ON RETURN FROM SUBROUTINE TEST_LOCATION SHOULD HAVE THE SAME DATA
644      ;* AS DDR AND THE SAME AS PATTERN
645      ;* THE PROGRAM HAS TO BE RUNNING IN DIAGNOSTIC MODE WITH DIAGNOSTIC
646      ;* NPR REGISTER SELECTED
647      ;
648      ;      BGNROUTINE
649      ;
650      ;      MOVE PATTERN TO DDR
651      ;      DO EXTERNAL WRITE FROM TEST_LOCATION
652      ;      RETURN
653      ;
654      ;      ENDROUTINE

```

SUBROUTINE - DIAGNOSTIC_DATA_OUT SUBROUTINE

```

654      ;
655      ;-----
656
657 002210 013737 001124 177732 DDOUT:  MOV    $GDDAT,DDR      ; STORE PATTERN IN DDR
658 002216 011111              1$:   MOV    (R1),(R1)        ; EXTENAL READ TO PROVIDE ADDRESS
659 002220 000207              RTS    PC
660

```

```

661
662      .SBTTL SUBROUTINE - DIAGNOSTIC_DATA_IN SUBROUTINE
663
664      ;* INPUTS: PATTERN TO BE WRITTEN TO MEMORY AND THEN TO DDR $GDDAT
665      ;*          TEST_LOCATION TO READ FROM (16 OR 22 BITS) (R1)
666
667      ;* ON RETURN FROM SUBROUTINE DDR SHOULD HAVE THE SAME DATA AS
668      ;* SPECIFIED MEMORY LOCATION AND THE SAME AS THE PATTERN
669      ;* THE PROGRAM HAS TO BE RUNNING IN DIAGNOSTIC MODE WITH DIAGNOSTIC
670      ;* NPR REGISTER SELECTED.
671

```

```

672      ; BGNROUTINE
673
674      ;       LET DCSR<0> = #1
675      ;       DO EXTERNAL WRITE FROM TEST_LOCATION
676      ;       RETURN
677
678      ; ENDRROUTINE
679

```

```

680      ;-----
681
682 002222 052737 000001 177730 DDIN:   BIS    #BIT00,DCSR      ; SET GO BIT
683 002230 011111              2$:   MOV    (R1),(R1)        ; PROVIDE ADDRESS FOR DMA
684 002232 000207              3$:   RTS    PC
685

```

```

686
687      .SBTTL SUBROUTINE - TIMEOUT_ROUTINE
688      ;* THIS ROUTINE IS USED TO FLAG AN UNEXPECTED TIMEOUT.
689
690      ; BGNROUTINE
691
692      ;       STORE ADDRESS THAT CAUSED TIMEOUT
693      ;       ERROR
694      ;       RETURN
695
696      ; ENDRROUTINE
697

```

```

698      ;-----
699
700 002234 011637 001122      TIMEOUT: MOV    (SP), $BDADR      ; STORE ADDRESS THAT TIMED OUT
701 002240 104035              ERROR   +35                    ; UNEXPECTED TIMEOUT
702 002242 000002              RTI
703

```

```

704
705      .SBTTL SUBROUTINE - MAP_PROGRAM_AREA
706
707      ;* THIS ROUTINE MAPS THE PROGRAM AREA TO THE FIRST 32K
708
709      ; BGNROUTINE
710

```

SUBROUTINE - MAP_PROGRAM_AREA

```

711      ;      MAP PROGRAM AREA THRU KIPAR'S TO FIRST 32 K
712      ;      NO CACHE BYPASS
713      ;      RETURN
714      ;
715      ;      ENDRoutine
716      ;
717      ;-----
718
719      002244      MAPPR:
720      002244      010046      MOV      R0,-(SP)      ; PUSH R0 ON STACK
721      002246      010146      MOV      R1,-(SP)      ; PUSH R1 ON STACK
722      002250      010246      MOV      R2,-(SP)      ; PUSH R2 ON STACK
723      002252      012700      172300      MOV      #172300,R0      ; R0 POINTS TO FIRST KIPDR
724      002256      012701      000010      MOV      #8,R1      ; DO FOR ALL 8 REGISTERS
725      002262      012720      077406      1$: MOV      #77406,(R0)+      ; . 4K PAGE, CACHE ON, READ/WRITE
726      002266      077103      SOB      R1,1$      ; . CONTINUE TILL DONE
727      002270      012700      172340      MOV      #172340,R0      ; R0 POINTS TO FIRST KIPAR
728      002274      012701      000006      MOV      #6,R1      ; DO FOR ALL 8 REGISTERS
729      002300      012702      000000      MOV      #0,R2      ; START WITH ADDRESS 0
730      002304      010220      2$: MOV      R2,(R0)+      ; . LOAD WITH ADDRESS
731      002306      062702      000200      ADD      #200,R2      ; . NEXT 4K
732      002312      077104      SOB      R1,2$      ; . CONTINUE TILL DONE
733      002314      005020      CLR      (R0)+      ; CLEAR KIPAR6
734      002316      012710      177600      MOV      #177600,(R0)+      ; I/O PAGE TO KIPAR7
735      002322      012602      MOV      (SP)+,R2      ; POP STACK INTO R2
736      002324      012601      MOV      (SP)+,R1      ; POP STACK INTO R1
737      002326      012600      MOV      (SP)+,R0      ; POP STACK INTO R0
738      002330      000207      RTS      PC
739
740      .SBTTL SUBROUTINE - MEMORY SIZE
741
742      ;* THIS ROUTINE SIZES MAIN (PMI) MEMORY IN 4K WORDS.
743      ;
744      ;      BGNROUTINE
745      ;
746      ;      REMAP TIMEOUT VECTOR AND PROGRAM AREA
747      ;      ENABLE MEMORY MANAGEMENT UNIT
748      ;      DO UNTIL TIMEOUT OR ALL 2M CHECKED
749      ;      . CHECK LOCATION 0 IN EACH 4K PAGE THRU KIPAR6
750      ;      ENDDO
751      ;
752      ;      ENDRoutine
753      ;-----
754
755      002332      013702      000004      MEMSIZ: MOV      ERRVEC,R2      ; SAVE TIME OUT VECTOR
756      002336      012737      002412      000004      MOV      #3$,ERRVEC      ; POINT NEW TO PROGRAM
757      002344      012737      000340      000006      MOV      #340,ERRVEC+2      ; AT PRIORITY 7
758
759      ;      SIZE MEMORY IN 4K WORDS
760      ;
761      002352      012737      000200      172354      MOV      #200,KIPAR6      ; FIRST 4K
762      002360      000403      BR      2$      ; GO TRY TO ACCESS
763      002362      062737      000200      172354      1$: ADD      #200,KIPAR6      ; . NEXT 4K
764      002370      005737      140000      2$: TST      #140000      ; . ACCESS THRU KIPAR6
765      002374      022737      170000      172354      CMP      #170000,KIPAR6      ; . LAST PAGE?

```

SUBROUTINE - MEMORY SIZE

```

763 002402 001367      BNE      1$                ; . IF NOT, BRANCH
764 002404 012701 000040      MOV      #40,R1        ; IF 2M PRESENT, ONLY 21ST MASKED
765 002410 000402      BR       4$                ; BRANCH
766 002412 005726      3$:      TST      (SP)+        ; RESTORE SIACK
767 002414 005726      TST      (SP)+        ;
768 002416 010237 000004      4$:      MOV      R2,ERRVEC    ; RESTORE TIMEOUT VECTOR
769 002422 000207      RTS      PC                ; RETURN

```

```

770
771      .SBTTL SUBROUTINE - SIZE UNIBUS MEMORY FROM KMCR
772
773      ;* ON RETURN FROM THIS SUBROUTINE R2 WOULD HAVE PAR VALUE
774      ;* OF UNIBUS MEMORY
775      ;

```

```

776 002424 013701 177734      UMSIZ:  MOV      KMCR,R1        ; SAVE KMCR
777 002430 042701 000040      1$:      BIC      #BIT05,R1    ; LEAVE ONLY # OF PAGES
778 002434 012702 007600      MOV      #7600,R2        ; START W/O UNIBUS MEMORY
779 002440 162702 000200      100$:    SUB      #200,R2      ; . SUBTRACT 4K
780 002444 077103      SOB      R1,100$        ; . FOR ALL PAGES PRESENT
781 002446 000207      RTS      PC
782
783

```

```

784      .SBTTL SUBROUTINE - INITIALIZE THE UBE'S
785
786      ;* THIS ROUTINE IS USED TO INITIALIZE THE UBE'S
787      ;
788      ; BGNROUTINE
789      ;
790      ;      SAVE R0,R1
791      ;      LET R0 := FIRST UBE ADDRESS
792      ;      DO FOR (R0) := BE108 TO BE1C1
793      ;      . LET (R0) := 0
794      ;      ENDDO
795      ;      RESTORE R1,R0
796      ;      RETURN
797      ;
798      ; ENDROUTINE
799      ;

```

```

800      ;-----
801
802 002450      IUBE:
803 002450 010046      MOV      R0,-(SP)    ;;PUSH R0 ON STACK
804 002452 010146      MOV      R1,-(SP)    ;;PUSH R1 ON STACK
805 002454 013700 002150      MOV      BE108,R0    ; POINT R0 TO UBE REGISTERS
806 002460 012701 000010      MOV      #10,R1    ; SET UP A LOOP COUNTER
807 002464 012720 000000      5$:      MOV      #0,(R0)+    ; CLEAR OUT A REGISTER
808 002470 077103      SOB      R1,5$        ; HAVE WE INITIALIZED THE UBE ?
809 002472 012601      MOV      (SP)+,R1    ;;POP STACK INTO R1
810 002474 012600      MOV      (SP)+,R0    ;;POP STACK INTO R0
811 002476 000207      RTS      PC
812
813
814
815 002500

```

```

;
; STARTING POINT OF PROGRAM
;

```

```

START:
;; LCP/ORION ROUTINE TO SAVE EMULATOR AND PRIORITY

```

SUBROUTINE - INITIALIZE THE UBE'S

```

002500 005737 002566      EMTSAV: TST      SAV30      ;; FIRST TIME THROUGH ?
002504 001034              BNE      VMKOR      ;; BRANCH IF BEEN HERE ALREADY
002506 032737 000040 000052      BIT      @BIT5,@52  ;; ARE WE IN UFD MODE ?
002514 001430              BEQ      VMKOR      ;; LEAVE IF NOT
002516 012737 177777 002572      MOV      #-1,UFDPLG  ;; SET UFD FLAG
002524 032737 000100 000052      BIT      @BIT6,@52  ;; ARE WE IN QUIET MODE ?
002532 001403              BEQ      1$        ;; BR IF NOT
002534 012737 177777 002574      MOV      #-1,UQUIET  ;; SET QUIET MODE
002542 104042 1$:          EMT      42          ;; GET ADDRESS OF XXDP DCA TABLE
002544 005060 000042              CLR      42(R0)    ;; CLR XXDP+ "DRSERR"
002550 013737 000030 002566      MOV      30,SAV30    ;; SAVE EMULATOR ADDRESS
002556 013737 000032 002570      MOV      32,SAV32    ;; SAVE EMULATOR PRIORITY LEVEL
002564 000404              BR      VMKOR      ;; GET AROUND TAG AREA
002566 000000      SAV30: .WORD 0      ;; PUT EMULATOR INFO HERE
002570 000000      SAV32: .WORD 0      ;; PUT PRIORITY LOCATION      HERE
002572 000000      UFDPLG: .WORD 0    ;; USER FRIENDLY MODE FLAG
002574 000000      UQUIET: .WORD 0    ;; UFD QUIET MODE FLAG
002576              VMKOR:

;*****
816      .SBTTL INITIALIZE THE COMMON TAGS
      ;;CLEAR THE COMMON TAGS (@CHTAG) AREA
      MOV      @CHTAG,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
      MOVB     @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      @64,@ERRVEC   ;;SET UP ERROR VECTOR
      MOV      @DSWR,SWR     ;;SETUP FOR A HARDWARE SWICH REGISTER
      MOV      @DISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
      CMP      #-1,@SWR      ;;TRY TO REFERENCE HARDWARE SWR
      BNE      66$          ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
      ;;AND THE HARDWARE SWR IS NOT -1
      BR      65$          ;;BRANCH IF NO TIMEOUT
      MOV      @65$,(SP)    ;;SET UP FOR TRAP RETURN
      MOV      @SWREG,SWR   ;;POINT TO SOFTWARE SWR
      MOV      @DISPREG,DISPLAY
      MOV      (SP)+,@ERRVEC ;;RESTORE ERROR VECTOR
      CLR      @PASS        ;;CLEAR PASS COUNT

```

INITIALIZE THE COMMON TAGS

```

      003030 132737 000200 001221      BITB  #APTSIZE,#ENVM  ;;TEST USER SIZE UNDER APT
      003036 001403                      BEQ  67$           ;;YES,USE NON-APT SWITCH
      003040 012737 001222 001140      MOV  #SWREG,SWR      ;;NO,USE APT SWITCH REGISTER
      003046                      67$:
817 003046 005737 002574      TST  UQUIET                ; ARE WE UNDER UFD QUIET MODE ?
818 003052 001056                      BNE  1$             ; YES,THEN SKIP THE DIAGNOSTIC TITLE PRINTOUT
819 003054 122737 000001 001220      CMPB  #APTENV,#ENV     ; APT?
820 003062 001452                      BEQ  1$             ; IF YES, SKIP PRINTOUT
821                      .SBTTL  TYPE PROGRAM NAME
                      ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
      003064 005227 177777      INC  #-1                 ;;FIRST TIME?
      003070 001047                      BNE  68$           ;;BRANCH IF NO
      003072 022737 023142 000042      CMP  #ENDAD,#42      ;;ACT-11?
      003100 001443                      BEQ  68$           ;;BRANCH IF YES
      003102 104401 003150      TYPE  ,69$               ;;TYPE ASCIZ STRING
                      .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
      003106 005737 000042      TST  #42                 ;;ARE WE RUNNING UNDER XXDP/ACT?
      003112 001012                      BNE  70$           ;;BRANCH IF YES
      003114 123727 001220 000001      CMPB  #ENV,#1        ;;ARE WE RUNNING UNDER APT?
      003122 001406                      BEQ  70$           ;;BRANCH IF YES
      003124 023727 001140 000176      CMP  SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
      003132 001005                      BNE  71$           ;;BRANCH IF NO
      003134 104406                      GTSWR              ;;GET SOFT-SWR SETTINGS
      003136 000403                      BR  71$
      003140 112737 000001 001134 70$: MOV  #1,#AUTOB      ;;SET AUTO-MODE INDICATOR
      003146 000420 71$:
      003146                      BR  68$
                      ;;69$: .ASCIZ <CRLF>* COKTABO KTJ11-B DIAGNOSTIC *<CRLF>
822 003210 012737 002234 000004 68$: MOV  #TIMOUT,#44      ; SETUP TIMEOUT VECTOR FOR UNEXPECTED TIMEOUT
823 003216 012737 000340 000006 1$: MOV  #340,#46
824 003224 005037 001716      CLR  TEST                ; TEST # FOR ERROR REPORTING
825                      ; SIZE MEMORY IN 4K PAGES
826 003230 004737 002244      JSR  PC,MAPPR              ; REMAP PROGRAM AREA
827 003234 052737 000060 172516      BIS  #BIT05:BIT04,MMR3 ; ENABLE RELOCATION AND 22BITS
828 003242 005237 177572      INC  MMR0                ; ENABLE MMU
829 003246 004737 002332      JSR  PC,MEMSIZ             ; GET HIGHEST 128K OF MEMORY
830 003252 005037 177572      CLR  MMR0                ; DISABLE MMU
831 003256 005037 172516      CLR  MMR3
832 003262 013737 172354 001720      MOV  KIPAR6,PMIS      ; STORE HIGHEST PAGE
833 003270      RESTART:
834 003270 012706 001100      MOV  #1100,SP              ; SET UP THE STACK POINTER
835
836
837

```

TEST - UNIBUS MAP REGISTER TESTS

```

839      .SBTTL TEST - UNIBUS MAP REGISTER TESTS
840
841      ; * THIS TEST WILL TRY TO ACCESS ALL THE MAP
842      ; * REGISTERS WITH BOTH MAPPING ENABLED AND
843      ; * DISABLED
844
845      ;   BGNTST
846
847      ;       LET 4:= ADDRESS OF ERROR ROUTINE
848      ;       LET 6:= PRIORITY OF 7
849      ;       DO FOR BOTH UNIBUS MAP ENABLED AND DISABLED
850      ;       . DO FOR ADDRESS := 170200 TO 170376
851      ;       .   READ ADDRESS
852      ;       .   IF ADDRESS CAN'T BE READ THEN
853      ;       .   . TRAP THRU VECTOR 4
854      ;       .   . ENDF
855      ;       . ENDDO
856      ;       ENDDO
857      ;       EXIT TEST
858
859      ;   ERROR:
860      ;       REPORT WHICH MAP REGISTER TIMED OUT
861
862      ;   ENDTST
863
864      ;-----
865
866      ;*****
867      ;*TEST 1      UNIBUS MAP REGISTER RESPONSE TEST
868      ;*****
869      TST1:  SCOPE
870
871      ;   SET UP MEMORY TIMEOUT VECTOR TO ERROR ROUTINE
872
873      ;       MOV      #4,R5                ; SAVE THE TIMEOUT VECTOR
874      ;       MOV      #201,R4             ; SET UP TIMEOUT VECTOR TO ERROR ROUTINE
875      ;       MOV      #340,R6             ; SET UP TIMEOUT PRIORITY
876
877      ;   DISABLE UNIBUS MAPPING
878
879      ;       BIC      #BIT5,R3            ; DISABLE UNIBUS MAPPING
880
881      ;   TRY TO ACCESS ALL THE UNIBUS MAP REGISTERS
882
883      ;       CLR      R1                  ; INITIALIZE LOOP COUNT
884      ;       MOV      #MAPLOO,R0          ; . R0 POINTS TO MAP REGISTER #0
885      ;       TST      (R0)                ; . . TRY TO READ MAP REGISTER
886      ;       CMP      R0,#170400          ; . . HAVE WE READ ALL THE MAP REGISTERS
887      ;       BNE      101                 ; . . NO THEN GO READ THE NEXT ONE
888      ;       TST      R1                  ; . HAVE WE DONE THIS TWICE ?
889      ;       BEQ      151                 ; . GO TURN ON UNIBUS MAP AND TRY AGAIN
890      ;       BIC      #BIT5,R3            ; . TURN OFF UNIBUS MAPPING
891      ;       MOV      R5,R4               ; RESTORE THE TIMEOUT VECTOR
892      ;       BR       TST2                ; . GO TO NEXT TEST
893      ;       INC      R1                  ; . INCREMENT LOOP COUNT
894      ;       BIS      #BIT5,R3            ; . TURN ON UNIBUS MAP

```

T1 UNIBUS MAP REGISTER RESPONSE TEST

```
893 003372 000755                    BR      5$                    ; . GO TRY TO ACCESS THE REGISTERS
894                    ;
895                    ; MAP REGISTER TIMEOUT ROUTINE
896                    ;
897 003374 005726                    20$:    TST      (SP)+                    ; CLEAN UP THE STACK
898 003376 005726                           TST      (SP)+                    ;
899 003400 162700 000002                           SUB      #2,R0                    ; GET ADDRESS THAT TIMED OUT
900 003404 010037 001122                           MOV      R0,$BDADR                    ; PUT IT IN $BDADR
901 003410 062700 000002                           ADD      #2,R0                    ; GET NEXT ADDRESS
902 003414 104001                           ERROR    +1                    ; REPORT THAT ACCESS HAS TIMED OUT
903 003416 000746                           BR      12$                    ;
904
```

TEST UNIBUS MAP REGISTER BIT PATTERN

```

906      .SBTTL  TEST - UNIBUS MAP REGISTER BIT PATTERN
907
908      ; * THIS TEST IS DESIGNED TO DETECT ANY STUCK
909      ; * AT BITS OR SHORTED BITS IN THE UNIBUS MAP
910      ; * REGISTERS
911      ;
912      ;   BGNTST
913      ;
914      ;       CLEAR ALL MAP REGISTERS
915      ;       DO FOR ADDRESS := 170200 TO 170376
916      ; *
917      ; * CHECK LOW MAPPING REGISTERS
918      ; *
919      ;       . READ ADDRESS
920      ;       . IF ADDRESS NE 0 THEN
921      ;       .   ERROR MAP REGISTER DID NOT CLEAR
922      ;       .   ENDF
923      ;       . DO FOR PATTERNS := 377,7417,1463,52525,125252
924      ;       .   WRITE PATTERN INTO ADDRESS
925      ;       .   READ PATTERN
926      ;       .   IF CONTENTS OF ADDRESS NE PATTERN THEN
927      ;       .   . ERROR IN BITS OF ADDRESS
928      ;       .   . ENDF
929      ;       . ENDDO
930      ;       . LET ADDRESS := ADDRESS + 2
931      ; *
932      ; * CHECK HIGH 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 := 17,14,25,52
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      ;       ENDDO
946      ;
947      ;   ENDTST
948      ;
949      ; .....
950
951      ; *****

```

```

; *****
; *TEST 2      UNIBUS MAP REGISTER BIT PATTERN TEST
; *****
TST2:  COPE

```

```

952      003420    000004
953
954
955

```

```

956      003422    012700    170200
957      003426    005020
958      003430    020027    170400
959      003434    001374

```

```

; CLEAR ALL MAP REGISTERS
;
51:      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      51              ; . NO THEN GO CLEAR THE NEXT ONE

```

T2 UNIBUS MAP REGISTER BIT PATTERN TEST

```

960      ;
961      ; DO A READ,WRITE,READ ON THE REGISTERS TO
962      ; VERIFY ALL THE BITS OF THE LOW MAP REGISTERS
963      ;
964 003436 012700 170200      ;
965 003442 005710      104:  MOV    @MAPL00,R0      ; GET ADDRESS TO 1ST MAP REGISTER
966 003444 001410      ;      TST    (R0)      ; . IS THE MAP REGISTER CLEAR ?
967 003446 011037 001126      ;      BEQ    154      ; . YES, THEN GO CHECK THE BITS
968 003452 012737 000000 001124  ;      MOV    (R0),@BDDAT      ; . NO, THEN GET THE BAD DATA
969 003460 010037 001122      ;      MOV    @0,@GDDAT      ; . GET THE GOOD DATA
970 003464 104002      ;      MOV    R0,@BDADR      ; . GET THE ADDRESS THAT DID NOT CLEAR
971 003466 012701 001774      ;      ERROR    +2      ; . ERROR - REGISTER DID NOT CLEAR
972 003472 011110      154:  MOV    @PTRN16,R1      ; . GET POINTER TO PATTERN TABLE
973 003474 012102      204:  MOV    (R1),(R0)      ; . . MOVE PATTERN TO MAP REGISTER
974 003476 042702 000001      ;      MOV    (R1),R2      ; . . MOVE COPY OF PATTERN TO R2
975 003502 020210      ;      BIC    @BIT0,R2      ; . . CLEAR BIT0 OF THE PATTERN
976 003504 001410      ;      CMP    R2,(R0)      ; . . DID PATTERN GET WRITTEN CORRECTLY ?
977 003506 011037 001126      ;      BEQ    254      ; . . YES, THEN GO SEE IF ALL PATTERNS HAVE BEEN WRIT
978 003512 005741      ;      MOV    (R0),@BDDAT      ; . . NO, GET THE BAD DATA
979 003514 012137 001124      ;      TST    -(R1)      ; . . POINT TO THE GOOD DATA
980 003520 010037 001122      ;      MOV    (R1),@GDDAT      ; . . GET THE GOOD DATA, AND POINT R1 TO THE NEXT DA
981 003524 104003      ;      MOV    R0,@BDADR      ; . . GET THE ADDRESS OF DATA FAULT
982 003526 005711      ;      ERROR    +3      ; . . ERROR - REGISTER COULD NOT HOLD PATTERN
983 003530 001360      254:  TST    (R1)      ; . . ARE WE THROUGH THE TABLE ?
984      ;      BNE    204      ; . . NO THEN GO WRITE NEXT PATTERN
985      ;
986      ; DO A READ,WRITE,READ ON THE REGISTERS TO
987      ; VERIFY ALL THE BITS OF THE HIGH MAP REGISTERS
988      ;
989 003532 062700 000002      ;
990 003536 005710      304:  ADD    @2,R0      ; . POINT TO THE HIGH MAPPING REGISTER
991 003540 001410      ;      TST    (R0)      ; . IS THE MAP REGISTER CLEAR ?
992 003542 011037 001126      ;      BEQ    354      ; . YES, THEN GO CHECK THE BITS
993 003546 012737 000000 001124  ;      MOV    (R0),@BDDAT      ; . NO, THEN GET THE BAD DATA
994 003554 010037 001122      ;      MOV    @0,@GDDAT      ; . GET THE GOOD DATA
995 003560 104002      ;      MOV    R0,@BDADR      ; . GET THE ADDRESS THAT DID NOT CLEAR
996 003562 012701 002010      ;      ERROR    +2      ; . ERROR - REGISTER DID NOT CLEAR
997 003566 011110      354:  MOV    @PTRN16,R1      ; . GET POINTER TO PATTERN TABLE
998 003570 012102      404:  MOV    (R1),(R0)      ; . . MOVE PATTERN TO MAP REGISTER
999 003572 042702 177700      ;      MOV    (R1),R2      ; . . MOVE COPY OF THE PATTERN TO R2
1000 003576 020210      ;      BIC    @177700,R2      ; . . MASK OUT THE UPPER 10 BITS
1001 003600 001410      ;      CMP    R2,(R0)      ; . . DID PATTERN GET WRITTEN CORRECTLY ?
1002 003602 011037 001126      ;      BEQ    454      ; . . YES, THEN GO SEE IF ALL PATTERNS HAVE BEEN WRIT
1003 003606 005741      ;      MOV    (R0),@BDDAT      ; . . NO, GET THE BAD DATA
1004 003610 012137 001124      ;      TST    -(R1)      ; . . POINT TO THE GOOD DATA
1005 003614 010037 001122      ;      MOV    (R1),@GDDAT      ; . . GET THE GOOD DATA, AND POINT R1 BACK TO THE NE
1006 003620 104003      ;      MOV    R0,@BDADR      ; . . GET THE ADDRESS OF DATA FAULT
1007 003622 005711      ;      ERROR    +3      ; . . ERROR - REGISTER COULD NOT HOLD PATTERN
1008 003624 001360      454:  TST    (R1)      ; . . ARE WE THROUGH THE TABLE ?
1009      ;      BNE    404      ; . . NO THEN GO WRITE NEXT PATTERN
1010      ;
1011      ; SEE IF WE HAVE CHECKED ALL THE MAP REGISTERS
1012      ;
1013 003626 062700 000002      ;
1014 003632 020027 170400      ;      ADD    @2,R0      ; . POINT TO NEXT REGISTER
1015 003636 001301      ;      CMP    R0,@170400      ; . ARE WE THROUGH ALL THE MAP REGISTERS ?
1016      ;      BNE    104      ; . GO CHECK THE NEXT REGISTER

```

TEST UNIBUS MAP REGISTER ADDRESS UNIQUENESS

```

1017      .SBTTL  TEST - UNIBUS MAP REGISTER ADDRESS UNIQUENESS
1018
1019      ;
1020      ; * THIS TEST IS DESIGNED TO DETECT ANY DUAL ADDRESSING
1021      ; * IN THE UNIBUS MAP ADDRESSING LOGIC
1022      ;
1023      ;   BGNTST
1024      ;
1025      ;       DO FOR ADDRESS := 170200 TO 170376
1026      ;       . WRITE ADDRESS TO ADDRESS
1027      ;       ENDDO
1028      ;       DO FOR ADDRESS := 170200 TO 170376
1029      ;       . READ ADDRESS
1030      ;       . IF CONTENTS OF ADDRESS NE ADDRESS THEN
1031      ;       .     REPORT ERROR IN ADDRESSING
1032      ;       .   ENDIF
1033      ;       ENDDO
1034      ;
1035      ;   ENDTST
1036      ;
1037      ; -----
1038
1039      ; *****
1040      ; *TEST 3      UNIBUS MAP REGISTER UNIQUENESS TEST
1041      ; *****
1042      TST3:  SCOPE
1043
1044      ; WRITE ADDRESS INTO ALL MAP REGISTERS
1045      ;
1046      ;       MOV      @MAPL00,R0      ; GET ADDRESS L. 1ST MAP REGISTER
1047      ;       MOV      @MAPL00,R1      ; GET ADDRESS OF 1ST MAP REGISTER
1048      ;       MOV      R0,(R1)+        ; . WRITE ADDRESS TO ITSELF
1049      ;       ADD      #2,R0            ; . GET THE NEXT ADDRESS
1050      ;       CMP      @170400,R0      ; . HAVE WE WRITTEN TO ALL MAP REGISTERS ?
1051      ;       BNE      54              ; . NO, THEN GO WRITE TO NEXT ADDRESS
1052      ;
1053      ; MAKE SURE EACH LOW MAP REGISTER CONTAINS IT'S ADDRESS
1054      ;
1055      ;       MOV      @MAPL00,R0      ; GET ADDRESS TO 1ST MAP REGISTER
1056      ;       MOV      @MAPL00,R1      ; GET ADDRESS TO 1ST MAP REGISTER
1057      ;       CMP      R0,(R1)+        ; . WAS THE ADDRESS WRITTEN TO THE MAP REGISTER ?
1058      ;       BEQ      154             ; . YES, THEN GO CHECK THE HIGH MAP REGISTER
1059      ;       TST      -(R0)           ; . NO GET THE ADDRESS THAT WAS WRITTEN
1060      ;       MOV      R0,@GLADR       ; . GET THE GOOD ADDRESS
1061      ;       MOV      (R0)+,@BADADR   ; . GET THE BAD ADDRESS, AND POINT BACK TO NEXT ADDR
1062      ;
1063      ;       ERROR    +4              ; . ERROR IN ADDRESSING OF MAP REGISTERS
1064      ;
1065      ; MAKE SURE EACH HIGH MAP REGISTER CONTAINS IT'S ADDRESS
1066      ;
1067      ;       ADD      #2,R0            ; . POINT R0 TO HIGH MAP REGISTER
1068      ;       MOV      R0,R2           ; . PUT A COPY OF IT INTO R2
1069      ;       BIC      @177700,R2      ; . MASK OUT THE UPPER 10 BITS
1070      ;       CMP      R2,(R1)+        ; . WAS THE ADDRESS WRITTEN TO THE HIGH MAP REGISTER
1071      ;
1072      ;       BEQ      204             ; . YES, THEN GO CHECK IF WE ARE THROUGH ALL THE MAP
1073      ;
1074      ;       TST      -(R0)           ; . NO GET THE ADDRESS THAT WAS WRITTEN
1075      ;       MOV      R0,@GDADR       ; . GET THE GOOD ADDRESS

```

003640	000004		
003642	012700	170200	
003646	012701	170200	
003652	010021		
003654	062700	000002	
003660	022700	170400	
003664	001372		
003666	012700	170200	
003672	012701	170200	
003676	020021		
003700	001406		
003702	005740		
003704	010037	001120	
003710	012037	001122	
003714	104004		
003716	062700	000002	
003722	010002		
003724	042702	177700	
003730	020221		
003732	001406		
003734	005740		
003736	010037	001120	

E3

KTJ11 B DIAGNOSTIC MACRO M1200 05-OCT-84 09:24 PAGE 13-1

SEQ 0030

T3 UNIBUS MAP REGISTER UNIQUENESS TEST

```

1071 003742 012037 001122      MOV      (R0)+,1BDADR      ; . GET THE BAD ADDRESS, AND POINT BACK TO NEXT ADDR
ESS 1072 003746 104004      ERROR    +4                ; . ERROR IN ADDRESSING OF MAP REGISTERS
1073                               ;
1074                               ; CHECK TO SEE IF WE HAVE CHECKED ALL MAP REGISTERS
1075                               ;
1076 003750 062700 000002      201:   ADD      #2,R0        ; . GET THE NEXT REGISTER
1077 003754 020027 170400      CMP      R0,#170400        ; . HAVE WE CHECKED ALL THE MAPPING REGISTERS
1078 003760 001346      BNE      101                ; . NO, THEN GO CHECK THE NEXT REGISTER
1079
1080
1081

```

TEST UNIQUENESS BETWEEN LOW MAP REGISTERS AND HIGH MAP REGIST

```

1083      .SBTTL   TEST - UNIQUENESS BETWEEN LOW MAP REGISTERS AND HIGH MAP REGISTERS
1084
1085      ;
1086      ; * THIS TEST IS DESIGNED TO TEST THE UNIQUENESS BETWEEN THE LOW
1087      ; * AND HIGH MAP REGISTERS
1088      ;
1089      ;    BGNTST
1090      ;
1091      ;        DO FOR ADDRESS := 170200 TO 170374
1092      ;        . WRITE #77777 TO ADDRESS
1093      ;        ENDDO
1094      ;        DO FOR ADDRESS := 170200 TO 170374 BY 4
1095      ;        . LET R2 := CONTENTS OF ADDRESS
1096      ;        . LET R3 := CONTENTS OF ADDRESS * 2
1097      ;        . LET R3 := R3 - R2
1098      ;        . IF R3 EQ 0 THEN
1099      ;        .    ERROR IN DIFFERENTIATING BETWEEN HI AND LO REGISTERS
1100      ;        .    ENDIF
1101      ;        ENDDO
1102      ;
1103      ;    ENDTST
1104      ;
1105      ; -----
1106      ; *****
1107      ; *TEST 4        HIGH AND LOW MAP REGISTER UNIQUENESS TEST
1108      ; *****
1109      ;
1110      ;    TST4:    SCOPE
1111      ;
1112      ;    WRITE #77777 TO ALL ADDRESSES
1113      ;
1114      ;        MOV        #MAPLOO,R0                    ; GET ADDRESS OF FIRST MAP REGISTER
1115      ;        50:    MOV        #77777,(R0).           ; . WRITE 77777 TO MAP REGISTER
1116      ;               CMP        R0,#170400           ; . HAVE WE WRITTEN TO ALL REGISTERS ?
1117      ;               BNE        50                    ; . NO, THEN GO WRITE TO THE NEXT REGISTER ?
1118      ;        ; MAKE SURE THAT HIGH MAP REGISTER BITS 6-15 ARE CLEARED
1119      ;        ;
1120      ;        100:    MOV        #MAPLOO,R0           ; GET ADDRESS OF FIRST MAP REGISTER
1121      ;               MOV        (R0)+,R2           ; . GET CONTENTS OF LOW MAP REGISTER
1122      ;               MOV        (R0)+,R3           ; . GET CONTENTS OF HIGH MAP REGISTER
1123      ;               SUB        R3,R2               ; . DID THE REGISTERS GET ADDRESSED CORRECTLY ?
1124      ;               BGT        150                ; . YES, THEN GO SEE IF WE HAVE CHECKED ALL THE MAP R
1125      ;               TST        -(R0)               ; . NO,
1126      ;               TST        -(R0)               ; . POINT TO THE LOW MAP REGISTER
1127      ;               MOV        (R0)+,BDDAT        ; . GET CONTENTS OF LOW MAP REGISTER
1128      ;               MOV        (R0)+,BDDAT        ; . GET CONTENTS OF HIGH MAP REGISTER
1129      ;               ERROR     *5                ; . ERROR - MAP REGISTERS WERE NOT ADDRESSED CORREC
1130      ;        ; SEE IF WE HAVE CHECKED ALL THE MAP REGISTER PAIRS
1131      ;        ;
1132      ;        150:    CMP        R0,#170400           ; . HAVE WE CHECKED ALL THE REGISTERS ?
1133      ;               BNE        100                ; . NO, THEN GO CHECK THE NEXT PAIR !!!
1134      ;

```

003762 000004

1108
1109
1110
1111
1112 003764 012700 170200
1113 003770 012720 077777
1114 003774 020027 170400
1115 004000 001373
1116
1117
1118
1119 004002 012700 170200
1120 004006 012002
1121 004010 012003
1122 004012 160302
1123 004014 003007
EGISTERS
1124 004016 005740
1125 004020 005740
1126 004022 012037 001124
1127 004026 012037 001126
1128 004032 104005
TLY
1129
1130
1131
1132 004034 020027 170400
1133 004040 001362
1134

TEST - DCSR REGISTER RESPONSE TEST

1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181

.SBTTL TEST - DCSR REGISTER RESPONSE TEST

```

;
; * THIS TEST IS DESIGNED TO SEE IF WE CAN ACCESS THE
; * DIAGNOSTIC CONTROLLER STATUS REGISTER. IF WE CAN IT
; * TESTS OUT BITS 3-6,9-14 OF THE DCSR AND MAKES SURE
; * THAT A BUS INIT HAS A PROPER RESPONSE IN THE DCSR.
;
;
;    BGNTST
;
;        LET 4 := ADDRESS OF ERROR ROUTINE
;        LET 6 := PRIORITY OF 7
;        LET ADDRESS := 177730
;        READ ADDRESS
;
; CHECK BITS 3-6,9-14 OF THE DCSR REGISTER
;
;        LET DCSR := DCSR .OR. #77170
;        READ DCSR
;        IF DCSR NE 0 THEN
;        . REPORT ERROR IN BITS 3-6,9-14 OF THE DCSR REGISTER
;        ENDIF
;
; CHECK OUT THE DCSR RESPONSE TO A RESET
;
;        CACHE THE RESET INSTRUCTION
;        LET DCSR := DCSR .OR. #206
;        DO A RESET TO INITIALIZE THE BUS
;        IF LOWER BYTE OF DCSR GT 0 THEN
;        . REPORT ERROR (BIT 7 DID NOT GET SET BY BUS INIT)
;        ELSE
;        . LET DCSR := DCSR .AND. #6
;        . IF DCSR NE 0 THEN
;        . . REPORT ERROR (BITS 1,2 DIDN'T CLEAR ON BUS INIT)
;        . ENDIF
;        ENDIF
;        EXIT TEST
;
; ERROR:
;        REPORT THAT ACCESS TO DCSR HAS TIMED OUT
;
;    ENDTST

```

```

;*****
; *TEST 5        DCSR REGISTER TEST
;*****
TST5:    SCOPE

```

1182
1183
1184
1185
1186 004042 000004
1187 004044 012737 004216 000004
1188 004052 012737 000340 000006
1189

```

;
; SET UP MEMORY TIMEOUT VECTOR
;
;        MOV        #201,B#4                    ; SET UP VECTOR PC TO ERROR ROUTINE
;        MOV        #340,B#6                    ; SET UP VECTOR PSW TO PRIORITY 7
;
; READ ADDRESS OF DCSR TO SEE IF IT TIMES OUT

```

T5 DCSR REGISTER TEST

```

1190
1191 004060 005737 177730      ;      TST      @DCSR      ; WILL THE DCSR TIMEOUT ?
1192 004064 012737 002234 000004      MOV      @TIMOUT,@A      ; RESTORE TIMEOUT VECTOR
1193
1194      ; TRY TO WRITE ONES TO BITS 3-6 AND 9-14 ( BIT 03 SHOULD STAY WRITTEN)
1195
1196 004072 052737 077170 177730      BIS      @77170,@DCSR      ; TRY TO WRITE TO READ ONLY BITS
1197 004100 032737 077160 177730      BIT      @77160,@DCSR      ; DID THEY GET WRITTEN TO ?
1198 004106 001412      BEQ      5:      ; NO, THEN GO CHECK THE EFFECT OF A RESET ON THE DCSR
1199 004110 013701 177730      MOV      @DCSR,R1      ; GET COPY OF DCSR INTO R1
1200 004114 042701 100607      BIC      @100607,R1      ; MASK OUT UNTESTED BITS
1201 004120 010137 001126      MOV      R1,@BDDAT      ; PUT THEM INTO BAD DATA
1202 004124 012737 000010 001124      MOV      @10,@GDDAT      ; GOOD DATA
1203 004132 104006      ERROR      +6      ; ERROR - IN BITS 3-6,9-14 OF DCSR
1204
1205      ; CACHE THE RESET INSTRUCTION
1206
1207 004134 012737 000200 001124 5:      MOV      @BIT7,@GDDAT      ; ONLY 7 SHOULD BE SET AFTER RESET
1208 004142 053737 177730 001124      BIS      DCSR,@GDDAT      ; SET BOOT BITS
1209 004150 005737 004162      TST      10:      ; CACHE THE RESET
1210
1211      ; SET UP DCSR, DO A RESET, MAKE SURE PROPER RESPONSE OCCURS
1212
1213 004154 052737 000006 177730      BIS      @6,@DCSR      ; SET BIT 7 AND CLEAR BIT 1,2
1214 004162 000005      RESET      ; DO A BUS INIT
1215 004164 032737 000200 177730 10:      BIT      @BIT7,@DCSR      ; DID BIT 7 GET SET ?
1216 004172 001001      BNE      15:      ; YES, THEN GO CHECK IF BITS 1,2 GOT CLEAR
1217 004174 104007      ERROR      +7      ; NO, THEN ERROR - BIT 7 DIDN'T GET SET
1218 004176 032737 000006 177730 15:      BIT      @6,@DCSR      ; DID BIT 1,2 GET CLEARED ?
1219 004204 001410      BEQ      TST6      ; YES, THEN GO TO NEXT TEST
1220 004206 013737 177730 001126      MOV      @DCSR,@BDDAT      ; NO, THEN GET CONTENTS OF THE DCSR
1221 004214 104007      ERROR      +7      ; ERROR - BIT 1,2 DID NOT GET CLEAR
1222
1223      ; ERROR ROUTINE
1224
1225 004216 012737 177730 001122 20:      MOV      @DCSR,@BDDAT      ; GET ADDRESS OF THE DCSR
1226 004224 104010      ERROR      +10      ; ERROR - DCSR REGISTER ACCESS HAS TIMED OUT
1227
1228

```

TEST - KMCR BITS TEST

.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

```

```

; SETUP TIMEOUT VECTOR TO ERROR ROUTINE
; VERIFY THAT AT LEAST SOME PMI MEMORY PRESENT

```

```

;
;     MOV     #104, @R4      ; SET VECTOR PC TO ERROR ROUTINE
;     MOV     #340, @R6      ; SET VECTOR PSW TO ERROR ROUTINE
;     MOV     KMCR, @TMP0     ; STORE KMCR
;     MOV     @TIMOUT, @R4    ; RESTORE TIMEOUT VECTOR
;     BIC     #177700, @TMP0  ; CLEAR ALL BUT U.MEMORY
;     CMP     #77, @TMP0      ; NO PMI MEMORY?
;     BNE     2$             ; IF SOME, GO TO THE NEXT TEST
;     CLR     PHIS           ; 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$:  MOV     KMCR, R1          ; STORE PATTERN FROM KMCR TO R1
;     COM     R1              ; COMPLEMENT THE PATTERN
;     BIC     #177700, R1     ; CLEAR BITS <15-6>
;     BIC     @BIT08, DCSR    ; MAKE SURE THAT NOT IN DIAGN. MODE
;     BIC     @BIT03, R1     ; ENABLE 32K, IN CASE OF ERROR
;     BIS     R1, KMCR        ; TRY TO WRITE TO KMCR<5-0>
;     CMP     R1, KMCR        ; WRITTEN OK?
;     BNE     1$             ; 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     @TMP0, KMCR     ; RESTORE KMCR
;     ERROR   +11            ; NO, THEN ERROR - BITS DID NOT GET SET CORRECTLY IN

```

```

1$:  BR      TST7             ; GO TO THE NEXT TEST

```

```

1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
004226 000004
1251
1252
1253
1254
1255
1256 004230 012737 004374 000004
1257 004236 012737 000340 000006
1258 004244 013737 177734 001160
1259 004252 012737 002234 000004
1260 004260 042737 177700 001160
1261 004266 022737 000077 001160
1262 004274 001003
1263 004276 005037 001720
1264 004302 000440
1265
1266
1267
1268 004304 013701 177734
1269 004310 005101
1270 004312 042701 177700
1271 004316 042737 000400 177730
1272 004324 042701 000010
1273 004330 050137 177734
1274 004334 020137 177734
1275 004340 001014
1276 004342 013737 177734 001124
1277 004350 010137 001126
1278 004354 012737 177734 001122
1279 004362 013737 001160 177734
1280 004370 104011
THE KMCR
1281 004372
004372 000404
1282

```

T6 KMCR MEMORY CONFIGURATION BITS TEST

SEQ 0035

```

1283                                     ; TIMEOUT ROUTINE
1284                                     ;
1285 004374 012737 177734 001122 104:      MOV      #KMCR, #BDADR      ; GET THE ADDRESS OF THE KMCR
1286 004402 104012                                ERROR      +12      ; ERROR - MEMORY TIMEOUT HAS OCCURED

```

TEST - UNIBUS TIMEOUT TEST

```

1288 .SBTTL TEST - UNIBUS TIMEOUT TEST
1289
1290 ;* THIS TEST CHECKS OUT THE UNIBUS TIMEOUT LOGIC
1291 ;* BEFORE IT IS USED IN ANY TESTS
1292 ;
1293 ; BGNTST
1294 ;
1295 ;     LET 4 := THE ADDRESS 104
1296 ;     LET 6 := #340
1297 ;     MAKE SURE THAT UBA IS IN DIAGNOSTIC MODE,DCSR<8> IS SET
1298 ;     READ UNIBUS LOCATION
1299 ;     IF NO TIMEOUT THEN
1300 ;         ERROR - UBA DIDN'T TIMEOUT ON UNIBUS ACCESS IN DIAGNOSTIC MODE
1301 ;     ENDIF
1302 ; 104: CLEAN UP THE STACK
1303 ;
1304 ; ENDTST
1305 ;
1306 ;-----
1307
1308 ;*****
1309 ;*TEST 7          UNIBUS TIMEOUT TEST
1310 ;*****
1311 ;TST7:  SCOPE
1312 ;
1313 ; SET UP MEMORY TIMEOUT VECTOR AND PUT UBA INTO DIAGNOSTIC MODE
1314 ;
1315 ;     TST     PMIS          ; ANY PMI MEMORY?
1316 ;     BEQ     TST10        ;; IF NONE, EXIT TEST
1317 ;     MOV     #104,804     ; SET UP TIMEOUT VECTOR TO END OF TEST
1318 ;     MOV     #340,806     ;
1319 ;     BIS     #BIT8,DCSR   ; PUT UBA INTO DIAGNOSTIC MODE
1320 ;
1321 ; READ A UNIBUS LOCATION
1322 ;
1323 ;     TST     #170002      ; TRY TO READ UBE REGISTER ?
1324 ;     ERROR   +36         ; ERROR - SHOULD TIMEOUT HERE
1325 ;     BR      114         ; GO TO THE NEXT TEST
1326 ;
1327 ;
1328 ; TIMEOUT ROUTINE
1329 ;
1330 ;
1331 ; 104:  TST     (SP)+       ; CLEAN UP THE STACK
1332 ;      TST     (SP)+       ;
1333 ; 114:  MOV     #TIMOUT,804 ; RESTORE TIMEOUT VECTOR
1334 ;      BIC     #BIT8,DCSR  ;
1335 ;
1336

```

TEST - DATA OUT WITHOUT RELOCATION

```

1338 .SBTTL TEST - DATA OUT WITHOUT RELOCATION
1339
1340 ;* THIS TEST WILL PERFORM DIAGNOSTIC DATA OUT CYCLES WITHOUT ENABLING
1341 ;* RELOCATION. THE CYCLES WILL BE VERIFIED TO WORK CORRECTLY AND
1342 ;* DIAGNOSTIC NPR REGISTER WILL BE CHECKED TO BE ABLE TO STORE PROPER
1343 ;* DATA.
1344 ;
1345 ; BGNTST
1346 ;
1347 ; MAKE SURE THAT DIGNOSTIC MODE IS ON DCSR<8>=1
1348 ; SELECT DIAGNOSTIC NPR THRU DDR BY DCSR<2-1>=<0,0>
1349 ; LET TEST_LOCATION = 0
1350 ; DO FOR PATTERN = 377,7417,31463,52525,125152 (BINARY DIVIDE)
1351 ; . CALL DIAGNOSTIC_DATA_OUT <TEST_LOCATION,PATTERN>
1352 ; . IF DDR NE PATTERN THEN
1353 ; . . ERROR IN PATH
1354 ; . . ENDF
1355 ; . IF TEST_LOCATION NE PATTERN THEN
1356 ; . . ERROR PERFORMINDG DATA OUT
1357 ; . . ENDF
1358 ; . ENDDO
1359 ;
1360 ; ENDTST
1361 ;
1362 ;-----
1363 ;*****
1364

```

PC	OPCODE	DISASSEMBLY	COMMENT
1365	004466 000004	TST10: SCOPE	
1366	004470 005737 001720	TST PMIS	; ANY PMI MEMORY?
1367	004474 001453	BEQ TST11	:: IF NONE, EXIT TEST
1368	004476 052737 000400 177730	BIS #BIT08,DCSR	; MAKE SURE DIAGNOSTIC MODE IS ON
1369	004504 042737 000006 177730	BIC #BIT02:BIT01,DCSR	; MAKE SURE NPR REGISTER IS SELECTED
1370	004512 005037 001160	CLR \$TMP0	; CLEAR TEST LOCATION
1371	004516 012702 001774	MOV #PTRN16,R2	; POINT TO BINARY DIVIDE PATTERN
1372	004522 012701 001160	MOV #TMP0,R1	; STORE TEST LOCATION
1373	004526 032737 000200 177730	BIT #BIT07,DCSR	; READY BIT SET?
1374	004534 001001	BNE 1\$	; IF SO, BRANCH
1375	004536 104017	ERROR +17	; DCSR<7> NOT SET
1376	004540 012237 001124 1\$:	MOV (R2)+,\$GDDAT	; . STORE PATTERN
1377	004544 004737 002210	JSR PC,DDOUT	; . PERFORM DIAGNOSTIC DATA OUT
1378	004550 023737 177732 001124	CMP DDR,\$GDDAT	; . DDR HAS PROPER PATTERN
1379	004556 001404	BEQ 2\$	; . IF YES, BRANCH
1380	004560 013737 177732 001126	MOV DDR,\$BDDAT	; . STORE DDR FOR ERROR REPORTS
1381	004566 104013	ERROR +13	; . ERROR IN DATA PATH
1382	004570 023737 001160 001124 2\$:	CMP \$TMP0,\$GDDAT	; . DATA OUT OK?
1383	004576 001404	BEQ 3\$	; . BRANCH, IF YES
1384	004600 013737 001160 001126	MOV \$TMP0,\$BDDAT	; . STORE MEMGRY FOR ERROR REPORTS
1385	004606 104014	ERROR +14	; . ERROR IN DATA OUT
1386	004610 022712 000000 3\$:	CMP #0,(R2)	; . LAST PATTERN?
1387	004614 001351	BNE 1\$	; . IF NOT, BRANCH
1388	004616 042737 000400 177730	BIC #BIT8,DCSR	

TEST - DATA IN WITHOUT RELOCATION

```

1390      .SBTTL TEST - DATA IN WITHOUT RELOCATION
1391
1392      ; * THIS TEST WILL PERFORM DIAGNOSTIC DATA IN CYCLE AND VERIFY ITS
1393      ; * OPERATION
1394      ;
1395      ;   BGNTST
1396      ;
1397      ;       MAKE SURE THAT DIGNOSTIC MODE IS ON DCSR<8>=1
1398      ;       SELECT DIAGNOSTIC NPR THRU DDR BY DCSR<2-1>=<0,0>
1399      ;       LET PATTERN = 52525
1400      ;       LET DCSR<0> = #1
1401      ;       WRITE PATTERN TO TEST_LOCATION
1402      ;       IF DDR NE PATTERN THEN
1403      ;           . ERROR PERFORMING DATA IN
1404      ;       ENDIF
1405      ;
1406      ;   ENDTST
1407      ;
1408      ;-----
1409
1410      ; *****
1411      ; *TEST 11      DATA IN WITHOUT RELOCATION
1412      ; *****
1413      TST11:  SCOPE
1414
1415      004624  000004
1416
1417      004626  005737  001720      TST      PHIS      ; ANY PHI MEMORY?
1418      004632  001433      BEQ      TST12      ;: IF NONE, EXIT TEST
1419      004634  052737  000400  177730  BIS      #BIT08,DCSR      ; MAKE SURE DIAGNOSTIC MODE IS ON
1420      004642  042737  000006  177730  BIC      #BIT02:BIT01,DCSR      ; MAKE SURE NPR REGISTER IS SELECTED
1421      004650  012737  052525  001124  MOV      #52525,$GDDAT      ; USE ALTERNATING 1'S AND 0'S
1422      004656  052737  000001  177730  BIS      #BIT00,DCSR      ; SET GO BIT
1423      004664  013737  001124  001160  MOV      $GDDAT,$TMP0      ; STORE PATTERN IN TEST LOCATON
1424
1425      ; NOW COMPARE THE RESULTS
1426      ;
1427      004672  023737  177732  001124  CMP      DDR,$GDDAT      ; DDR GOT DATA OK?
1428      004700  001004      BNE      1$      ; IF NO, ERROR
1429      004702  042737  000400  177730  BIC      #BIT08,DCSR
1430      004710  000404      BR      TST12      ;:EXIT
1431      004712  013737  177732  001126  1$: MOV      DDR,$BDDAT      ; STORE DDR FOR ERROR REPORTS
1432      004720  104015      ERROR  +15      ; ERROR IN DATA IN

```

TEST - CONTENT OF DDR

```

1429 .SBTTL TEST - CONTENT OF DDR
1430
1431 ;* THIS TEST VERIFIES THAT IF DCSR<2-1> ARE NOT ZERO, DDR IS ALWAYS
1432 ;* READ AS 0, MEANING THAT NONE OF THE UNIBUS LINES ARE STUCK.
1433 ;
1434 ; BGNTST
1435 ;
1436 ; DO FOR DCSR<2-1> FROM <0,1> TO <1,1>
1437 ; . DO DATI
1438 ; . IF DDR NE 0 THEN
1439 ; . . ERROR IN UNIBUS LINES
1440 ; . ENDF
1441 ; ENDDO
1442 ;
1443 ; ENDTST
1444 ;
1445 ;-----
1446 ;*****
1447 ;*TEST 12 CONTENT OF DDR
;*****
TST12: SCOPE
1448 004722 000004 TST PHIS ; ANY PMI MEMORY?
1449 004724 005737 001720 BEQ TST13 ;: IF NONE, EXIT TEST
1450 004732 052737 000400 177730 BIS #BIT08,DCSR ; MAKE SURE IN DIAG. MODE
1451 004740 052737 000002 177730 BIS #BIT01,DCSR ; START WITH 1 IN DCSR
1452 004746 012703 000003 MOV #3,R3 ; DO 3 TIMES
1453 004752 005037 001124 CLR $GDDAT ; RECIEVED DATA 0
1454 004756 005001 1#: CLR R1 ; . ADDRESS 0
1455 004760 004737 002222 JSR PC,DOIN ; . DO DIAGNOSTIC DATI
1456 004764 022703 000001 CMP #1,R3 ; . CONTROL LINES SELECTED?
1457 004770 001403 BEQ 2# ; . IF YES, CHECK IT
1458 004772 005737 177732 TST DDR ; . ALL ZEROES?
1459 004776 001413 BEQ 3# ; . IF YES, BRANCH
1460 005000 012737 177400 001124 2#: MOV #177400,$GDDAT ; . EXPECTED PATTERN
1461 005006 032737 000373 177732 BIT #373,DDR ; . SELECT ONLY USED
1462 005014 001404 BEQ 3# ; . IF ALL ZEROES, BRANCH
1463 005016 013737 177732 001126 MOV DDR,$BDDAT ; . STORE RECIEVED DATA
1464 005024 104016 ERROR +16 ; . ERROR IN UNIBUS LINES
1465 005026 062737 000002 177730 3#: ADD #BIT01,DCSR ; . GET NEXT SET OF UNIBUS LINES
1466 005034 077330 SOB R3,1# ; . DO FOR ALL COMBINATIONS
1467 005036 042737 000400 177730 BIC #BIT8,DCSR ;
1468

```

TEST - INDIRECT ACCESSING OF UNIBUS MAP REGISTERS

1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526

.SBTTL TEST - INDIRECT ACCESSING OF UNIBUS MAP REGISTERS

* THIS TEST WILL CHECK THAT EACH OF THE UNIBUS MAP REGISTER PAIRS CAN BE
* ACCESSED INDIRECTLY AND THAT RELOCATION WORKS PROPERLY
* DATA OUT CYCLES WILL BE PERFORMED. EACH REGISTER EXCEPT THE ONE
* UNDER TEST WILL POINT TO NXH 17760000. THE REGISTER PAIR UNDER TEST
* WILL POINT TO 0, AND DIAGNOSTIC DATA OUT CYCLES WILL BE PERFORMED
* TO PHYSICAL LOCATION 0. IN UFD MODE THIS TEST WILL RUN ONLY IF
* KMCR<5>=0, I.E. 22 BIT MODE.

.BGNTST

```

IF UFD MODE AND KMCR<5> EQ #1 THEN
  . EXIT TEST (RELOCATION DOES NOT HAVE TO BE PERFORMED IN 18 BIT)
ENDIF
MAKE SURE THAT DIAGNOSTIC MODE IS ON DCSR<8>=1
SELECT DIAGNOSTIC NPR THRU DOR BY DCSR<2-1>=0,0>
DO FOR 32. MAPPING REGISTERS
  . LET LO REGISTER = #160000
  . LET HI REGISTER = #77
ENDDO (ALL REGISTERS POINT TO NXH 17760000)
POINT TIMEOUT ROUTINE TO VECTOR 4
CALL MAP_PROGRAM_AREA
LET MMRO<0> = 1 TO ENABLE MMU
LET MMR3<5> = 1 TO ENABLE RELOCATION
POINT R0 TO ADDRESS OF MAP REGISTER 0
LET KIPAR6 = #0 TO ACCESS REGISTER 0
LET (R0) = #0
LET (R0)*2 = #0 TO USE ONLY 16 BITS
LET R1 = #140000 TO ACCESS THRU KIPAR6
IF NOT UFD THEN
  . SAVE KMCR
  . CLEAR KMCR<4-0>
  . LET R4 = #31. (DO FOR ALL REGISTERS)
ELSE
  . LET R4 = 31. - (KMCR<4-1>) ONLY NOT DISABLED REGISTERS
ENDIF
DO FOR R3 FROM #0 TO R4 FOR ALL REGISTERS
  . CLEAR #00 (THIS LOCATION IS ACTUALLY USED)
  . LET PATTERN = R0 (ADDRESS OF LOW REGISTER)
  . CALL DIAGNOSTIC_DATA_OUT <(R1),PATTERN>
  . IF TIMEOUT THEN
    . ERROR IN UNIQUE ADDRESSING OF REGISTERS
  . ENDIF
  . IF #00 NE PATTERN THEN
    . ERROR PERFORMING DATA OUT
  . ENDIF
  . LET (R0) = #160000
  . LET (R0) = #77 TO MAP JUST USED PAIR TO NXH
  . LET (R0) = #0
  . LET (R0)*2 = #0 TO MAP NEXT PAIR TO TEST LOCATION
  . LET KIPAR6 = KIPAR6 + #200 TO ACCESS NEXT PAIR
ENDDO
DISABLE MMU

```

.ENDTST

TEST INDIRECT ACCESSING OF UNIBUS MAP REGISTERS

1527
1528
1529

```

.....
;*****
;TEST 13            INDIRECT ACCESSING OF UNIBUS MAP REGISTERS
;*****
TST13:  SCOPE

```

1530 005044 000004

```

1531 005046    005737    001720            TST        PHIS                    ; ANY PHI MEMORY?
1532 005052    001576                    BEQ        TST14                ; IF NONE, EXIT TEST
1533 005054    032737    000040    000052    BIT        @BIT05,B#52            ; UFD MODE ?
1534 005062    001406                    BEQ        11                    ; BRANCH, IF NOT
1535 005064    032737    000040    177734    BIT        @BIT05,KMCR           ; 18 BIT MODE ?
1536 005072    001402                    BEQ        11                    ; BRANCH, IF NOT
1537 005074    000137    005430            JMP        101                  ; EXIT TEST
1538 005100    052737    000400    177730    11:    BIS        @BIT08,DCSR           ; SET DIAGNOSTIC MODE
1539 005106    042737    000006    177730    BIC        @BIT02!BIT01,DCSR   ; SELECT NPR THRU DDR
1540
1541            ; POINT EACH OF MAPPING REGISTER PAIRS TO 17760000
1542
1543 005114    012701    000040            MOV        @32.,R1                ; DO FOR 32. REGISTER PAIRS
1544 005120    012702    170200            MOV        @MAPLO0,R2            ; POINT TO ADDRESS OF THE FIRST REGISTER
1545 005124    012722    160000    21:    MOV        @160000,(R2).        ; . 160000 -> LO REGISTER
1546 005130    012722    000077            MOV        @77,(R2).           ; . 77 -> HI REGISTER
1547 005134    077105                    SOB        R1,21                ; . CONTINUE UNTILL ALL DONE
1548
1549            ; INITIALISE TO DO RELOCATION IN 22 BIT MODE
1550
1551 005136    012737    002234    000004    MOV        @TIMEOUT,B#4           ; POINT TIMEOUT ROUTINE
1552 005144    004737    002244            JSR        PC,MAPPR            ; REMAP PROGRAM AREA
1553 005150    005237    177572            INC        MPRO                ; TURN ON MPU
1554 005154    052737    000040    172516    BIS        @BIT05,M#R3           ; ENABLE RELOCATION
1555 005162    012700    170200            MOV        @MAPLO0,R0           ; POINT R0 TO FIRST REGISTER
1556 005166    012737    000000    172354    MOV        @0, KIPAR6          ; LET KIPAR6 SELECT MAP PAIR 0
1557 005174    012710    000000            MOV        @0,(R0)             ; MOVE 0 TO LOW ADDRESS
1558 005200    012760    000000    000002    MOV        @0,2(R0)           ; MOVE 0 TO HI ADDRESS
1559 005206    012701    140000            MOV        @140000,R1          ; TO ACCESS THRU KIPAR6
1560
1561            ; STORE # OF REGISTERS TO TEST IN R4
1562
1563 005212    032737    000040    000052    BIT        @BIT05,B#52           ; UFD MODE ?
1564 005220    001010                    BNE        31                    ; IF YES, BRANCH
1565 005222    013737    177734    001162    MOV        KMCR,@TMP1          ; SAVE KMCR
1566 005230    005037    177734            CLR        KMCR                ; DO NOT DISABLE ANY REGISTERS
1567 005234    012704    000036            MOV        @30.,R4             ; DO FOR ALL REGISTERS
1568 005240    000407                    BR        41                    ; BRANCH AROUND
1569 005242    113703    177734    31:    MOV8       KMCR,R3             ; STORE # OF DISABLED REGISTERS
1570 005246    042703    177740            BIC        @177740,R3          ; LEAVE JUST BITS <4-0>
1571 005252    012704    000036            MOV        @30.,R4             ; STORE MAX. # OF REGISTERS
1572 005256    160304                    SUB        R3,R4                ; GET # OF ACCESSABLE REGISTERS
1573
1574            ; DO DATA OUT FOR EACH REGISTER PAIR UNDER TEST
1575
1576 005260    005003                    41:    CLR        R3                    ; START WITH REGISTER PAIR 0
1577 005262    005037    000600    51:    CLR        @00                ; . CLEAR LOCATION UNDER TEST
1578 005266    005037    001730            CLR        TOUT                ; . CLEAR TIMEOUT FLAG
1579 005272    010037    001124            MOV        R0,@GDOAT          ; . PATTERN IS ADDRESS OF LO REGISTER
1580 005276    004737    002210            JSR        PC,DDOUT           ; . DO DIAGN. DATA OUT

```

T13 INDIRECT ACCESSING OF UNIBUS MAP REGISTERS

1581	005302	005737	001730			TST	TOUT
1582	005306	001411				BEG	6:
1583	005310	013737	172354	001122		MOV	KIPAR6, #BDADR
1584	005316	012705	000007			MOV	#7,R5
1585	005322	006237	001122		11:	ASR	#BDADR
1586	005326	077503				SQB	R5,11:
1587	005330	104020				ERROR	+20
1588	005332	023737	000000	001124	6:	CMP	#0,#GDDAT
1589	005340	001404				BEG	7:
1590	005342	013737	000000	001126		MOV	#0,#BDDAT
1591	005350	104013				ERROR	+13
1592	005352	012720	160000		7:	MOV	#160000,(R0).
1593	005356	012720	000077			MOV	#77,(R0).
1594	005362	012710	000000			MOV	#0,(R0)
1595	005366	012760	000000	000002		MOV	#0,2(R0)
1596	005374	062737	000200	172354		ADD	#200,KIPAR6
1597	005402	062703	000001			ADD	#1,R3
1598	005406	020304				CMP	R3,R4
1599	005410	003724				BLE	5:
1600	005412	032737	000040	000052		BIT	#BIT05,#52
1601	005420	001003				BNE	10:
1602	005422	013737	001162	177734		MOV	#TMP1,KHCR
1603	005430	005037	177572		10:	CLR	MHR0
1604	005434	042737	000040	172516		BIC	#BIT05,MHR3
1605	005442	042737	000400	177730		BIC	#BIT8,DCSR

```

1 . TIMEOUT FLAG SET?
2 . IF NOT BRANCH
3 . STORE # OF PAIR FOR ERRORS
4 . PREPARE TO SHIFT TO GET TO LOWER BITS
5 . . SHIFT ONCE
6 . . SHIFT 7 TIMES
7 . IN UNIQUE ADDRESSING OF REGISTERS
8 . DATA OUT OK?
9 . IF YES, BRANCH
10 . STORE FOR ERROR REPORTS
11 . PERFORMING DATA OUT
12 . POINT JUST USED REGISTER
13 . TO MMX
14 . POINT NEXT REGISTER PAIR
15 . TO LOCATION 0
16 . ACCESS NEXT REGISTER PAIR
17 . INCREMENT COUNT
18 . ALL AVAILABLE PAIRS DONE?
19 . BRANCH IF NOT
20 CHECK TO SEE IF KMCR WAS SAVED
21 NO, DON'T RESTORE KMCR
22 RESTORE KMCR
23 DISABLE MPU
24 DISABLE MAPPING
25 EXIT DIAGNOSTIC MODE

```

TEST - DISABLING OF THE MAPPING REGISTERS

```

1607 .SBTTL TEST - DISABLING OF THE MAPPING REGISTERS
1608
1609 ; * THIS TEST WILL CHECK THAT NONE OF THE REGISTERS DISABLED BY KMCR<4-0>
1610 ; * PERFORM RELOCATION. IN UFD MODE ONLY THE BITS ACTUALLY SET IN THE KMCR WILL
1611 ; * BE CHECKED, OTHERWISE ALL OF THEM EXCEPT THE FIRST 4 ARE CHECKED.
1612 ; * IN UFD MODE THIS TEST WILL RUN ONLY IF KMCR<5>=0. (IN 22 BIT MODE)
1613 ;
1614 ; BGNTEST
1615 ;
1616 ; IF UFD MODE AND KMCR<5> EQ #1 THEN
1617 ; . EXIT TEST (RELOCATION DOES NOT HAVE TO BE PERFORMED IN 18 BIT)
1618 ; ENDF
1619 ; MAKE SURE THAT DIAGNOSTIC MODE IS ON DCSR<8>=1
1620 ; SELECT DIAGNOSTIC NPR THRU DOR BY DCSR<2-1>=0,0
1621 ; CALL MAP_PROGRAM_AREA
1622 ; LET MMR3<5> = #1 TO ENABLE RELOCATION
1623 ; LET MMR0<0> = #1 TO ENABLE MMU
1624 ; LET R1 = #140000 TO ACCESS THRU KIPAR6
1625 ; LET KIPAR6 = #7600 TO ACCESS MAP REG. 32.
1626 ; SET 4 TO IGNORE TIMEOUT
1627 ; LET PATTERN = #125252
1628 ; LET B#0 = #0
1629 ; LET MAPL37 = #0
1630 ; LET MAPH37 = #0
1631 ; CALL DIAGNOSTIC_DATA_OUT<(R0),PATTERN>
1632 ; IF B#0 EQ PATTERN THEN
1633 ; . ERROR REG. PAIR 37 PERFORMS RELOCATION
1634 ; ENDF
1635 ;
1636 ; ENDTST
1637 ;
1638 ; -----
1639 ; *****
1640 ; *TEST 14            DISABLING REGISTERS
1641 ; *****
1642 ; TST14: SCOPE
1643
1644 ;
1645 ;            IST            PHIS            ; ANY PHI MEMORY?
1646 ;            BEQ            TST15            ; IF NONE, EXIT TEST
1647 ;            BIT            #BIT05,B#52            ; UFD MODE ?
1648 ;            BEQ            11            ; BRANCH, IF NOT
1649 ;            BIT            #BIT05,KMCR            ; 18 BIT MODE ?
1650 ;            BNE            TST15            ; EXIT TEST IF 18 BIT MODE
1651
1652 ; INITIALISE MMU AND DIAGNOSTIC REGISTERS
1653 ;
1654 ; 14:            BIS            #BIT08,DCSR            ; SET DIAGNOSTIC MODE
1655 ;            BIC            #BIT02!BIT01,DCSR            ; SELECT NPR THRU DOR
1656 ;            JSR            PC,MAPPR            ; REMAP PROGRAM AREA
1657 ;            INC            MMR0            ; TURN ON MMU
1658 ;            BIS            #BIT05,MMR3            ; ENABLE RELOCATION
1659
1660 ; TRY TO DO RELOCATION THRU REGISTER PAIR 31.
1661 ;
1662 ;            MOV            #21,B#4            ; INITIALISE TIMEOUT ROUTINE
1663 ;            MOV            #140000,R1            ; ACCESS THRU KIPAR6

```

T14 DISABLING REGISTERS

1661	005544	012737	007600	172354	MOV	#7600,KIPAR6	; ACCESS REG. PAIR 31.
1662	005552	012737	125252	001124	MOV	#125252,#GDDAT	; STORE PATTERN
1663	005560	005037	000000		CLR	#00	; CLEAR LOCATION UNDER TEST
1664	005564	012737	000000	170374	MOV	#0,MAPL37	; LET MAP PAIR 31.
1665	005572	012737	000000	170376	MOV	#0,MAPH37	; POINT TO 0
1666	005600	013737	001124	177732	MOV	#GDDAT,DDR	; STORE PATTERN IN DDR
1667	005606	005011			CLR	(R1)	; EXTENAL READ TO PROVIDE ADDRESS
1668	005610	104021			ERROR	+21	; RELOCATION PERFORMED THRU REG. 31.
1669	005612	000402			BR	31	; EXIT TEST
1670	005614	005726			21: TST	(SP)+	; RESTORE STACK
1671	005616	005726			TST	(SP)+	
1672	005620	005037	177572		31: CLR	MMR0	; DISABLE MMU
1673	005624	042737	000040	172516	BIC	#BIT05,MMR3	; DISABLE MAPPING
1674	005632	012737	002234	000004	MOV	#TIMOUT,004	; RESTORE TIMEOUT
1675	005640	042737	000400	177730	BIC	#BIT8,DCSR	; EXIT DIAGNOSTIC MODE

TEST - NXM MEMORY TIMEOUT

```

1677 .SBTTL TEST - NXM MEMORY TIMEOUT
1678
1679 ;* THIS TEST WILL VERIFY THAT WHENEVER LOCATION 17760000 IS ACCESSED
1680 ;* DIAGNOSTIC NPR CYCLES RESULT IN A TIMEOUT SETTING BIT <15> IN DCSR.
1681 ;
1682 ; BGNTST
1683 ;
1684 ; MAKE SURE THAT DIGNOSTIC MODE IS ON DCSR<8>=1
1685 ; SELECT DIAGNOSTIC NPR THRU DDR BY DCSR<2-1>=<0,0>
1686 ; CALL MAP_PROGRAM_AREA
1687 ; LET MMRO<0> = 1 TO ENABLE MMU
1688 ; IF NOT UFD MODE OR KMCR<5> EQ #0
1689 ; . LET TEST_LOCATION = #140000 TO ACCESS THRU KIPAR6
1690 ; . LET KIPAR6 = #0 TO ACCESS REGISTER PAIR 0
1691 ; . LET LO REGISTER = #160000 TO SET REG. 0 TO NXM
1692 ; . LET HI REGISTER = #77
1693 ; . LET MMR3<5> = 1 TO ENABLE RELOCATION
1694 ; ELSE
1695 ; . IF KMCR<5> EQ #1 THEN(18 BIT MODE)
1696 ; . . LET TEST_LOCATION = #140000 TO ACCESS THRU KIPAR6
1697 ; . . LET KIPAR6 = #177600 TO ACCESS NXM
1698 ; . ENDIF
1699 ; ENDIF
1700 ; START DIAGNOSTIC_DATA_OUT TO #00
1701 ; IF DCSR<15> NE 0 THEN
1702 ; . ERROR NXM BIT SET
1703 ; ENDIF
1704 ; PUT ACTUAL EXTERNAL ADDRESS ON THE BUS
1705 ; IF DCSR<15> NE 1 THEN
1706 ; . ERROR NXM MEMORY DOES NOT SET DCSR<15>
1707 ; ENDIF
1708 ;
1709 ; ENDTST
1710 ;
1711 ;-----
1712 ;*****
1713 ;*TEST 15            NXM MEMORY TIMEOUT
1714 ;*****

```

```

005646 000004
1714
1715 005650 005737 001720            TST    PHIS            ; ANY PHI MEMORY?
1716 005654 001476            BEQ    TST16            ; IF NONE, EXIT TEST
1717 005656 052737 000400 177730    BIS    #BIT8,DCSR       ; SELECT DIAGNOSTIC MODE
1718 005664 004737 002244            JSR    PC,HAPPR       ; REMAP PROGRAM AREA
1719 005670 005237 177572            INC    MMRO           ; ENABLE MMU
1720
1721 ; DECIDE WHETHER TO ACCESS WITH RELOCATION OR NOT
1722 ;
1723 005674 012701 140000            MOV    #140000,R1       ; ACCESS THRU KIPAR6
1724 005700 032737 000040 000052    BIT    #BIT05,#52       ; UFD MODE?
1725 005706 001410            BEQ    1#            ; IF NOT, BRANCH
1726 005710 032737 000040 177734    BIT    #BIT05,KMCR       ; 3 BIT MODE?
1727 005716 001412            BEQ    2#            ; IF NOT BRANCH
1728 005720 012737 177600 172354    MOV    #177600,KIPAR6     ; KIPAR6 HAS NXM
1729 005726 000422            BR    3#            ; GO DO IT
1730 005730 013737 177734 001162 1# : MOV    KMCR,#TMP1       ; SAVE KMCR

```

T15 NXM MEMORY TIMEOUT

```

1731 005736 042737 000040 177734      BIC      #BIT05,KMCR      ; IN NOT UFD, DO IN 22 BITS
1732 005744 012737 000000 172354 24:  MOV      #0,KIPAR6      ; KIPAR6 ACCESS MAP REG. 0
1733 005752 012737 160000 170200      MOV      #160000,MAPL00    ; MAP PAIR HAS NXM
1734 005760 012737 000077 170202      MOV      #77,MAPH00
1735 005766 052737 000040 172516      BIS      #BIT05,MMR3      ; ENABLE RELOCATION
1736
1737      ; DO ACTUAL DATA OUT CYCLE
1738
1739 005774 005037 177732      34:  CLR      DOR      ; PROMT DIAG. DATO
1740 006000 005011      CLR      (R1)      ; WRITE TO NXM
1741 006002 032737 100000 177730      BIT      #BIT15,DCSR      ; NXM SET?
1742 006010 001001      BNE      54      ; IF YES, BRANCH
1743 006012 104022      ERROR      +22      ; NXM ACCESS DOES NOT SET DCSR<15>
1744 006014 005037 177572      54:  CLR      MMR0      ; DISABLE MMU
1745 006020 042737 000040 172516      BIC      #BIT05,MMR3      ; AND MAPPING TOO
1746 006026 032737 000040 000052      BIT      #BIT05,B#52      ; UFD MODE
1747 006034 001006      BNE      TST16      ; IF NOT UFD, GO TO NEXT TEST
1748 006036 013737 001162 177734      MOV      #TMP1,KMCR      ; OTHERWISE, RESTORE KMCR
1749 006044 042737 000400 177730      BIC      #BIT8,DCSR      ; EXIT DIAGNOSTIC MODE
1750

```

TEST - CARRY PROPOGATION TEST

```

1752 .SBTTL TEST - CARRY PROPOGATION TEST
1753
1754 ;* THIS TEST CHECKS THE CARRY PROPOGATION THRU THE ALU. MAPPING REGISTER PAIR 0
1755 ;* IS LOADED WITH ALL 1'S. LOCATION 1 IS ACCESSED. THE RESULT OF THE DATO
1756 ;* CYCLE IS SUPPOSED TO BE LOADED IN LOCATION 0.
1757
1758 ; BGNTEST
1759
1760 ; IF UFD MODE AND KMCR<5> EQ #1 THEN
1761 ; . EXIT TEST (RELOCATION DOES NOT HAVE TO BE PERFORMED IN 18 BIT)
1762 ; ENDF
1763 ; MAKE SURE THAT DIGNOSTIC MODE IS ON DCSR<8>=1
1764 ; SELECT DIAGNOSTIC NPR THRU DDR BY DCSR<2-1>=<0,0>
1765 ; POINT TIMEOUT ROUTINE TO VECTOR 4
1766 ; CALL MAP_PROGRAM AREA
1767 ; LET MMRO<0> = 1 TO ENABLE MMU
1768 ; LET MMR3<5> = 1 TO ENABLE RELOCATION
1769 ; LET KIPAR6 = #0 TO ACCESS THRU MAP REG.0
1770 ; LET R1 = #140001 TO ACCESS THRU KIPAR6
1771 ; LET LO MAP REG. = #177777
1772 ; LET HI MAP REG. = #77
1773 ; LET PATTERN = #125252
1774 ; CALL DIAGNOSTIC_DATA_OUT<(R1),PATTERN>
1775 ; LET MMRO<0> = #0 TO DISABLE MMU
1776 ; IF #0 NE #125252 THEN
1777 ; . ERROR IN CARRY PROPOGATION
1778 ; ENDF
1779
1780 ; ENDTST
1781
1782
1783
1784

```

```

;*****
;*TEST 16      CARRY PROPOGATION TEST
;*****
TST16:  SCOPE

1785      006052    000004
1786 006054    005737    001720      TST      PHIS      ; ANY PHI MEMORY?
1787 006060    001470                BEQ      TST17      ; IF NONE, EXIT TEST
1788 006062    052737    000040    000052      BIT      #BIT05,#52      ; UFD MODE ?
1789 006070    001404                BEQ      14      ; BRANCH, IF NOT
1790 006072    032737    000040    177734      BIT      #BIT05,KMCR      ; 18 BIT MODE ?
1791 006100    001060                BNE      TST17      ; GO TO NEXT TEST, IF YES
1792
1793 ; INITIALISE MMU AND DIAGNOSTIC REGISTERS
1794
1795 006102    052737    000400    177730    14:      BIS      #BIT08,DCSR      ; SET DIAGNOSTIC MODE
1796 006110    042737    000006    177730      BIC      #BIT02!BIT01,DCSR      ; SELECT NPR THRU DDR
1797 006116    004737    002244                JSR      PC,MAPPR      ; REMAP PROGRAM AREA
1798 006122    005237    177572                INC      MMRO      ; TURN ON MMU
1799 006126    052737    000040    172516      BIS      #BIT05,MMR3      ; ENABLE RELOCATION
1800
1801 ; ACCESS LOCATION 0 THRU RELOCATION REGISTERS
1802
1803 006134    005037    172354                CLR      KIPAR6      ; CLEAR KIPAR6
1804 006140    012701    140002                MOV      #140002,R1      ; ACCESS THRU KIPAR6
1805 006144    012737    177777    170200      MOV      #177777,MAPLOO      ; ALL ONE'S TO LO REG

```

T16 CARRY PROPOGATION TEST

1806	006152	012737	000077	170202	MOV	#77,MAPH00	; ALL ONE'S TO HI REG
1807	006160	012737	125252	001124	MOV	#125252,\$GDDAT	; THE PATTERN TO BE STORED
1808	006166	005037	000000		CLR	#0	; CLEAR TEST LOCATION
1809	006172	004737	002210		JSR	PC,DDOUT	; DO DATA OUT
1810	006176	005037	177572		CLR	MMR0	; DISABLE MMU
1811	006202	042737	000040	172516	BIC	#BIT05,MMR3	; AND MAPPING
1812	006210	042737	000400	177730	BIC	#BIT8,DCSR	; EXIT DIAGNOSTIC MODE
1813	006216	022737	125252	000000	CMP	#125252,#0	; DATA OUT OK?
1814	006224	001406			BEQ	TST17	:: IF YES, GO TO NEXT TEST
1815	006226	013737	000000	001126	MOV	#0,\$BDDAT	; STORE FOR ERROR REPORTS
1816	006234	005037	001122		CLR	\$BDADR	; ADDRESS 0
1817	006240	104023			ERROR	+23	; IN CARRY PROPOGATION
1818							

TEST - EXTENSIVE CARRY PROPOGATION TEST

```

1820 .SBTTL TEST - EXTENSIVE CARRY PROPOGATION TEST
1821
1822 ; * THIS TEST PERFORMS EXTENSIVE CHECKING OF CARRY PROPOGATION. FIRST 1K IS
1823 ; * CHECKED IN WORD INCREMENTS, AFTER THAT EACH 1K UP TO 4K. DIAGNOSTIC DATI
1824 ; * CYCLES ARE PERFORMED AND A MAPPING REGISTER PAIR USED IS ALWAYS AT ALL 1'S.
1825
1826 ; BGNTEST
1827
1828 ; IF UFD MODE AND KMCR<5> EQ #1 THEN
1829 ; . EXIT TEST (RELOCATION DOES NOT HAVE TO BE PERFORMED IN 18 BIT)
1830 ; ENDF
1831 ; CALL MAP_PROGRAM_AREA
1832 ; ENABLE MAPPING
1833 ; INITIALISE ALL MAPPING REGISTERS TO POINT TO ALL 1'S
1834 ; DO FOR 4K OF MEMORY
1835 ; . IF MEMORY ACCESSED LESS THEN 1K
1836 ; . . DO WORD INCREMENTS
1837 ; . ELSE
1838 ; . . DO 1K INCREMENTS
1839 ; . ENDF
1840 ; . DO DATI INCREMENTS
1841 ; . IF DOR NE TO LOCATION IN MEMORY THEN
1842 ; . . ALU ERROR
1843 ; . ENDF
1844 ; ENDDO
1845
1846 ; ENDTST
1847
1848 -----
1849
1850 ; *****

```

```

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

1851 006242 000004
1852 006244 005737 001720 TST PMIS ; ANY PMI MEMORY?
1853 006250 001521 BEQ TST20 ; IF NONE, EXIT TEST
1854 006252 032737 000040 000052 BIT #BIT05,B#52 ; UFD MODE
1855 006260 001404 BEQ 14 ; IF NOT, BRANCH
1856 006262 032737 000040 177734 BIT #BIT05,KMCR ; 18 BIT MODE
1857 006270 001111 BNE TST20 ; IF 18 BIT MODE, DON'T DO THIS TEST
1858
1859 ; INITIALISE MMU AND RELOCATION
1860
1861 006272 052737 000400 177730 14: BIS #BIT08,DCSR ; SET DIAGNOSTIC MODE
1862 006300 042737 000006 177730 BIC #BIT02:BIT01,DCSR ; SELECT NPR THRU DDR
1863 006306 004737 002244 JSR PC,MAPPR ; REMAP PROGRAM AREA
1864 006312 052737 000060 172516 BIS #BIT05:BIT04,MMR3 ; ENABLE RELOCATION AND 22BITS
1865 006320 005237 177572 INC MMR0 ; ENABLE MMU
1866
1867 ; POINT ALL MAPPING REGISTERS TO ALL 1'S
1868
1869 006324 012701 170200 MOV #MAPL00,R1 ; START WITH 0
1870 006330 012703 000036 MOV #36,R3 ; DO FOR ALL REGISTERS
1871 006334 012721 177776 24: MOV #177776,(R1)+ ; LOW MAP REGISTER
1872 006340 012721 000077 MOV #77,(R1)+ ; HIGH MAP REGISTER
1873 006344 077305 SOB R3,24 ; DO FOR ALL

```

T17 EXTENSIVE CARRY PROPOGATION TEST

```

1874                                     ;
1875                                     ; START WITH DATI ON A WORD BOUNDARY
1876                                     ;
1877 006346 005037 172354                CLR    KIPAR6                ; ACCESS REGISTER PAIR 0
1878 006352 012701 140002                MOV    #140002,R1           ; START WITH 0 THRU KIPAR6
1879 006356 000401                        BR     4$                  ;
1880 006360 005721                        3$: TST    (R1)+           ; . DO IN WORD INCREMENTS
1881 006362 004737 002222                4$: JSR    PC,DOIN          ; . DO DATI
1882 006366 010137 001122                MOV    R1,#BDADR          ; . STORE JUST ACCESSED ADDRESS
1883 006372 162737 000002 001122        SUB    #2,#BDADR          ; . GET RID OF OVERFLOW
1884 006400 027737 172516 177732        CMP    #BDADR,DDR         ; . PROPER DATA?
1885 006406 001412                        BEQ     7$                  ; . IF EQUAL, BRANCH
1886 006410 017737 172506 001124        MOV    #BDADR,#GDDAT       ; . STORE RECEIVED DATA
1887 006416 013737 177732 001126        MOV    DDR,#BDDAT        ; . STORE EXPECTED DATA
1888 006424 042737 160000 001122        BIC     #BIT15:BIT14:BIT13,#BDADR ; . STRIP OF PAR BITS
1889 006432 104023                        ERROR  +23                ; . ERROR IN RELOCATION
1890                                     ;
1891                                     ; DECIDE WHICH CYCLE SHOULD OCCUR NEXT
1892                                     ;
1893 006434 005737 172354                7$: TST    KIPAR6           ; . FIRST TIME 1K BOUNDARY?
1894 006440 001011                        BNE     10$                ; . IF NOT, BRANCH
1895 006442 022701 144000                CMP     #144000,R1          ; . LESS THAN 1K DONE?
1896 006446 103744                        BLO     3$                  ; . DO IN WORD INCREMENTS
1897 006450 012701 140002                MOV     #140002,R1          ; . NOW DO INCREMENTING THRU PAR
1898 006454 012737 000040 172354        MOV     #40,KIPAR6          ; . FIRST 1K
1899 006462 000737                        BR     4$                  ; . GO DO COMPARE
1900 006464 062737 000040 172354        10$: ADD    #40,KIPAR6       ; . ACCESS NEXT 1K
1901 006472 022737 000200 172354        CMP     #200,KIPAR6        ; . OVER 4K BOUNDARY?
1902 006500 003330                        BGT     4$                  ; . IF NOT, GO DO DATI
1903 006502 005037 177572                CLR     MMRO              ; DISABLE MMU
1904 006506 042737 000400 177730        BIC     #BIT8,DCSR         ; EXIT DIAGNOSTIC MODE
1905

```

TEST - ALU TEST

1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937

.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

006514 000004

1938
1939 006516 005737 001720
1940 006522 001574
1941 006524 032737 000040 177734
1942 006532 001170
1943
1944
1945
1946 006534 052737 000400 177730
1947 006542 042737 000006 177730
1948 006550 004737 002244
1949 006554 052737 000060 172516
1950 006562 005237 177572
1951 006566 013702 001720
1952 006572 010237 001160
1953 006576 012703 004200
1954 006602 040203
1955 006604 072227 177766
1956
1957
1958
1959 006610 012701 140000
1960 006614 005037 172354

TST PHIS ; ANY 22-BIT PHI MEMORY?
BEQ TST21 ; IF NONE, EXIT TEST
BIT #BIT05,KMCR ; 18 BIT MODE
BNE TST21 ; IF 18 BIT MODE, DON'T DO THIS TEST
; INITIALISE MMU AND RELOCATION
14: BIS #BIT08,DCSR ; SET DIAGNOSTIC MODE
BIC #BIT02!BIT01,DCSR ; SELECT NPR THRU DOR
JSR PC,MAPPR ; REMAP PROGRAM AREA
BIS #BIT05!BIT04,MMR3 ; ENABLE RELOCATION AND 22BITS
INC MMR0 ; ENABLE MMU
MOV PHIS,R2 ; STORE SIZE OF PHI MEMORY
MOV R2,#TMP0 ; CREATE A PATTERN
MOV #4200,R3 ; CONSTANT FOR PAR
BIC R2,R3 ; MASK OUT NXM
ASH #-10.,R2 ; 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 ALJ TEST

```

1961 006620 005037 170200      CLR      MAPL00      ; CLEAR MAP REGISTERS
1962 006624 005037 170202      CLR      MAPH00      ;
1963 006630 005037 001162      CLR      $TMP1      ; CLEAR STORAGE FOR PAR
1964 006634 005037 001122      CLR      $BDADR      ; FIRST ADDRESS USED
1965 006640 005037 001124      CLR      $GDDAT      ; CLEAR PATTERN
1966 006644 004737 002222      2$: JSR      PC,DDIN      ; . DO DATI
1967 006650 013737 001162 172354 MOV      $TMP1,KIPAR6 ; . GET ADDED PAR
1968 006656 023777 177732 172236 CMP      DDR,$BDADR   ; . DATI OK?
1969 006664 001435          BEQ      10$      ; . IF SO, BRANCH
1970 006666 013737 177732 001126 MOV      DDR,$BDAT   ; . STORE RECIEVED DATA
1971 006674 017737 172222 001124 MOV      $BDADR,$GDDAT ; . STORE EXPECTED DATA
1972 006702 042737 160000 001122 BIC      $BIT15:$BIT14:$BIT13,$BDADR ; . STRIP OF PAR BITS
1973 006710 104023          ERROR    +23      ;
1974 006712 005037 001124      CLR      $GDDAT      ;
1975 006716 000420          BR       10$      ;
1976 006720 004737 002210      5$: JSR      PC,DDOUT     ; . DO DATO
1977 006724 013737 001162 172354 MOV      $TMP1,KIPAR6 ; . GET ADDED PAR
1978 006732 023777 001124 172167 CMP      $GDDAT,$BDADR ; . DATI OK?
1979 006740 001407          BEQ      10$      ; . IF SO, BRANCH
1980 006742 017737 172154 001126 MOV      $BDADR,$BDAT   ; . STORE EXPECTED DATA
1981 006750 042737 160000 001122 BIC      $BIT15:$BIT14:$BIT13,$BDADR ; . STRIP OF PAR BITS
1982 006756 104023          ERROR    +23      ;
1983          ;
1984          ; DECIDE WHAT ACCESS NEXT
1985          ;
1986 006760 005037 172354      10$: CLR      KIPAR6      ; . CLEAR PAR
1987 006764 062701 001042      ADD      $1042,R1      ; . START INCREMENTING
1988 006770 062737 021042 170200 ADD      $21042,MAPL00 ; . EVERY 4TH BIT
1989 006776 005537 170202      ADC      MAPH00      ; . DON'T FORGET CARRY
1990 007002 062737 000002 170202 ADD      $2,MAPH00    ; . FOR ALL REGISTERS
1991 007010 010137 001122      MOV      R1,$BDADR     ; . SAVE ADDRESS USED
1992 007014 063737 170200 001122 ADD      MAPL00,$BDADR ; . ADD UNIBUS MAP REG.
1993 007022 052737 140000 001122 BIS      $140000,$BDADR ; . MAKE SURE KIPAR6 SELECTED
1994 007030 042737 020000 001122 BIC      $20000,$BDADR  ;
1995 007036 022701 150420      CMP      $150420,R1    ; . OVERFLOW FROM ADDITION?
1996 007042 001003          BNE      15$      ; . IF NO, BRANCH
1997 007044 062737 000200 001162 ADD      $200,$TMP1   ; . ADD OVERFLOW FOR PAR
1998 007052 060337 001162      15$: ADD      R3,$TMP1   ; . ADD INCR. TO PAR
1999 007056 022737 002000 001162 CMP      $2000,$TMP1   ; . LESS THAN 32K?
2000 007064 003267          BGT      25$      ; . IF YES, DO DATI
2001 007066 005237 001124      INC      $GDDAT      ; . INCREMENT PATTERN
2002 007072 023737 001160 001162 CMP      $TMP0,$TMP1   ; . OVER AVAILABLE MEMORY?
2003 007100 003307          BGT      5$        ; . IF NOT, DO ANOTHER DATO
2004 007102 005037 177572      25$: CLR      MMRO      ; DISABLE MMU
2005 007106 042737 000400 177730 BIC      $BIT8,DCSR   ; EXIT DIAGNOSTIC MODE

```

TEST - MAIN MEMORY DISABLE

```

2007 .SBTTL TEST - MAIN MEMORY DISABLE
2008
2009 ;* THIS TEST WILL CHECK THAT A MAIN MEMORY RESPONSE IS DISABLED WHENEVER
2010 ;* THE APPROPRIATE BITS IN THE KMCR REGISTER ARE SET. IN UFD MODE ONLY BITS SET
2011 ;* IN KMCR WILL BE TESTED. IN ALL OTHER ENVIRONMENTS ALL BITS WILL BE TESTED.
2012 ;
2013 ; BGNTEST
2014 ;
2015 ; CALL MAP_PROGRAM_AREA
2016 ; LET MMRO<0> = 1 TO ENABLE MMU
2017 ; LET R1 = KMCR_LOW_BYTE
2018 ; CLEAR BITS <5,6,7> IN R1
2019 ; IF KMCR<5> EQ #1 THEN (18 BIT MODE)
2020 ; . LET KIPAR6 = #7400
2021 ; ELSE (22 BIT MODE)
2022 ; . LET KIPAR6 = #177400
2023 ; ENDIF
2024 ; LET TEST_LOCATION = #140000 TO ACCESS THRU KIPAR6
2025 ; DO FOR R0 FROM #1 TO R1 ALL PAGES SPECIFIED IN KMCR
2026 ; . LET R2 = #TEST_LOCATION
2027 ; . IF NO TIMEOUT THEN
2028 ; . . ERROR KMCR<4-5> DOESNOT DISABLE MAIN MEMORY
2029 ; . ENDIF
2030 ; . LET KIPAR6 = KIPAR6 - #200
2031 ; ENDDO
2032 ; IF NOT UFD MODE THEN
2033 ; . SAVE KMCR
2034 ; . CHECK KMCR<5-0> TO BE READ-WRITE
2035 ; . LET KMCR<5-0> = #0 (ALL NON-UNIBUS MEMORY)
2036 ; . LET KIPAR6 = #171600
2037 ; . IF #140000 DOES NOT TIMEOUT THEN
2038 ; . . LET KMCR<4-0> = <1000> TO DISABLE HIGHER THAN 28K
2039 ; . . IF #140000 DOESNOT TIMEOUT THEN
2040 ; . . . ERROR SETTING KMCR<5-0> DOESNOT DISABLE MEMORY
2041 ; . . ENDIF
2042 ; . ENDIF
2043 ; . RESTORE KMCR
2044 ; ENDIF
2045 ;
2046 ; ENDTST
2047 ;
2048 ;-----
2049 ;*****
2050 ;*****

```

```

007114 000004
2051 007116 005737 001720
2052 007122 001002
2053 007124 000137 007574
2054 007130 052737 000400 177730 201:
2055 007136 004737 002244
2056 007142 005237 171572
2057 007146 113701 177734
2058 007152 042701 177740
2059
2060

```

```

;*****
;TEST 21 MAIN MEMORY DISABLE
;*****
TST21: SCOPE
      TST     PHIS          ; ANY PHI MEMORY?
      BNE     201          ; IF YES, DO THE TEST
      JMP     1001         ; OTHERWISE EXIT TEST
      BIS     @BIT08,DCSR   ; STANDALONE MODE
      JSR     PC,MAPPR     ; REMAP PROGRAM AREA
      INC     MMRO         ; ENABLE MMU
      MOV8    KMCR,R1      ; SAVE KMCR
      BIC     #177740,R1   ; R1 HAS # OF PAGES DISABLED
;
; GET THE HIGHEST NXM LOCATION

```

T21 MAIN MEMORY DISABLE

```

2061
2062 007156 032737 000040 177734      ; BIT      #BIT05,KMCR      ; 18 BIT MODE?
2063 007164 001404      ; BEQ      11      ; IF NOT, BRANCH
2064 007166 012737 007400 172354      ; MOV      #7400,KIPAR6    ; NXM FOR 18 BITS
2065 007174 000403      ; BR       21      ; GO DO IT
2066 007176 012737 177400 172354 11:  ; MOV      #177400,KIPAR6  ; NXM FOR 22 BITS
2067
2068      ; GO THRU ALL PAGES OF MEMORY DISABLED
2069
2070 007204 105701      21:  ; TSTB     R1      ; ANY UNIBUS MEMORY?
2071 007206 001420      ; BEQ      61      ; IF NO, BRANCH
2072 007210 012737 007226 000004      ; MOV      #41,804      ; POINT TIMEOUT VECTOR TO PROGRAM
2073 007216 005737 140000      31:  ; TST      #0140000      ; . ACCESS THRU KIPAR6
2074 007222 104025      ; ERROR    +25      ; . KMCR<4-0> DOES NOT DISABLE MAIN MEMORY
2075 007224 000402      ; BR       51      ; .
2076 007226 005726      41:  ; TST      (SP).      ; . ADJUST STACK
2077 007230 005726      ; TST      (SP).      ; .
2078 007232 162737 000200 172354 51:  ; SUB      #200,KIPAR6    ; . ACCESS NEXT LOWER PAGE
2079 007240 077112      ; SOB      R1,31      ; . DO FOR ALL PAGES IN KMCR
2080 007242 012737 002234 000004      ; MOV      #TIMOUT,804    ; RESTORE TIMEOUT VECTOR
2081
2082      ; IN NON-UFD MODE, CHECK KMCR<5-0>
2083
2084 007250 032737 000040 000052 61:  ; BIT      #BIT05,8052    ; UFD MODE?
2085 007256 001146      ; BNE      TST22      ; IF YES, GO TO NEXT TEST
2086 007260 013737 177734 001162      ; MOV      KMCR,#TMP1      ; SAVE KMCR
2087 007266 012701 000067      ; MOV      #67,R1      ; HIGHEST VALUE (32K OF MEMORY)
2088 007272 110137 177734      71:  ; MOVB     R1,KMCR      ; . WRITE TO KMCR
2089 007276 120137 177734      ; CMPB     R1,KMCR      ; . WRITTEN OK?
2090 007302 001426      ; BEQ      81      ; . IF SO, BRANCH
2091 007304 010137 001124      ; MOV      R1,#GDDAT      ; . NO, THEN GET THE GOOD DATA
2092 007310 052737 000200 001124      ; BIS      #BIT07,#GDDAT  ; . IN CASE REBOOT BIT SET
2093 007316 123737 001124 177734      ; CMPB     #GDDAT,KMCR    ; . WAS THAT THE ONLY WRONG?
2094 007324 001415      ; BEQ      81      ; . IF YES, BRANCH
2095 007326 010137 001124      ; MOV      R1,#GDDAT      ; . IF NOT, RESTORE PATTERN WRITTEN
2096 007332 013737 177734 001126      ; MOV      KMCR,#GDDAT    ; . GET THE BAD DATA
2097 007340 012737 177734 001122      ; MOV      #KMCR,#BADADR  ; . GET THE ADDRESS OF THE KMCR
2098 007346 013737 001160 177734      ; MOV      #TMP0,KMCR     ; . RESTORE REGISTER
2099 007354 104011      ; ERROR    +11      ; . ERROR IN KMCR<5-0>
2100 007356 000446      ; BR       121      ; . EXIT TEST ON ERROR
2101 007360 005301      81:  ; DEC      R1      ; . LAST LOCATION (0)?
2102 007362 002343      ; BGE      71      ; . IF NOT, BRANCH
2103
2104      ; IN NON-UFD MODE, GO THRU THE PAGE JUST ABOVE 32K
2105
2106
2107 007364 052737 000020 172516      ; BIS      #BIT04,MHR3    ; ENABLE 22-BIT
2108 007372 013702 001720      ; MOV      PHIS,R2      ; STORE HIGHEST PAGE
2109 007376 005037 170200      ; CLR      MAPL00      ; CLEAR MAP REG. 0
2110 007402 012737 000001 170202      ; MOV      #1,MAPH00     ; POINTS TO 32K
2111 007410 052737 000040 172516      ; BIS      #BIT05,MHR3    ; ENABLE MAPPING
2112 007416 005001      ; CLR      R1      ; ADDRESS FOR DMA CYCLES
2113 007420 012737 007452 000004      ; MOV      #101,804      ; TIMEOUT TO PROGRAM
2114 007426 012737 002000 172354      ; MOV      #2000,KIPAR6   ; PAGE JUST ABOVE 32K
2115 007434 012737 000067 177734      ; MOV      #67,KMCR      ; DISABLE HIGHER THEN 32K
2116
2117      ; DO A CPU CYCLE TO DISABLED MEMORY

```

T21 MAIN MEMORY DISABLE

```

2118
2119 007442 005737 140000      91:   TST      @140000      ; . ACCESS DISABLED MEMORY
2120 007446 104025              ERROR    +25      ; . IN DISABLING MEMORY THRU KMCR<4-0>
2121 007450 000411              BR       121      ;
2122 007452 005726      101:   TST      (SP).      ; . RESTORE STACK
2123 007454 005726              TST      (SP).      ;
2124
2125      ; DO A DMA DATA CYCLE TO DISABLED MEMORY
2126
2127 007456 004737 002222      111:   JSR      PC,DDIN      ; . DIAGNOSTIC DATA
2128 007462 032737 100000 177730      BIT      @BIT15,DCSR      ; . NXM SET?
2129 007470 001001              BNE      121      ; . IF SET, BRANCH
2130 007472 104025              ERROR    +25      ; . IN DMA MEMORY NOT DISABLED
2131
2132      ; CHECK WHETHER ALL PAGES IN MEMORY VERIFIED
2133
2134 007474 023702 172354      121:   CMP      KIPAR6,R2      ; . IS IT IN MEMORY?
2135 007500 103020              BHIS     141      ; . IF NOT, BRANCH
2136 007502 062737 020000 170200      ADD      @20000,MAPL00      ; . ACCESS NEXT 4K
2137 007510 103002              BCC      131      ; . IF DIDN'T EXCEED 32K, BRANCH
2138 007512 005237 170202              INC      MAPH00      ; . OTHERWISE, INCREMENT MAP REGISTER
2139 007516 005337 177734      131:   DEC      KMCR      ; . ENABLE NEXT 4K OF MEMORY
2140 007522 032737 000040 177734      BIT      @BIT05,KMCR      ; . ALL 128K DONE?
2141 007530 001404              BEQ      141      ; . IF SO, BRANCH
2142 007532 062737 000200 172354      ADD      @200,KIPAR6      ; . ACCESS NEXT 4K
2143 007540 000740              BR       91      ; . TRY TO DO IT
2144 007542 012737 002234 000004      141:   MOV      @TIMOUT,@4      ; RESTORE TIMEOUT VECTOR
2145 007550 013737 001162 177734      MOV      @TMP1,KMCR      ; RESTORE KMCR
2146 007556 005037 172516              CLR      MMR3      ; DISABLE 22-BIT MODE
2147 007562 005037 177572              CLR      MMR0      ; DISABLE MMU
2148 007566 042737 000400 177730      BIC      @BIT8,DCSR      ; EXIT DIAGNOSTIC MODE
2149 007574      1001:

```

TEST - CACHE PRESENCE

2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166

.SBTTL TEST - CACHE PRESENCE

; * THIS TEST WILL FIND OUT WHETHER THE CACHE IS PRESENT

; BGNTST

```

;       LET KMCR<6>=01
;       IF KMCR<6> NE 01 THEN
;         CACHE NOT PRESENT OR NOT SEEN
;       ENDIF

```

; ENDTST

```

;*****
;TEST 22      CACHE PRESENCE
;*****

```

TST22: SCOPE

007574 000004

2167
2168 007576 005737 001720
2169 007602 001440
2170 007604 052737 000100 177734
2171 007612 032737 000100 177734
2172 007620 001031
2173 007622 005737 001206
2174 007626 001014
2175 007630 122737 000001 001220
2176 007636 001410
2177 007640 032737 000040 000052
2178 007646 001004
2179 007650 104401 007664
2180 007654 104401 001175
2181 007660 000137 013210
2182
2183 007664 040 116 117
007667 040 103 101
007672 103 110 105
007675 040 123 105
007700 105 116 000

```

TST      PHIS          ; ANY PHI MEMORY?
BEQ      TST23          ; IF NONE, EXIT TEST
BIS      @BIT06,KMCR    ; SET CACHE ENABLE BIT
BIT      @BIT06,KMCR    ; DID IT GET SET?
BNE      TST23          ; IF CACHE PRESENT, EXIT TEST
TST      @PASS          ; FIRST PASS?
BNE      11             ; IF YES, SKIP PRINTOUT
CMPB     @APTENV, @ENV   ; IN APT MODE?
BEQ      11             ; IF SO, SKIP PRINTOUT
BIT      @BIT05, @52     ; UFD MODE ?
BNE      11             ; IF YES, BRANCH
TYPE     ,NOCAH          ; TYPE NO CACHE MESSAGE
TYPE     ,@CRLF          ;
JMP      UBETST          ; IF NO CACHE, SKIP ALL CACHE TSTS

```

11: NOCAH: .ASCIZ / NO CACHE SEEN/

.EVEN

2184

TEST - CACHE DISABLED AND KMCR

```

2186 .SBTTL TEST - CACHE DISABLED AND KMCR
2187
2188 ;* THIS TEST WILL CHECK THAT WHENEVER THE DMA CACHE IS DISABLED
2189 ;* THE STATUS BITS IN KMCR REGISTER ARE INITIALISED TO POWER UP
2190 ;* CONDITIONS: THE VALID BITS ARE CLEARED, A IS THE NEXT AVAILABLE
2191 ;* SET, FOLLOWED BY B, C, AND D.
2192
2193 ; BGNSTST
2194
2195 ; DO FOR MMR3<5>=0,1 WITH RELOCATION ENABLED AND DISABLED
2196 ; . LET KMCR<6> = #0 TO DISABLE CACHE
2197 ; . LET KMCR<8>=0
2198 ; . IF KMCR<12!11!10!9> NE #0 THEN
2199 ; . ERROR IN VALID BITS WITH CACHE DISABLED
2200 ; . ENDF
2201 ; . LET KMCR<8>=1
2202 ; . IF KMCR<14!13!12!11!10!9!> NE #1 THEN
2203 ; . ERROR IN LRU BITS SELECTION
2204 ; . ENDF
2205 ; ENDDO
2206
2207 ; ENDTST
2208
2209
2210
2211

```

```

2212 007704 000004
2213 007706 005737 001720
2214 007712 001471
2215 007714 052737 000400 177730
2216 007722 042737 000040 172516
2217 007730 000403
2218 007732 052737 000040 172516 14:
2219 007740 042737 000100 177734 24:
2220 007746 042737 000400 177734
2221 007754 032737 017000 177734
2222 007762 001412
2223 007764 013737 177734 001126
2224 007772 013737 177734 001124
2225 010000 042737 017000 001124
2226 010006 104026
2227 010010 052737 000400 177734 34:
2228 010016 013737 177734 001160
2229 010024 122737 000177 001161
2230 010032 001412
2231 010034 013737 177734 001126
2232 010042 013737 177734 001124
2233 010050 152737 000177 001124
2234 010056 104026
2235 010060 032737 000040 172516 44:
2236 010066 001721
2237 010070 042737 000400 177730
2238
2239

```

```

;*****
;*TEST 23      CACHE DISABLED AND KMCR
;*****
TST23: SCOPE

```

```

TST      PHIS      ; ANY PHI MEMORY?
BEQ      TST24      ; IF NONE, EXIT TEST
BIS      #BIT8,DCSR      ; SELECT DIAGNOSTIC MODE
BIC      #BIT05,MMR3      ; FIRST DO WITH RELOCATION DISABLED
BR      24      ; GO DO IT
BIS      #BIT05,MMR3      ; NOW DO WITH RELOCATION ENABLED
BIC      #BIT06,KMCR      ; MAKE SURE THAT CACHE IS DISABLED
BIC      #BIT08,KMCR      ; READ VALID BITS FIRST
BIT      #BIT12!BIT11!BIT10!BIT9,KMCR      ; ALL VALID BITS CLEAR?
BEQ      34      ; IF YES, BRANCH
MOV      KMCR,#BDDAT      ; STORE KMCR
MOV      KMCR,#GDDAT      ; STORE TO PRESERVE LOW BYTE
BIC      #BIT12!BIT11!BIT10!BIT9,#GDDAT      ; HIGH BYTE CLEAR
ERROR      +26      ; VALID BITS NOT CLEAR WITH KMCR<6>=0
BIS      #BIT08,KMCR      ; NOW READ AVAILABILITY BITS
MOV      KMCR,#TMP0      ; STORE KMCR
CMPB      #177,#TMP0+1      ; ALL SET?
BEQ      44      ; IF YES, BRANCH
MOV      KMCR,#BDDAT      ; STORE KMCR
MOV      KMCR,#GDDAT      ; STORE TO PRESERVE LOW BYTE
BISB      #177,#GDDAT      ; HIGH BYTE SET
ERROR      +26      ; AVAIL. BITS NOT SET WITH KMCR<6>=0
BIT      #BIT05,MMR3      ; DONE WITH RELOCATION ENABLED?
BEQ      14      ; IF NOT, BRANCH
BIC      #BIT8,DCSR      ; EXIT DIAGNOSTIC MODE

```

TEST - AVAILABILITY OF SETS

```

2241 .S8TTL TEST - AVAILABILITY OF SETS
2242
2243 ; * THIS TEST WILL CHECK KMCR<15-8>. FIRST, EACH SET IN THE CACHE IS ALLOCATED.
2244 ; * THEN EACH SET IS MADE MOST RECENTLY USED (MRU), AND THEN EACH SET IS
2245 ; * INVALIDATED BY READING THE 8TH REGISTER OF THE SET.
2246 ;
2247 ; B8TST
2248 ;
2249 ;     ENABLE CACHE
2250 ;     POINT R2 TO VALTBL FOR VALID BITS (KMCR<8>=0)
2251 ;     POINT R3 TO TOPTBL FOR AVAILABILITY (KMCR<8>=1)
2252 ;     ENABLE RELOCATION
2253 ; *
2254 ; * ALLOCATE SETS IN CACHE MAKING EACH OF THEM VALID AND GETTING MISSES
2255 ; *
2256 ;     DO FOR ALL 4 SETS ALLOCATING THEM IN CACHE
2257 ;     . LET KMCR<8>=0
2258 ;     . DO DMA READ ON OCTAL BOUNDARY USING DIAGNOSTIC_DATA_IN
2259 ;     . IFB (R2)+ NE IN KMCR+1 THEN
2260 ;     . . ERROR IN READING VALID BITS ON MISS
2261 ;     . ENDF
2262 ;     . LET KMCR<8>=1
2263 ;     . IFB KMCR+1 NE (R3)+ THEN
2264 ;     . . ERROR READING AVAILABILITY BITS ON MISS
2265 ;     . ENDF
2266 ;     ENDDC
2267 ; *
2268 ; * READ A REGISTER FROM A VALID SET MAKING EACH OF THE SETS MOST RECENTLY USED
2269 ; *
2270 ;     DO FOR ALL 4 SETS MAKING THEM MRU AND THEN MOST AVAILABLE
2271 ;     . LET KMCR<8>=0
2272 ;     . DO DMA READ FROM 3RD LOCATION USING DIAGNOSTIC_DATA_IN
2273 ;     . IFB KMCR+1 NE (R2)+ THEN
2274 ;     . . ERROR IN READING VALID BITS ON HIT
2275 ;     . ENDF
2276 ;     . LET KMCR<8>=1
2277 ;     . IFB KMCR+1 NE (R3)+ THEN
2278 ;     . . ERROR READING AVAILABILITY BITS ON HIT
2279 ;     . ENDF
2280 ; *
2281 ; * READ THE LAST REGISTER FROM A SET MAKING EACH SET INVALID AND MOST AVAILABLE
2282 ; *
2283 ;     . LET KMCR<8>=0
2284 ;     . DO DMA READ FROM 8TH LOCATION USING DIAGNOSTIC_DATA_IN
2285 ;     . IFB KMCR+1 NE (R2)+ THEN
2286 ;     . . ERROR IN READING VALID BITS INVALIDATING A SET
2287 ;     . ENDF
2288 ;     . LET KMCR<8>=1
2289 ;     . IFB KMCR+1 NE (R3)+ THEN
2290 ;     . . ERROR READING AVAILABILITY BITS INVALIDATING A SET
2291 ;     . ENDF
2292 ;     ENDDO
2293 ;
2294 ; ENDTST
2295 ;
2296 ; -----
2297

```

T24 AVAILABILITY OF SETS

2298

```

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

```

010076 000004

2299

```

2300 010100 005737 001720      TST      PHIS      ; ANY PHI MEMORY?
2301 010104 001002              BNE      204      ; IF SOME, DO THE TEST
2302 010106 000137 010524      JMP      1004      ; OTHERWISE EXIT
2303 010112 052737 000400 177730 204:  BIS      @BIT8,DCSR  ; SELECT DIAGNOSTIC MODE
2304 010120 052737 000100 177734      BIS      @BIT06,KMCR  ; ENABLE CACHE
2305 010126 012702 010526      MOV      @VALTBL,R2  ; POINT TO VALID BITS
2306 010132 012703 010542      MOV      @TOPTBL,R3  ; POINT TO AVAILABILITY BITS
2307 010136 052737 000040 172516      BIS      @BIT05,MHR3  ; ENABLE RELOCATION
2308 010144 012737 000200 170200      MOV      @20C,MAPL00 ; POINT MAP00 TO FIRST 8KB
2309 010152 012737 000000 170202      MOV      @0,MAPH00  ; IN HIGH AND LOW MAP REGISTERS
2310 010160 013737 177734 001124      MOV      KMCR,@GDDAT ; PRESERVE LOW BYTE
2311 010166 105037 001125      CLR      @GDDAT+1  ; CLEAR TO REPORT ERRORS, IF ANY
2312
2313      ; ALLOCATE 4 SETS BY READING THEM THRU DIAGNOSTIC_DATA_IN
2314
2315 010172 012701 000000      MOV      @0,R1      ; START READING MEMORY FROM 0
2316 010176 004737 002222 14:  JSR      PC,DOIN  ; . ALLOCATE IN CACHE NEXT 8 WORDS
2317 010202 042737 000400 177734      BIC      @BIT08,KMCR  ; . READ VALID BITS
2318 010210 013737 177734 001126      MOV      KMCR,@BDDAT ; . STORE KMCR
2319 010216 122237 001127      CMP      (R2)+,@BDDAT+1 ; . VALID BITS OK?
2320 010222 001404      BEQ      24      ; . IF YES, BRANCH
2321 010224 105742      TST      -(R2)      ; . GET THE PATTERN JUST WRITTEN
2322 010226 112237 001125      MOV      (R2)+,@GDDAT+1 ; . STORE HIGH BYTE
2323 010232 104026      ERROR      +26      ; . ERROR IN VALID BITS
2324 010234 052737 000400 177734 24:  BIS      @BIT08,KMCR  ; . READ AVAILABILITY BIT
2325 010242 013737 177734 001126      MOV      KMCR,@BDDAT ; . STORE KMCR
2326 010250 122337 001127      CMP      (R3)+,@BDDAT+1 ; . AVAILABILITY BITS OK?
2327 010254 001404      BEQ      34      ; . IF YES, BRANCH
2328 010256 105743      TST      -(R3)      ; . GET THE PATTERN JUST WRITTEN
2329 010260 112337 001125      MOV      (R3)+,@GDDAT+1 ; . STORE HIGH BYTE
2330 010264 104026      ERROR      +26      ; . ERROR IN AVAILABILITY BITS
2331 010266 062701 000020 34:  ADD      @20,R1      ; . DO FOR NEXT OCTAL BOUNDARY
2332 010272 022701 000100      CMP      @100,R1    ; . ALL 4 DONE?
2333 010276 003337      BGT      14      ; . IF NOT, BRANCH
2334
2335      ; NOW READ 3RD AND THEN 8TH LOCATION FROM CACHE
2336
2337 010300 012701 000004      MOV      @4,R1      ; START WITH 3RD LOCATION A SET
2338 010304 042737 000400 177734 44:  BIC      @BIT08,KMCR  ; . READ VALID BITS
2339 010312 004737 002222      JSR      PC,DOIN  ; . READ FROM CACHE
2340 010316 013737 177734 001126      MOV      KMCR,@BDDAT ; . STORE KMCR
2341 010324 122237 001127      CMP      (R2)+,@BDDAT+1 ; . VALID BITS OK?
2342 010330 001404      BEQ      54      ; . IF YES, BRANCH
2343 010332 105742      TST      -(R2)      ; . GET THE PATTERN JUST WRITTEN
2344 010334 112237 001125      MOV      (R2)+,@GDDAT+1 ; . STORE HIGH BYTE
2345 010340 104026      ERROR      +26      ; . ERROR IN VALID BITS
2346 010342 052737 000400 177734 54:  BIS      @BIT08,KMCR  ; . READ AVAILABILITY BIT
2347 010350 013737 177734 001126      MOV      KMCR,@BDDAT ; . STORE KMCR
2348 010356 122337 001127      CMP      (R3)+,@BDDAT+1 ; . AVAILABILITY BITS OK?
2349 010362 001404      BEQ      64      ; . IF YES, BRANCH
2350 010364 105743      TST      -(R3)      ; . GET THE PATTERN JUST WRITTEN
2351 010366 112337 001125      MOV      (R3)+,@GDDAT+1 ; . STORE HIGH BYTE

```

T24 AVAILABILITY OF SETS

```

2352 010372 104026          ERROR +26          ; . ERROR IN AVAILABILITY BITS
2353 010374 062701 000012 64:  ADD #12,R1      ; . READ THE 8TH WORD
2354 010400 052737 000400 177734  BIS #BIT08,KMCR ; . READ AVAILABILITY BIT
2355 010406 004737 002222      JSR PC,DDIN    ; . READ FROM CACHE
2356 010412 013737 177734 001126  MOV KMCR,#BDDAT ; . STORE KMCR
2357 010420 122337 001127      CMPB (R3)+,#BDDAT+1 ; . AVAILABILITY BITS OK?
2358 010424 001404          BEQ 7#           ; . IF YES, BRANCH
2359 010426 105743          TSTB -(R3)        ; . GET THE PATTERN JUST WRITTEN
2360 010430 112337 001125      MOVB (R3)+,#GDDAT+1 ; . STORE HIGH BYTE
2361 010434 104026          ERROR +26          ; . ERROR IN AVAILABILITY BITS
2362 010436 042737 000400 177734 74:  BIC #BIT08,KMCR ; . READ VALID BITS
2363 010444 013737 177734 001126  MOV KMCR,#BDDAT ; . STORE KMCR
2364 010452 122237 001127      CMPB (R2)+,#BDDAT+1 ; . VALID BITS OK?
2365 010456 001404          BEQ 8#           ; . IF YES, BRANCH
2366 010460 105742          TSTB -(R2)        ; . GET THE PATTERN JUST WRITTEN
2367 010462 112237 001125      MOVB (R2)+,#GDDAT+1 ; . STORE HIGH BYTE
2368 010466 104026          ERROR +26          ; . ERROR IN VALID BITS
2369 010470 062701 000006 84:  ADD #6,R1      ; . DO FOR NEXT OCTAL BOUNDARY
2370 010474 022701 000100      CMP #100,R1    ; . ALL 4 DONE?
2371 010500 003301          BGT 4#           ; . NOT, BRANCH
2372 010502 042737 000040 172516  BIC #BIT05,MMR3 ; . DISABLE RELOCATION
2373 010510 042737 000100 177734  BIC #BIT06,KMCR ; . DISABLE CACHE
2374 010516 042737 000400 177730  BIC #BIT8,DCSR  ; . EXIT DIAGNOSTIC MODE
2375 010524          1004:  BR TST25          ; . EXIT TEST
      010524 000414
2376
2377          ;* THIS TABLE HAS BITS KMCR<15-8> WHEN KMCR<8>=0
2378          ;* KMCR<15>=0 FLAGGING CACHE MISS
2379
2380 010526 020 030 034 VALTBL: .BYTE 20,30,34,36 ; ORDER FOR VALID: A, AB, ABC, ABCD
      010531 036
2381
2382 010532 236 016 .BYTE 236,16 ; NOW ALL VALID, HITS-MISS(WRITE TO I/O)
2383 010534 216 006 .BYTE 216,6  ; HIT-MISS, A: VALID - NOT VALID
2384 010536 206 002 .BYTE 206,2  ; HIT-MISS, B: VALID - NOT VALID
2385 010540 202 000 .BYTE 202,0  ; HIT-MISS, C: VALID - NOT VALID
2386
2387          ;* THIS TABLE HAS BITS KMCR<15-8> WHEN KMCR<8>=1
2388          ;* AT FIRST NO HITS ARE RECORDED, THE COMMENT FIELD SHOWS SETS
2389          ;* STARTING WITH LEAST AVAILABLE
2390
2391 010542 017 TOPTBL: .BYTE 17 ; ADCB - 000111
2392 010543 103 .BYTE 103 ; BADC - 100001
2393 010544 151 .BYTE 151 ; CBAD - 110100
2394 010545 177 .BYTE 177 ; DCBA - 111111
2395          ; NOW MISS-HITS
2396 010546 017 377 .BYTE 17,377 ; ->ADCB->DCBA - 000111.111111
2397 010550 163 277 .BYTE 163,277 ; ->BDCA->DCAB - 111001.011111
2398 010552 075 227 .BYTE 75,227 ; ->CDAB->DABC - 011110.001011
2399 010554 027 201 .BYTE 27,201 ; ->DABC->ABCD - 001011.000000
2400
2401          .EVEN

```

TEST - DEALLOCATION OF SETS

```

2403 .SBTTL TEST - DEALLOCATION OF SETS
2404
2405 ; * THIS TEST CHECKS THAT EACH SET CAN BE MADE MOST AVAILABLE AND
2406 ; * DEALLOCATED ON THE NEXT READ ON THE OCTAL BOUNDARY.
2407 ;
2408 ;     DISABLE CACHE TO MAKE A MOST AVAILABLE
2409 ;     ENABLE CACHE
2410 ; *
2411 ; * ALLOCATE SETS IN CACHE MAKING EACH OF THEM VALID AND GETTING MISSES
2412 ; *
2413 ;     DO FOR ALL 4 SETS ALLOCATING THEM IN CACHE
2414 ;     . DO DMA READ ON OCTAL BOUNDARY USING DIAGNOSTIC_DATA_IN
2415 ;     ENDDO - THIS LEAVES A AS THE MOST AVAILABLE SET
2416 ; *
2417 ; * TRY TO BRING A NEW LOCATIONS TO SET A
2418 ; *
2419 ;     DO DMA READ ON OCTAL BOUNDARY FROM A FIFTH LOCATION TO SET A AGAIN
2420 ;     LET KMCR<8> = #0
2421 ;     IF KMCR<15!12!11!10!9!> NE <0,1,1,1,1> THEN
2422 ;     . ERROR DEALLOCATIONING LRU SET (A)
2423 ;     ENDIF
2424 ;     LET KMCR<8> = #1
2425 ;     IF KMCR<15-9> NE <0,0,0,0,1,1,1> THEN
2426 ;     . ERROR DEALLOCATING LRU SET (A)
2427 ;     ENDIF
2428 ; *
2429 ; * TRY TO BRING A NEW LOCATIONS TO SET B
2430 ; *
2431 ;     DO DMA READ ON OCTAL BOUNDARY FROM 1ST LOCATION TO SET B
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 (B)
2435 ;     ENDIF
2436 ;     LET KMCR<8> = #1
2437 ;     IF KMCR<15-9> NE <0,1,0,0,0,0,1> THEN
2438 ;     . ERROR DEALLOCATING LRU SET (B)
2439 ;     ENDIF
2440 ; *
2441 ; * TRY TO BRING A NEW LOCATIONS TO SET C
2442 ; *
2443 ;     DO DMA READ ON OCTAL BOUNDARY FROM 1ST LOCATION TO SET C
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 (C)
2447 ;     ENDIF
2448 ;     LET KMCR<8> = #1
2449 ;     IF KMCR<15-9> NE <0,1,1,0,1,0,0> THEN
2450 ;     . ERROR DEALLOCATING LRU SET (C)
2451 ;     ENDIF
2452 ; *
2453 ; * TRY TO BRING A NEW LOCATIONS TO SET D
2454 ; *
2455 ;     DO DMA READ ON OCTAL BOUNDARY FROM 1ST LOCATION TO SET D
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 (D)
2459 ;     ENDIF

```

TEST - DEALLOCATION OF SETS

```

2460      ;      LET KMCR<8> = #1
2461      ;      IF KMCR<15-9> NE <0,1,1,1,1,1,1> THEN
2462      ;      .      ERROR DEALLOCATING LRU SET (D)
2463      ;      ENDIF
2464      ;
2465      ;      ENDTST
2466      ;
2467      ;-----
2468
2469      ;*****
;*TEST 25      DEALLOCATION OF SETS
;*****
TST25:  SCOPE

2470      010556  000004
2471      010560  005737  001720      TST      PHIS      ; ANY PHI MEMORY?
2472      010564  001002      BNE      204      ; IF SOME, DO THE TEST
2473      010566  000137  011266      JMP      1004      ; OTHERWISE, EXIT
2474      010572  052737  000400  177730  204:  BIS      #BIT8,DCSR      ; SELECT DIAGNOSTIC MODE
2475      010600  052737  000040  172516      BIS      #BIT05,MMR3      ; ENABLE RELOCATION
2476      010606  052737  000100  177734      BIS      #BIT06,KMCR      ; ENABLE CACHE
2477      010614  012737  000200  170200      MOV      #200,MAPL00      ; POINT MAP00 TO FIRST 8KB
2478      010622  012737  000000  170202      MOV      #0,MAPH00      ; IN HIGH AND LOW MAP REGISTERS
2479      010630  013737  177734  001124      MOV      KMCR,#GDDAT      ; STORE LOW BYTE OF KMCR
2480
2481      ;      ALLOCATE ALL SETS MAKING A MOST AVAILABLE
2482      ;
2483      010636  012701  000000      MOV      #0,R1      ; START WITH 0 IN SET A
2484      010642  004737  002222  14:  JSR      PC,DOIN      ; . ALLOCATE A SET
2485      010646  062701  000020      ADD      #20,R1      ; . GET READY FOR NEXT SET
2486      010652  022701  000100      CMP      #100,R1      ; . ALL 4 DONE?
2487      010656  003371      BGT      14      ; . IF NOT, BRANCH
2488
2489      ;      TRY TO BRING A NEW SET TO A
2490      ;
2491      010660  012701  000100      MOV      #100,R1      ; TRY TO DO DATI
2492      010664  042737  000400  177734      BIC      #BIT08,KMCR      ; READ VALID BITS
2493      010672  004737  002222      JSR      PC,DOIN      ; ALLOCATE IN CACHE
2494      010676  013737  177734  001126      MOV      KMCR,#BDDAT      ; STORE KMCR
2495      010704  122737  000036  001127      CMPB     #36,#BDDAT+1      ; ALL SETS VALID?
2496      010712  001404      BEQ      24      ; IF OK, BRANCH
2497      010714  112737  000036  001125      MOVB     #36,#GDDAT+1      ; STORE EXPECTED PATTERN
2498      010722  104026      ERROR     +26      ; DEALLOCATING A
2499      010724  052737  000400  177734  24:  BIS      #BIT08,KMCR      ; READ AVAILABILITY BITS
2500      010732  013737  177734  001126      MOV      KMCR,#BDDAT      ; STORE KMCR
2501      010740  122737  000017  001127      CMPB     #17,#BDDAT+1      ; MRU:A,D,C,B
2502      010746  001404      BEQ      34      ; IF OK, BRANCH
2503      010750  112737  000017  001125      MOVB     #17,#GDDAT+1      ; STORE EXPECTED PATTERN
2504      010756  104026      ERROR     +26      ; DEALLOCATING A
2505
2506      ;      TRY TO BRING A NEW SET TO B
2507      ;
2508      010760  012701  000120  34:  MOV      #120,R1      ; PREPARE FOR NEXT READ
2509      010764  042737  000400  177734      BIC      #BIT08,KMCR      ; READ VALID BITS
2510      010772  004737  002222      JSR      PC,DOIN      ; READ INTO CACHE
2511      010776  013737  177734  001126      MOV      KMCR,#BDDAT      ; STORE KMCR
2512      011004  122737  000036  001127      CMPB     #36,#BDDAT+1      ; ALL VALID?
2513      011012  001404      BEQ      44      ; IF YES, BRANCH

```

T25 DEALLOCATION OF SETS

```

2514 011014 112737 000036 001125      MOV      #36, #GDDAT+1      ; STORE EXPECTED PATTERN
2515 011022 104026                      ERROR      +26          ; DEALLOCATING B
2516 011024 052737 000400 177734 44:    BIS      #BIT08, KMCR      ; READ AVAILABILITY BITS
2517 011032 013737 177734 001126          MOV      KMCR, #BDDAT      ; STORE KMCR
2518 011040 122737 000103 001127          CMPB     #103, #BDDAT+1    ; MRU: B, A, D, C
2519 011046 001404                      BEQ       54            ; IF OK, BRANCH
2520 011050 112737 000103 001125          MOV      #103, #GDDAT+1    ; STORE EXPECTED PATTERN
2521 011056 104026                      ERROR      +26          ; DEALLOCATING B
2522                                     ;
2523                                     ; TRY TO BRING A NEW SET TO C
2524                                     ;
2525 011060 012701 000140 177734 54:    MOV      #140, R1          ; PREPARE FOR NEXT READ
2526 011064 042737 000400 177734          BIC      #BIT08, KMCR      ; READ VALID BITS
2527 011072 004737 002222                      JSR      PC, DDIN          ; READ INTO CACHE
2528 011076 013737 177734 001126          MOV      KMCR, #BDDAT      ; STORE KMCR
2529 011104 122737 000036 001127          CMPB     #36, #BDDAT+1    ; ALL VALID?
2530 011112 001404                      BEQ       64            ; IF YES, BRANCH
2531 011114 112737 000036 001125          MOV      #36, #GDDAT+1    ; STORE EXPECTED PATTERN
2532 011122 104026                      ERROR      +26          ; DEALLOCATING C
2533 011124 052737 000400 177734 64:    BIS      #BIT08, KMCR      ; READ AVAILABILITY BITS
2534 011132 013737 177734 001126          MOV      KMCR, #BDDAT      ; STORE KMCR
2535 011140 122737 000151 001127          CMPB     #151, #BDDAT+1   ; MRU: C, B, A, D
2536 011146 001404                      BEQ       74            ; IF OK, BRANCH
2537 011150 112737 000151 001125          MOV      #151, #GDDAT+1  ; STORE EXPECTED PATTERN
2538 011156 104026                      ERROR      +26          ; DEALLOCATING C
2539                                     ;
2540                                     ; TRY TO BRING A NEW SET TO D
2541                                     ;
2542 011160 012701 000160 177734 74:    MOV      #160, R1          ; PREPARE FOR NEXT READ
2543 011164 042737 000400 177734          BIC      #BIT08, KMCR      ; READ VALID BITS
2544 011172 004737 002222                      JSR      PC, DDIN          ; READ INTO CACHE
2545 011176 013737 177734 001126          MOV      KMCR, #BDDAT      ; STORE KMCR
2546 011204 122737 000036 001127          CMPB     #36, #BDDAT+1    ; ALL VALID?
2547 011212 001404                      BEQ       84            ; IF YES, BRANCH
2548 011214 112737 000036 001125          MOV      #36, #GDDAT+1    ; STORE EXPECTED PATTERN
2549 011222 104026                      ERROR      +26          ; DEALLOCATING D
2550 011224 052737 000400 177734 84:    BIS      #BIT08, KMCR      ; READ AVAILABILITY BITS
2551 011232 013737 177734 001126          MOV      KMCR, #BDDAT      ; STORE KMCR
2552 011240 042737 000400 177730          BIC      #BIT08, DCSR      ; EXIT DIAGNOSTIC MODE
2553 011246 122737 000177 001127          CMPB     #177, #BDDAT+1   ; MRU: D, C, B, A
2554 011254 001404                      BEQ       TST26          ; IF OK, EXIT TEST
2555 011256 112737 000177 001125          MOV      #177, #GDDAT+1  ; STORE EXPECTED PATTERN
2556 011264 104026                      ERROR      +26          ; DEALLOCATING D
2557 011266                                     ;
2558                                     ;

```


TEST - WRITE CYCLES AND CACHE

2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633

.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

011436 000004

2634	011440	005737	001720	TST	PHIS	; ANY PMI MEMORY?
2635	011444	001002		BNE	201	; IF SOME, DO THE TEST
2636	011446	000137	012120	JMP	1001	; OTHERWISE EXIT
2637	011452	052737	000400 177730	201: BIS	#BIT08,DCSR	; SELECT DIAGNOSTIC MODE
2638	011460	052737	000100 177734	BIS	#BIT06,KMCR	; CACHE STILL ENABLED
2639	011466	052737	000040 172516	BIS	#BIT05,MMR3	; ENABLE RELOCATION
2640	011474	012737	000200 170200	MOV	#200,MAPLOO	; POINT MAP00 TO FIRST 8KB
2641	011502	012737	000000 170202	MOV	#0,MAPH00	; IN HIGH AND LOW MAP REGISTERS
2642	011510	042737	000400 177734	BIC	#BIT08,KMCR	; SELECT VALID BITS
2643	011516	013702	177734	MOV	KMCR,R2	; SAVE VALID BITS OF KMCR
2644	011522	052737	000400 177734	BIS	#BIT08,KMCR	; SELECT AVAILABILITY BITS
2645	011530	013703	177734	MOV	KMCR,R3	; SAVE AVAILABILITY BITS
2646						
2647						
2648						
2649				; ACCESS LOCATION NOT IN CACHE		
2650	011534	012701	000200	MOV	#200,R1	; ACCESS ADDRESS
2651	011540	013737	000400 001124	MOV	#400,\$GDDAT	; THE PATTERN TO BE WRITTEN
2652	011546	004737	002210	JSR	PC,DOOUT	; DO DIAGNOSTIC DATA OUT
2653	011552	042737	000400 177734	BIC	#BIT08,KMCR	; SELECT VALID BITS
2654	011560	020237	177734	CMP	R2,KMCR	; KMCR CHANGED?
2655	011564	001406		BEQ	11	; IF NOT, BRANCH
2656	011566	010237	001124	MOV	R2,\$GDDAT	; EXPECTED PATTERN
2657	011572	013737	177734 001126	MOV	KMCR,\$BDDAT	; RECIEVED PATTERN
2658	011600	104026		ERROR	+26	; WRITE MISSES AFFECTS CACHE
2659	011602	052737	000400 177734	11: BIS	#BIT08,KMCR	; SELECT AVAILABILITY BITS
2660	011610	020337	177734	CMP	R3,KMCR	; KMCR CHANGED?
2661	011614	001406		BEQ	21	; IF OK, BRANCH

PC	OP	OP2	OP3	OP4	OP5	OP6	OP7	OP8	OP9	OP10	OP11	OP12	OP13	OP14	OP15	OP16	OP17	OP18	OP19	OP20	OP21	OP22	OP23	OP24	OP25	OP26	OP27	OP28	OP29	OP30	OP31	OP32	OP33	OP34	OP35	OP36	OP37	OP38	OP39	OP40	OP41	OP42	OP43	OP44	OP45	OP46	OP47	OP48	OP49	OP50	OP51	OP52	OP53	OP54	OP55	OP56	OP57	OP58	OP59	OP60	OP61	OP62	OP63	OP64	OP65	OP66	OP67	OP68	OP69	OP70	OP71	OP72	OP73	OP74	OP75	OP76	OP77	OP78	OP79	OP80	OP81	OP82	OP83	OP84	OP85	OP86	OP87	OP88	OP89	OP90	OP91	OP92	OP93	OP94	OP95	OP96	OP97	OP98	OP99	OP100	OP101	OP102	OP103	OP104	OP105	OP106	OP107	OP108	OP109	OP110	OP111	OP112	OP113	OP114	OP115	OP116	OP117	OP118	OP119	OP120	OP121	OP122	OP123	OP124	OP125	OP126	OP127	OP128	OP129	OP130	OP131	OP132	OP133	OP134	OP135	OP136	OP137	OP138	OP139	OP140	OP141	OP142	OP143	OP144	OP145	OP146	OP147	OP148	OP149	OP150	OP151	OP152	OP153	OP154	OP155	OP156	OP157	OP158	OP159	OP160	OP161	OP162	OP163	OP164	OP165	OP166	OP167	OP168	OP169	OP170	OP171	OP172	OP173	OP174	OP175	OP176	OP177	OP178	OP179	OP180	OP181	OP182	OP183	OP184	OP185	OP186	OP187	OP188	OP189	OP190	OP191	OP192	OP193	OP194	OP195	OP196	OP197	OP198	OP199	OP200	OP201	OP202	OP203	OP204	OP205	OP206	OP207	OP208	OP209	OP210	OP211	OP212	OP213	OP214	OP215	OP216	OP217	OP218	OP219	OP220	OP221	OP222	OP223	OP224	OP225	OP226	OP227	OP228	OP229	OP230	OP231	OP232	OP233	OP234	OP235	OP236	OP237	OP238	OP239	OP240	OP241	OP242	OP243	OP244	OP245	OP246	OP247	OP248	OP249	OP250	OP251	OP252	OP253	OP254	OP255	OP256	OP257	OP258	OP259	OP260	OP261	OP262	OP263	OP264	OP265	OP266	OP267	OP268	OP269	OP270	OP271	OP272	OP273	OP274	OP275	OP276	OP277	OP278	OP279	OP280	OP281	OP282	OP283	OP284	OP285	OP286	OP287	OP288	OP289	OP290	OP291	OP292	OP293	OP294	OP295	OP296	OP297	OP298	OP299	OP300	OP301	OP302	OP303	OP304	OP305	OP306	OP307	OP308	OP309	OP310	OP311	OP312	OP313	OP314	OP315	OP316	OP317	OP318	OP319	OP320	OP321	OP322	OP323	OP324	OP325	OP326	OP327	OP328	OP329	OP330	OP331	OP332	OP333	OP334	OP335	OP336	OP337	OP338	OP339	OP340	OP341	OP342	OP343	OP344	OP345	OP346	OP347	OP348	OP349	OP350	OP351	OP352	OP353	OP354	OP355	OP356	OP357	OP358	OP359	OP360	OP361	OP362	OP363	OP364	OP365	OP366	OP367	OP368	OP369	OP370	OP371	OP372	OP373	OP374	OP375	OP376	OP377	OP378	OP379	OP380	OP381	OP382	OP383	OP384	OP385	OP386	OP387	OP388	OP389	OP390	OP391	OP392	OP393	OP394	OP395	OP396	OP397	OP398	OP399	OP400	OP401	OP402	OP403	OP404	OP405	OP406	OP407	OP408	OP409	OP410	OP411	OP412	OP413	OP414	OP415	OP416	OP417	OP418	OP419
----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

TEST - DMA READ WITH INDEX NOT ZERO

```

2708 .SBTTL TEST - DMA READ WITH INDEX NOT ZERO
2709
2710 ; * THIS TEST CHECKS THAT A DMA READ MISS DOES NOT AFFECT THE CACHE IF THE INDEX
2711 ; * FIELD IS NOT ZERO.
2712 ;
2713 ; BGNTEST
2714 ;
2715 ;     SAVE KMCR<15-9> FOR KMCR<8>=0,1
2716 ;     DO DIAGNOSTIC_DATA_IN FOR A LOCATION WITH NON-ZERO INDEX
2717 ;     DO FOR KMCR<8>=0,1
2718 ;     . IF KMCR<15-9> NOT SAVED THEN
2719 ;     .     ERROR IN COMPARING INDEXES
2720 ;     . ENDOF
2721 ;     ENDDO
2722 ;
2723 ; ENDTST
2724 ;
2725 ;-----
2726 ;*****
2727 ;*TEST 30      DMA READ WITH INDEX NOT ZERO
;*****
TST30: SCOPE

012120 000004
2728
2729 012122 005737 001720      TST      PHIS      ; ANY PHI MEMORY?
2730 012126 001502      BEQ      TST31      ; IF NONE, EXIT TEST
2731
2732 ;
2733 ; ENABLE CACHE AND ALLOCATE SET A
2734 ;
2735 012130 052737 000400 177730      BIS      @BIT08,DCSR      ; SELECT DIAGNOSTIC MODE
2736 012136 052737 000040 172516      BIS      @BIT05,MMR3      ; ENABLE RELOCATION
2737 012144 052737 000100 177734      BIS      @BIT06,KMCR      ; ENABLE CACHE
2738 012152 012737 000200 170200      MOV      @200,MAPLO0      ; POINT MAP00 TO FIRST 8KB
2739 012160 012737 000000 170202      MOV      @0,MAPH00      ; IN HIGH AND LOW MAP REGISTERS
2740 012166 012701 000000      MOV      @0,R1      ; ALLOCATE 200-216
2741 012172 004737 002222      JSR      PC,DOIN      ; IN CACHE
2742
2743 ;
2744 ; TRY TO CHECK WHETHER NON-ZERO INDEX EFFECTS CACHE
2745 ;
2745 012176 042737 000400 177734      BIC      @BIT08,KMCR      ; SELECT VALID BITS
2746 012204 013737 177734 001124      MOV      KMCR,@GDDAT      ; SAVE VALID BITS OF KMCR
2747 012212 052737 000400 177734      BIS      @BIT08,KMCR      ; SELECT AVAILABILITY BITS
2748 012220 013703 177734      MOV      KMCR,R3      ; SAVE AVAILABILITY BITS
2749 012224 012701 000102      MOV      @102,R1      ; TRY CACHING 102
2750 012230 004737 002222      JSR      PC,DOIN      ; DO DIAGNOSTIC DATA IN
2751 012234 042737 000400 177734      BIC      @BIT08,KMCR      ; SELECT VALID BITS
2752 012242 023737 001124 177734      CMP      @GDDAT,KMCR      ; KMCR CHANGED?
2753 012250 001404      BEQ      11      ; IF NOT, BRANCH
2754 012252 013737 177734 001126      MOV      KMCR,@BDDAT      ; RECEIVED PATTERN
2755 012260 104026      ERROR      *26      ; NON-ZERO INDEX AFFECTS CACHE
2756 012262 052737 000400 177734 11: BIS      @BIT08,KMCR      ; SELECT AVAILABILITY BITS
2757 012270 020337 177734      CMP      R3,KMCR      ; KMCR CHANGED?
2758 012274 001406      BEQ      21      ; IF NOT, EXIT TEST
2759 012276 010337 001124      MOV      R3,@GDDAT      ; EXPECTED DATA
2760 012302 013737 177734 001126      MOV      KMCR,@BDDAT      ; RECEIVED DATA
2761 012310 104026      ERROR      *26      ; NON-ZERO INDEX AFFECTS CACHE

```

DE

SEQ 0068

T30 DMA READ WITH INDEX NOT ZERO

```

2762 012312 042737 000100 177734 24: BIC #BIT06,KMCR ; DISABLE CACH
2763 012320 042737 000040 172516 BIC #BIT05,MHR3 ; AND MAPPING
2764 012326 042737 000400 177730 BIC #BIT08,DCSR ; EXIT DIAGNOSTIC MODE
2765

```

TEST - TAG REGISTERS

```

2767 .SBTTL TEST - TAG REGISTERS
2768
2769 ;* THIS TEST WILL AUTOSIZE MEMORY IN 128K WORDS. THEN DEPENDING
2770 ;* ON THE AMOUNT OF MEMORY AVAILABLE, A PATTERN OF ALTERNATING 0'S
2771 ;* AND 1'S WILL BE CONSTRUCTED TO VERIFY THE TAG REGISTERS.
2772
2773 ;
2774 ; BGNST
2775 ;
2776 ;     SAVE TIMEOUT VECTOR
2777 ;     ENABLE MMU
2778 ;     LET R1 = KIPAR6
2779 ;     REPEAT TO CONSTRUCT A MASK FOR NXM FOR MAPH01
2780 ;     . LET R1 = R1 SHIFT.LEFT BY #1
2781 ;     . LET R1 = R1 + KIPAR6
2782 ;     UNTIL CARRY SET
2783 ;     SWAP R1
2784 ;     LET R1 = R1 SHIFT.RIGHT BY #3 TO GET BITS<21-16> FROM <21-13>
2785 ;     LET KMCR<6> = #1 TO ENABLE CACHE
2786 ;     LET R2 = #TBLML FOR PATTERNS FOR LOW REGISTER
2787 ;     LET R3 = #TBLMH FOR PATTERNS FOR HIGH REGISTER
2788 ;     DO FOR ALL 16 PATTERNS
2789 ;     . DO FOR 4 PATTERNS
2790 ;     . . LET MAPL01 = (R2).
2791 ;     . . LET MAPH01 = (R3).
2792 ;     . . LET MAPH01 = MAPH01 CLEAR.BY R1 NOT TO GET NXM
2793 ;     . . DO DIAGNOSTIC_DATA_IN TO ALLOCATE TAG PATTERN
2794 ;     . ENDDO
2795 ;     MOVE POINTERS THRU PATTERN TABLE TO PREVIOUS 4 WORDS
2796 ;     ENABLE 22-BIT MMU
2797 ;     DO FOR 4 PATTERNS
2798 ;     . LET MAPL01 = (R2).
2799 ;     . LET MAPH01 = (R3).
2800 ;     . LET MAPH01 = MAPH01 CLEAR.BY R1 NOT TO GET NXM
2801 ;     . . DO DIAGNOSTIC_DATA_IN FROM IN CACHE
2802 ;     . . IF NOT A HIT THEN
2803 ;     . . . ERROR IN TAG REGISTERS
2804 ;     . ENF
2805 ;     DISABLE 22-BIT MMU
2806 ;     ENDDO
2807 ;
2808 ; ENDTST
2809
2810 -----
2811 ;*****
;*TEST 31 TAG REGISTERS
;*****
TST31: SCOPE
012334 000004
2812
2813 012336 005737 001720 TST PHIS ; ANY PHI MEMORY?
2814 012342 001002 BNE 18 ; YES
2815 012344 000137 012654 JMP 1004 ; IF NONE, EXIT TEST
2816 012350 032737 000040 177734 18: BIT #BIT05,KMCR ; 18-BIT MODE?
2817 012356 001177 BNE TST32 ; IF YES, EXIT TEST
2818 012360 052737 000400 177730 BIS #BIT08,DCSR ; SELECT DIAGNOSTIC MODE
2819 012366 004737 002244 JSR PC,MAPPR ; REMAP PROGRAM AREA
2820 012372 005237 177572 INC MMRO ; ENABLE MMU

```

T31 TAG REGISTERS

```

2821 012376 052737 000020 172516      BIS      #BIT04,MMR3      ; ENABLE 22-BIT
2822 012404 022737 000040 001720      CMP      #40,PMIS      ; IF 2M PRESENT, ONLY 21ST MASKED
2823 012412 001412                      BEQ      $#              ; DON'T CONSTRUCT MASK
2824                                     ;
2825                                     ; IF LESS THEN 2M OF MEMORY, MAKE UP A MASK FOR MAP HIGH REGISTERS
2826                                     ;
2827 012414 000241                      CLC                      ; CLEAR CARRY
2828 012416 013701 001720      MOV      PMIS,R1      ; SAVE KIPAR6
2829 012422 006101      44:      ROL      R1          ; . ROTATE LEFT R1
2830 012424 053701 001720      BIS      PMIS,R1      ; . AND ADD BIT AT FIRST POSITION
2831 012430 103374                      BHS      $#          ; . IF CARRY NOT SET, BRANCH
2832 012432 000301                      SWAB     R1          ; <15-8><----><7-0>
2833 012434 006001                      ROR      R1          ; NOW GET ADDRESS BITS <21-16>
2834 012436 006001                      ROR      R1          ; FROM <21-14>
2835                                     ;
2836                                     ; ALLOCATE DIFFERENT PATTERNS IN TAG REGISTERS
2837                                     ;
2838 012440 042737 000100 177734      54:      BIC      #BIT06,KMCR      ; DISABLE CACHE
2839 012446 052737 000100 177734      BIS      #BIT06,KMCR      ; ENABLE CACHE
2840 012454 042737 000400 177734      BIC      #BIT08,KMCR      ; MAKE SURE THAT VALID READ
2841 012462 052737 000040 172516      BIS      #BIT05,MMR3      ; ENABLE MAPPING
2842 012470 012702 012656      MOV      #TBLML,R2      ; POINTER FOR LOW MAP PATTERNS
2843 012474 012703 012716      MOV      #TBLMH,R3      ; POINTER FOR HIGH MAP PATTERNS
2844 012500 010137 001160      MOV      R1,#TMP0      ; SAVE THE MASK
2845 012504 012704 000004      MOV      #4,R4          ; COUNTER FOR 16 PATTERNS
2846 012510 012701 020000      64:      MOV      #20000,R1      ; . TO ACCESS THRU MAP REG. 1
2847 012514 012705 000004      MOV      #4,R5          ; . DO EACH FOR AT A TIME
2848 012520 012237 170204      74:      MOV      (R2)+,MAPL01      ; . . LOAD LOW MAP
2849 012524 012337 170206      MOV      (R3)+,MAPH01      ; . . LOAD HIGH MAP
2850 012530 043737 001160 170206      BIC      #TMP0,MAPH01      ; . . MASK OUT NXM
2851 012536 004737 002222      JSR      PC,DOIN      ; . . DO DIAGNOSTIC DATA IN
2852 012542 077512                      SOB      R5,74      ; . . DO FOR ALL 4 PATTERNS
2853 012544 013701 177734      MOV      KMCR,R1      ; . STORE SETS VALID?
2854 012550 005101                      COM      R1          ; . COMPLEMENT VALID BITS
2855 012552 032701 017000      BIT      #17000,R1      ; . THAT'S ALL VALID BITS
2856 012556 001401                      BEQ      $#          ; . IF ALL SET BEFORE COM, BRANCH
2857 012560 104027                      ERROR     +27      ; . NOT ALL VALID BITS SET
2858                                     ;
2859                                     ; NOW INVALIDATE BY WRITING TO THE SAME LOCATIONS
2860                                     ;
2861 012562 162702 000010      84:      SUB      #10,R2      ; . MOVE POINTER TO -4
2862 012566 162703 000010      SUB      #10,R3      ; . MOVE HIGH MAP POINTER TOC
2863 012572 012705 000004      MOV      #4,R5          ; . VALIDATE ALLOCATED PATTERNS
2864 012576 012701 020000      MOV      #20000,R1      ; . POINTER TO MAP 1
2865 012602 012237 170204      94:      MOV      (R2)+,MAPL01      ; . . LOAD LOW MAP
2866 012606 012337 170206      MOV      (R3)+,MAPH01      ; . . LOAD HIGH MAP
2867 012612 043737 001160 170206      BIC      #TMP0,MAPH01      ; . . MASK OUT NXM
2868 012620 004737 002222      JSR      PC,DOIN      ; . . DO DAIGN. DATA IN
2869 012624 032737 100000 177734      BIT      #BIT15,KMCR      ; . . HIT?
2870 012632 071001                      BNE      104      ; . . IF HIT, BRANCH
2871 012634 104027                      ERROR     +27      ; . . IN TAG REGISTERS
2872 012636 077517      104:      SOB      R5,94      ; . . DO FOR ALL 4 PATTERNS
2873 012640 077455                      SOB      R4,64      ; . REPEAT FOR ALL 16 PATTERNS
2874 012642 005037 177572                      CLR      MMR0      ; DISABLE MMU
2875 012646 042737 000400 177730      BIC      #BIT08,DCSR      ; EXIT DIAGNOSTIC MODE
2876 012654 000440      1004:      BR       TST32      ;

```

11 EXIT TEST

T31 TAG REGISTERS

```

2877
2878
2879
2880
2881 012656 125240 052520 000000 TBLML: .WORD 125240,52520,0,177760 ; FIRST PATTERN THRU 4 TAGS
      012664 177760
2882 012666 052520 000000 177760 .WORD 52520,0,177760,125240 ; SECOND PATTERN
      012674 125240
2883 012676 000000 177760 125240 .WORD 0,177760,125240,52520 ; THIRD
      012704 052520
2884 012706 177760 125240 052520 .WORD 177760,125240,52520,0 ; FOURTH
      012714 000000
2885
2886
2887
2888 012716 000052 000025 000000 TBLMH: .WORD 52,25,0,77 ; FIRST FOR HIGH MAP REGISTER
      012724 000077
2889 012726 000025 000000 000077 .WORD 25,0,77,52 ; SECOND
      012734 000052
2890 012736 000000 000077 000052 .WORD 0,77,52,25 ; THIRD
      012744 000025
2891 012746 000077 000052 000025 .WORD 77,52,25,0 ; FOURTH
      012754 000000

```

TEST - CACHE RAM BIT PATTERN TEST

```

2893 .SBTTL TEST - CACHE RAM BIT PATTERN TEST
2894
2895 ;* THIS TEST IS DONE TO CHECK THE OPERATION OF
2896 ;* THE CACHE RAM . BIT PATTERNS 0,177777,52525
2897 ;* AND 125252 ARE WRITTEN TO ALL OF THE MEMORY
2898 ;* USED AND THEN VERIFIED
2899
2900 ; BGNTST
2901
2902 ;*
2903 ;* GO SET UP THE CACHE IMAGE TABLE
2904 ;*
2905 ; LET RC := #CTBLE
2906 ; REPEAT
2907 ; . IF RC POINTS TO OCTAL BOUNDARY ADDRESS THEN
2908 ; . . WE HAVE FOUND THE BEGINNING OF THE TABLE
2909 ; . ELSE
2910 ; . . POINT RC TO THE NEXT ADDRESS
2911 ; . . ENDF
2912 ; . UNTIL WE HAVE FOUND BEGINNING OF THE TABLE
2913 ;*
2914 ;* SET UP BOARD FOR THE TRANSFERS
2915 ;*
2916 ; ENABLE UNIBUS MAPPING
2917 ; ENABLE DIAGNOSTIC MODE
2918 ; ENABLE CACHE
2919 ; DO FOR PATTERN := 0,177777,52525,125252
2920 ;*
2921 ;* INITIALIZE THE CACHE TABLE
2922 ;*
2923 ; . DO FOR ALL OF THE CACHE TABLE
2924 ; . . WRITE PATTERN TO TABLE
2925 ; . . ENDDO
2926 ;*
2927 ;* ALLOCATE ALL THE CACHE
2928 ;*
2929 ; . DO UNTIL ALL CACHE ALLOCATED
2930 ; . . DO A DIAGNOSTIC DATA NPR CYCLE (ON OCTAL BOUNDARY)
2931 ; . . ENDDO
2932 ;*
2933 ;* CHECK THAT THE DATA GOT CACHED CORRECTLY
2934 ;*
2935 ; . DO FOR ALL OF THE CACHE TABLE
2936 ; . . DO A DIAGNOSTIC DATA NPR CYCLE
2937 ; . . IF DDR NEQ PATTERN THEN
2938 ; . . . ERROR IN CACHE RAM
2939 ; . . ENDF
2940 ; . . ENDDO
2941 ; . ENDDO
2942 ;
2943 ; ENDTST
2944
2945 -----
2946
2947 ;*****
;*TEST 32 CACHE RAM BIT TEST
;*****

```

T32 CACHE RAM BIT TEST

```

012756 000004
2948 012760 005737 001720
2949 012764 001511
2950
2951
2952
2953 012766 052737 000400 177730
2954 012774 012700 002022
CHE TABLE
2955 013000 010001
2956 013002 042701 177760
2957 013006 005701
2958 013010 001402
2959 013012 005720
2960 013014 000771
2961 013016 012702 002140
2962 013022 012703 000004
2963 013026 010022
2964 013030 062700 000020
2965 013034 077304
2966
2967
2968
2969 013036 052737 000040 172516
2970 013044 005037 170202
2971 013050 052737 000400 177730
2972 013056 052737 000100 177734
2973 013064 012700 001774
2974 013070 013701 002140
2975
2976
2977
2978 013074 012702 000040
2979 013100 011021
2980 013102 077202
2981
2982
2983
2984 013104 012702 002140
2985 013110 012703 000004
2986 013114 005001
2987 013116 012237 170200
2988 013122 004737 002222
2989 013126 077305
2990
2991
2992
2993 013130 012703 000040
2994 013134 013702 002140
2995
2996 013140 005001
2997 013142 010237 170200
2998 013146 004737 002222
2999 013152 032737 100000 177734
3000 013160 001001
3001 013162 104030
3002 013164 023722 177732
3003 013170 001401

TST32: SCOPE
TST PMIS ; ANY PMI MEMORY?
BEQ TST33 ; IF NONE, EXIT TEST

; FIND THE FIRST OCTAL BOUNDARY ADDRESS IN THE TABLE
;
; BIS #BIT08,DCSR ; SELECT DIAGNOSTIC MODE
; MOV #CTBLE,R0 ; GET POINTER TO THE START OF THE SPACE ALLOCATED FOR THE CA
54: MOV R0,R1 ; SAVE IT IN R1 FOR WORKING
BIC #177760,R1 ; MASK IN THE LEAST SIGNIFICANT DIGIT
TST R1 ; IS THE ADDRESS ON AN OCTAL BOUNDARY ?
BEQ 104 ; YES, THEN GO SET UP THE TABLE ADDRESSES
TST (R0)+ ; NO, THEN GET THE NEXT ADDRESS
BR 54 ; GO CHECK IF IT FALLS ON AN OCTAL BOUNDARY
104: MOV #OBADR,R2 ; GET POINTER TO THE OCTAL BOUNDARY TABLE
MOV #4,R3 ; SET UP THE LOOP COUNTER
124: MOV R0,(R2)+ ; GO PUT ADDRESS INTO THE TABLE
ADD #20,R0 ; GET NEXT OCTAL BOUNDARY
SOB R3,124 ; FILL UP THE TABLE ?

; INITIALIZE THE BOARD FOR THE TRANSFERS
;
; BIS #BIT5,BMPPR3 ; ENABLE UNIBUS MAPPING
; CLR MAPH00 ; CLEAR HIGH MAP REGISTER
; BIS #BIT8,DCSR ; ENABLE DIAGNOSTIC MODE
; BIS #BIT6,KMCR ; ENABLE THE CACHE
; MOV #PTRN16,R0 ; GET POINTER TO THE CACHE PATTERN TABLE
154: MOV OBADR,R1 ; GET POINTER TO THE CACHE IMAGE TABLE

; INITIALIZE THE CACHE IMAGE TABLE TO CURRENT PATTERN
;
; MOV #40,R2 ; SET UP LOOP COUNTER
204: MOV (R0),(R1)+ ; MOVE PATTERN TO TABLE
SOB R2,204 ; HAVE WE DONE IT 40 TIMES ?

; ALLOCATE ALL THE CACHE MEMORY AVAILABLE
;
; MOV #OBADR,R2 ; GET ADDRESS OF OCTAL BOUNDARIES WITHIN THE TABLE
; MOV #4,R3 ; SET UP LOOP COUNTER
; CLR R1 ; NON-MAPPED ADDRESS=0
254: MOV (R2)+,MAPL00 ; SET UP MAP FOR DIAG NPR DATA CALL
JSR PC,DDIN ; GO DO A DIAGNOSTIC DATA IN CYCLE
SOB R3,254 ; HAVE WE DONE IT 4 TIMES ?

; CHECK THE CACHE RAM IMAGE PATTERN
;
; MOV #40,R3 ; SET UP LOOP COUNTER
; MOV OBADR,R2 ; GET ADDRESS OF CACHE IMAGE TABLE
; ; (THIS WHOLE TABLE SHOULD BE CACHED)
; ; USE ADDRESS 0
304: MOV R2,MAPL00 ; GET ADDRESS FOR DIAG. DATA IN CYCLE
JSR PC,DDIN ; GO DO A DIAGNOSTIC DATA IN
; ; HIT?
; ; IF YES, BRANCH
; ; OTHERWISE, ERROR
354: CMP DOR,(R2)+ ; DID IT GET STORED CORRECTLY ?
BEQ 404 ; YES, THEN GO CHECK THE NEXT LOCATION

```

J6

KTJ11-B DIAGNOSTIC MACRO M1200 05-OCT-84 09:24 PAGE 36-2

SEQ 0074

T32 CACHE RAM BIT TEST

3004	013172	104030		ERROR	+30		; . . NO, THEN ERROR IN CACHE RAM
3005	013174	077316	40\$:	S08	R3,30\$		; . . HAVE WE CHECKED ALL THE RAM ?
3006	013176	005720		TST	(R0)+		; . ALL PATTERNS DONE?
3007	013200	001333		BNE	15\$		; . IF NOT HAVE DONE, BRANCH
3008	013202	042737	000400 177730	BIC	#BIT08,DCSR		; EXIT DIAGNOSTIC MODE
3009							

TEST - BOOT ROMS TEST

```

3011 .SBTTL TEST - BOOT ROMS TEST
3012
3013 ; * THIS TEST CHECKS FOR THE PRESENCE OF BOOT ROMS. IF THEY ARE FOUND,
3014 ; * A CRC TEST IS PERFORMED FOR ALL THE ROM'S.
3015
3016 ; BGNST
3017
3018 ; LET BCSR = BCSR SET BY #BIT05 TO ENABLE ACCESSING OF ROM'S
3019 ; LET PCR = #0 TO ACCESS FIRST PAGE
3020 ; LET R4 = #173000 FOR STARTING ADDRESS
3021 ; LET $TMP0 = #0 TO CLEAR COUNT FOR EMPTY ROM'S
3022 ; DO FOR ALL 4 POSSIBLE ROMS
3023 ; . IF (R4) EQ #161777 (EMPTY SOCKET) THEN
3024 ; . . INCREMENT $TMP0 TO COUNT EMPTY ROM'S
3025 ; . . IDENTIFY POSITION OF EMPTY SOCKET
3026 ; . ELSE
3027 ; . . LET R1 = #0 TO COUNT BYTES IN ROMS
3028 ; . . DO FOR EACH LOCATION IN A ROM UNTILL R1=128
3029 ; . . . IF LOCATION 24 ACCESSED THEN
3030 ; . . . . IF (R4) NE #173000 THEN
3031 ; . . . . . ERROR IN LOCATION 24
3032 ; . . . . . ENDF
3033 ; . . . . INCREMENT R1 NOT TO DO BYTE 25
3034 ; . . . ELSE
3035 ; . . . . CALCULATE CRC FOR THE BYTE
3036 ; . . . . ENDF
3037 ; . . . INCREMENT R1 FOR THE NEXT BYTE
3038 ; . . ENDDO
3039 ; ENDDO
3040 ; IF $TMP0 NE #4 NOT ALL EMPTY
3041 ; . TRY TO WRITE TO #173000 IN DIAGNOSTIC MODE AND OUT
3042 ; . IF NO TIMEOUT THEN
3043 ; . . ERROR WRITE ACCESS TO ROMS DOES NOT TIMEOUT
3044 ; . ENDF
3045 ; ENDF
3046
3047 ; ENDTST
3048
3049
3050
3051 013210
3052
3053
3054
3055
3056 013212 042737 000400 177730
3057 013220 052737 000200 177520
3058 013226 005037 177522
3059 013232 012704 173000
3060 013236 005037 001160
3061
3062
3063

```

UBETST:

```

;*****
; *TEST 33      BOOT ROMS TEST
;*****
TST33:  SCOPE

; PREPARE TO DO CRC

; BIC      #BIT08,DCSR      ; LEAVE STANDALONE MODE
; BIS      #BIT07,BCSR      ; TO ENABLE BOOT ROMS
; CLR      PCR              ; TO READ PAGE 0
; MOV      #173000,R4       ; R4 POINTS TO FIRST ADDRESS
; CLR      $TMP0            ; CLEAR E PTY SOCKET COUNT

; DO FOR EACH POSSIBLE ROM

```

T33 BOOT ROMS TEST

```

3064 013242 022704 173776      1$:  CMP      #173776,R4      ; . ALL ROM'S DONE?
3065 013246 103475             BLO      11$      ; . IF SO, EXIT
3066 013250 005001             CLR      R1      ; . CLEAR COUNTER THRU A ROM
3067 013252 005005             CLR      R5      ; . INITIALIZE PARTIAL CRC
3068
3069      ; CHECK FOR EMPTY SOCKETS AND EXIT IF SO
3070
3071 013254 022714 161777      :  CMP      #161777,(R4)      ; . EMPTY SOCKET?
3072 013260 001027             BNE      3$      ; . IF NOT, BRANCH TO DO CRC
3073 013262 005737 001206      :  TST      $PASS      ; . FIRST PASS?
3074 013266 001017             BNE      2$      ; . IF NOT, BRANCH
3075 013270 122737 000001 001220 :  CMPSB   #APTENV,$ENV      ; IN APT MODE?
3076 013276 001413             BEQ      2$      ; IF SO, SKIP PRINTOUT
3077 013300 032737 000040 000052 :  BIT      #BIT05,$052      ; . IN UFD MODE?
3078 013306 001007             BNE      2$      ; . IF IN UFD, BRANCH
3079 013310 010446             MOV      R4,-(SP)      ; . PUT NUMBER TO TYPE OUT
3080 013312 104401 013576      :  TYPE     ,NROM      ; . TYPE ASCII MESSAGE
3081 013316 104403             TYPPOS      ; . CALL TYPE OUT ROUTINE
3082 013320      006             .BYTE     6      ; . TYPE 6 DIGITS
3083 013321      000             .BYTE     0      ; . SUPPRESS LEADING 0'S
3084 013322 104401 001175      :  TYPE     ,CRLF      ; . CARRIAGE RETURN
3085 013326 005237 001160      2$:  INC      $TMP0      ; . INCREMENT EMPTY SOCKET COUNT
3086 013332 062704 000200      :  ADD      #200,R4      ; . PREPARE TO DO NEXT ROM
3087 013336 000741             BR       1$      ; . BRANCH TO DO NEXT ROM
3088
3089      ; IN EACH ROM CHECK LOCATION 24 AND DON'T DO CRC ON IT
3090
3091 013340 022701 000024      3$:  CMP      #24,R1      ; . LOCATION 24 ACCESSED?
3092 013344 001007             BNE      5$      ; . IF NO, GO DO CRC
3093 013346 022724 173000      :  CMP      #173000,(R4)+      ; . PROPER DATA AT 24?
3094 013352 001401             BEQ      4$      ; . IF YES, BRANCH
3095 013354 104031             ERROR     +31      ; . WRONG DATA AT 24
3096 013356 005201 013420      4$:  INC      R1      ; . INCREMENT FOR AN EXTRA BYTE
3097 013360 000137             JMP      8$      ; . DO NOT DO CRC FOR 24
3098
3099      ; DO CRC FOR EACH BYTE
3100
3101 013364 112403 000010      5$:  MOVB     (R4)+,R3      ; . STORE CORRECT DATA IN R3
3102 013366 012702             MOV      #8.,R2      ; . NUMBER OF BITS PER BYTE
3103 013372 000241      6$:  CLC      ; . CLEAR CARRY
3104 013374 006005             ROR      R5      ; . . LOW BIT PARTIAL TO CARRY
3105 013376 006003             ROR      R3      ; . . CARRY TO BYTE AND BYTE TO CARRY
3106 013400 102006             BVC      7$      ; . . XOR OF PARTIAL AND BYTE LOW BITS
3107 013402 012746 120001      :  MOV      #POLY,-(SP)      ; . . XOR POLY TO PARTIAL (4 INSTRUCTIONS)
3108 013406 040516             BIC      R5,(SP)      ; . . NOT PARTIAL AND POLY
3109 013410 042705 120001      :  BIC      #POLY,R5      ; . . NOT POLY AND PARTIAL
3110 013414 052605             BIS      (SP)+,R5      ; . . POLY XOR PARTIAL
3111 013416 077213      7$:  SOB      R2,6$      ; . . DECREMENT BIT COUNT AND CONTINUE
3112 013420 005201      8$:  INC      R1      ; . COUNT BYTES
3113
3114      ; IF A ROM DONE, CHECK FOR 0 IN R5
3115
3116 013422 022701 000200      :  CMP      #128.,R1      ; . ALL 64 WORDS DONE?
3117 013426 003344             BGT      3$      ; . IF NOT, BRANCH
3118 013430 005705             TST      R5      ; . IF YES, CRC 0?
3119 013432 001401             BEQ      10$      ; . IF CRC = 0, BRANCH
3120 013434 104031             ERROR     +31      ; . CRC FOR A ROM NOT EQUAL TO 0

```

T33 BOOT ROMS TEST

```

3121 013436 000137 013242      104:  JMP      14      ; . DO FOR NEXT ROM
3122                               ;
3123                               ; TRY TO WRITE TO GET A TIMEOUT
3124                               ;
3125 013442 013737 000004 001160 114:  MOV      ERRVEC,$TMP0      ; SAVE TIMEOUT VECTOR
3126 013450 012737 013466 000004      MOV      $154,ERRVEC      ; POINT NEW TO PROGRAM
3127 013456 005037 173000      144:  CLR      $0173000      ; TRY TO WRITE TO 1ST PAGE OF ROM
3128 013462 104031      ERROR      +31      ; WRITE ACCESS TO ROM'S DIDN'T TIMEOUT
3129 013464 000402      BR      164      ;
3130 013466 005726      154:  TST      (SP)+      ; RESTORE STACK
3131 013470 005726      TST      (SP)+      ;
3132 013472 032737 000400 177730 164:  BIT      @BIT08,DCSR      ; OUT OF STANDALONE MODE?
3133 013500 001013      BNE      204      ; IF NOT, EXIT TEST
3134 013502 013702 177734      MOV      KMCR,R2      ; MAKE SURE THAT
3135 013506 042702 177700      BIC      $177700,R2      ; AT LEAST SOME OF
3136 013512 022702 000077      CMP      $77,R2      ; MEMORY PMI
3137 013516 001404      BEQ      204      ; IF NOT, BRANCH
3138 013520 052737 000400 177730      BIS      @BIT08,DCSR      ; DO 1 MORE TIME IN STANDALONE MODE
3139 013526 000753      BR      144      ;
3140 013530 052737 000010 177730 204:  BIS      @BIT03,DCSR      ; DISABLE UBA ROM RESPONSE
3141 013536 012737 013554 000004      MOV      $254,ERRVEC      ; POINT NEW TIMEOUT VECTOR
3142 013544 005737 173000      TST      $0173000      ; READ BOOT ROM
3143 013550 104037      ERROR      +37      ; DSCR<3> DIDN'T DISABLE UBA ROM
3144 013552 000402      BR      264      ;
3145 013554 005726      254:  TST      (SP)+      ; RESTORE STACK
3146 013556 005726      TST      (SP)+      ;
3147 013560 042737 000410 177730 264:  BIC      @BIT08!BIT03,DCSR      ; LEAVE STANDALONE MODE
3148 013566 013737 001160 000004      MOV      $TMP0,ERRVEC      ; RESTORE STACK
3149 013574 000417      BR      TST34      ; EXIT TEST
3150
3151 013576      101      104      104  NROM:  .ASCIZ  /ADDRESS OF EMPTY UBA SOCKET /
      013601      122      105      123
      013604      123      040      117
      013607      106      040      105
      013612      115      120      124
      013615      131      040      125
      013620      102      101      040
      013623      123      117      103
      013626      113      105      124
      013631      040      000
3152                               .EVEN

```

TEST UNIBUS MEMORY TEST

3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197

.SBTTL TEST - UNIBUS MEMORY TEST

; * THIS TEST READS KMCR<5-0> TO FIND OUT HOW MUCH UNIBUS MEMORY IS
; * AVAILABLE. THEN ALTERNATING 0'S AND 1'S WILL BE WRITTEN AND READ
; * FROM MEMORY. IF THE SYSTEM CONTAINS ALL UNIBUS MEMORY, THE FIRST
; * 32K ARE NOT GOING TO TESTED.

```

;
; BGNTST
;   IF KMCR<5-0> = <0-0> THEN NO UNIBUS MEMORY
;   . EXIT TEST
;   . ENDF
;   IF KMCR<5-0> = <1-1> THEN ALL UNIBUS MEMORY
;   . LET R1 = #1600 LOWER BOUNDARY FOR PAR
;   . LET R2 = #7600 HIGH BOUNDARY FOR PAR
;   ELSE
;   . LET R1 = COMPLEMENT(KMCR<4-0>)
;   . LET R1 = R1 SHIFT LEFT BY #7 TO GET LOWER BOUNDARY PAR
;   . LET R2 = #7600 HIGH BOUNDARY
;   . IF KMCR<5> = #0 THEN 22 BIT MODE
;   . . LET R1 = R1 SET BY #BIT15:BIT14:BIT13:BIT12 FOR 22 BITS
;   . . LET R2 = R2 SET BY #BIT15:BIT14:BIT13:BIT12
;   . ENDF
;   ENDF
;   LET MMRO<0> = #1 TO ENABLE MMU
;   REMAP PROGRAM AREA
;   DO FOR KIPAR6 FROM R1 TO R2
;   . DO FOR R4 FROM #140000 TO #157776 BY #2 FOR 4K THRU KIPAR6
;   . . LET (R4) = #125252 TO WRITE A PATTERN
;   . . IF (R4) NE #125252 THEN
;   . . . ERROR IN UNIBUS MEMORY
;   . . ENDF
;   . . LET (R4) = COMPLEMENT <(R4)>
;   . . IF (R4) NE #52525 THEN
;   . . . ERROR IN UNIBUS MEMORY
;   . . ENDF
;   . ENDDO
;   ENDDO
;   DISABLE MMU
;
; ENDTST

```

```

;*****
; *TEST 34      UNIBUS MEMORY TEST
;*****
TST34:  SCOPE

```

013634 000004
3198
3199
3200
3201
3202 013636 C32737 G00037 177734
3203 013644 001473
3204
3205
3206
3207 013646 013701 177734

```

;
; 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

```

3208 013652 042701 177700      BIC      #177700,R1      ; LEAVE ONLY <5-0>
3209 013656 022701 000077      CMP      #77,R1      ; BITS <5-0> ALL SET?
3210 013662 001005              BNE      11      ; IF NOT, BRANCH
3211 013664 012701 001600      MOV      #1600,R1     ; DON'T TEST FIRST 32K
3212 013670 012702 007600      MOV      #7600,R2     ; 248KB MAXIMUM CONFIGURATION
3213 013674 000420              BR       21      ; GO DO TEST
3214 013776 004737 002424      11:      JSR      PC,UMSIZ      ; SIZE UNIBUS MEMORY
3215                          ;
3216                          ; ON RETURN R2 HAS PAR VALUE FOR UNIBUS MEMORY
3217                          ;
3218 013702 010201              MOV      R2,R1      ; STORE LOWER BOUNDARY
3219 013704 012702 007600      MOV      #7600,R2     ; HIGHER BOUNDARY
3220 013710 032737 000040 177734 BIT      #BIT05,KMCP      ; 18 BIT MODE?
3221 013716 001007              BNE      21      ; IF YES, GO DO TEST
3222 013720 052701 170000      BIS      #170000,R1    ; EXTEND TO 22 BITS
3223 013724 052702 170000      BIS      #170000,R2    ; FOR HIGHER BOUNDARY TOO
3224 013730 052737 000020 172516 BIS      #BIT04,MMR3      ; ENABLE 22 BIT MAPPING
3225                          ;
3226                          ; NOW WRITE MEMORY WITH 01 PATTERN, VERIFY IT AND DO THE SAME FOR 10 PATTERN
3227                          ;
3228 013736 004737 002244      21:      JSR      PC,MAPP8      ; REMAP PROGRAM TO FIRST 32K
3229 013742 005237 177572      INC      MMR0      ; ENABLE MMU
3230 013746 010137 172354      MOV      R1,KIPAR6      ; START AT LOWER BOUNDARY
3231 013752 012704 140000      31:      MOV      #140000,R4      ; . START 4K PAGE
3232 013756 012714 125252      41:      MOV      #125252,(R4)      ; . . WRITE FIRST PATTERN
3233 013762 022714 125252      CMP      #125252,(R4)      ; . . WRITEN OK?
3234 013766 001401              BEQ      51      ; . . IF OK, BRANCH
3235 013770 104034              ERROR    .34      ; . . IN UNIBUS MEMORY
3236 013772 005114              51:      COM      (R4)      ; . . COMPLEMENT INITIAL PATTERN
3237 013774 022714 052525      CMP      #52525,(R4)      ; . . NEW PATTERN OK?
3238 014000 001401              BEQ      61      ; . . IF OK, BRANCH
3239 014002 104034              ERROR    .34      ; . . IN UNIBUS MEMORY
3240 014004 005724              61:      TST      (R4).      ; . . GET NEXT MEMORY LOCATION
3241 014006 022704 160000      CMP      #160000,R4      ; . . LAST ONE IN 4K PAGE?
3242 014012 101361              BHI      41      ; . . IF NOT, BRANCH
3243 014014 062737 000200 172354 ADD      #200,KIPAR6      ; . GET NEXT PAGE
3244 014022 020237 172354      CMP      R2,KIPAR6      ; . LAST PAGE DONE?
3245 014026 101351              BHI      31      ; . IF NOT, BRANCH
3246 014030 005037 177572      CLR      MMR0      ; DISABLE MMU
3247                          ;
3248                          .SBTTL UBE TESTS FOR THE UNIBUS ADAPTER BOARD
3249                          ;
3250                          ;*
3251                          ;* THE FOLLOWING TESTS REQUIRE THE USE OF THE UNIBUS EXERCISER
3252                          ;* TO TEST OUT THE UNIBUS ADAPTER BOARD.SOME TESTS REQUIRE ONE
3253                          ;* UBE OTHER TESTS REQUIRE TWO UBE'S.
3254                          ;*
3255                          ;* THE PROGRAM WILL AUTOSIZE TO SEE HOW MANY UBE'S ARE IN THE
3256                          ;* SYSTEM. DEPENDING ON THE RESULTS OF THE AUTOSIZING CERTAIN
3257                          ;* TESTS WILL BE SELECTED OR DESELECTED.
3258                          ;*
3259                          ;-----
3260

```

TEST UBE AUTOSIZING ROUTINE

3262
3263
3264
3265
3266
3267
3268
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310

.SBTTL TEST UBE AUTOSIZING ROUTINE

```

; *
; * THE FOLLOWING ROUTINE IS USED TO AUTOSIZE THE NUMBER OF
; * UBE'S PRESENT IN THE SYSTEM.
; *
;
; BGNTEST
;
;   LET R0 := #170000           ;1ST UBE ADDRESS
;   LET R2 := #510             ;1ST UBE VECTOR
;   LET R1 := #BE10B           ;TABLE FOR 1ST UBE
;   LET R3 := #BE1VEC          ;LOCATION FOR FIRST UBE VECTOR
;   LET 4 := #TOUT             ;SETUP TIMEOUT VECTOR
;   DO FOR R0 := #170000 TO 170160 BY 20
;   . TEST FOR (R0)             ;WILL TIMEOUT IF NOT THERE
;   . IF TIMEOUT OCCURS THEN
;   .   LET R0 := R0 + 20       ;GET NEXT UBE ADDRESS
;   .   LET R2 := R2 + 4       ;GET NEXT UBE VECTOR LOCATION
;   .   LET (SP) := #CHECK
;   . ELSE
; *
; * ASSIGN THE UBE ADDRESSES TO THE CURRENT UBE TABLE
; *
;   . DO FOR R4 := 1 TO 5 BY 1
;   .   . LET (R1) := R0
;   .   . LET R0 := R0 + 2
;   .   ENDDO
;   .   LET R0 := R0 + 4       ;POINT TO LAST DEVICE ADDRESS
;   .   LET (R1) := R0
;   .   LET R0 := R0 + 2       ;POINT R0 TO NEXT UBE ADDRESSES
; *
; * ASSIGN UBE VECTORS TO CURRENT UBE TABLE
; *
;   . LET (R3) := R2           ;GET POINTER TO VECTOR ADDRESS
;   . LET R2 := R2 + 2         ;GET VECTOR PSM LOCATION
;   . LET (R3) := R2           ;GET POINTER TO VECTOR PSM
;   . LET R2 := R2 + 2         ;GET NEXT VECTOR ADDRESS
;   . IF WE HAVE FOUND TWO UBE'S THEN
;   .   EXIT TEST
;   .   ENDF
;   . ENDF
;   ENDDO
; ENDTST

```

```

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

```

014034 000004

3311
3312 014036 042737 000400 177730
3313 014044 042737 000040 172516
3314 014052 005737 001206
3315 014056 001051

```

BIC    #BIT08,DCSR    ; MAKE SURE THAT OUT OF DIAGN. MODE
BIC    #BIT05,PMR3    ; MAPPING DISABLED
TST    #00PASS        ; IS THIS THE FIRST PASS ?
BNE    101            ; NO, THEN NO NEED TO SIZE AGAIN

```

T35 UNIBUS EXERCISER AUTOSIZING ROUTINE

```

3316      ;
3317      ; INITIALIZE POINTERS FOR AUTO-SIZING
3318      ;
3319 014060 012700 170000      MOV    #170000,R0      ; GET ADDRESS OF FIRST POSSIBLE UBE
3320 014064 012702 000510      MOV    #510,R2      ; GET VECTOR OF FIRST POSSIBLE UBE
3321 014070 012701 002150      MOV    @BE10B,R1      ; TABLE HEADER FOR 1ST UBE
3322 014074 012737 014106 000004      MOV    #10,B04      ; SET UP TIMEOUT VECTOR
3323 014102 005710      100:: TST    (R0)      ; . IS THERE A UBE HERE ?
3324 014104 000412      BR      24      ; . . YES, THEN BRANCH AROUND TIMEOUT ROUTINE
3325      ;
3326      ; TIMEOUT ROUTINE
3327      ;
3328      ;
3329 014106 062706 000004      18: ADD    #4,SP      ; . . READJUST THE STACK
3330 014112 020027 170160      CMP     R0,#170160      ; . . HAVE WE CHECKED ALL THE ADDRESSES
3331 014116 001431      BEQ     101      ; . . YES THEN GO SEE IF WE FOUND ANY UBE'S
3332 014120 062700 000020      ADD     #20,R0      ; . . NO, THEN GET THE NEXT UBE ADDRESS
3333 014124 062702 000004      ADD     #4,R2      ; . . GET THE NEXT UBE VECTOR
3334 014130      BR      1001      ; . . GO SEE IF THE NEXT UBE IS THERE
3335      ;
3336      ; ASSIGN THE ADDRESSES TO THE CURRENT UBE TABLE
3337      ;
3338      ;
3339 014132 012704 000005      24: MOV     #5,R4      ; . SET UP LOOP COUNTER
3340 014136 010021      34: MOV     R0,(R1)+      ; . . ASSIGN AN ADDRESS TO THE POINTER
3341 014140 005720      TST     (R0)+      ; . . GET THE NEXT ADDRESS
3342 014142 077403      SOB     R4,34      ; . . ARE WE DONE ? NO THEN GO GET NEXT ADDRESS
3343 014144 062700 000004      ADD     #4,R0      ; . . POINT R0 TO LAST UBE ADDRESS
3344 014150 010021      MOV     R0,(R1)+      ; . . PUT THE ADDRESS INTO THE POINTER TABLE
3345 014152 005720      TST     (R0)+      ; . . POINT R0 TO NEXT UBE ADDRESS
3346      ;
3347      ;
3348      ; ASSIGN UBE VECTORS TO CURRENT UBE TABLE
3349      ;
3350      ;
3351 014154 010221      MOV     R2,(R1)+      ; . GET POINTER TO VECTOR ADDRESS
3352 014156 005722      TST     (R2)+      ; . GET THE VECTOR PSW LOCATION
3353 014160 010221      MOV     R2,(R1)+      ; . GET POINTER TO VECTOR PSW
3354 014162 005722      TST     (R2)+      ; . GET THE NEXT UBE'S VECTOR ADDRESS
3355 014164 005237 001722      INC     UBECT      ; . FLAG WE HAVE FOUND ANOTHER UBE
3356      ;
3357      ;
3358      ; SEE IF WE HAVE FOUND TWO UBE'S
3359      ;
3360      ;
3361 014170 022737 000002 001722      CMP     #2,UBECT      ; . HAVE WE FOUND TWO UBE'S
3362 014176 001401      BEQ     101      ; . GO DO THE UBE TESTS
3363 014200 000740      BR      1001
3364      ;
3365      ;
3366      ; PROGRAM WILL CHECK IF ANY UBE'S WERE FOUND
3367      ;
3368      ;
3369 014202 012737 002234 000004      101: MOV     #TIMOUT,B04      ; RESTORE TIMEOUT VECTOR
3370 014210 005737 001722      TST     UBECT      ; HAVE ANY UBE'S BEEN FOUND ?
3371 014214 001002      BNE     TST36      ; GO DO THE SELECTED TESTS FOR 1 UBE
3372 014216 000137 022740      CMP     UBEH      ; SKIP ALL THE UBE TESTS

```

TEST - NPG ARBITRATION

```

3374 .SBTTL TEST - NPG ARBITRATION
3375 ;
3376 ; * THIS TEST CHECKS THAT A NPG CAN BE GRANTED AT ANY PRIORITY OF THE CPU.
3377 ; *
3378 ;
3379 ; BGNTEST
3380 ;
3381 ; DO FOR PSW FROM #340 DOWNT0 #0 BY #40
3382 ; . LET SBE1CC = -1 TO DO 1 CYCLE
3383 ; . LET SBE1BA = #1TMP0 FOR THE ADDRESS
3384 ; . LET SBE1CR1 = #2041 TO DO 1 DATI
3385 ; . WAIT FOR SBE1CR1<7>=1 READY
3386 ; . IF SBE1CC EQ -1 THEN
3387 ; . . ERROR NPG DIDN'T HAPPEN
3388 ; . ENDOF
3389 ; ENDOO
3390 ;
3391 ; ENDTST
3392 ;
3393 ; -----
3394 ; *****
3395 ; *TEST 36 NPG ARBITRATION
3396 ; *****
3397 ; TST36: SCOPE
3398 ;
3399 ; 014222 000004
3400 ;
3401 ; 014224 004737 002450 JSR PC,IUBE ; INITIALIZE THE UBE
3402 ; 014230 012737 000340 177776 MOV #340,PSW ; STORE PRIORITY 7
3403 ; 014236 012777 177777 165706 1#: MOV #-1,SBE1CC ; . DO FOR 1 CYCLE
3404 ; 014244 012777 001160 165702 MOV #1TMP0,SBE1BA ; . STORE ADDRESS FOR DATI
3405 ; 014252 012777 002041 165676 2#: MOV #2041,SBE1CR1 ; . DO 1 DATI AT NPG
3406 ; 014260 105777 165672 TSTB SBE1CR1 ; . . READY ON?
3407 ; 014264 100375 BPL 2# ; . . WAIT FOR READY
3408 ; 014266 022777 177777 165656 CMP #-1,SBE1CC ; . CYCLE COUNT INCREMENTED?
3409 ; 014274 001001 BNE 3# ; . IF YES, BRANCH
3410 ; 014276 104032 ERROR +32 ; . IF NO, ERROR IN NPG
3411 ; 014300 162737 000040 177776 3#: SUB #40,PSW ; . DECREMENT PRIORITY
3412 ; 014306 022737 000000 177776 CMP #0,PSW ; . LAST ONE?
3413 ; 014314 003750 BLE 1# ; . IF NOT, BRANCH
3414 ; 014316 012737 000340 177776 MOV #340,PSW ; RESTORE PRIORITY

```

TEST NO BUS GRANTS WITH PROCESSOR AT HIGHER PRIORITY

```

3413 .SCTL TEST - NO BUS GRANTS WITH PROCESSOR AT HIGHER PRIORITY
3414 ;
3415 ; * THIS TEST CHECKS THAT NO BUS REQUESTS ARE GOING TO BE HONORED
3416 ; * WHEN THE PROCESSOR HAS HIGHER PRIORITY THAN THE REQUESTING DEVICE.
3417 ; *
3418 ;
3419 ; BGTST
3420 ;
3421 ;     SET UP SBE1VEC TO POINT TO ERROR_CHECK_ROUTINE
3422 ;     SET UP SBE1PSW TO #340
3423 ; *
3424 ; * TRY TO DO BR7 WITH CPU PRIORITY AT 7
3425 ; *
3426 ;     LET SBE1CR1 = #21 TO DO 1 DATI
3427 ;     LET PSW = #340 TO SET PRIORITY AT 7
3428 ;     DO "NOP"
3429 ;     IF INTERRUPT THEN
3430 ;         . ERROR BG7 GRANTED WITH PROCESSOR AT HIGHER PRIORITY
3431 ;     . ENDF
3432 ;
3433 ; *
3434 ; * TRY TO DO BR6 WITH CPU PRIORITY AT 7 - 6
3435 ; *
3436 ;     DO FOR R3 FROM #340 DOWNT0 #300 BY #40
3437 ;     . LET SBE1CR1 = #11 TO DO 1 DATI
3438 ;     . LET PSW = R3 TO CHANGE PRIORITY
3439 ;     . DO "NOP"
3440 ;     . IF INTERRUPT THEN
3441 ;     .     . ERROR BG6 GRANTED WITH PROCESSOR AT HIGHER PRIORITY
3442 ;     .     . ENDF
3443 ;     . ENDDO
3444 ;     LET PSW = #340 FOR THE NEXT PART
3445 ; *
3446 ; * TRY TO DO BR5 WITH CPU PRIORITY AT 7 - 5
3447 ; *
3448 ;     DO FOR R3 FROM #340 DOWNT0 #240 BY #40
3449 ;     . LET SBE1CR1 = #5 TO DO 1 DATI
3450 ;     . LET PSW = R3 TO CHANGE PRIORITY
3451 ;     . DO "NOP"
3452 ;     . IF INTERRUPT THEN
3453 ;     .     . ERROR BG5 GRANTED WITH PROCESSOR AT HIGHER PRIORITY
3454 ;     .     . ENDF
3455 ;     . ENDDO
3456 ;     LET PSW = #340
3457 ; *
3458 ; * TRY TO DO BR4 WITH CPU PRIORITY AT 7 - 4
3459 ; *
3460 ;     DO FOR R3 FROM #340 DOWNT0 #200 BY #40
3461 ;     . LET SBE1CR1 = #2003 TO DO 1 DATI
3462 ;     . LET PSW = R3 TO CHANGE PRIORITY
3463 ;     . DO "NOP"
3464 ;     . IF INTERRUPT THEN
3465 ;     .     . ERROR BR4 GRANTED WITH PROCESSOR AT HIGHER PRIORITY
3466 ;     .     . ENDF
3467 ;     . ENDDO
3468 ;
3469 ; ENDTST

```

TEST - NO BUS GRANTS WITH PROCESSOR AT HIGHER PRIORITY

```

3470
3471
3472
3473
;-----
;*****
;*TEST 37      NO BUS GRANTS WITH PROCESSOR AT HIGHER PRIORITY
;*****
;TST37:  SCOPE
014324 000004
3474
3475 01432 004737 002450      JSR      PC,IUBE      ; INITIALIZE THE UBE
3476 01433 012737 000340 177776      MOV      #340,PSW      ; SET CPU PRIORITY AT 7
3477 014340 012777 000340 165620      MOV      #340,8BE1PSW      ; AT PRIORITY 7
3478
3479      ; TRY TO DO BR7 WITH CPU PRIORITY AT 7
3480
3481 014346 012777 014366 165610      MOV      #18,8BE1VEC      ; POINT UBE VECTOR TO PROGRAM
3482 014354 012777 000021 165574      MOV      #21,8BE1CR1      ; BR7 WITH FUNO
3483 014362 000240      NOP      ; JUST IN CASE
3484 014364 000405      BR      21      ; BRANCH AROUND IF NO INTERRUPT
3485 014366 005077 165564 14:      CLR      8BE1CR1      ; CLEAR ANY OTHER REQUESTS
3486 014372 062706 000004      ADD      #4,SP      ; ADJUST STACK POINTER
3487 014376 104032      ERROR      +32      ; BG7 GRANTED WITH CPU AT 7
3488
3489      ; TRY TO DO BR6 WITH CPU PRIORITY AT 7-6
3490
3491 014400 012777 014430 165556 24:      MOV      #44,8BE1VEC      ; POINT UBE VECTOR TO PROGRAM
3492 014406 012703 000340      MOV      #340,R3      ; START WITH PRIORITY AT 7
3493 014412 012777 000011 165536 34:      MOV      #11,8BE1CR1      ; BR6 WITH FUNO
3494 014420 010337 177776      MOV      R3,PSW      ; CHANGE PRIORITY
3495 014424 000240      NOP      ; JUST IN CASE OF INTERRUPTS
3496 014426 000405      BR      54      ; IF NO INTERRUPTS, BRANCH
3497 014430 005077 165522 44:      CLR      8BE1CR1      ; CLEAR ANY OTHER REQUESTS
3498 014434 062706 000004      ADD      #4,SP      ; ADJUST STACK POINTER
3499 014440 104032      ERROR      +32      ; BG6 GRANTED WITH CPU AT 6-7
3500 014442 162703 000040 54:      SUB      #40,R3      ; LOWER PRIORITY
3501 014446 020327 000300      CMP      R3,#300      ; LAST ONE TO CHECK?
3502 014452 002357      BGE      34      ; IF NOT, BRANCH
3503
3504      ; TRY TO DO BR5 WITH CPU PRIORITY AT 7-5
3505
3506 014454 012777 014504 165502      MOV      #74,8BE1VEC      ; POINT UBE VECTOR TO PROGRAM
3507 014462 012703 000340      MOV      #340,R3      ; START WITH PRIORITY AT 7
3508 014466 012777 000005 165462 64:      MOV      #5,8BE1CR1      ; BR5 WITH FUNO
3509 014474 010337 177776      MOV      R3,PSW      ; CHANGE PRIORITY
3510 014500 000240      NOP      ; JUST IN CASE OF INTERRUPTS
3511 014502 000405      BR      84      ; IF NO INTERRUPTS, BRANCH
3512 014504 005077 165446 74:      CLR      8BE1CR1      ; CLEAR ANY OTHER REQUESTS
3513 014510 062706 000004      ADD      #4,SP      ; ADJUST STACK POINTER
3514 014514 104032      ERROR      +32      ; BG5 GRANTED WITH CPU AT 5-7
3515 014516 162703 000040 84:      SUB      #40,R3      ; LOWER PRIORITY
3516 014522 020327 000240      CMP      R3,#240      ; LAST ONE TO CHECK?
3517 014526 002357      BGE      64      ; IF NOT, BRANCH
3518
3519      ; TRY TO DO BR4 WITH CPU PRIORITY AT 7-4
3520
3521 014530 012777 014560 165426      MOV      #104,8BE1VEC      ; POINT UBE VECTOR TO PROGRAM
3522 014536 012703 000340      MOV      #340,R3      ; START WITH PRIORITY AT 7
3523 014542 012777 000003 165406 94:      MOV      #3,8BE1CR1      ; BR6 WITH FUNO

```

H7

T37 NO BUS GRANTS WITH PROCESSOR AT HIGHER PRIORITY

3524	014550	010337	177776		MOV	R3,PSW
3525	014554	000240			NOP	
3526	014556	000405			BR	114
3527	014560	005077	165372	104:	CLR	0BE1CR1
3528	014564	062706	000004		ADD	#4,SP
3529	014570	104032			ERROR	+32
3530	014572	162703	000040	114:	SUB	#40,R3
3531	014576	020327	000200		CMF	R3,#200
3532	014602	002357			BGE	94
3533						
3534						

```

; . CHANGE PRIORITY
; . JUST IN CASE OF INTERRUPTS
; . IF NO INTERRUPTS, BRANCH
; . CLEAR ANY OTHER REQUESTS
; . ADJUST STACK POINTER
; . BGA GRANTED WITH CPL AT 4-7
; . LOWER PRIORITY
; . LAST ONE TO CHECK?
; . IF NOT, BRANCH

```

TEST - BR7-BR4 ARBITRATION

```

3536      .SBTTL TEST - BR7-BR4 ARBITRATION
3537
3538      ; * THIS TEST CHECKS THAT BR7-BR4 INTERRUPTS CAN OCCUR WITH A PROCESSOR
3539      ; * PRIORITY AT A LOWER LEVEL
3540      ; *
3541      ;
3542      ;   BGNTST
3543      ;
3544      ;       SET UP BBE1PSW TO #340
3545      ; *
3546      ; * TRY TO DO BR7 WITH CPU PRIORITY AT 6 - 0
3547      ; *
3548      ;       DO FOR R3 FROM #300 DOWNT0 #0 BY #40
3549      ;       . LET PSW = #340
3550      ;       . LET BBE1CR1 = #21 TO DO 1 DATI
3551      ;       . LET PSW = R3 TO CHANGE PRIORITY
3552      ;       . DO "NOP"
3553      ;       . IF NO INTERRUPT THEN
3554      ;       .     ERROR BG7 DOESN'T OCCUR
3555      ;       . ENDF
3556      ;       ENDDO
3557      ; *
3558      ; * TRY TO DO BR6 WITH CPU PRIORITY AT 5 - 0
3559      ; *
3560      ;       DO FOR R3 FROM #240 DOWNT0 #0 BY #40
3561      ;       . LET PSW = #340
3562      ;       . LET BBE1CR1 = #11 TO DO 1 DATI
3563      ;       . LET PSW = R3 TO CHANGE PRIORITY
3564      ;       . DO "NOP"
3565      ;       . IF NO INTERRUPT THEN
3566      ;       .     ERROR BG6 DOESN'T OCCUR
3567      ;       . ENDF
3568      ;       ENDDO
3569      ; *
3570      ; * TRY TO DO BR5 WITH CPU PRIORITY AT 4    0
3571      ; *
3572      ;       DO FOR R3 FROM #200 DOWNT0 #0 BY #40
3573      ;       . LET PSW = #340
3574      ;       . LET BBE1CR1 = #5 TO DO 1 DATI
3575      ;       . LET PSW = R3 TO CHANGE PRIORITY
3576      ;       . DO "NOP"
3577      ;       . IF NO INTERRUPT THEN
3578      ;       .     ERROR BG5 DOESN T OCCUR
3579      ;       . ENDF
3580      ;       ENDDO
3581      ; *
3582      ; * TRY TO DO BR4 WITH CPU PRIORITY AT 3    0
3583      ; *
3584      ;       DO FOR R3 FROM #140 DOWNT0 #0 BY #40
3585      ;       . LET PSW = #340
3586      ;       . LET BBE1CR1 = #3 TO DO 1 DATI
3587      ;       . LET PSW = R3 TO CHANGE PRIORITY
3588      ;       . DO "NOP"
3589      ;       . IF NO INTERRUPT THEN
3590      ;       .     ERROR BG4 DOESN T OCCUR
3591      ;       . ENDF
3592      ;       ENDDO

```

J7

TEST BR7-BR4 ARBITRATION

```
3593 ;
3594 ;   ENDTST
3595 ;
3596 ;-----
3597 ;
3598 ;*****
;*TEST 40      BR7-BR4 ARBITRATION
;*****
TST40:  SCOPE

3599 014604 000004
3600 014606 004737 002450      JSR      PC,IUBE      ; INITIALIZE THE UBE
3601 014612 012777 000340 165346  MOV      #340,8BE1PSW      ; AT PRIORITY 7
3602 ;
3603 ; TRY TO DO BR7 WITH CPU PRIORITY AT 6-0
3604 ;
3605 014620 012777 014664 165336      MOV      #21,8BE1VEC      ; POINT UBE VECTOR TO PROGRAM
3606 014626 012703 000300      MOV      #300,R3      ; START WITH PRIORITY AT 6
3607 014632 012737 000340 177776 14:  MOV      #340,PSW      ; . RAISE PRIORITY TO 7
3608 014640 012777 000021 165310      MOV      #21,8BE1CR1      ; . BR7 WITH FUNO
3609 014646 010337 177776      MOV      R3,PSW      ; . LOWER PRIORITY
3610 014652 000240      NOP      ; . JUST IN CASE OF INTERRUPTS
3611 014654 005077 165276      CLR      8BE1CR1      ; . CLEAR ANY OTHER REQUESTS
3612 014660 104032      ERROR    +32      ; . BR7 NOT GRANTED WITH CPU AT 6-0
3613 014662 000402      BR        34      ; . DON'T ADJUST STACK IF NO INTERRUPT
3614 014664 062706 000004 24:  ADD      #4,SP      ; . ADJUST STACK POINTER
3615 014670 162703 000040 34:  SUB      #40,R3      ; . LOWER PRIORITY
3616 014674 022703 000000      CMP      #0,R3      ; . LAST ONE TO CHECK?
3617 014700 002354      BGE      14      ; . IF NOT, BRANCH
3618 ;
3619 ; TRY TO DO BR6 WITH CPU PRIORITY AT 5-0
3620 ;
3621 014702 004737 002450      JSR      PC,IUBE      ; INITIALIZE THE UBE
3622 014706 012777 014752 165250  MOV      #51,8BE1VEC      ; POINT UBE VECTOR TO PROGRAM
3623 014714 012703 000240      MOV      #240,R3      ; START WITH PRIORITY AT 5
3624 014720 012737 000340 177776 44:  MOV      #340,PSW      ; . RAISE PRIORITY TO 7
3625 014726 012777 000011 165222      MOV      #11,8BE1CR1      ; . BR6 WITH FUNO
3626 014734 010337 177776      MOV      R3,PSW      ; . LOWER PRIORITY
3627 014740 000240      NOP      ; . JUST IN CASE OF INTERRUPTS
3628 014742 005077 165210      CLR      8BE1CR1      ; . CLEAR ANY OTHER REQUESTS
3629 014746 104032      ERROR    +32      ; . BR6 NOT GRANTED WITH CPU AT 5-0
3630 014750 000402      BR        64      ; . DON'T ADJUST STACK
3631 014752 062706 000004 54:  ADD      #4,SP      ; . ADJUST STACK POINTER
3632 014756 162703 000040 64:  SUB      #40,R3      ; . LOWER PRIORITY
3633 014762 022703 000000      CMP      #0,R3      ; . LAST ONE TO CHECK?
3634 014766 002354      BGE      44      ; . IF NOT, BRANCH
3635 ;
3636 ; TRY TO DO BR5 WITH CPU PRIORITY AT 4-0
3637 ;
3638 014770 004737 002450      JSR      PC,IUBE      ; INITIALIZE THE UBE
3639 014774 012777 015040 165162  MOV      #61,8BE1VEC      ; POINT UBE VECTOR TO PROGRAM
3640 015002 012703 000200      MOV      #200,R3      ; START WITH PRIORITY AT 4
3641 015006 012737 000340 177776 74:  MOV      #340,PSW      ; . RAISE PRIORITY TO 7
3642 015014 012777 000005 165134      MOV      #5,8BE1CR1      ; . BR5 WITH FUNO
3643 015022 010337 177776      MOV      R3,PSW      ; . LOWER PRIORITY
3644 015026 000240      NOP      ; . JUST IN CASE OF INTERRUPTS
3645 015030 005077 165122      CLR      8BE1CR1      ; . CLEAR ANY OTHER REQUESTS
3646 015034 104032      ERROR    +32      ; . BR5 NOT GRANTED WITH CPU AT 4-0
```

T40 BR7-BR4 ARBITRATION

```

3647 015036 000402          BR      9:
3648 015040 062706 000004    8: ADD  #4,SP
3649 015044 162703 000040    9: SUB  #40,R3
3650 015050 022703 000000      CMP  #0,R3
3651 015054 002354          BGE   7:
3652
3653      ; TRY TO DO BR4 WITH CPU PRIORITY AT 3-0
3654
3655 015056 004737 002450      JSR   PC,IUBE
3656 015062 012777 015126 165074  MOV  #11#,BBE1VEC
3657 015070 012703 000140      MOV  #140,R3
3658 015074 012737 000340 177776 10: MOV  #340,PSW
3659 015102 012777 000003 165046  MOV  #3,BBE1CR1
3660 015110 010337 177776      MOV  R3,PSW
3661 015114 000240      NOP
3662 015116 005077 165034      CLR  BBE1CR1
3663 015122 104032      ERROR  +32
3664 015124 000402          BR      12:
3665 015126 062706 000004    11: ADD  #4,SP
3666 015132 162703 000040    12: SUB  #40,R3
3667 015136 022703 000000      CMP  #0,R3
3668 015142 002354          BGE   10:
3669
3670
; . DON'T TOUCH STACK
; . ADJUST STACK POINTER
; . LOWER PRIORITY
; . LAST ONE TO CHECK?
; . IF NOT, BRANCH

; INITIALIZE THE UBE
; POINT UBE VECTOR TO PROGRAM
; START WITH PRIORITY AT 3
; . RAISE PRIORITY TO 7
; . BR4 WITH FUNO
; . LOWER PRIORITY
; . JUST IN CASE OF INTERRUPTS
; . CLEAR ANY OTHER REQUESTS
; . BR4 NOT GRANTED WITH CPU AT 3-0
; . DON'T ADJUST STACK
; . ADJUST STACK POINTER
; . LOWER PRIORITY
; . LAST ONE TO CHECK?
; . IF NOT, BRANCH

```

TEST ARBITRATION BETWEEN INTERRUPTS AND PIRQ'S

```

3672 .SMTL TEST - ARBITRATION BETWEEN INTERRUPTS AND PIRQ'S
3673
3674 ;* THIS TEST CHECKS THAT PIRQ REQUESTS OVERRIDE INTERRUPT REQUESTS
3675 ;* OF THE SAME LEVEL.
3676 ;*
3677 ;   SET UP SBE1VEC AND PIRQVEC AND PRIORITY FOR BOTH
3678 ;   LET R1 = #20 FOR BG7 FROM UBE
3679 ;   LET R2 = #100000 FOR PIRQ7
3680 ;   DO FOR R3 FROM #300 DOWNT0 #140 BY #40
3681 ;   . LET PSW = #340
3682 ;   . LET SBE1CR1 = R1 FOR INTERRUPT LEVEL
3683 ;   . LET SBE1CR1 = SBE1CR1 SET BY #1 TO DO 1 DATI
3684 ;   . LET PIRQ = PIRQ SET BY R2 FOR PIRQ LEVEL
3685 ;   . LET PSW = R3 TO CHANGE PRIORITY
3686 ;   . WAIT FOR INTERRUPT
3687 ;   . IF PIRQ INTERRUPT DIDN'T HAPPEN OR SBE1CC NE -1 THEN
3688 ;   . . ERROR IN ARBITRATION BETWEEN PIRQ'S AND INTERRUPTS
3689 ;   . ENDF
3690 ;   . LET R1 = R1 SHIFT RIGHT 1
3691 ;   . LET R2 = R2 SHIFT RIGHT 1
3692 ;   ENDDO
3693 ;
3694 ;   ENDTST
3695 ;
3696 ;-----
3697 ;*****
3698 ;*TEST 41 ARBITRATION BETWEEN INTERRUPTS AND PIRQ'S
3699 ;*****

```

015144	000004				TST41: SCOPE			
3699	015146	J04737	002450		JSR	PC,IUBE	; INITIALIZE THE UBE	
3700	015152	012777	015242	164776	MOV	#24,SBE1CR1	; POINT UBE VECTOR TO PROGRAM	
3701	015160	012777	000340	165000	MOV	#340,SBE1PSW	; AT PRIORITY 7	
3702	015166	012737	015252	000240	MOV	#34,PIRQVEC	; POINT PIRQ VECTOR TO PROGRAM	
3703	015174	012737	000340	000242	MOV	#340,#0242	; AT PRIORITY 7	
3704	015202	012701	000020		MOV	#20,R1	; STORE FOR BR7	
3705	015206	012702	100000		MOV	#100000,R2	; STORE FOR PIRQ 7	
3706	015212	012703	000300		MOV	#300,R3	; START WITH PRIORITY 6	
3707								
3708					; DO ARBITRATION FOR LEVEL 7-4			
3709								
3710	015216	012737	000340	177776	14:	MOV	#340,PSW	; . START WITH PRIORITY 7
3711	015224	010177	164726			MOV	R1,SBE1CR1	; . CHANGE UBE PRIORITY
3712	015230	010237	177772			MOV	R2,PIRQ	; . CHANGE PIRQ PRIORITY
3713	015234	010337	177776			MOV	R3,PSW	; . LOWER CPU PRIORITY
3714	015240	000001				WAIT		; . WAIT FOR INTERRUPT
3715	015242	062706	000004		24:	ADD	#4,SP	; . CLEAN UP STACK
3716	015246	104032				ERROR	#32	; . PIRQ'S DON'T TAKE PRIORITY
3717	015250	000402				BR	#4	; . DON'T CLEAN UP STACK TWICE
3718	015252	062706	000004		34:	ADD	#4,SP	; . CLEAN UP STACK
3719	015256	006001			44:	ROR	R1	; . ADJUST BR FOR UBE
3720	015260	006002				ROR	R2	; . ADJUST PIRQ'S
3721	015262	162703	000040			SUB	#40,R3	; . LOWER FOR CPU PRIORITY
3722	015266	022703	000140			CMP	#140,R3	; . PRIORITY 3?
3723	015272	001351				BNE	#1	; . BRANCH IF NOT YET
3724	015274	005037	177772			CLR	PIRQ	; . CLEAR ANY REQUESTS

TEST - ARBITRATION BETWEEN INTERRUPTS FROM 2 UBE

```

3726 .SBTTL TEST - ARBITRATION BETWEEN INTERRUPTS FROM 2 UBE
3727
3728 ; * THIS TEST CHECKS THAT WHEN TWO INTERRUPTS FROM DIFFERENT UBE'S
3729 ; * COME IN AT THE SAME TIME, THE BUS IS GRANTED TO THE REQUEST WITH HIGHER
3730 ; * PRIORITY.
3731 ; *
3732 ;
3733 . BGNTST
3734
3735 . IF UBECT NE #2 THEN THERE'S NO 2ND UBE
3736 . . EXIT TEST
3737 . ENDF
3738 . SET UP VECTORS AND PSW FOR BOTH UBE'S
3739 . LET R1 = #20 FOR BR7
3740 . DO 3 TIMES FOR BG7-BG6, BG6-BG5, BG5-BG4
3741 . . LET PSW = #340 FOR PRIORITY 7
3742 . . LET BBE1CR1 = R1
3743 . . LET R1 = R1 SHIFT RIGHT 1
3744 . . LET BBE2CR1 = R1 TO SET PRIORITY OF 2ND UBE
3745 . . LET BSIMLGO = #1 TO DO SIMULTANEOUS "GO"
3746 . . LET PSW = #140 TO LOWER PRIORITY FOR INTERRUPTS
3747 . . WAIT FOR INTERRUPTS
3748 . . IF INTERRUPTS FROM 1ST HAPPENED AFTER 2ND THEN
3749 . . . ERROR IN BR ARBITRATION
3750 . . ENDF
3751 . ENDDO
3752
3753 . ENDTST
3754
3755 -----
3756
3757 ; *****
3758 ; *TEST 42 ARBITRATION BETWEEN INTERRUPTS FROM 2 UBE
3759 ; *****
3760 015300 000004 TST42: SCOPE
3761
3762 015302 022737 000002 001722 CMP #2,UBECT ; 2 UBE'S?
3763 015310 001112 BNE TST43 ; EXIT, IF NO
3764 015312 004737 002450 JSR PC,IUBE ; INITIALIZE THE UBE
3765
3766 ; INITIAL SETUP
3767
3768 015316 012703 000240 MOV #240,R3 ; START AT PRIORITY 6
3769 015322 012701 000020 MOV #20,R1 ; UBE AT LEVEL 7
3770 015326 012777 015476 164630 MOV #BE1SV,BBE1VEC ; POINT UBE1 VECTOR TO PROGRAM
3771 015334 012777 000340 164624 MOV #340,BBE1PSW ; AT PRIORITY 7
3772 015342 012777 015516 164634 MOV #BE2SV,BBE2VEC ; POINT UBE2 VECTOR TO PROGRAM
3773 015350 012777 000340 164630 MOV #340,BBE2PSW ; AT PRIORITY 7
3774
3775 ; ARBITRATE BETWEEN 2 INTERRUPTS OF DIFFERENT LEVEL
3776
3777 015356 012737 000340 177776 14: MOV #340,PSW ; . RAISE PRIORITY TO 7
3778 015364 010177 164566 MOV R1,BBE1CR1 ; . HIGHER PRIORITY TO UBE1
3779 015370 006001 ROR R1 ; . GET LOWER PRIORITY
3780 015372 010177 164600 MOV R1,BBE2CR1 ; . LOWER PRIORITY TO UBE2
3781 015376 052777 000001 164552 BIS #BIT00,BBE1CR1 ; . SET GO BIT OF UBE #1

```

T42 ARBITRATION BETWEEN INTERRUPTS FROM 2 UBE

```

3780 015404 052777 000001 164564      BIS      #BIT00, @BE2CR1      ; . SET GO BIT OF UBE #2
3781 015412 010337 177776              MOV      R3, PSW          ; . LOWER CPU PRIORITY
3782 015416 000240                      NOP                      ; . GIVE UBE TIME TO INTERRUPT
3783 015420 000240                      NOP                      ;
3784 015422 000240                      NOP                      ;
3785 015424 005737 001724              TST      BE1INT          ; . DID UBE1 INTERRUPT
3786 015430 001002                      BNE      5$              ; . YES, THEN GO MAKE SURE UBE2 DIDN'T
3787 015432 104032                      ERROR   +32             ; . LOWER ORDER INTERRUPTS HAPPEN BEFORE
3788 015434 000404                      BR       10$             ; . GO SEE IF WE ARE DONE
3789 015436 005737 001726      5$:      TST      BE2INT          ; . DID UBE2 INTERRUPT
3790 015442 001401                      BEQ      10$             ; . NO, THEN GO SEE IF WE ARE DONE
3791 015444 104032                      ERROR   +32             ; . YES, THEN ERROR IN ARBITRATION
3792 015446 005037 001724      10$:     CLR      BE1INT          ; . INITIALIZE INTERRUPT FLAGS
3793 015452 005037 001726              CLR      BE2INT
3794 015456 162703 000040              SUB      #40, R3
3795 015462 022703 000100              CMP      #100, R3
3796 015466 001333                      BNE      1$              ; . DO THE NEXT LEVEL
3797 015470 005077 164504              CLR      @BE2CLR          ; . CPU AT 2 (LAST ONE)?
3798 015474 000420                      BR       TST43          ; . BRANCH IF NOT YET
3799                                ; . CLEAR ERRORS ON 2ND UBE
3800 015476 012777 000000 164472 BE1SV: MOV      #0, @BE2CR1      ; . CLEAR PENDING UBE #2 INTERRUPTS
3801 015504 005237 001724              INC      BE1INT          ; . SET BE1 INTERRUPT FLAG
3802 015510 005037 001726              CLR      BE2INT          ; . CLEAR BE2 INTERRUPT FLAG
3803 015514 000002                      RTI
3804
3805 015516 012777 000000 164432 BE2SV: MOV      #0, @BE1CR1      ; . CLEAR PENDING UBE #1 INTERRUPTS
3806 015524 005237 001726              INC      BE2INT          ; . SET BE2 INTERRUPT FLAG
3807 015530 005037 001724              CLR      BE1INT          ; . CLEAR BE1 INTERRUPT FLAG
3808 015534 000002                      RTI
3809

```

TEST - POWER DOWN TEST

3811
3812
3813
3814
3815
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3830
3831
3832
3833
3834
3835
3836
3837

.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 EARM ON CPU BOARD
; *

; BGNTEST

; LET BCSR<6,5>=<0,1> TO ACCESS EARM
; IF B0165002<7,6> NE <1,1> FOR POWER UP CODE 00 THEN
; . EXIT TEST
; ENDF
; SAVE PWVEC
; POINT PWVEC TO PROGRAM AREA
; LET BBE1CR2<4> = #1
; IF NO TRAP TO 24 THEN
; . ERROR EXECUTING POWER DOWN THRU UNIBUS
; ENDF
; LET BBE1CR2<4> = #0 FOR POWER UP
; RESTORE PWVEC

; ENDTST

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

; RESTRICTIONS ON RUNNING THIS TEST

3838
3839
3840
3841 015540 005737 001206
3842 015544 001102
3843 015546 032737 000040 000052
3844 015554 001076
3845 015556 122737 000001 001220
3846 015564 001472
3847 015566 032737 000010 177524
3848 015574 001066
3849 015576 042737 000100 177520
3850 015604 052737 000040 177520
3851 015612 032737 000100 165000
3852 015620 001454
3853 015622 032737 000200 165002
3854 015630 001450
3855 015632 032737 000100 165002
3856 015640 001444
3857 015642 004737 002450
3858
3859
3860
3861 015646 013737 000024 001160
3862 015654 012737 015706 000024
3863 015662 013737 000026 000026
3864 015670 052777 000020 164264

; TST #PASS ; FIRST PASS
; BNE TST44 ; IF NOT 1ST PASS, DON'T DO IT
; BIT #BIT05,B052 ; UFD MODE?
; BNE TST44 ; EXIT TEST, IF SO
; CMPB #APTEHV,ENV ; APT?
; BEQ TST44 ; EXIT TEST IF APT
; BIT #BIT03,B0177524 ; FORCED CONSOLE MODE?
; BNE TST44 ; IF YES, EXIT TEST
; BIC #BIT06,BCSR ; ENABLE ACCESS THRU 165000
; BIS #BIT05,BCSR ; READ EARM
; BIT #BIT06,B0165000 ; BATTERY OVERRIDE?
; BEQ TST44 ; IF SO, EXIT TEST
; BIT #BIT07,B0165002 ; POWER UP CODE 00?
; BEQ TST44 ; EXIT TEST, IF NOT
; BIT #BIT06,B0165002 ; POWER UP CODE 00?
; BEQ TST44 ; EXIT TEST, IF NOT
; JSR PC,IUBE ; INITIALIZE THE UBE

; DO POWER DOWN SEQUENCE

; MOV PWVEC, #TMP0 ; STORE POWER DOWN VECTOR
; MOV #1, PWVEC ; POINT NEW ONE TO PROGRAM
; MOV PWVEC+2, B026 ; AT PRIORITY 7
; BIS #BIT04, BBE1CR2 ; START POWER DOWN

T43 POWER DOWN TEST

3865	015676	000240				NOP			
3866	015700	000240				NOP			
3867	015702	104032				ERROR	.32		; WAIT WHAT HAPPENED
3868	015704	000417				BR	101		; NO POWER DOWN FOR THRU UNIBUS
3869	015706	042777	000020	164246	11:	BIC	#BIT04, BBE1CR2		
3870	015714	062706	000004			ADD	#4, SP		; CLEAR POWER DOWN BIT
3871	015720	012737	015740	000024		MOV	#31, PWRVEC		; ADJUST STACK POINTER
3872	015726	012701	177777			MOV	#177777, R1		; POINT POWER UP VECTOR
3873	015732	077101			21:	SUB	R1, 21		; TIMEOUT ROUTINE
3874	015734	104032				ERROR	.32		; WAIT A WHILE
3875	015736	000402				BR	101		; NO POWER UP
3876	015740	012706	001100		31:	MOV	#1100, SP		
3877	015744	013737	001160	000024	101:	MOV	#TMP0, PWRVEC		; ADJUST STACK
3878									; RESTORE POWER DOWN VECTOR

TEST WRONG PARITY TEST

3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903

.SBTTL TEST - WRONG PARITY TEST

```

; * THIS TEST CHECKS THAT A WRONG PARITY TRAP CAN BE GENERATED ON
; * DATA CYCLES.
; *
; *
; * BGNTST
; *
; *   SAVE 114 WRONG PARITY VECTOR AND POINT IT TO PROGRAM AREA
; *   LET BBE1CC = -1 TO DO 1 CYCLE
; *   LET BBE1BA = #1TMP0 FOR THE ADDRESS
; *   LET BBE1CR2 = BBE1CR2 SET BY #BIT12 TO ENABLE WRONG PARITY
; *   LET BBE1CR1 = #13041 TO DO 1 DATA FROM BE1CC
; *   IF NO TRAP TO 114 THEN
; *     ERROR GENERATING WRONG PARITY TRAP THRU UNIBUS
; *   ENDIF
; *   LET BBE1CR2 = #0 TO CLEAR WRONG PARITY BIT
; *   RESTORE VECTOR 114
; *
; * ENDTST

```

```

;*****
; *TEST 44      WRONG PARITY TEST
;*****
TST44:  SCOPE

```

3904 015752 000004

3905 015754 004737 002450
3906 015760 013701 000114
3907 015764 012737 016032 000114
3908 015772 012737 000340 000116
3909 016000 012777 177777 164144
3910 016006 012777 001160 164140
3911 016014 052777 010000 164140
3912 016022 005777 164130
3913 016026 000240
3914 016030 104032
3915 016032 012777 000000 164122 11:
3916 016040 062706 000004
3917

```

JSR PC,IUBE ; INITIALIZE THE UBE
MOV #114,R1 ; SAVE PARITY TRAP
MOV #11,R1 ; POINT TO PROGRAM AREA
MOV #340,R1 ; AT PRIORITY 7
MOV #-1,BBE1CC ; DO 1 CYCLE
MOV #1TMP0,BBE1BA ; SETUP ADDRESS REGISTER
BIS #BIT12,BBE1CR2 ; SET WRONG PARITY BIT
TST BBE1CR1 ; READ A UBE REGISTER
NOP ; SHOULD CAUSE A PARITY TRAP
ERROR .32 ; NO PARITY TRAP
MOV #0,BBE1CR2 ; CLEAR WRONG PARITY BIT
ADD #4,SP ; ADJUST STACK POINTER

```

TEST - NO SACK TIMEOUT

```

3919 .SBTTL TEST - NO SACK TIMEOUT
3920
3921 ; * THIS TEST SURES THAT THE CPU TIMES OUT AND DROPS A GRANT IF NO
3922 ; * SACK SIGNAL IS RECEIVED. IF THE CPU DOES NOT TIMEOUT, THE UBE
3923 ; * WILL TIME OUT AND SEND SACK TO PREVENT THE BUS FROM HANGING AND WILL SET
3924 ; * THE ERROR BIT IN CR2.
3925 ; *
3926 ;
3927 ; BGNTST
3928 ;
3929 ; LET BBE1CC = -1 TO DO 1 TRANSFER
3930 ; LET BBE1CR2 = BBE1CR2 SET BY #BIT03 TO INHIBIT SACK
3931 ; LET BBE1CR1 = #6003 TO DO FUN 3
3932 ; WAIT FOR TIMEOUT OR BBE1CC NE -1
3933 ; IF NO TIMEOUT AND BBE1CR2 SET BY #BIT07 THEN
3934 ; . ERROR CPU FAILED TO DO NO SACK TIMEOUT
3935 ; ENDF
3936 ;
3937 ; ENDTST
3938 ;
3939 ; -----
3940 ; *****
3941 ; *TEST 45 NO SACK TIMEOUT
3942 ; *****

```

```

016044 000004
3942
3943 016046 004737 002450 JSR PC,IUBE ; INITIALIZE THE UBE
3944 016052 012777 000010 164102 MOV #BIT03,BBE1CR2 ; INHIBIT SACK BIT
3945 016060 005037 177776 CLR PSW ; LOWER PRIORITY
3946 016064 012777 006003 164064 MOV #6003,BBE1CR1 ; GIVE UP BUS AFTER BECOMING MASTER
3947 016072 012701 007777 MOV #7777,R1 ; DELAY CONSTANT
3948 016076 077101 16: SOB R1,16 ; WAIT IN A LOOP
3949 016100 105777 164056 TSTB BBE1CR2 ; NO NO SACK TIMEOUT?
3950 016104 100001 BPL 26 ; IF NO, BRANCH
3951 016106 104032 ERROR +32 ; NO NO SACK TIMEOUT
3952 016110 012777 000000 164044 26: MOV #0,BBE1CR2 ; CLEAR INHIBIT SACK BIT
3953 016116 012737 000340 177776 MOV #340,PSW ; RESTORE PRIORITY

```

TEST - NO INTERRUPT TEST

```

3955      .SBTTL TEST - NO INTERRUPT TEST
3956
3957      ; * THIS TEST CHECKS THAT NO INTERRUPT CONDITION DOES NOT HANG THE BUS.
3958      ; *
3959
3960      ;   BGNTST
3961
3962      ;       PROGRAM UBE TO DO DMA ON BR7
3963      ;       IF CYCLE NOT DONE OR INTERRUPT THEN
3964      ;       .   ERROR IN NO INTERRUPT LOGIC
3965      ;       ENDIF
3966
3967      ;   ENDTST
3968
3969      ; -----
3970
3971      ; *****
3972      ; *TEST 46      PASSIVE RELEASE
3973      ; *****
3974      TST46:  SCOPE
3975
3976      016124  000004
3977
3978      016126  004737  002450      JSR      PC,IUBE      ; INITIALISE UBE
3979      016132  012777  177777  164012  MOV      #-1,8BE1CC      ; DO 1 CYCLE
3980      016140  012777  001160  164006  MOV      @TMP0,8BE1BA      ; AT ADDRESS $TMP0
3981      016146  012777  016220  164010  MOV      @10$,8BE1VEC      ; POINT VECTOR
3982      016154  012737  000003  177776  MOV      @3,PSW      ; LOWER PRIORITY TO 3
3983      016162  012777  002021  163766  MOV      @2021,8BE1CR1      ; BR7, DATI
3984      016170  105777  163762      1$:  TSTB      8BE1CR1      ; DONE?
3985      016174  100375      BPL      1$      ; IF NOT, WAIT
3986      016176  012737  000340  177776  MOV      @340,PSW      ; RESTORE PRIORITY
3987      016204  023777  001160  163736  CMP      $TMP0,8BE1D8      ; DATA OK?
3988      016212  001404      BEQ      TST47      ;; IF OK, EXIT TEST
3989      016214  104032      ERROR   +32      ; DATI ON BR7 IS WRONG
3990      016216  000402      BR      TST47      ;; GO TO NEXT TEST
3991
3992      016220  104032      10$:  ERROR   +32      ; NO INTERRUPT LOGIC IS WRONG
3993      016222  000002      RTI
3994

```

TEST - UNIBUS DEVICE DATO CYCLE

```

3991 .SBTTL TEST - UNIBUS DEVICE DATO CYCLE
3992
3993 ; * THE FOLLOWING TEST WILL SEE IF A UNIBUS DEVICE DATA OUT
3994 ; * CAN OCCUR ON THE KTJ11-B MODULE. THE UNIBUS DATA PATH
3995 ; * ON THE BOARD IS ALSO TESTED OUT.
3996 ; *
3997 ;
3998 ; BGNST
3999 ;
4000 ;     DISABLE UNIBUS MAPPING
4001 ;     LET RO := POINTER TO WRITE BUFFER
4002 ;     LET R1 := POINTER TO PATTERN TABLE
4003 ; *
4004 ; * GO DO THE TRANSFERS
4005 ; *
4006 ;     DO FOR PATTERNS 377,7417,31463,52525,125252
4007 ;     . LET BE1DB := (R1). ; DATA IS CURRENT PATTERN
4008 ;     . LET BE1BA := (RO). ; ADDRESS IS CURRENT ADDRESS OF WRITE BUFFER
4009 ;     . LET BE1CC := -1 ; SET UBE FOR 1 DATA XFER
4010 ;     . SET UBE FOR NPR-DATO-1 XFER
4011 ;     . SET OFF THE TRANSFER
4012 ;     ENDDO
4013 ;     LET RO := POINTER TO WRITE BUFFER
4014 ;     LET R1 := POINTER TO PATTERN TABLE
4015 ; *
4016 ; * CHECK IF THE TRANSFERS WERE CORRECT
4017 ; *
4018 ;     DO UNTIL TABLE IS COMPLETE
4019 ;     . IF (RO). NEQ (R1). THEN
4020 ;     . . ERROR DATO DID NOT OCCUR CORRECTLY
4021 ;     . ENDOF
4022 ;     ENDDO
4023 ;
4024 ; ENDTST
4025 ;
4026 ; -----
4027 ; *****
4028 ; *TEST 47            UNIBUS DEVICE DATO CYCLE TEST
4029 ; *****
4030 ; TST47: SCOPE
4031
4032        016224    000004
4033                                JSR    PC,IUBE            ; INITIALIZE THE UBE
4034                                ;
4035                                ; INITIALIZE POINTERS FOR THE TRANSFERS
4036                                ;
4037        016232    042737    000040    172516        BIC    #BIT5,MHR3            ; MAKE SURE UNIBUS MAPPING IS DISABLED
4038        016240    012700    001734                MOV    #WRTBUF,RO            ; POINTER TO WRITE BUFFER
4039        016244    012701    001774                MOV    #PTRN16,R1           ; POINTER TO PATTERN TABLE
4040                                ;
4041                                ; EXECUTE THE LOOP TO DO THE TRANSFERS
4042                                ;
4043        016250    012737    000002    177730        MOV    #BIT01,DCSR        ; SELECT UNIBUS LINES

```

T47 UNIBUS DEVICE DATO CYCLE TEST

```

4045 016256 012704 000005      MOV    #5,R4      ; SET UP LOOP COUNT
4046 016262 012177 163662      14:  MOV    (R1)+,R0E1DB  ; . GET CURRENT DATA
4047 016266 010077 163662      MOV    R0,R0E1BA  ; . GET ADDRESS IN THE WRITE BUFFER
4048 016272 005077 163664      CLR     R0E1CR2  ; . CLEAR ADDRESS BITS 16,17
4049 016276 012777 177777 163646  MOV    #-1,R0E1CC  ; . SET UBE FOR 1 DATA TRANSFER
4050 016304 012777 003041 163644  MOV    #3041,R0E1CR1 ; . SET UBE FOR NPR-DATO-1 XFER
4051
4052
4053      ; WAIT FOR THE TRANSFER TO BE COMPLETE
4054
4055
4056 016312 032777 000200 163636 54:  BIT     #BIT7,R0E1CR1 ; . . IS THE TRANSFER COMPLETE ?
4057 016320 001774          BEQ     54      ; . . NO, THEN GO WAIT FOR IT
4058 016322 005737 177732      TST     DOR      ; . . UNIBUS LINES 0?
4059 016326 001415          BEQ     74      ; . . IF SO, BRANCH
4060 016330 022704 000003      CMP     #3,R4      ; . . SELECTING CONTROL LINES
4061 016334 001004          BNE     64      ; . . IF NOT, BRANCH
4062 016336 032737 000373 177732  BIT     #373,DOR      ; . . ONLY USED LINES 0'S?
4063 016344 001406          BEQ     74      ; . . IF SO, BRANCH
4064 016346 013737 177732 001126 64:  MOV     DOR,R0DOAT  ; . . RECIEVED DATA
4065 016354 005037 001124      CLR     R0DOAT  ; . . EXPECTED DATA
4066 016360 104016          ERROR    +16      ; . . ERROR IN UNIBUS LINES
4067 016362 062737 000002 177730 74:  ADD     #BIT01,DCSR  ; . . THRU ALL COMBINATIONS
4068 016370 032737 000006 177730  BIT     #6,DCSR      ; . . ALL DONE?
4069 016376 001003          BNE     84      ; . . IF NOT, BRANCH
4070 016400 012737 000002 177730  MOV     #BIT01,DCSR  ; . . OTHERWISE, START OVER
4071 016406 062700 000002      84:  ADD     #2,R0      ; . . GET THE NEXT ADDRESS TO TRANSFER TO
4072 016412 077455          SOB     R4,14      ; . HAVE WE GONE THROUGH THE TABLE ?
4073
4074
4075      ; CHECK IF THE TRANSFERS WERE COMPLETED CORRECTLY
4076
4077
4078 016414 012700 001734      MOV     #WRTBUF,R0  ; POINTER TO WRITE BUFFER
4079 016420 012701 001774      MOV     #PTRN16,R1  ; POINTER TO PATTERN TABLE
4080 016424 012704 000005      MOV     #5,R4      ; SET UP LOOP COUNT
4081 016430 022021          104:  CMP     (R0)+,(R1)+  ; . WAS THE TRANSFER CORRECT ?
4082 016432 001401          BEQ     154      ; . YES, THEN SKIP THE ERROR MESSAGE
4083 016434 104032          ERROR    +32      ; . NO, THEN ERROR IN THE TRANSFER
4084 016436 077404          154:  SOB     R4,104      ; . HAVE WE CHECKED ALL 5 XFERS ?
4085
4086
4087

```

TEST - UNIBUS DEVICE DATI CYCLE

```

4089      .SBTTL TEST - UNIBUS DEVICE DATI CYCLE
4090
4091      ; * THIS TEST WILL SEE IF THE KTJ11-B MODULE CAN EXECUTE A
4092      ; * UNIBUS DEVICE DATI CYCLE. THE UNIBUS ADDRESS PATH ON THE
4093      ; * MODULE IS ALSO TESTED OUT.
4094      ; *
4095      ;
4096      ;   BGNST
4097      ;
4098      ; *
4099      ; *   INITIALIZE THE POINTERS FOR THE TRANSFERS
4100      ; *
4101      ;       DISABLE UNIBUS MAPPING
4102      ;       LET R0 := POINTER TO 16 BIT PATTERNS
4103      ;       LET R1 := (R0)                                ;CURRENT PATTERN
4104      ; *
4105      ; *   GO DO THE TRANSFER THEN CHECK IF IT OCCURED CORRECTLY
4106      ; *
4107      ;       LET BE1DB := 0                                ;CLEAR DATA BUFFER
4108      ;       LET BE1BA := R0                                ;READ FROM DATA PATTERN
4109      ;       LET BE1CC := -1                                ;1 DATA XFER
4110      ;       SETUP UBE TO DO NPR-DATI-1 DATA XFER
4111      ;       SET OFF THE TRANSFER
4112      ;       IF BE1DB NEQ #377 THEN
4113      ;       . ERROR DATA READ IN WAS INCORRECT
4114      ;       ENDIF
4115      ;
4116      ;   ENDST
4117      ;
4118      ; -----
4119      ; *****
4120      ; *TEST 50      UNIBUS DEVICE DATI CYCLE TEST
4121      ; *****
4122      ;TST50:  SCOPE
4123      ;
4124      ;
4125      ;
4126      ;   INITIALIZE POINTERS FOR THE TRANSFERS
4127      ;
4128      ;
4129      ;       MOV      #PTRN16,R0      ; GET POINTER TO PATTERN TABLE
4130      ;       MOV      (R0),R1        ; GET CURRENT PATTERN
4131      ;
4132      ;
4133      ;   GO DO THE TRANSFER
4134      ;
4135      ;
4136      ;T50:  CLR      $BE1DB          ; . INITIALIZE THE DATA BUFFER
4137      ;       MOV      R0,$BE1BA      ; . GET ADDRESS WE WANT TO READ
4138      ;       CLR      $BE1CR2        ; . CLEAR ADDRESS BITS 16,17
4139      ;       MOV      #-1,$BE1CC     ; . SET UBE FOR 1 TRANSFER
4140      ;       MOV      #2041,$BE1CR1 ; . SET UBE FOR NPR-DATI-1 XFER
4141      ;
4142      ;

```

T50 UNIBUS DEVICE DATA CYCLE TEST

```
4143                               ; WAIT FOR THE TRANSFER TO COMPLETE
4144                               ;
4145
4146 016512 032777 000200 163436 104:   BIT     #BIT7,8BE1CR1   ; . . IS THE TRANSFER COMPLETE ?
4147 016520 001774               BEQ     104               ; . . WAIT FOR IT TO BE COMPLETE
4148
4149                               ;
4150                               ; CHECK THE TRANSFER
4151                               ;
4152
4153 016522 027720 163422           CMP     8BE1DB,(R0)+   ; . WAS THE TRANSFER CORRECT ?
4154 016526 001401               BEQ     TST51           ;;     YES, THEN GO TO THE NEXT TEST
4155 016530 104032               ERROR   +32           ; . NO, THEN ERROR IN THE TRANSFER
4156
4157
4158
4159
```

TEST - UNIBUS DEVICE DATO CYCLE WITH RELOCATION ENABLED

```

4161      .SBTTL TEST - UNIBUS DEVICE DATO CYCLE WITH RELOCATION ENABLED
4162
4163      ;* THIS TEST WILL SEE IF THE KTJ11-B MODULE CAN EXECUTE
4164      ;* A UNIBUS DEVICE DATO CYCLE THROUGH THE UNIBUS MAP.
4165      ;*
4166      ;
4167      ;   BGNTST
4168      ;       IF ALL UNIBUS MEMORY THEN GO TO END OF PASS
4169      ;*
4170      ;*   SET UP UBE AND UNIBUS MAP REGISTERS FOR TRANSFER
4171      ;*
4172      ;       LET BE1BA := 0                      ;POINT TO UMRO
4173      ;       LET BE1DB := 125252                ;DATA PATTERN IS 125252
4174      ;       LET BE1CC := -8.                   ;DO 8 XFERS
4175      ;       SET UBE TO DO NPR-DATO-2 DATA XFERS
4176      ;       LET MAPLOO := ADDRESS OF WRITE BUFFER
4177      ;       LET MAPHOO := 0
4178      ;*
4179      ;*   GO DO THE TRANSFER
4180      ;*
4181      ;       SET OFF TRANSFER
4182      ;       WAIT TILL XFER IS DONE
4183      ;*
4184      ;*   CHECK IF THE TRANSFER OCCURED CORRECTLY
4185      ;*
4186      ;       LET RO := POINTER TO WRITE BUFFER
4187      ;       DO FOR R1 := 1 TO 8
4188      ;           IF (RO)+ NEQ #125252 THEN
4189      ;               ERROR DATO DID NOT EXECUTE CORRECTLY
4190      ;           .ENDIF
4191      ;       ENDDO
4192      ;
4193      ;   ENDTST
4194      ;
4195      ;-----
4196      ;*****
4197      ;*TEST 51      UNIBUS DEVICE DATO CYCLE WITH RELOCATION ENABLED
4198      ;*****
4199      TST51:  SCOPE
4200
4201      016532  000004
4202
4203      016534  013701  177734      MOV      KMCR,R1      ; STORE KMCR
4204      016540  042701  177700      BIC      #177700,R1    ; LEAVE ONLY BITS <5-0>
4205      016544  022701  000077      CMP      #77,R1        ; ALL UNIBUS MEMORY?
4206      016550  001002      BNE      1#                  ; IF NOT, BRANCH
4207      016552  000137  022766      JMP      #EOP          ; IF ALL UNIBUS, SKIP TILL END OF PASS
4208      016556  004737  002450      JSR      PC,IUBE       ; INITIALIZE THE UBE
4209
4210      ;
4211      ;   SET UP UBE AND UNIBUS MAP REGISTERS FOR TRANSFER
4212      ;
4213      ;
4214      ;       CLR      #BE1BA                      ; POINT UBE ADDRESS TO UNIBUS MAP
4215      ;       CLR      #BE1CR2                     ; REGISTER #0
4216      ;       MOV      #125252,#BE1DB              ; DATA PATTERN IS 125252
4217      ;       MOV      #-8.,#BE1CC                 ; DO 8. TRANSFERS
4218      ;       MOV      #WRTBUF,MAPLOO              ; SET UP LOW MAP REGISTER

```

T51 UNIBUS DEVICE DATA CYCLE WITH RELOCATION ENABLED

```

4215 016614 005037 170202          CLR      MAPH00          ; SET UP HIGH MAP REGISTER
4216 016620 052737 000040 172516    BIS      #BIT5,MMR3      ; ENABLE UNIBUS MAPPING
4217
4218                                ;
4219                                ; GO DO THE TRANSFER
4220                                ;
4221
4222 016626 012777 003041 163322      MOV      #3041,8BE1CR1    ; DO A NPR-DATO-1 XFER
4223 016634 032777 000200 163314 10$: BIT      #BIT7,8BE1CR1    ; IS THE TRANSFER COMPLETE ?
4224 01664. 001774                      BEQ      10$           ; WAIT FOR IT TO COMPLETE
4225
4226                                ;
4227                                ; CHECK IF THE TRANSFER OCCURED CORRECTLY
4228                                ;
4229
4230 016644 042737 000040 172516      BIC      #BIT5,MMR3      ; DISABLE UNIBUS MAPPING
4231 016652 012700 001734                      MOV      #WRTBUF,RO    ; GET POINTER TO WRITE BUFFER
4232 016656 012701 000010                      MOV      #8.,R1      ; SET UP LOOP COUNTER
4233 016662 022027 125252      15$:  CMP      (RO)+,#125252    ; . SEE IF TRANSFER OCCURED CORRECTLY ?
4234 016666 001401                      BEQ      20$           ; . YES, THEN SKIP ERROR MESSAGE
4235 016670 104032                      ERROR   +32            ; . NO, THEN ERROR IN TRANSFER
4236 016672 077105      20$:  SOB      R1,15$           ; . HAVE WE CHECKED ALL THE TRANSFERS
4237
4238

```

TEST UNIBUS DEVICE DATI CYCLE WITH RELOCATION ENABLED

```

4240      .SBTTL TEST - UNIBUS DEVICE DATI CYCLE WITH RELOCATION ENABLED
4241
4242      ; * THIS TEST WILL SEE IF THE KTJ11-B MODULE CAN EXECUTE
4243      ; * A UNIBUS DEVICE DATI CYCLE THROUGH THE UNIBUS MAP WITH
4244      ; * CACHE DISABLED.
4245      ; *
4246      ;
4247      ; BGNST
4248      ;
4249      ; *
4250      ; * SET UP THE UBE FOR A DATI CYCLE
4251      ; *
4252      ;       LET BE1BA := 0           ;POINT ADDRESS TO MAPX00
4253      ;       LET BE1CR2 := 0         ;
4254      ;       LET BE1DB := 0         ;INITIALIZE DATA BUFFER
4255      ;       LET BE1CC := -1        ;SET FOR 1 XFER
4256      ;       SET UBE FOR A NPR-DATI-1 XFER
4257      ; *
4258      ; * SET UP THE MAP REGISTER FOR THE TRANSFER
4259      ; *
4260      ;       LET MAPLOO := R0         ;POINT UMR #0 TO DATA PATTERN
4261      ;       LET MAPHOO := 0
4262      ;       ENABLE UNIBUS MAPPING
4263      ; *
4264      ; * GO DO THE TRANSFER
4265      ; *
4266      ;       SET OFF THE TRANSFER
4267      ;       WAIT FOR TRANSFER TO COMPLETE
4268      ;       DISABLE UNIBUS MAPPING
4269      ; *
4270      ; * CHECK IF THE TRANSFER OCCURED CORRECTLY
4271      ; *
4272      ;       IF BE1DB NEQ (R0) THEN
4273      ;       . ERROR DATI THROUGH UNIBUS MAP DID NOT EXECUTE CORRECTLY
4274      ;       ENDIF
4275      ;
4276      ; ENDTST
4277      ;
4278      ; -----
4279
4280      ; *****
4281      ; *TEST 52            UNIBUS DEVICE DATI CYCLE WITH RELOCATION ENABLED
4282      ; *****

```

```

016674  G00004
4282
4283 016676 004737 002450      JSR    PC,IUBE            ; INITIALIZE THE UBE
4284
4285      ;
4286      ; SET UP UBE AND UNIBUS MAP REGISTERS FOR A DATI CYCLE
4287      ;
4288
4289 016702 005077 163246      CLR    @BE1BA            ; POINT UBE TO THE FIRST UNIBUS MAP
4290 016706 005077 163250      CLR    @BE1CR2          ; REGISTER
4291 016712 005077 163232      CLR    @BE1DB            ; INITIALIZE THE DATA BUFFER
4292 016716 012777 177777 163226  MOV    @-1,@BE1CC        ; SET FOR 1 TRANSFER
4293 016724 012737 001774 170200  MOV    @PTRN16,MAPLOO    ; POINT MAP REGISTERS TO PATTERN TABLE

```

T52 UNIBUS DEVICE DATI CYCLE WITH RELOCATION ENABLED

```

4294 016732 005037 170202          CLR      MAPH00          ;
4295 016736 052737 000040 172516    BIS      #BIT5,MMR3      ; ENABLE UNIBUS MAPPING
4296
4297                                ;
4298                                ; GO DO THE TRANSFER
4299                                ;
4300
4301 016744 012777 002041 163204      MOV      #2041,8BE1CR1    ; SET UBE FOR MPR-DATI-1 XFER
4302 016752 032777 000200 163176 5:  BIT      #BIT7,8BE1CR1    ; IS THE TRANSFER COMPLETE
4303 016760 001774          BEQ      5:                        ; NO, THEN WAIT FOR IT TO BE COMPLETE
4304 016762 042737 000040 172516      BIC      #BIT5,8MMR3      ; YES, THEN DISABLE UNIBUS MAPPING
4305
4306                                ;
4307                                ; CHECK THE TRANSFER
4308                                ;
4309
4310 016770 027727 163154 000377      CMP      8BE10B, #377      ; DID THE TRANSFER HAPPEN CORRECTLY ?
4311 016776 001401          BEQ      TST53                    ; GO TO THE NEXT TEST
4312 017000 104032          ERROR     +32                      ; TRANSFER DID NOT OCCUR CORRECTLY
4313
4314
4315

```

TEST - ALU TEST USING UBE

4317
4318
4319
4320
4321
4322
4323
4324
4325
4326
4327
4328
4329
4330
4331
4332
4333
4334
4335
4336
4337
4338
4339
4340
4341
4342
4343
4344
4345
4346
4347
4348
4349
4350
4351
4352
4353
4354
4355
4356
4357
4358
4359
4360
4361
4362
4363
4364
4365
4366
4367
4368
4369
4370
4371
4372
4373

```
.SBTTL TEST - ALU TEST USING UBE

; * THIS TEST WILL CHECK THE FIRST 3 ALU'S EACH ONE AT A TIME. AT FIRST THE
; * PATTERN THAT DOES NOT RESULT IN OVERFLOW WILL BE USED, THE SECOND PATTERN
; * GENERATES OVERFLOW INTO THE NEXT ALU.
; *
; *
;   BGNTST
;
;   IN 18-BIT MODE DON'T RUN THIS TEST
;   LET R1 := 1001 (PATTERN WITHOUT OVERFLOW FROM ALU)
;   LET R2 := 0101 (MAP REGISTER PATTERN)
;   LET R3 := 0MAP00
;   LET R4 := 0110 (OVERFLOW)
;   LET R5 := 1011
;   DO FOR 3 PATTERNS
;   . LET BE1BA := R1                ;POINT ADDRESS TO MAPX00
;   . LET BE1CR2 := 0                ;
;   . LET BE1DB := 0                ;INITIALIZE DATA BUFFER
;   . LET BE1CC := -1                ;SET FOR 1 XFER
;   . SET UBE FOR A NPR-DATI-1 XFER
; *
; * SET UP THE MAP REGISTER FOR THE TRANSFER WITHOUT OVERFLOW FROM ALU
; *
;   . LET (R3) := R2                ;POINT UMR #0 TO DATA PATTERN
;   . LET 2(R3) := 0
;   . ENABLE UNIBUS MAPPING
; *
; * GO DO THE TRANSFER
; *
;   . SET OFF THE TRANSFER
;   . WAIT FOR TRANSFER TO COMPLETE
;   . DISABLE UNIBUS MAPPING
; *
; * CHECK IF THE TRANSFER OCCURED CORRECTLY
; *
;   . IF BE1DB NEQ (R1,R2) THEN
;   .   ERROR DATI THROUGH UNIBUS MAP DID NOT EXECUTE CORRECTLY
;   . ENDIF
; *
; * NOW CHECK CARRY OUT FROM ALU
; *
;   . LET BE1BA := R4                ;POINT ADDRESS TO MAPX00
;   . LET BE1CR2 := 0                ;
;   . LET BE1DB := 0                ;INITIALIZE DATA BUFFER
;   . LET BE1CC := -1                ;SET FOR 1 XFER
;   . SET UBE FOR A NPR-DATI-1 XFER
; *
; * SET UP THE MAP REGISTER FOR THE TRANSFER
; *
;   . LET (R3) := R5                ;POINT UMR #0 TO DATA PATTERN
;   . LET 2(R3) := 0
;   . ENABLE UNIBUS MAPPING
; *
; * GO DO THE TRANSFER
; *
;   . SET OFF THE TRANSFER
```

TEST - ALU TEST USING UBE

```

4374      ;      . WAIT FOR TRANSFER TO COMPLETE
4375      ;      . DISABLE UNIBUS MAPPING
4376      ;*
4377      ;* CHECK IF THE TRANSFER OCCURED CORRECTLY
4378      ;*
4379      ;      IF BE1DB NEQ (R4,R5) THEN
4380      ;      . ERROR DATA THROUGH UNIBUS MAP DID NOT EXECUTE CORRECTLY
4381      ;      . ENDF
4382      ;*
4383      ;* CHANGE THE PATTERN
4384      ;*
4385      ;      . SHIFT R1,R2,R4,R5 4 TIMES AND POINT TO MAPLO3 AND MAPLO6
4386      ;      . POINT R3 TO MAPLO3 AND THEN TO MAPLO6
4387      ;      ENDDO
4388      ;
4389      ; ENDTST
4390      ;
4391      ;-----
4392
4393
4394

```

```

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

```

4395	017002	000004					
4396	017004	032737	000040	177734	BIT	#BIT05,KMCR	; 18 BIT MODE?
4397	017012	001402			BEQ	104	; IF NOT, CONTINUE
4398	017014	000137	017432		JMP	204	; OTHERWISE, EXIT
4399	017020	005037	172354		104: CLR	KIPAR6	; FOR ERROR REPORTING
4400	017024	004737	002450		JSR	PC,IUBE	; INITIALIZE THE UBE
4401	017030	012700	000022		MOV	#22,R0	; INITIAL PATTERN FOR ADDRESS LINES
4402	017034	012701	000012		MOV	#12,R1	; INITIAL PATTERN FOR MAP REGISTER
4403	017040	012703	170200		MOV	#MAPLO0,R3	; FIRST REGISTER PAIR TO BE USED
4404	017044	012704	000014		MOV	#14,R4	; PATTERN FOR THE OVERFLOW FOR A. LINES
4405	017050	012705	000026		MOV	#26,R5	; PATTERN FOR MAP PAIR
4406	017054	005037	001162		CLR	#TMP1	; LOCATION TO SELECT REGISTER PAIR
4407							
4408							
4409							
4410							
4411	017060	010077	163070		16: MOV	R0,#BE1BA	; POINT UBE TO THE PATTERN
4412	017064	005077	163072		CLR	#BE1CR2	; REGISTER
4413	017070	005077	163054		CLR	#BE1DB	; INITIALIZE THE DATA BUFFER
4414	017074	012777	177777	163050	MOV	#-1,#BE1CC	; SET FOR 1 TRANSFER
4415	017102	010113			MOV	R1,(R3)	; POINT MAP REGISTERS TO PATTERN
4416	017104	005063	000002		CLR	2(R3)	; CLEAR HIGH MAP REGISTER
4417	017110	052737	000040	172516	BIS	#BIT5,MNR3	; ENABLE UNIBUS MAPPING
4418							
4419							
4420							
4421							
4422							
4423	017116	012777	002041	163032	MOV	#2041,#BE1CR1	; SET UBE FOR NPR-DATI-1 XFER
4424	017124	032777	000200	163024	24: BIT	#BIT7,#BE1CR1	; IS THE TRANSFER COMPLETE
4425	017132	001774			BEQ	24	; NO, THEN WAIT FOR IT TO BE COMPLETE
4426	017134	042737	000040	172516	BIC	#BIT5,#MNR3	; YES, THEN DISABLE UNIBUS MAPPING
4427							

T53 ALU TEST USING UBE

```

4428
4429      ; CHECK THE TRANSFER
4430
4431
4432 017142 010037 001160      MOV      R0, $TMP0      ; FIND ADDRESS ACCESSED
4433 017146 042737 160000 001160 BIC      $160000, $TMP0      ; MAKE SURE THAT USING ALL 16 BITS FOR ADDRESS
4434 017154 060137 001160      ADD      R1, $TMP0
4435 017160 027777 162764 161772 CMP      $BE1DB, $TMP0      ; DID THE TRANSFER HAPPEN CORRECTLY ?
4436 017166 001412      BEQ      34      ; IF SO, BRANCH
4437 017170 017737 162754 001126 MOV      $BE1DB, $BDDAT      ; WRONG DATA
4438 017176 017737 161756 001124 MOV      $TMP0, $GDDAT      ; EXPECTED DATA
4439 017204 013737 001160 001122 MOV      $TMP0, $BADDR      ; ADDRESS USED
4440 017212 104023      ERROR      +23      ; TRANSFER DID NOT OCCUR CORRECTLY
4441
4442      ; NOW CHECK THE SAME ALU FOR OVERFLOW
4443
4444 017214 010477 162734      34: MOV      R4, $BE1BA      ; POINT UBE TO THE PATTERN
4445 017220 005077 162736      CLR      $BE1CR2      ; REGISTER
4446 017224 005077 162720      CLR      $BE1DB      ; INITIALIZE THE DATA BUFFER
4447 017230 012777 177777 162714 MOV      0-1, $BE1CC      ; SET FOR 1 TRANSFER
4448 017236 010513      MOV      R5, (R3)      ; POINT MAP REGISTERS TO PATTERN
4449 017240 005063 000002      CLR      2(R3)      ; CLEAR HIGH MAP REGISTER
4450 017244 052737 000040 172516 BIS      $BIT5, $MMR3      ; ENABLE UNIBUS MAPPING
4451
4452
4453      ; GO DO THE TRANSFER
4454
4455
4456 017252 012777 002041 162676      MOV      $2041, $BE1CR1      ; SET UBE FOR NPR-DAT: 1 XFER
4457 017260 032777 000200 162670 44: BIT      $BIT7, $BE1CR1      ; IS THE TRANSFER COMPLETE
4458 017266 001774      BEQ      44      ; NO, THEN WAIT FOR IT TO BE COMPLETE
4459 017270 042737 000040 172516 BIC      $BIT 5, $MMR3      ; YES, THEN DISABLE UNIBUS MAPPING
4460
4461
4462      ; CHECK THE TRANSFER
4463
4464
4465 017276 010437 001160      MOV      R4, $TMP0      ; FIND ADDRESS ACCESSED
4466 017302 042737 160000 001160 BIC      $160000, $TMP0      ; MAKE SURE THAT USING ALL 16 BITS FOR ADDRESS
4467 017310 060537 001160      ADD      R5, $TMP0
4468 017314 027777 162630 161636 CMP      $BE1DB, $TMP0      ; DID THE TRANSFER HAPPEN CORRECTLY ?
4469 017322 001412      BEQ      54      ; IF SO, BRANCH
4470 017324 017737 162620 001126 MOV      $BE1DB, $BDDAT      ; WRONG DATA
4471 017332 017737 161622 001124 MOV      $TMP0, $GDDAT      ; EXPECTED DATA
4472 017340 013737 001160 001122 MOV      $TMP0, $BADDR      ; ADDRESS USED
4473 017346 104023      ERROR      +23      ; TRANSFER DID NOT OCCUR CORRECTLY
4474
4475      ; CHANGE THE PATTERN
4476
4477 017350 022737 002000 001720 54: CMP      $2000, $PHIS      ; MORE THAN 32K OF PHI MEMORY?
4478 017356 103025      BHS      TST54      ; IF NOT, EXIT TEST
4479 017360 072027 000004      ASH      $4, R0      ; TO GET TO NEXT ALU
4480 017364 072127 000004      ASH      $4, R1      ; BY SHIFYING 4 TIMES
4481 017370 072427 000004      ASH      $4, R4
4482 017374 072527 000004      ASH      $4, R5
4483 017400 062703 000014      ADD      $12, R3      ; GET NEXT REGISTER PAIR
4484 017404 062737 060000 001162 ADD      $60000, $TMP1

```

T53 ALU TEST USING UBE

4485	017412	053700	001162		BIS	\$TMP1,R0	; SET TO SELECT NEXT REGISTER PAIR
4486	017416	053704	001162		BIS	\$TMP1,R4	;
4487	017422	022737	020000	001162	CMP	#20000,\$TMP1	; ALL 3 ALU'S DONE?
4488	017430	001213			BNE	1;	; IF NOT, BRANCH
4489	017432			204:			

TEST - CARRY PROPOGATION TEST USING UBE

```

4491 .SBTTL TEST - CARRY PROPOGATION TEST USING UBE
4492
4493 ; * THIS TEST VALIDATES THAT CARRY CAN BE PROPOGATED THRU ALL ALU'S CORRECTLY.
4494 ; * THE RESULT IS OBTAINED FROM LOCATION 0.
4495 ;
4496 ; BGNTST
4497 ;
4498 ; *
4499 ; * SET UP THE UBE FOR A DATI CYCLE
4500 ; *
4501 ; LET B#0 := #52525
4502 ; LET BE1BA := #2
4503 ; LET BE1CR2 := 0
4504 ; LET BE1DB := 0 ; INITIALIZE DATA BUFFER
4505 ; LET BE1CC := -1 ; SET FOR 1 XFER
4506 ; SET UBE FOR A NPR-DATI-1 XFER
4507 ;
4508 ; * SET UP THE MAP REGISTER FOR THE TRANSFER
4509 ; *
4510 ; LET MAPL00 := #177777 ; POINT UMR #0 TO DATA PATTERN
4511 ; LET MAPH00 := #77
4512 ; ENABLE UNIBUS MAPPING
4513 ;
4514 ; * GO DO THE TRANSFER
4515 ; *
4516 ; SET OFF THE TRANSFER
4517 ; WAIT FOR TRANSFER TO COMPLETE
4518 ; DISABLE UNIBUS MAPPING
4519 ;
4520 ; * CHECK IF THE TRANSFER OCCURED CORRECTLY
4521 ; *
4522 ; IF BE1DB NEQ B#0 (52525) THEN
4523 ; . ERROR DATI THROUGH UNIBUS MAP DID NOT EXECUTE CORRECTLY
4524 ; ENDIF
4525 ;
4526 ; ENDTST
4527 ;
4528 ;
4529 ; -----

```

```

017432 000004
4530
4531 017434 005037 172354 CLR KIPAR6 ; CLEAR FOR ERROR REPORTS
4532 017440 004737 002450 JSR PC,IUBE ; INITIALIZE THE UBE
4533
4534 ;
4535 ; SET UP UBE AND UNIBUS MAP REGISTERS FOR A DATI CYCLE
4536 ;
4537
4538 017444 012737 052525 000000 MOV #52525,B#0 ; TEST LOCATION
4539 017452 012777 000002 162474 MOV #2,BE1BA ; POINT UBE TO THE FIRST UNIBUS MAP
4540 017460 005077 162476 CLR BE1CR2 ; REGISTER
4541 017464 005077 162460 CLR BE1DB ; INITIALIZE THE DATA BUFFER
4542 017470 012777 177777 162454 MOV #-1,BE1CC ; SET FOR 1 TRANSFER
4543 017476 012737 177777 170200 MOV #177777,MAPL00 ; POINT MAP REGISTERS TO OVERFLOW
4544 017504 012737 000077 170202 MOV #77,MAPH00 ;

```

T54 CARRY PROPOGATION TEST USING UBE

```

4545 017512 052737 000040 172516      BIS      #BIT5,MRR3      ; ENABLE UNIBUS MAPPING
4546
4547                                      ;
4548                                      ; GO DO THE TRANSFER
4549                                      ;
4550
4551 017520 012777 002041 162430      MOV      #2041,0BE1CR1      ; SET UBE FOR NPR-DATI-1 XFER
4552 017526 032777 000200 162422 51:      BIT      #BIT7,0BE1CR1      ; IS THE TRANSFER COMPLETE
4553 017534 001774                      BEQ      51                      ; NO, THEN WAIT FOR IT TO BE COMPLETE
4554 017536 042737 000040 172516      BIC      #BIT5,0MRR3      ; YES, THEN DISABLE UNIBUS MAPPING
4555
4556                                      ;
4557                                      ; CHECK THE TRANSFER
4558                                      ;
4559
4560 017544 027727 162400 052525      CMP      0BE1DB,#52525      ; DID THE TRANSFER HAPPEN CORRECTLY ?
4561 017552 001411                      BEQ      TST55                      ; GO TO THE NEXT TEST
4562 017554 017737 162370 001126      MOV      0BE1DB,#BDDAT      ; WRONG DATA
4563 017562 012737 052525 001124      MOV      #52525,#GDDAT      ; EXPECTED DATA
4564 017570 005037 001122                      CLR      #BDAOR                      ; ADDRESS USED
4565 017574 104023                      ERROR      +23                      ; TRANSFER DID NOT OCCUR CORRECTLY
4566

```

TEST - NXM TEST USING UBE

```

4568 .SBTTL TEST - NXM TEST USING UBE
4569 ; * THIS TEST CHECKS THAT ACCESSING NXM MEMORY CAN BE DONE CORRECTLY THRU
4570 ; * UBE. TEST LOCATION USED IS 17760000.
4571 ;
4572 ; BGNTEST
4573 ;
4574 ; *
4575 ; * SET UP THE UBE FOR A DATI CYCLE
4576 ; *
4577 ;     LET BE1BA := 0           ; POINT ADDRESS TO MAPX00
4578 ;     LET BE1CR2 := 0         ;
4579 ;     LET BE1DB := 0          ; INITIALIZE DATA BUFFER
4580 ;     LET BE1CC := -1         ; SET FOR 1 XFER
4581 ;     SET UBE FOR A NPR-DATI-1 XFER
4582 ; *
4583 ; * SET UP THE MAP REGISTER FOR THE TRANSFER
4584 ; *
4585 ;     LET MAPL00 := #160000    ; POINT UMR #0 TO DATA PATTERN
4586 ;     LET MAPH00 := #77
4587 ;     ENABLE UNIBUS MAPPING
4588 ; *
4589 ; * GO DO THE TRANSFER
4590 ; *
4591 ;     SET OFF THE TRANSFER
4592 ; *
4593 ; * CHECK IF THE TRANSFER OCCURED CORRECTLY
4594 ; *
4595 ;     IF #BIT08 NOTSET IN BBE1CR2 THEN
4596 ;     . ERROR DATI TO NXM IS WRONG
4597 ;     ENDIF
4598 ;
4599 ; ENDTST
4600 ;
4601 ; -----
4602 ; *****
4603 ; *TEST 55      NXM TEST USING UBE WITH RELOCATION ENABLED
4604 ; *****
4605 TST55: SCOPE
4606 JSR      PC,IUBE          ; INITIALIZE THE UBE
4607 ;
4608 ; SET UP UBE AND UNIBUS MAP REGISTERS FOR A DATI CYCLE
4609 ;
4610 MOV      #10#,BBE1VEC     ; POINT INTERRUPT VECTOR
4611 MOV      #340,BBE1PSW    ;
4612 CLR      BBE1BA          ; POINT UBE TO THE FIRST UNIBUS MAP
4613 CLR      BBE1CR2         ; REGISTER
4614 CLR      BBE1DB          ; INITIALIZE THE DATA BUFFER
4615 MOV      #-1,BBE1CC      ; SET FOR 1 TRANSFER
4616 MOV      #160000,MAPL00  ; POINT MAP REGISTERS TO NXM 17760000
4617 MOV      #77,MAPH00     ;
4618 BIS      #BIT5,MMR3       ; ENABLE UNIBUS MAPPING
4619 ;
4620 ; GO DO THE TRANSFER
4621 ;
4622 MOV      #2041,BBE1CR1    ; SET UBE FOR NPR-DATI-1 XFER

```

4604	017576	000004		
4605	017600	004737	002450	
4606				
4607				
4608				
4609	017604	012777	017734	162352
4610	017612	012777	000340	162346
4611	017620	005077	162330	
4612	017624	005077	162332	
4613	017630	005077	162314	
4614	017634	012777	177777	162310
4615	017642	012737	160000	170200
4616	017650	012737	000077	170202
4617	017656	052737	000040	172516
4618				
4619				
4620				
4621	017664	012777	002041	162264

T55 NXM TEST USING UBE WITH RELOCATION ENABLED

```

4622 017672 032777 000200 162256 54:    BIT    @BIT7,@BE1CR1    ; IS THE TRANSFER COMPLETE
4623 017700 001774                        BEQ    54                ; NO, THEN WAIT FOR IT TO BE COMPLETE
4624 017702 032777 000400 162252                BIT    @BIT08,@BE1CR2    ; NXM ?
4625 017710 001001                        BNE    64                ; IF SO, BRANCH
4626 017712 104022                        ERROR    +22                ; NXM NOT SET
4627 017714 106427 000140                MTPS    @140                ; ALLOW INTERRUPTS
4628 017720 000240                        NOP
4629 017722 000240                        NOP
4630 017724 104022                        ERROR    +22                ; NXM WITHOUT INTERRUPT
4631 017726 000404                        BR       124                ;
4632 017730 106427 000340                MTPS    @340                ; RESTORE PRIORITY
4633                                        ;
4634                                        ; CHECK THE TRANSFER
4635                                        ;
4636 017734 062706 000004                104:    ADD    @4,SP                ; ADJUST STACK
4637 017740 042737 000040 172516 124:    BIC    @BIT5,@MPPR3    ; YES, THEN DISABLE UNIBUS MAPPING
4638

```

TEST - RELOCATION WITH UNIBUS MEMORY

```

4640 .S8TTL TEST - RELOCATION WITH UNIBUS MEMORY
4641
4642 ;* THIS TEST WILL CHECK THAT MAPPING REGISTER PAIRS THAT ARE
4643 ;* ASSOCIATED WITH UNIBUS MEMORY DO NOT PERFORM RELOCATION.
4644 ;* IF ANY UNIBUS MEMORY PRESENT, DATI CYCLES TO THAT MEMORY
4645 ;* WILL BE CHECKED.
4646 ;*
4647 ;
4648 ; BGNTST
4649 ;
4650 ; DO FOR KMCR = #67,27 (18-BIT AND 22-BIT MODES, 32K OF PMI MEMORY)
4651 ; DO WITH MAPPING PAIRS 10 TO 36
4652 ; . DO DATI TO 0 USING SELCTED MAPPING PAIR
4653 ; . IF NO TIMEOUT THEN
4654 ; . . ERROR DISABLED REGISTER PAIRS DO RELOCATION
4655 ; . ENDDIF
4656 ; ENDDO
4657 ; IF ANY UNIBUS MEMORY PRESENT THEN
4658 ; . DO DATI WITH RELOCATION ENABLED
4659 ; . IF RELOCATED THEN
4660 ; . . ERROR
4661 ; . ENDDIF
4662 ; ENDDIF
4663 ;
4664 ; ENDTST
4665 ;
4666 ;-----
4667 ;*****
4668 ;*TEST 56            RELOCATION WITH UNIBUS MEMORY

```

```

017746 000004
4669
4670 017750 013737 177734 001162            MOV    KMCR,#TMP1            ; STORE KMCR
4671 017756 013701 177734            MOV    KMCR,R1            ; STORE KMCR
4672 017762 042701 177740            BIC    #177740,R1            ; LEAVE ONLY <4-0>
4673 017766 012737 170374 001160            MOV    #MAPL37,#TMP0            ; START WITH HIGHEST ADDRESS
4674 017774 032701 000037            BIT    #37,R1            ; ANY UNIBUS MEMRORY?
4675 020000 001404            BEQ    101$            ; IF NOT, BRANCH
4676 020002 162737 000004 001160 100$: SUB    #4,#TMP0            ; GET LOWER REGISTER PAIR
4677 020010 077104            SOB    R1,100$            ; LEAVE IN #TMP0 HIGHEST AVAILABLE PAIR
4678 020012 052737 000400 177730 101$: BIS    #BIT08,DCSR            ; ENABLE DIAGN. MODE
4679 020020 012737 000067 177734            MOV    #67,KMCR            ; 18-BIT, 32K PMI
4680 020026 042737 000400 177730            BIC    #BIT08,DCSR            ; DISABLE DIAGN. MODE
4681 ;
4682 ; SETUP MAPPING REGISTERS FOR INITIAL CONDITIONS
4683 ;
4684 020034 052737 000040 172516            BIS    #BIT5,MMR3            ; ENABLE UNIBUS MAPPING
4685 020042 012701 170240 1$: MOV    #MAPL10,R1            ; . PCINT TO MAPL10
4686 020046 005002            CLR    R2            ; . ADDRESS BITS <15-0> REG. 10
4687 020050 012703 000001            MOV    #1,R3            ; . ADDRESS FOR CR2<17-16>
4688 020054 004737 002450 2$: JSR    PC,IUBE            ; . . INITIALIZE THE UBE
4689 ;
4690 ;
4691 ; SET UP UBE AND UNIBUS MAP REGISTERS FOR A DATI CYCLE
4692 ;
4693 ;

```

T56 RELOCATION WITH UNIBUS MEMORY

```

4694 020060 010277 162070      MOV      R2,8BE1BA      ; . . POINT UBE TO FIRST MAP REGISTER
4695 020064 005077 162072      CLR      8BE1CR2      ; . . REGISTER
4696 020070 010377 162066      MOV      R3,8BE1CR2      ; . . LOAD ADDRESS <17-16>
4697 020074 005077 162050      CLR      8BE1D8      ; . . INITIALIZE THE DATA BUFFER
4698 020100 012777 177777 162044  MOV      #-1,8BE1CC      ; . . SET FOR 1 TRANSFER
4699 020106 005021          CLR      (R1)+      ; . . CLEAR LOW MAP REGISTER
4700 020110 005021          CLR      (R1)+      ; . . AND HIGH MAP TOO
4701 020112 012777 020200 162044  MOV      #104,8BE1VEC      ; . . LOAD INTERRUPT VECTOR
4702 020120 012777 000340 162040  MOV      #340,8BE1PSW      ;
4703
4704      ; GO DO THE TRANSFER
4705
4706
4707 020126 012777 002041 162022      MOV      #2041,8BE1CR1      ; . . SET UBE FOR NPR-DATI-1 XFER
4708 020134 032777 000200 162014 54:  BIT      #BIT7,8BE1CR1      ; . . IS THE TRANSFER COMPLETE
4709 020142 001774          BEQ      54      ; . . NO, THEN WAIT FOR IT TO BE COMPLETE
4710 020144 032777 000400 162010      BIT      #BIT08,8BE1CR2      ; . . NXM SET?
4711 020152 001002          BNE      64      ; . . IF SO, BRANCH
4712 020154 104022          ERROR      +22      ; . . NO NXM
4713 020156 000412          BR      114      ;
4714 020160 106427 000140          64:  MTPS      #140      ; . . LOWER PRIORITY
4715 020164 000240          NOP          ; . . WAIT FOR INTERRUPT
4716 020166 000240          NOP          ;
4717 020170 104022          ERROR      +22      ; . . NO INTERRUPT ON NXM
4718 020172 106427 000340          MTPS      #340      ; . . RAISE PRIORITY
4719 020176 000402          BR      114      ;
4720 020200 062706 000004          104:  ADD      #4,SP      ; . . ADJUST STACK
4721 020204 062702 020000          114:  ADD      #20000,R2      ; . . SELECT NEXT MAP PAIR REGISTER
4722 020210 103001          BCC      124      ; . . IF NO CARRY, BRANCH
4723 020212 005203          INC      R3      ; . . OTHERWISE INCREMENT ADDRESS <17-16>
4724 020214 023701 001160          124:  CMP      #TMP0,R1      ; . . ALL DONE?
4725 020220 101315          BHI      24      ; . . IF LESS THAN, BRANCH
4726
4727      ; CHECK IF DONE WITH 22-BIT MODE AND 18-BIT
4728
4729 020222 032737 000040 177734      BIT      #BIT05,KMCR      ; . . 22-BIT MODE DONE?
4730 020230 001412          BEQ      134      ; . . IF YES, BRANCH TO EXIT
4731 020232 052737 000400 177730      BIS      #BIT08,DCSR      ; . . ENABLE DIAGN. MODE
4732 020240 012737 000027 177734      MOV      #27,KMCR      ; . . 22-BIT, 32K PHI
4733 020246 042737 000400 177730      BIC      #BIT08,DCSR      ; . . DISABLE DIAGN. MODE
4734 020254 000672          BR      14      ; . . DO AGAIN IN 22-BIT MODE
4735
4736      ; IF ANY UNIBUS MEMORY PRESENT, DO RELOCATION THRU IT
4737
4738 020256 022737 170374 001160 134:  CMP      #MAPL37,#TMP0      ; ANY UNIBUS MEMORY?
4739 020264 001432          BEQ      2004      ; IF NO, EXIT
4740 020266 012777 140000 161660      MOV      #140000,8BE1BA      ; TRY TO DO DATI THRU
4741 020274 012777 000003 161660      MOV      #3,8BE1CR2      ; THRU MAP 36 REGISTER
4742 020302 005037 170370          CLR      MAPL36      ; CLEAR MAPPING REGISTER
4743 020306 005037 170372          CLR      MAPH36      ; PAIR
4744 020312 012737 123456 000000      MOV      #123456,#0      ; SET UP PATTERN AT 0
4745 020320 005077 161624          CLR      8BE1D8      ; CLEAR DATA REGISTER
4746 020324 012777 002071 161624      MOV      #2071,8BE1CR1      ; START DATI
4747 020332 105777 161620          204:  TSTB      8BE1CR1      ; WAIT TILL
4748 020336 100375          BPL      204      ; DONE
4749 020340 022777 123456 161602      CMP      #123456,8BE1D8      ; DATA CAME FROM #0?
4750 020346 001001          BNE      2004      ; IF NOT, BRANCH

```

T56 RELOCATION WITH UNIBUS MEMORY

4751	020350	104033				ERROR	.33		; RELOCATED UNIBUS MEMORY
4752	020352	052737	000400	177730	2004:	BIS	#BIT08,DCSR		; ENABLE DIAGN. MODE
4753	020360	013737	001162	177734		MOV	\$TMP1,KMCR		; RESTORE KMCR
4754	020366	042737	000400	177730		BIC	#BIT08,DCSR		; DISABLE DIAGN. MODE

MAIN MEMORY DISABLE THRU UBE

```

4756 .SBTTL MAIN MEMORY DISABLE THRU UBE
4757
4758 ;* THIS TEST CHECKS THAT A MAIN MEMORY RESPONSE IS DISABLED WHENEVER
4759 ;* THE APPROPRIATE BITS IN KMCR ARE SET. THE FIRST 32K ARE NOT GOING
4760 ;* TO BE CHECKED. THE DMA DATI CYCLES WILL BE DONE THAT SHOULD CAUSE
4761 ;* NXM BIT TO BE SET IN THE UBE REGISTER.
4762
4763 ; BGNTST
4764
4765 ;     SIZE THE MEMORY AVAILABLE
4766 ;     DO FOR ALL MEMORY PAGES
4767 ;     . LET KMCR DISABLE THAT PAGE IN MEMORY
4768 ;     DO DATI TO THAT LOCATION
4769 ;     IF NO TIMEOUT THEN
4770 ;     . ERROR DISABLED MEMORY DOES NOT TIMEOUT
4771 ;     . ENDF
4772 ;     ENDDO
4773
4774 ; ENDTST
4775
4776 -----
4777
4778 ;*****
4779 ;*TEST 57      MAIN MEMORY DISABLE THRU UBE
4780 ;*****
4781 TST57: SCOPE
4782
4783 ;
4784 ;
4785 ; TST      PHIS      ; ANY PHI MEMORY?
4786 BEQ      TST60      ; IF NONE, EXIT TEST
4787
4788 ;
4789 ; ENABLE MMU AND SIZE MEMORY
4790 ;
4791 ; JSR      PC,MAPPR   ; REMAP PROGRAM AREA
4792 ; BIS      @BIT00,MMR0 ; ENABLE MMU
4793 ; BIS      @BIT04,MMR3 ; ENABLE 22-BIT MODE
4794 ; MOV      PHIS,R2    ; STORE HIGHEST PAGE
4795 ; BIC      @BI*00,MMR0 ; DISABLE MMU
4796
4797 ;
4798 ; SETUP MAPPING REGISTERS AND KMCR
4799 ;
4800 ; BIS      @BIT5,@MMR3 ; ENABLE UNIBUS MAPPING
4801 ; MOV      KMCR,@TMP0  ; SAVE KMCR
4802 ; BIS      @BIT08,DCSR ; ENABLE DIAGNOSTIC MODE
4803 ; MOV      @67,KMCR    ; 18-BIT, >32K DISABLED
4804 ; BIC      @BIT08,DCSR ; DISABLE DIAGNOSTIC MODE
4805 ; CLR      MAPL00      ; CLEAR LOW MAP REGISTER
4806 ; MOV      @1,MAPH00   ; POINT TO 32K
4807 ; MOV      @20@,@BE1VEC ; . LOAD INTERRUPT VECTOR
4808 ; MOV      @340,@BE1PSW ;
4809 ; MOV      @2000,KIPAR6 ; INITIAL POINTER TO ABOVE 32K
4810
4811 ;
4812 ; DO DATI TO DISABLED MEMORY
4813 ;
4814 ;
4815 ; 10@: JSR      PC,IUBE ; INITIALISE UBE
4816 ; CLR      @BE1BA      ; . ADDRESS TO ACCESS THRU MAP 0
4817 ; MOV      @-1,@BE1CC  ; . ONE CYCLE
4818 ; MOV      @2041,@BE1CR1 ; . SET UBE FOR NPR DATI 1 XFER

```

T57 MAIN MEMORY DISABLE THRU UBE

```

4810 020554 032777 000200 161374 14: BIT #BIT7,BBE1CR1 ; . IS THE TRANSFER COMPLETE
4811 020562 001774 BEQ 14: ; . NO, THEN WAIT FOR IT TO BE COMPLETE
4812 020564 032777 000400 161370 BIT #BIT08,BBE1CR2 ; . NXM SET?
4813 020572 001002 BNE 15: ; . IF SO, BRANCH
4814 020574 104022 ERROR +22 ; . NO NXM
4815 020576 000412 BR 25: ;
4816 020600 106427 000140 15: MTPS #140 ; . LOWER PRIORITY
4817 020604 000240 NOP ; . WAIT FOR INTERRUPT
4818 020606 000240 NOP ;
4819 020610 104022 ERROR +22 ; . NO INTERRUPT ON NXM
4820 020612 106427 000340 MTPS #340 ; . RAISE PRIORITY
4821 020616 000402 BR 25: ; . BRANCH
4822 020620 062706 000004 20: ADD #4,SP ; . ADJUST STACK
4823 ;
4824 ; CHANGE PAGE ACCESSED AND CHECK END CONDITIONS
4825 ;
4826 020624 023702 172354 25: CMP KIPAR6,R2 ; . ALL PAGES DONE?
4827 020630 103032 BHIS 30: ; . IF YES, EXIT
4828 020632 062737 004000 170200 ADD #4000,MAPL00 ; . INCREMENT IN 1K
4829 020640 103002 BCC 26: ; . IF NO CARRY, BRANCH
4830 020642 005237 170202 INC MAPH00 ; . OTHERWISE INCR. HIGH MAP
4831 020646 032737 017777 170200 26: BIT #17777,MAPL00 ; . IS IT ON 4K BOUNDARY?
4832 020654 001017 BNE 27: ; . IF NOT, BRANCH
4833 020656 062737 000200 172354 ADD #200,KIPAR6 ; . INCREMENT COUNTER
4834 020664 052737 000400 177730 BIS #BIT08,DCSR ; . ENABLE DIAG. MODE
4835 020672 005337 177734 DEC KMCR ; . ACCESS NEXT HIGHER LOCATION
4836 020676 032737 000040 177734 BIT #BIT05,KMCR ; . ALL 128K DONE?
4837 020704 001404 BEQ 30: ; . IF SO, BRANCH
4838 020706 042737 000400 177730 BIC #BIT08,DCSR ; . DISABLE DIAG. MODE
4839 020714 000705 BR 10: ; . DO FOR THE NEXT PAGE
4840 020716 042737 000040 172516 30: BIC #BIT05,MPR3 ; . DISABLE MAPPING
4841 020724 052737 000400 177730 BIS #BIT08,DCSR ; . ENABLE DIAG. MODE
4842 020732 013737 001160 177734 MOV $TMP0,KMCR ; . RESTORE KMCR
4843 020740 042737 000400 177730 BIC #BIT08,DCSR ; . DISABLE DIAG. MODE
4844

```

TEST UNIBUS DEVICE DATOB CYCLE

```

4846 .SBTTL TEST - UNIBUS DEVICE DATOB CYCLE
4847
4848 ;* THIS TEST WILL SEE IF THE KTJ11-B MODULE CAN EXECUTE
4849 ;* A UNIBUS DEVICE DATOB .
4850 ;*
4851 ;
4852 ; BGNTST
4853 ;
4854 ;*
4855 ;* SET UP UBE FOR A DATOB CYCLE
4856 ;*
4857 ;       LET R1 := ADDRESS OF THE WRITE BUFFER
4858 ;       LET (R1) := 0 ;CLEAR OUT THE LOCATION
4859 ;       LET BE1DB := 77777 ;DATA PATTERN IS 77777
4860 ;       LET BE1CC := -1 ;SET FOR 1 DATA XFER
4861 ;       LET BE1BA := R1 ;ADDRESS IS FIRST LOCATION IN WRITE BUFFER
4862 ;       SET UBE FOR NPR-DATOB-1 XFER
4863 ;       SET OFF THE TRANSFER
4864 ;       WAIT FOR THE TRANSFER TO COMPLETE
4865 ;*
4866 ;* CHECK IF THE TRANSFER OCCURED CORRECTLY
4867 ;*
4868 ;       IF (R1) NEQ #377 THEN
4869 ;       . ERROR TRANSFER DID NOT EXECUTE CORRECTLY
4870 ;       ENDIF
4871 ;
4872 ; ENDTST
4873 ;
4874 ;-----
4875 ;*****
4876 ;*TEST 60      UNIBUS DEVICE DATOB CYCLE TEST
4877 ;*****
4878 ;TST60:  SCOPE
4879
4880 020746 000004
4881
4882 020750 004737 002450      JSR      PC,IUBE      ; INITIALIZE THE UBE
4883 020754 012701 001734      MOV      @WRTBUF,R1      ; GET POINTER TO WRITE BUFFER
4884 020760 005011              CLR      (R1)          ; CLEAR OUT THE LOCATION
4885 020762 012777 077777 161160 MOV      @77777, @BE1DB      ; SET DATA BUFFER UP
4886 020770 012777 177777 161154 MOV      @-1, @BE1CC      ; SET FOR 1 DATA TRANSFER
4887 020776 010177 161152      MOV      R1, @BE1BA      ; POINT UBE XFER TO WRITE BUFFER
4888
4889 ; DO THE TRANSFER
4890 ;
4891
4892 021002 012777 003441 161146 MOV      @3441, @BE1CR1      ; SET UP UBE FOR NPR-DATOB-1 XFER
4893 021010 032777 000200 161140 51: BIT      @BIT7, @BE1CR1      ; IS THE TRANSFER COMPLETE ?
4894 021016 001774              BEQ      51             ; NO, THEN WAIT FOR IT TO COMPLETE
4895
4896 ; CHECK THE TRANSFER
4897 ;
4898
4899 021020 021127 000377      CMP      (R1), #377      ; WAS THE TRANSFER CORRECT ?
4900 021024 001401              BEQ      TST61          ; YES GO TO THE NEXT TEST
4901 021026 104033              ERROR      +33          ; ERROR TRANSFER WAS INCORRECT
4902
4903
4904
4905
4906
4907

```

TEST UNIBUS DEVICE DATIP CYCLE

```

4899 .SBTTL TEST UNIBUS DEVICE DATIP CYCLE
4900
4901 ; THIS TEST WILL SEE IF THE KTJ11-B MODULE CAN EXECUTE
4902 ; A UNIBUS DEVICE DATIP CYCLE.
4903 ;
4904 ;
4905 ; BGNST
4906 ;
4907 ; INITIALIZE WRITE BUFFER FOR DATIP CYCLE
4908 ;
4909 ; LET R1 := POINTER TO WRITE BUFFER
4910 ; DO FOR R2 := 1 TO 8
4911 ; . LET (R1). := #52525
4912 ; ENDDO
4913 ;
4914 ; SET UP UBE TO DO 8 DATIP'S TO WRITE BUFFER
4915 ;
4916 ; LET BE1BA := ADDRESS OF WRITE BUFFER
4917 ; LET BE1CC := -8. ; SET FOR 8 TRANSFERS
4918 ; SET UBE FOR NPR-DATIP-1 XFER
4919 ; SET OFF THE TRANSFER
4920 ; WAIT FOR TRANSFER TO BE COMPLETE
4921 ;
4922 ; CHECK IF DATIP WAS EXECUTED CORRECTLY
4923 ;
4924 ; LET R1 := ADDRESS OF WRITE BUFFER
4925 ; DO FOR R2 := 1 TO 8
4926 ; . IF (R1). NEQ 125252 THEN
4927 ; . . ERROR TRANSFER DID NOT EXECUTE CORRECTLY
4928 ; . ENDF
4929 ; ENDDO
4930 ;
4931 ; ENDTST
4932 ;
4933 ;-----
4934 ;*****
4935 ;TEST 61 UNIBUS DEVICE DATIP CYCLE TEST
4936 ;*****
4937 TST61: SCOPE
4938
4939 ;
4940 ; INITIALIZE WRITE BUFFER FOR DATIP CYCLE.
4941 ;
4942 ;
4943 021036 012701 001734 MOV @WRTBUF,R1 ; GET POINTER TO WRITE BUFFER
4944 021042 012702 000010 MOV @6.,R2 ; SET UP LOOP COUNTER
4945 021046 012721 052525 51: MOV #52525,(R1). ; . INITIALIZE WRITE BUFFER LOCATION
4946 021052 077203 SOB R2,51 ; . HAVE WE DONE IT 8 TIMES ?
4947 ;
4948 ;
4949 ; SET UP UBE TO 8 DATIP'S TO WRITE BUFFER
4950 ;
4951 ;
4952 021054 012777 001734 161072 MOV @WRTBUF,@BE1BA ; SET UP UBE TO WRITE BUFFER

```

T61 UNIBUS DEVICE DATIP CYCLE TEST

```

4953 021062 005077 161074          CLR    BBE1CR2          ;
4954 021066 012777 177770 161056    MOV    0-8.,BBE1CC      ; SET UP FOR 8 TRANSFERS
4955 021074 012777 002441 161054    MOV    02441,BBE1CR1    ; SET UP UBE FOR NPR-DATIP-1 XFER
4956 021102 032777 000200 161046 104: BIT    0BIT7,BBE1CR1    ; IS THE TRANSFER COMPLETE ?
4957 021110 001774          BEQ     104          ; NO, THEN WAIT FOR IT TO BE COMPLETE
4958
4959
4960          ; CHECK IF DATIP WAS EXECUTED CORRECTLY
4961          ;
4962
4963 021112 012701 001734          MOV    WRTBUF,R1          ; GET POINTER TO WRITE BUFFER
4964 021116 012702 000010          MOV    00.,R2            ; SET UP LOOP COUNTER
4965 021122 022127 125252          154: CMP    (R1),0125252    ; . WAS THE TRANSFER CORRECT ?
4966 021126 001401          BEQ     204          ; . YES, THEN GO SEE IF WE ARE DONE CHECKING
4967 021130 104033          ERROR   .33            ; . NO, THEN ERROR IN THE TRANSFER
4968 021132 077205          204: SOB     R2,154          ; . HAVE WE CHECKED ALL THE XFERS
4969
4970

```

TEST - UNIBUS DEVICE I/O PAGE READ CYCLE

```

4972 .SBTTL TEST - UNIBUS DEVICE I/O PAGE READ CYCLE
4973
4974 ;* THIS TEST WILL SEE IF THE KTJ11-B MODULE CAN EXECUTE
4975 ;* A UNIBUS DEVICE I/O PAGE READ CYCLE. THIS CONSISTS OF
4976 ;* THREE DIFFERENT TYPES OF READS :
4977 ;*
4978 ;*     1. UNIBUS DEVICE PHI I/O PAGE READ CYCLE
4979 ;*     2. UNIBUS DEVICE BOOT ROM READ CYCLE
4980 ;*     3. UNIBUS DEVICE MAP REGISTER READ CYCLE.
4981 ;*
4982 ;* IT ALSO WILL CHECK THAT A UNIBUS DEVICE READ CYCLE NOT
4983 ;* OF THE ABOVE THREE TYPES SHOULD CAUSE A TIMEOUT.
4984 ;*
4985 ;
4986 ; BGNTST
4987 ;
4988 ;* DO A UNIBUS DEVICE PHI I/O PAGE READ CYCLE
4989 ;*
4990 ;     LET BE1BA := ADDRESS OF CSR OF MEMORY BOARD
4991 ;     LET BE1CC := -1 ;SET FOR 1 TRANSFER
4992 ;     SET UBE FOR A NPR-DATI-1 DATA XFER
4993 ;     SET OFF THE TRANSFER
4994 ;     WAIT FOR THE TRANSFER TO BE COMPLETE
4995 ;     IF BE1DB NEQ CONTENTS OF MEMORY CSR THEN
4996 ;     . ERROR UNIBUS DEVICE PHI I/O PAGE READ CYCLE DID NOT EXECUTE CORRECTLY
4997 ;     ENDIF
4998 ;*
4999 ;* DO A UNIBUS DEVICE BOOT ROM READ CYCLE
5000 ;*
5001 ;     LET BE1BA := 773000
5002 ;     LET BE1CC := -1 ;SET FOR 1 TRANSFER
5003 ;     SET UBE FOR A NPR-DATI-1 DATA XFER
5004 ;     SET OFF THE TRANSFER
5005 ;     WAIT FOR THE TRANSFER TO BE COMPLETE
5006 ;     IF BE1DB NEQ 00173000 THEN
5007 ;     . ERROR UNIBUS DEVICE BOOT ROM READ CYCLE DID NOT EXECUTE CORRECTLY
5008 ;     ENDIF
5009 ;*
5010 ;* DO A UNIBUS DEVICE MAP REGISTER READ CYCLE
5011 ;*
5012 ;     LET BE1BA := 770200 ;UNIBUS MAP REGISTER #0
5013 ;     LET BE1CC := -1 ;SET FOR 1 TRANSFER
5014 ;     SET UBE FOR A NPR-DATI-1 DATA XFER
5015 ;     SET OFF THE TRANSFER
5016 ;     WAIT FOR THE TRANSFER TO BE COMPLETE
5017 ;     IF BE1DB NEQ 00770200 THEN
5018 ;     . ERROR UNIBUS DEVICE MAP REGISTER READ CYCLE DID NOT EXECUTE CORRECTLY
5019 ;     ENDIF
5020 ;*
5021 ;* DO AN ILLEGAL UNIBUS DEVICE I/O PAGE READ CYCLE
5022 ;*
5023 ;     LET BE1BA := 777730,777732,777734 ;DCSR,DDR,KMCR ADDRESS
5024 ;     LET BE1CC := -1 ;1 DATA TRANSFER
5025 ;     SET UBE FOR NPR-DATI-1 DATA XFER
5026 ;     SET OFF THE TRANSFER
5027 ;     WAIT FOR THE TRANSFER TO BE COMPLETE
5028 ;     IF TRANSFER DID NOT TIMEOUT THEN

```

TEST - UNIBUS DEVICE I/O PAGE READ CYCLE

```

5029      ;      . ERROR AN ILLEGAL UNIBUS DEVICE CYCLE HAS OCCURED
5030      ;      ENDIF
5031      ;
5032      ;      ENDTST
5033      ;
5034      ;-----
5035      ;*****
5036      ;*TEST 62      UNIBUS DEVICE I/O PAGE READ CYCLE
5037      ;*****
5038      ;TST62:  SCOPE
5039      ;
5040      ;      JSR      PC,IUBE      ; INITIALIZE THE UBE
5041      ;
5042      ;      DO A UNIBUS DEVICE PMI I/O PAGE CYCLE
5043      ;
5044      ;      MOV      MCSR,8BE1BA      ; GET ADDRESS OF MEMORY CSR ADDRESS
5045      ;      MOV      #3,8BE1CR2      ; SET THE UPPER TWO ADDRESS BITS
5046      ;      MOV      #-1,8BE1CC      ; SET FOR 1 TRANSFER
5047      ;      MOV      #2041,8BE1CR1      ; SET UBE FOR NPR-DATI-1 XFER
5048      ;      BIT      #BIT7,8BE1CR1      ; ARE WE DONE THE TRANSFER ?
5049      ;      BEQ      54      ; NO, THEN FOR WAIT FOR IT TO
5050      ;
5051      ;      CHECK OUT THE TRANSFER
5052      ;
5053      ;      CMP      BMCSR,8BE1DB      ; WAS THE TRANSFER CORRECT
5054      ;      BEQ      104      ; YES, THEN GO DO BOOT ROM READ
5055      ;      ERROR      +33      ; NO, THEN ERROR IN READ CYCLE
5056      ;
5057      ;      DO A UNIBUS DEVICE BOOT ROM READ CYCLE
5058      ;
5059      ;      MOV      #173000,8BE1BA      ; GET ADDRESS OF OF THE BOOT ROM
5060      ;      MOV      #3,8BE1CR2      ; SET THE UPPER TWO ADDRESS BITS
5061      ;      MOV      #-1,8BE1CC      ; SET FOR 1 TRANSFER
5062      ;      MOV      #2041,8BE1CR1      ; SET UBE FOR NPR-DATI-1 XFER
5063      ;      BIT      #BIT7,8BE1CR1      ; ARE WE DONE THE TRANSFER ?
5064      ;      BEQ      154      ; NO, THEN FOR WAIT FOR IT TO
5065      ;
5066      ;      CHECK OUT THE TRANSFER
5067      ;
5068      ;      BIS      #BIT07,BCSR      ; SELECT RIGHT ROM
5069      ;      CMP      #173000,8BE1DB      ; WAS THE TRANSFER CORRECT
5070      ;      BEQ      204      ; YES, THEN GO DO UNIBUS DEVICE MAP REGISTER READ
5071      ;      ERROR      +33      ; NO, THEN ERROR IN READ CYCLE
5072      ;
5073      ;      DO A UNIBUS DEVICE MAP REGISTER READ
5074      ;
5075      ;      MOV      #170200,8BE1BA      ; GET ADDRESS OF THE MAP REGISTER
5076      ;      MOV      #3,8BE1CR2      ; SET THE UPPER TWO ADDRESS BITS
5077      ;      MOV      #-1,8BE1CC      ; SET FOR 1 TRANSFER
5078      ;      MOV      #2041,8BE1CR1      ; SET UBE FOR NPR-DATI-1 XFER
5079      ;      BIT      #BIT7,8BE1CR1      ; ARE WE DONE THE TRANSFER ?
5080      ;      BEQ      254      ; NO, THEN FOR WAIT FOR IT TO
5081      ;
5082      ;      CHECK OUT THE TRANSFER

```

T62 UNIBUS DEVICE I/O PAGE READ CYCLE

```

5083 021334 023777 170200 160606      CMP      #170200,8BE1D8 ; WAS THE TRANSFER CORRECT
5084 021342 001401                      BEQ      304 ; YES, THEN GO DO ILLEGAL UNIBUS DEVICE I/O READ
5085 021344 104033                      ERROR   +33 ; NO, THEN ERROR IN READ CYCLE
5086
5087 ; DO AN ILLEGAL UNIBUS DEVICE I/O PAGE READ
5088
5089 021346 012777 021450 160610 304:  MOV      #404,8BE1VEC ; SET UP INTERRUPT VECTOR
5090 021354 012777 000340 160604      MOV      #340,8BE1PSW ;
5091 021362 012701 000003              MOV      #3,R1 ; DO FOR 3 REGISTERS
5092 021366 012702 177730              MOV      #177730,R2 ; STARTING WITH DCSR (THEN DDR, KMCR)
5093 021372 010277 160556 314:  MOV      R2,8BE1BA ; GET ADDRESS OF THE DIAGNOSTIC REGISTER
5094 021376 012777 000003 160556      MOV      #3,8BE1CR2 ; SET THE UPPER TWO ADDRESS BITS
5095 021404 012777 177777 160540      MOV      #-1,8BE1CC ; SET FOR 1 TRANSFER
5096 021412 012777 002041 160536      MOV      #2041,8BE1CR1 ; SET UBE FOR NPR-DATI-1 XFER
5097 021420 106427 000140              MTPS     #140 ; LOWER PRIORITY
5098 021424 032777 000200 160524 354:  BIT      #BIT7,8BE1CR1 ; ARE WE DONE THE TRANSFER ?
5099 021432 001774                      BEQ      354 ; NO, THEN FOR WAIT FOR IT TO
5100 021434 000240                      NOP ; WAIT FOR INTERRUPT
5101 021436 000240                      NOP ;
5102 021440 104033                      ERROR   +33 ; ERROR KTJ11 RESPONDED TO AN ILLEGAL ADDRESS
5103 021442 106427 000340              MTPS     #340 ; RAISE PRIORITY BACK
5104 021446 000407                      BR       454 ;
5105 021450 062706 000004 404:  ADD      #4,SP ; CLEAN UP THE STACK AND THEN GO DO NEXT TEST
5106 021454 032777 000400 160500      BIT      -BIT08,8BE1CR2 ; NXM SET?
5107 021462 001001                      BNE      454 ; IF SET, BRANCH
5108 021464 104033                      ERROR   +33 ; NXM NOT SET ON ILLEGAL REFERENCE
5109 021466 005722 454:  TST      (R2) ; GET ADDRESS OF NEXT REGISTER
5110 021470 004737 002450              JSR      PC,IUBE ; CLEAN UP UBE
5111 021474 077142                      SOB      R1,314 ; DO FOR ALL 3 REGISTERS
5112

```

TEST - UNIBUS DEVICE I/O PAGE WRITE CYCLE

```

5114      .SBTTL TEST - UNIBUS DEVICE I/O PAGE WRITE CYCLE
5115
5116      ;* THIS TEST WILL SEE IF THE KTJ11-B MODULE CAN EXECUTE
5117      ;* A UNIBUS DEVICE I/O PAGE WRITE CYCLE. THIS CONSISTS OF
5118      ;* TWO DIFFERENT TYPES OF WRITES :
5119      ;*
5120      ;*     1. UNIBUS DEVICE PMI I/O PAGE WRITE CYCLE
5121      ;*     2. UNIBUS DEVICE MAP REGISTER WRITE CYCLE.
5122
5123      ;* IT ALSO WILL CHECK THAT A UNIBUS DEVICE WRITE CYCLE NOT
5124      ;* OF THE ABOVE TWO TYPES SHOULD CAUSE A TIMEOUT.
5125      ;*
5126      ;
5127      BGNTEST
5128      ;
5129      ;*
5130      ;* DO A UNIBUS DEVICE PMI I/O PAGE WRITE CYCLE
5131      ;*
5132      ;       LET BE1DB := 52525
5133      ;       LET BE1BA := ADDRESS OF CSR OF MEMORY BOARD
5134      ;       LET BE1CC := -1                               ;SET FOR 1 TRANSFER
5135      ;       SET UBE FOR A NPR-DATO-1 DATA XFER
5136      ;       SET OFF THE TRANSFER
5137      ;       WAIT FOR THE TRANSFER TO BE COMPLETE
5138      ;       IF CONTENTS OF MEMORY CSR NEQ 52525 THEN
5139      ;       . ERROR UNIBUS DEVICE PMI I/O PAGE WRITE CYCLE DID NOT EXECUTE CORRECTLY
5140      ;       ENDIF
5141      ;*
5142      ;* DO A UNIBUS DEVICE MAP REGISTER WRITE CYCLE
5143      ;*
5144      ;       LET BE1DB := 52525                               ;DATA PATTERN
5145      ;       LET BE1BA := 770200                             ;UNIBUS MAP REGISTER #0
5146      ;       LET BE1CC := -1                               ;SET FOR 1 TRANSFER
5147      ;       SET UBE FOR A NPR-DATO-1 DATA XFER
5148      ;       SET OFF THE TRANSFER
5149      ;       WAIT FOR THE TRANSFER TO BE COMPLETE
5150      ;       IF MAP REGISTER 0 NEQ #52525 THEN
5151      ;       . ERROR UNIBUS DEVICE MAP REGISTER READ CYCLE DID NOT EXECUTE CORRECTLY
5152      ;       ENDIF
5153      ;*
5154      ;* DO A UNIBUS DEVICE TO BOOT ROM WRITE CYCLE
5155      ;*
5156      ;       LET BE1BA := 773000
5157      ;       LET BE1CC := -1                               ;SET FOR 1 TRANSFER
5158      ;       SET UBE FOR A NPR-DATO-1 DATA XFER
5159      ;       SET OFF THE TRANSFER
5160      ;       WAIT FOR THE TRANSFER TO BE COMPLETE
5161      ;       IF NO TIMEOUT THEN
5162      ;       . ERROR UNIBUS DEVICE BOOT ROM WRITE CYCLE DID NOT EXECUTE CORRECTLY
5163      ;       ENDIF
5164      ;*
5165      ;* DO AN ILLEGAL UNIBUS DEVICE I/O PAGE WRITE CYCLE
5166      ;*
5167      ;       LET BE1DB := 52525                               ;DATA PATTERN
5168      ;       LET BE1BA := 777730,777732,777734             ;DCSR, DDR, KMCR ADDRESSES
5169      ;       LET BE1CC := -1                               ;1 DATA TRANSFER
5170      ;       SET UBE FOR NPR-DATO-1 DATA XFER

```

TEST - UNIBUS DEVICE I/O PAGE WRITE CYCLE

```

5171      ; SET OFF THE TRANSFER
5172      ; WAIT FOR THE TRANSFER TO BE COMPLETE
5173      ; IF TRANSFER DID NOT TIMEOUT THEN
5174      ; . ERROR AN ILLEGAL UNIBUS DEVICE CYCLE HAS OCCURED
5175      ; ENDF
5176      ; DISABLE UNIBUS MAPPING
5177      ;
5178      ; ENDTST
5179      ;
5180      ; -----
5181      ; *****
5182      ; *TEST 63      UNIBUS DEVICE I/O PAGE WRITE CYCLE
5183      ; *****
5184      ; TST63: SCOPE
5185      ;
5186      ; JSR      PC.TUBE      ; INITIALIZE THE UBE
5187      ; DO A UNIBUS DEVICE PHI I/O PAGE CYCLE
5188      ;
5189      ; MOV      @MCSR,@TMP0      ; STORE MEMORY CSR
5190      ; MOV      @MCSR,@BE18A      ; GET ADDRESS OF MEMORY CSR
5191      ; MOV      @3,@BE1CR2      ; SET UPPER TWO ADDRESS BITS
5192      ; MOV      @BIT14,@BE1DB      ; GET WRITE PATTERN
5193      ; MOV      @-1,@BE1CC      ; SET FOR 1 XFER
5194      ; MOV      @3041,@BE1CR1      ; SET UBE FOR NPR-DATO-1 XFER
5195      ; BIT      @BIT7,@BE1CR1      ; IS THE TRANSFER COMPLETE ?
5196      ; BEQ      51      ; NO, THEN WAIT FOR IT TO COMPLETE
5197      ;
5198      ; CHECK THE TRANSFER
5199      ;
5200      ; BIT      @BIT14,@MCSR      ; WAS THE TRANSFER CORRECT ?
5201      ; BNE      101      ; YES, THEN GO DO A MAP REGISTER WRITE
5202      ; ERROR      +33      ; NO, THEN ERROR IN THE TRANSFER
5203      ;
5204      ; DO A UNIBUS DEVICE MAP REGISTER WRITE CYCLE
5205      ;
5206      ; 101: MOV      @TMP0,@MCSR      ; RESTORE MEMORY CSR
5207      ; CLR      MAPL00      ; CLEAR MAP REGISTER PAIR
5208      ; CLR      MAPH00      ;
5209      ; MOV      @MAPL00,@BE18A      ; GET ADDRESS OF MAP REGISTER
5210      ; MOV      @3,@BE1CR2      ; SET UPPER TWO ADDRESS BITS
5211      ; MOV      @52525,@BE1DB      ; GET WRITE PATTERN
5212      ; MOV      @-2,@BE1CC      ; SET FOR 2 XFER
5213      ; MOV      @3041,@BE1CR1      ; SET UBE FOR NPR-DATO-1 XFER
5214      ; BIT      @BIT7,@BE1CR1      ; IS THE TRANSFER COMPLETE ?
5215      ; BEQ      151      ; NO, THEN WAIT FOR IT TO COMPLETE
5216      ;
5217      ; CHECK THE TRANSFER
5218      ;
5219      ; CMP      MAPL00,@52524      ; WAS THE TRANSFER CORRECT ?
5220      ; BEQ      161      ; YES, THEN GO DO A MAP REGISTER WRITE
5221      ; ERROR      +33      ; NO, THEN ERROR IN THE TRANSFER
5222      ; 161: CMP      @25,MAPH00      ; SECOND WORD OK TOO?
5223      ; BEQ      171      ; IF SO, BRANCH
5224      ; ERROR      +33      ; NO, THEN ERROR IN THE TRANSFER

```

T63 UNIBUS DEVICE I/O PAGE WRITE CYCLE

```

5225 ; DO A WRITE CYCLE TO BOOT ROM THAT SHOULD CAUSE A TIMEOUT
5226 ;
5227 021702 012777 173000 160244 17: MOV #173000,8BE1BA ; ADDRESS TO WRITE TO
5228 021710 012777 177777 160234 MOV #-1,8BE1CC ; 1 CYCLE
5229 021716 012777 021764 160240 MOV #19,8BE1VEC ; POINT INTERRUPT VECTOR TO PROGRAM
5230 021724 012777 000340 160234 MOV #340,8BE1PSW ; AT PRIORITY 7
5231 021732 012777 003041 160216 MOV #3041,8BE1CR1 ; 1 DATO-NPR
5232 021740 106427 000140 MTPS #140 ; LOWER PRIORITY
5233 021744 105777 160206 18: TSTB 8BE1CR1 ; DONE?
5234 021750 100375 BPL 18 ; WAIT TILL DONE
5235 021752 000240 NOP
5236 021754 104033 ERROR +33 ; DIDN'T TIMEOUT ON BOOT ROM WRITE
5237 021756 106427 000340 MTPS #340 ; RAISE PRIORITY
5238 021762 000407 BR 20 ;
5239 021764 062706 000004 19: ADD #4,SP ; ADJUST STACK
5240 021770 032777 000400 160164 BIT #BIT08,8BE1CR2 ; NXM SET?
5241 021776 001001 BNE 20 ; IF SET, BRANCH
5242 022000 104033 ERROR +33 ; NXM NOT SET ON BOOT ROM WRITE
5243 ;
5244 ; DO AN ILLEGAL UNIBUS DEVICE I/O PAGE WRITE CYCLE
5245 ;
5246 022002 004737 002450 20: JSR PC,IUBE ; CLEAN UP UBE
5247 022006 012701 000003 MOV #3,R1 ; DO FOR DCSR,DDR,KMCR
5248 022012 012702 177730 MOV #177730,R2 ; START WITH DCSR
5249 022016 012777 022124 160140 MOV #30,8BE1VEC ; SET UP TIMEOUT VECTOR
5250 022024 012777 000340 160134 MOV #340,8BE1PSW
5251 022032 010277 160116 21: MOV R2,8BE1BA ; GET ADDRESS OF THE DIAGN. REGISTER
5252 022036 012777 000003 160116 MOV #3,8BE1CR2 ; SET UPPER TWO ADDRESS BITS
5253 022044 012777 052525 160076 MOV #52525,8BE1DB ; GET WRITE PATTERN
5254 022052 012777 177777 160072 MOV #-1,8BE1CC ; SET FOR 1 XFER
5255 022060 052737 000040 172516 BIS #BIT5,MMR3 ; ENABLE UNIBUS MAPPING
5256 022066 012777 003041 160062 MOV #3041,8BE1CR1 ; SET UBE FOR NPR-DATO-1 XFER
5257 022074 005037 177776 CLR PSW ; LOWER PRIORITY
5258 022100 032777 000200 160050 25: BIT #BIT7,8BE1CR1 ; IS THE TRANSFER COMPLETE ?
5259 022106 001774 BEQ 25 ; NO, THEN WAIT FOR IT TO COMPLETE
5260 022110 000240 NOP ; WAIT FOR INTERRUPT
5261 022112 000240 NOP
5262 022114 104033 ERROR +33 ; ERROR - AN ILLEGAL CYCLE DID NOT TIMEOUT
5263 022116 106427 000340 MTPS #340 ; RAISE PRIORITY BACK
5264 022122 000407 BR 35 ;
5265 022124 062706 000004 30: ADD #4,SP ; ADJUST THE STACK AND GO TO NEXT TEST
5266 022130 032777 000400 160024 BIT #BIT08,8BE1CR2 ; NXM SET?
5267 022136 001001 BNE 35 ; IF SET, BRANCH
5268 022140 104033 ERROR +33 ; NXM NOT SET ON ILLEGAL REFERENCE
5269 022142 005722 35: TST (R2) ; GET NEXT DIAGNOSTIC REGISTER
5270 022144 004737 002450 JSR PC,IUBE ; INITIALISE UBE
5271 022150 077150 SOB R1,21 ; DO FOR ALL OF THEM

```

TEST - MAPPING REGISTERS TEST USING UBE

```

5273 .SBTTL TEST - MAPPING REGISTERS TEST USING UBE
5274
5275 ; * THIS TEST WRITES DIFFERENT PATTERNS TO MAPPING REGISTERS USING
5276 ; * DATI AND DATO CYCLES THRU UBE.
5277
5278 ;   BGNTST
5279
5280 ;       ENABLE UNIBUS MAPPING
5281 ;       WRITE ALL MAPPING REGISTERS WITH "10" PATTERN
5282 ;       READ ALL REGISTERS BACK CLEARING THE ONE JUST READ
5283 ;       IF ANY REGISTER DOES NOT HAVE THE PATTERN
5284 ;       . ERROR IN WRITING TO MAP REGISTERS
5285 ;       ENDF
5286
5287 ;   ENDTST
5288
5289
5290
5291
;*****
; *TEST 64      MAPPING REGISTERS TEST USING UBE
;*****
TST64:  SCOPE
        JSR      PC,IUBE          ; CLEAN UP UBE
;
; DO DATO TO ALL 100 REGISTERS WITH 125252 AS PATTERN
;
        MOV      #MAPLOO,8BE18A  ; THE STARTING ADDRESS
        MOV      #0-100,8BE1CC  ; 80 WORDS TO WRITE TO
        MOV      #125252,8BE10B ; THE PATTERN TO BE WRITTEN
        MOV      #3,8BE1CR2     ; ADDRESS <17-16>
        MOV      #3041,8BE1CR1  ; GO DO DATO'S
14:     TSTB     8BE1CR1         ; DONE BIT SET?
        BPL      14             ; WAIT
;
; CHECK THAT ALL REGISTERS WRITTEN OK
;
        MOV      #MAPLOO,R1      ; START WITH MAP REGISTER 1
        MOV      R1,8BE18A       ; STARTING ADDRESS
24:     MOV      #0-1,8BE1CC     ; . DO JUST 1 CYCLE
        MOV      #2041,8BE1CR1   ; . DATI FROM LOW MAP REGISTER
34:     TSTB     8BE1CR1         ; . DONE BIT SET?
        BFL      34             ; . IF NOT, WAIT
        CMP      #125252,8BE10B ; . READ OK?
        BEQ      44             ; . IF SO, BRANCH
        ERROR    +33            ; . ERROR IN WRITING AND READING MAP REGISTERS
        MOV      #0-1,8BE1CC     ; . NOW DO HIGH MAP REGISTER
        MOV      #2041,8BE1CR1   ; . DATI FROM HIGH REGISTER
54:     TSTB     8BE1CR1         ; . DONE BIT SET?
        BPL      54             ; . IF NOT, WAIT
        CMP      #52,8BE10B     ; . READ OK?
        BEQ      64             ; . IF SO, BRANCH
        ERROR    +33            ; . ERROR IN WRITING AND READING MAP REGISTERS
64:     CLR      (R1)+           ; . CLEAR REGISTER JUST WRITTEN
        CLR      (R1)+
        CMP      #MAPH37,R1     ; . ALL DONE?
        BHI      24             ; . IF NOT, BRANCH

```

5292	022152	000004		
5293	022154	004737	002450	
5294				
5295				
5296	022160	012777	170200	157766
5297	022166	012777	177700	157756
5298	022174	012777	125252	157746
5299	022202	012777	000003	157752
5300	022210	012777	003041	157740
5301	022216	105777	157734	
5302	022222	100375		
5303				
5304				
5305				
5306	022224	012701	170200	
5307	022230	010177	157720	
5308	022234	012777	177777	157710
5309	022242	012777	002041	157706
5310	022250	105777	157702	
5311	022254	100375		
5312	022256	022777	125252	157664
5313	022264	001401		
5314	022266	104033		
5315	022270	012777	177777	157654
5316	022276	012777	002041	157652
5317	022304	105777	157646	
5318	022310	100375		
5319	022312	022777	000052	157630
5320	022320	001401		
5321	022322	104033		
5322	022324	005021		
5323	022326	005021		
5324	022330	022701	170376	
5325	022334	101337		

TEST - UNIBUS DEVICE DATI CYCLE WITH CACHE ENABLED

```

5327      .SBTTL TEST - UNIBUS DEVICE DATI CYCLE WITH CACHE ENABLED
5328
5329      ; * THIS TEST WILL SEE IF THE CACHE OPERATES CORRECTLY
5330      ; * WHEN A UNIBUS DEVICE READ CYCLE IS DONE
5331      ; *
5332      ;
5333      ; BGNTST
5334      ;
5335      ;     ENABLE THE CACHE
5336      ;     ENABLE UNIBUS MAPPING
5337      ; *
5338      ; * CHECK INITIAL SETTING OF THE CACHE VALID BITS
5339      ; *
5340      ;     SELECT THE CACHE VALID BITS
5341      ;     IF KMCR <15-9> NEQ +B00000000 THEN
5342      ;         . ERROR IN THE CACHE
5343      ;     ENDIF
5344      ; *
5345      ; * CHECK INITIAL SETTING OF THE CACHE AVAILABILITY BITS
5346      ; *
5347      ;     SELECT THE CACHE AVAILABLE SET BITS
5348      ;     IF KMCR <14-9> NEQ +B1111111 THEN
5349      ;         . ERROR IN THE CACHE
5350      ;     ENDIF
5351      ; *
5352      ; * GO DO TWO CONSECUTIVE READS
5353      ; * THE FIRST WILL ALLOCATE SET A
5354      ; * THE SECOND WILL CAUSE A CACHE HIT IN SET A
5355      ; *
5356      ;     LET MAP REGISTER 1 POINT TO LOCATION 2000
5357      ;     LET BE1BA := 20000                                ;OCTAL BOUNDARY READ
5358      ;     LET BE1CR2 := 0                                    ;CLEAR UPPER 2 ADDRESS BITS
5359      ;     LET BE1CC := -2                                    ;SET FOR TWO TRANSFERS
5360      ;     SET UBE FOR NPR-DATI-1 XFER
5361      ;     SET OFF THE TRANSFER
5362      ;     WAIT FOR THE TRANSFER TO COMPLETE
5363      ;     DISABLE UNIBUS MAPPING
5364      ;     TURN OFF THE CACHE
5365      ;     SELECT THE CACHE VALID BITS
5366      ; *
5367      ; * CHECK IF THE CACHE OPERATED CORRECTLY
5368      ; *
5369      ;     IF KMCR <15-9> NEQ +B1001000 THEN
5370      ;         . ERROR CACHE DID NOT GET UPDATED CORRECTLY
5371      ;     ENDIF
5372      ;     SELECT THE CACHE AVAILBILITY BITS
5373      ;     IF KMCR <15-9> NEQ +B1000111 THEN
5374      ;         . ERROR CACHE DID NOT OPERATE CORRECTLY
5375      ;     ENDIF
5376      ;
5377      ; ENDTST
5378      ;
5379      ; -----
5380      ; *****
5381      ; *TEST 65      UNIBUS DEVICE CYCLE WITH CACHE ENABLED
5382      ; *****

```

T65 UNIBUS DEVICE CYCLE WITH CACHE ENABLED

```

022336 000004          TST65: SCOPE
5382
5383 022340 004737 002450          JSR      PC,IUBE          ; INITIALIZE THE UBE
5384 022344 052737 000100 177734    BIS      @BIT6,KMCR      ; ENABLE THE DMA CACHE
5385 022352 032737 000100 177734    BIT      @BIT6,KMCR      ; IS THE CACHE PRESENT ?
5386 022360 001002          BNE      14          ; IF YES, CONTINUE WITH TEST
5387 022362 000137 022766          JMP      $EOP          ; OTHERWISE, DO END OF PASS
5388 022366 012737 160000 170374 14:  MOV      @160000,MAPL37    ; INITIALIZE MAP REGISTERS
5389 022374 012737 000077 170376    MOV      @77,MAPH37
5390 022402 005037 170200          CLR      MAPL00
5391 022406 005037 170202          CLR      MAPH00
5392 022412 052737 000040 172516    BIS      @BIT5,MPIR3    ; ENABLE UNIBUS MAPPING
5393 022420 052737 000400 177730    BIS      @BIT08,DCSR    ; ENABLE DIAGNOSTIC MODE
5394 022426 042737 000400 177734    BIC      @BIT08,KMCR    ; MAKE SURE THAT VALID BITS SET
5395 022434 042737 000400 177730    BIC      @BIT08,DCSR    ; DISABLE DIAGNOSTIC MODE
5396 022442
5397
5398          ; GO DO TWO CONSECUTIVE READS STARTING AT A OCTAL BOUNDARY.
5399          ; THE FIRST READ WILL ALLOCATE CACHE SET A. THE SECOND READ
5400          ; WILL CAUSE A CACHE HIT IN SET A
5401
5402 022442 012737 000200 170204 104:  MOV      @200,MAPL01    ; POINT MAP REGISTER TO LOCATION 200
5403 022450 005037 170206          CLR      MAPH01
5404 022454 012777 020000 157472    MOV      @20000,@BE18A  ; SET TRANSFER ADDRESS TO OCTAL BOUNDARY
5405 022462 012777 000000 157472    MOV      @0,@BE1CR2    ; CLEAR UPPER 2 ADDRESS BITS
5406 022470 012777 177776 157454    MOV      @-2,@BE1CC    ; SET UBE FOR TWO TRANSFERS
5407
5408          ; GO DO THE TRANSFER
5409
5410 022476 012777 002041 157452    MOV      @2041,@BE1CR1  ; SET UBE FOR MPIR-DATI-1 XFER
5411 022504 032777 000200 157444 154:  BIT      @BIT7,@BE1CR1  ; IS THE TRANSFER COMPLETE ?
5412 022512 001774          BEQ      154          ; NO, THEN WAIT FOR IT TO BE COMPLETE
5413
5414          ; CHECK THE VALID BITS IN THE KMCR
5415
5416 022514 013703 177734          MOV      KMCR,R3          ; PUT KMCR INTO R3 FOR MASKING
5417 022520 042703 000777          BIC      @777,R3          ; MASK OUT BITS 0-8
5418 022524 022703 110000          CMP      @110000,R3    ; ARE THE BITS SET CORRECTLY ?
5419 022530 001401          BEQ      204          ; YES, THEN GO CHECK THE AVAILABILITY BITS
5420 022532 104033          ERROR    +33          ; NO, THEN ERROR IN DMA CACHE
5421 022534 005037 172516 204:  CLR      MPIR3          ; DISABLE MAPPING
5422 022540 042737 000100 177734    BIC      @BIT06,KMCR    ; AND CACHE
5423 022546 023777 000202 157374    CMP      @202,@BE1DB    ; DATA READ OK?
5424 022554 001401          BEQ      TST66          ; IF OK, EXIT TEST
5425 022556 104033          ERROR    +33          ; OTHERWISE ERROR IN CACHE
5426

```

TEST WRONG PARITY AND CACHE

5428
5429
5430
5431
5432
5433
5434
5435
5436
5437
5438
5439
5440
5441
5442
5443
5444

.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
; ENDIF

; ENDTST

;*****

; *TEST 66 WRONG PARITY AND CACHE

;*****

TST66: SCOPE

5445 022560 000004

5446	022562	004737	002450		JSR	PC,IUBE		; CLEAR UBE
5447	022566	052737	000005	172100	BIS	#BIT02!BIT00,#172100		; SET WRONG PARITY IN MEMORY CSR
5448	022574	042737	003740	172100	BIC	#3740,#172100		; CLEAR CHECK BITS
5449	022602	005037	000000		CLR	#0		; WRITE LOCATION 0 WITH WRONG PARITY
5450	022606	042737	000004	172100	BIC	#BIT02,#172100		; CLEAR WRONG PARITY
5451	022614	052737	000040	172516	BIS	#BIT05,MMR3		; ENABLE MEMORY MAPPING
5452	022622	042737	000100	177734	BIC	#BIT06,KMCR		; DISABLE JUST TO MAKE SURE
5453	022630	052737	000100	177734	BIS	#BIT06,KMCR		; ENABLE CACHE
5454	022636	012737	000000	170204	MOV	#0,MAPL01		; POINT MAP REGISTER TO LOCATION 200
5455	022644	005037	170206		CLR	MAPH01		
5456	022650	012777	020000	157276	MOV	#20000,8BE18A		; SET TRANSFER ADDRESS TO OCTAL BOUNDARY
5457	022656	012777	000000	157276	MOV	#0,8BE1CR2		; CLEAR UPPER 2 ADDRESS BITS
5458	022664	012777	177777	157260	MOV	#-1,8BE1CC		; SET UBE FOR TWO TRANSFERS

5459

5460

5461

5462	022672	012777	002041	157256		MOV	#2041,8BE1CR1		; SET UBE FOR NPR-DATI-1 XFER
5463	022700	032777	000200	157250	154:	BIT	#BIT7,8BE1CR1		; IS THE TRANSFER COMPLETE ?
5464	022706	001774				BEQ	154		; NO, THEN WAIT FOR IT TO BE COMPLETE
5465	022710	032737	177000	177734		BIT	#177000,KMCR		; WAS ALLOCATED?
5466	022716	001401				BEQ	164		; IF NOT, BRANCH
5467	022720	104032				ERROR	+32		; IF WAS, REPORT ERROR
5468	022722	005037	000000		164:	CLR	#0		; REWRITE 0 WITH RIGHT PARITY
5469	022726	005037	172516			CLR	MMR3		; CLEAR MAPPING ENABLED
5470	022732	042737	000001	172100		BIC	#BIT00,#172100		

5471

5472

5473 022740

UBEM:

;*****

; *TEST 67 DUMMY TEST

;*****

TST67: SCOPE

5474	022742	022737	000001	001206		CMP	#1,#PASS		; SECOND PASS?
5475	022750	001006				BNE	#EOP		; IF NOT, GOTO EOP
5476	022752	012737	000012	023014		MOV	#12,#EOPCT		; AFTER 1ST PASS
5477	022760	012737	000012	023022		MOV	#12,#ENDCT		; PRINT EVERY 10TH

T67 DUMMY TEST

5478
5479
5480
5481
5482

022766
022766 000004
022770 005037 001102
022774 005037 001164
023000 005237 001206
023004 042737 100000 001206
023012 005327
023014 000001
023016 003055
023020 012737
023022 000001
023024 023014
023026 005737 002574
023032 001037
023034 104401 023042
023040 000407

023060
023060 013746 001206

023064 104405
023066 104401 023074
023072 000412

023120
023120 013746 001112

023124 104405
023126 104401 001175
023132
023132 013700 000042
023136 001405
023140 000005
023142 004710
023144 000240
023146 000240
023150 000240
023152
023152 000137
023154 003270
023156 377 377 000

```

;
; END OF PASS ROUTINE
;
.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:
        SCOPE
        CLR      $TSTNM          ;;ZERO THE TEST NUMBER
        CLR      $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
        INC      $PASS           ;;INCREMENT THE PASS NUMBER
        BIC      $100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
        DEC      (PC)+           ;;LOOP?
$EOPCT: .WORD    1
        BGT      $DOAGN          ;;YES
        MOV      (PC)+,$(PC)+    ;;RESTORE COUNTER
$ENDCT: .WORD    1
        TST      UQUIET          ;;NO MESSAGE IF RUNNING UFD QUIET MODE
        BNE      1$
        TYPE     ,65$            ;;TYPE ASCIZ STRING
        BR       64$            ;;GET OVER THE ASCIZ
1$: .ASCIZ      <12><15>/END PASS 0/
64$:
        MOV      $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
                                ;;TYPE PASS NUMBER
                                ;;GO TYPE--DECIMAL ASCII WITH SIGN
        TYPDS
        TYPE     ,67$            ;;TYPE ASCIZ STRING
        BR       66$            ;;GET OVER THE ASCIZ
1$: .ASCIZ      /      TOTAL ERRORS:/
66$:
        MOV      $ERTTL,-(SP)    ;;SAVE $ERTTL FOR TYPEOUT
                                ;;TOTAL NUMBER OF ERRORS
                                ;;GO TYPE--DECIMAL ASCII WITH SIGN
        TYPDS
        TYPE     , $CRLF         ;;TYPE CARRIAGE RETURN, LINE FEED
1$:
$GET42: MOV      $M42,R0         ;;GET MONITOR ADDRESS
        BEQ      $DOAGN         ;;BRANCH IF NO MONITOR
        RESET
        $ENDAD: JSR      PC,(R0)  ;;CLEAR THE WORLD
                                ;;GO TO MONITOR
                                ;;SAVE ROOM
                                ;;FOP
                                ;;ACT11
        $DOAGN:
        JMP      $(PC)+         ;;RETURN
$RTNAD: .WORD    RESTART
$ENULL: .BYTE    -1,-1,0
        .EVEN
        ;;NULL CHARACTER STRING

```

5483
5484
5485
5486
5487

```

;
; SYSMAC ROUTINES
;

```

END OF PASS ROUTINE

5488
5489
5490
5491
5492

SCOPE - ENABLE HALT-ON-BREAK BETWEEN TESTS, SO APT CAN CATCH IT. DISABLE MOB
AT END SO TESTS WILL NOT BE INTERRUPTED BY APT BREAKS WHICH CAUSES CERTAIN
TESTS TO REPORT ERRORS WHEN THERE ARE REALLY NONE.

.SBTTL SCOPE HANDLER ROUTINE

THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
AND LOAD THE TEST NUMBER (ITSTNM) INTO THE DISPLAY REG. (DISPLAY<7:0>)
AND LOAD THE ERROR FLAG (IERFLG) 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

023162				SCOPE:		
023162	104407			CKSWR		TEST FOR CHANGE IN SOFT-SWR
023164	052737	001000	177520	BIS	#BIT09,BCSR	ENABLE
023172	032777	040000	155740	11: BIT	#BIT14,BSWR	LOOP ON PRESENT TEST?
023200	001117			BNE	OVER	YES IF SW14=1
				*****START OF CODE FOR THE XOR	TESTER*****	
023202	000416			XTSTR: BR	61	IF RUNNING ON THE "XOR" TESTER CHANGE
						THIS INSTRUCTION TO A "NOP" (NOP=24C)
023204	013746	000004		MOV	#ERRVEC, -(SP)	SAVE THE CONTENTS OF THE ERROR VECTOR
023210	012737	023230	000004	MOV	#5, #ERRVEC	SET FOR TIMEOUT
023216	005737	177060		TST	#177060	TIME OUT ON XOR?
023222	012637	000004		MOV	(SP), #ERRVEC	RESTORE THE ERROR VECTOR
023226	000466			BR	SVLAD	GO TO THE NEXT TEST
023230	022626			51: CMP	(SP), (SP)	CLEAR THE STACK AFTER A TIME OUT
023232	012637	000004		MOV	(SP), #ERRVEC	RESTORE THE ERROR VECTOR
023236	000426			BR	71	LOOP ON THE PRESENT TEST
023240				61: *****END OF CODE FOR THE XOR	TESTER*****	
023240	032777	0004C^	155672	BIT	#BIT08,BSWR	LOOP ON SPEC. TEST?
023246	001407			BEQ	21	IF NO
023250	017746	155664		MOV	BSWR, -(SP)	SET DESIRED TEST NUM. FROM SWR
023254	042716	000300		BIC	#SWRMK, (SP)	STRIP AWAY UNDESIRED BITS
023260	122637	001102		CMPS	(SP), ITSTNM	ON THE RIGHT TEST?
023264	001465			BEQ	OVER	IF YES
023266	105737	001103		21: TSTB	IERFLG	HAS AN ERROR OCCURRED?
023272	001421			BEQ	31	IF NO
023274	123737	001115	001103	CMPS	IERMAX, IERFLG	MAX. ERRORS FOR THIS TEST OCCURRED?
023302	101015			BHI	31	IF NO
023304	032777	001000	155626	BIT	#BIT09,BSWR	LOOP ON ERROR?
023312	001404			BEQ	41	IF NO
023314	013737	001110	001106	71: MOV	LPERR, LPADR	SET LOOP ADDRESS TO LAST SCOPE
023322	000446			BR	OVER	
023324	105037	001103		41: CLRB	IERFLG	ZERO THE ERROR FLAG
023330	005037	001164		CLR	ITIMES	CLEAR THE NUMBER OF ITERATIONS TO MAKE
023334	000415			BR	11	ESCAPE TO THE NEXT TEST
023336	032777	004000	155574	31: BIT	#BIT11,BSWR	INHIBIT ITERATIONS?
023344	001011			BNE	11	IF YES
023346	005737	001206		TST	IPASS	IF FIRST PASS OF PROGRAM
023352	001406			BEQ	11	INHIBIT ITERATIONS
023354	005237	001104		INC	ICNT	INCREMENT ITERATION COUNT
023360	023737	001164	001104	CMPS	ITIMES, ICNT	CHECK THE NUMBER OF ITERATIONS MADE
023366	002024			BGE	OVER	IF MORE ITERATION REQUIRED

SCOPE HANDLER ROUTINE

```

023370 012737 000001 001104 14: MOV #1,ICNT ;REINITIALIZE THE ITERATION COUNTER
023376 013737 023462 001164 MOV #MXCNT,ITIMES ;SET NUMBER OF ITERATIONS TO DO
023404 105237 001102 ;SVLAD: INCB ITSTNM ;COUNT TEST NUMBERS
023410 113737 001102 001204 MOVB ITSTNM,ITESTN ;SET TEST NUMBER IN APT MAILBOX
023416 011637 001106 MOV (SP),LPADR ;SAVE SCOPE LOOP ADDRESS
023422 011637 001110 MOV (SP),LPERR ;SAVE ERROR LOOP ADDRESS
023426 005037 001166 CLR #ESCAPE ;CLEAR THE ESCAPE FROM ERROR ADDRESS
023432 112737 000011 001115 MOVB #1,IERMAX ;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
023440 013777 001107 155474 ;OVER: MOV ITSTNM,DISP ;DISPLAY TEST NUMBER
023446 013716 001106 MOV LPADR,(SP) ;FUDGE RETURN ADDRESS
023452 042737 001000 177520 BIC #BIT09,BCSR ;DIS MOB
023460 000002 RTI
023462 000001 ;MXCNT: 1 ;MAX. NUMBER OF ITERATIONS

```

5493

.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
;
;TYPE: TSTB #TPFLG ;IS THERE A TERMINAL?

```

```

023464 105737 001157 ;TYPE: TSTB #TPFLG ;IS THERE A TERMINAL?
023470 100002 BPL 14 ;BR IF YES
023472 000000 HALT ;HALT HERE IF NO TERMINAL
023474 000430 BR 34 ;LEAVE
023476 010046 14: MOV R0,-(SP) ;SAVE R0
023500 017600 000002 MOV #2(SP),R0 ;GET ADDRESS OF ASCIZ STRING
023504 122737 000001 001220 CMPB #APTENV,ENV ;RUNNING IN APT MODE
023512 001011 BNE 624 ;NO,GO CHECK FOR APT CONSOLE
023514 132737 000100 001221 BITB #APTSPOOL,ENVH ;SPOOL MESSAGE TO APT
023522 001405 BEQ 624 ;NO,GO CHECK FOR CONSOLE
023524 010037 023534 MOV R0,614 ;SETUP MESSAGE ADDRESS FOR APT
023530 004737 026150 JSR PC,ATY3 ;SPOOL MESSAGE TO APT
023534 000000 614: .WORD ;MESSAGE ADDRESS
023536 132737 000040 001221 624: BITB #TCSUP,ENVH ;APT CONSOLE SUPPRESSED
023544 001003 BNE 604 ;YES,SKIP TYPE OUT
023546 112046 24: MOVB (R0),-(SP) ;PUSH CHARACTER TO BE TYPED ONTO STACK
023550 001005 BNE 44 ;BR IF IT ISN'T THE TERMINATOR
023552 005726 TST (SP), ;IF TERMINATOR POP IT OFF THE STACK
023554 012600 604: MOV (SP),R0 ;RESTORE R0
023556 062716 000002 34: ADD #2,(SP) ;ADJUST RETURN PC
023562 000002 RTI ;RETURN
023564 122716 000011 44: CMPB #HT,(SP) ;BRANCH IF <HT>
023570 001430 BEQ 84 ;BRANCH IF NOT <CRLF>
023572 122716 000200 CMPB #CRLF,(SP)
023576 001006 BNE 54 ;
023600 005726 TST (SP), ;POP <CR><LF> EQUIV
023602 104401 TYPE ;TYPE A CR AND LF
023604 001175 #CRLF
023606 105037 024014 CLRB #CHARCNT ;CLEAR CHARACTER COUNT

```

TYPE ROUTINE

```

023612 000755      BR      2#      ;;GET NEXT CHARACTER
023614 004737 023676 5# : JSR      PC,#TYPEC  ;;GO TYPE THIS CHARACTER
023620 123726 001156 6# : CMPB     #FILLC,(SP)+  ;;IS IT TIME FOR FILLER CHARS.?
023624 001350      BNE      2#      ;;IF NO GO GET NEXT CHAR.
023626 013746 001154      MOV      #NULL,-(SP)  ;;GET # OF FILLER CHARS. NEEDED
                                           ;;AND THE NULL CHAR.
023632 105366 000001 7# : DECB      1(SP)      ;;DOES A NULL NEED TO BE TYPED?
023636 002770      BLT      6#      ;;BR IF NO--GO POP THE NULL OFF OF STACK
023640 004737 023676      JSR      PC,#TYPEC  ;;GO TYPE A NULL
023644 105337 024014      DECB      #CHARCNT  ;;DO NOT COUNT AS A COUNT
023650 000770      BR      7#      ;;LOOP

023652 112716 000040      ;HORIZONTAL TAB PROCESSOR
023656 004737 023676 8# : MOVB      #' ,(SP)      ;;REPLACE TAB WITH SPACE
023662 132737 000007 9# : JSR      PC,#TYPEC  ;;TYPE A SPACE
023670 001372      BITB      #7,#CHARCNT  ;;BRANCH IF NOT AT
023672 005726      BNE      9#      ;;TAB STOP
023674 000724      TST      (SP)+      ;;POP SPACE OFF STACK
023676      BR      2#      ;;GET NEXT CHARACTER

023676 105777 155242      #TYPEC: TSTB      #TKS      ;;CHAR IN KYBD BUFFER?
023702 100022      BPL      10#      ;;BR IF NOT
023704 017746 155236      MOV      #TKB,-(SP)  ;;GET CHAR
023710 042716 177600      BIC      #177600,(SP)  ;;STRIP EXTRANEIOUS BITS
023714 122716 000023      CMPB     #XOFF,(SP)  ;;WAS CHAR XOFF
023720 001012      BNE      102#      ;;BR IF NOT
023722      101# : TSTB      #TKS      ;;WAIT FOR CHAR
023726 100375      BPL      101#      ;;BR IF NOT
023730 117716 155212      MOVB      #TKB,(SP)  ;;GET CHAR
023734 042716 177600      BIC      #177600,(SP)  ;;STRIP IT
023740 122716 000021      CMPB     #XON,(SP)  ;;WAS IT XON?
023744 001366      BNE      101#      ;;BR IF NOT
023746      102# : TST      (SP)+      ;;FIX STACK
023750      10# : TSTB      #TPS      ;;WAIT UNTIL PRINTER IS READY
023754 100375      BPL      10#      ;;BR IF NOT
023756 116677 000002 155166      MOVB      2(SP),#TPB  ;;LOAD CHAR TO BE TYPED INTO DATA REG.
023764 122766 000015 000002      CMPB     #CR,2(SP)  ;;IS CHARACTER A CARRIAGE RETURN?
023772 001003      BNE      1#      ;;BRANCH IF NO
023774 105037 024014      CLRB      #CHARCNT  ;;YES--CLEAR CHARACTER COUNT
024000 000406      BR      #TYPEX  ;;EXIT
024002 122766 000012 000002 1# : CMPB     #LF,2(SP)  ;;IS CHARACTER A LINE FEED?
024010 001402      BEQ      #TYPEX  ;;BRANCH IF YES
024012 105227      INCB      (PC)+      ;;COUNT THE CHARACTER
024014 000000      #CHARCNT: .WORD 0      ;;CHARACTER COUNT STORAGE
024016 000207      #TYPEX: RTS      PC

;SHTTL 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

```

BINARY TO OCTAL (ASCII) AND TYPE

```

; *                                     ;: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     8(SP), -(SP)      ;:PICKUP THE MODE
; *     MOV     1(SP), $OFILL        ;:LOAD ZERO FILL SWITCH
; *     MOV     (SP), $OMODE+1       ;:NUMBER OF DIGITS TO TYPE
; *     ADD     #2, (SP)             ;:ADJUST RETURN ADDRESS
; *     BR      $TYPON
; * $TYPOC: MOV     #1, $OFILL        ;:SET THE ZERO FILL SWITCH
; *     MOV     #6, $OMODE+1         ;:SET FOR SIX(6) DIGITS
; * $TYPON: MOV     #5, $OCNT         ;:SET THE ITERATION COUNT
; *     MOV     R3, -(SP)            ;:SAVE R3
; *     MOV     R4, -(SP)            ;:SAVE R4
; *     MOV     R5, -(SP)            ;:SAVE R5
; *     MOV     $OMODE+1, R4         ;:GET THE NUMBER OF DIGITS TO TYPE
; *     NEG     R4
; *     ADD     #6, R4               ;:SUBTRACT IT FOR MAX. ALLOWED
; *     MOV     R4, $OMODE           ;:SAVE IT FOR USE
; *     MOV     $OFILL, R4          ;:GET THE ZERO FILL SWITCH
; *     MOV     12(SP), R5           ;:PICKUP THE INPUT NUMBER
; *     CLR     R3                  ;:CLEAR THE OUTPUT WORD
; * 1$: ROL     R5                  ;:ROTATE MSB INTO "C"
; *     BR      3$
; * 2$: ROL     R5                  ;:GO DO MSB
; *     ROL     R5                  ;:FORM THIS DIGIT
; *     MOV     R5, R3
; * 3$: ROL     R3                  ;:GET LSB OF THIS DIGIT
; *     DECB    $OMODE              ;:TYPE THIS DIGIT?
; *     BPL     7$
; *     BIC     #177770, R3         ;:BR IF NO
; *     BNE     4$
; *     TST     R4                  ;:GET RID OF JUNK
; *     BEQ     5$
; *     INC     R4                  ;:TEST FOR 0
; *     INC     R4                  ;:SUPPRESS THIS 0?
; * 4$: INC     R4                  ;:BR IF YES
; *     BIS     #'0, R3             ;:DON'T SUPPRESS ANYMORE 0'S
; *     BIS     #' , R3             ;:MAKE THIS DIGIT ASCII
; *     MOV     R3, 8$              ;:MAKE ASCII IF NOT ALREADY
; *     TYPE    , 8$               ;:SAVE FOR TYPING
; * 5$: MOV     $OCNT, 8$           ;:GO TYPE THIS DIGIT
; *     DECB    $OCNT              ;:COUNT BY 1
; *     BGT     2$                  ;:BR IF MORE TO DO
; *     BLT     6$                  ;:BR IF DONE
; *     INC     R4                  ;:INSURE LAST DIGIT ISN'T A BLANK
; *     BR      2$
; * 6$: MOV     (SP), R5            ;:GO DO THE LAST DIGIT
; *     MOV     (SP), R4            ;:RESTORE R5
; *     MOV     (SP), R3            ;:RESTORE R4
; *     MOV     (SP), R3            ;:RESTORE R3

```

BINARY TO OCTAL (ASCII) AND TYPE

```

024226 016666 000002 000004      MOV      2(SP),4(SP)      ;;SET THE STACK FOR RETURNING
024234 012616      MOV      (SP)+,(SP)
024236 000002      RTI                      ;;RETURN
024240      000      81:      .BYTE      0      ;;STORAGE FOR ASCII DIGIT
024241      000      .BYTE      0      ;;TERMINATOR FOR TYPE ROUTINE
024242      000      %CNT:      .BYTE      0      ;;OCTAL DIGIT COUNTER
024243      000      %FILL:      .BYTE      0      ;;ZERO FILL SWITCH
024244 000000      %MODE:      .WORD      0      ;;NUMBER OF DIGITS TO TYPE
5495      .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
;*****
;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
;SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
;NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
;BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
;REPLACED WITH SPACES.
;CALL:
;*      MOV      NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
;*      TYPDS      ;;GO TO THE ROUTINE
%TYPDS:
MOV      R0,-(SP)      ;;PUSH R0 ON STACK
MOV      R1,-(SP)      ;;PUSH R1 ON STACK
MOV      R2,-(SP)      ;;PUSH R2 ON STACK
MOV      R3,-(SP)      ;;PUSH R3 ON STACK
MOV      R5,-(SP)      ;;PUSH R5 ON STACK
MOV      #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
MOV      20(SP),R5      ;;GET THE INPUT NUMBER
BPL      11:      ;;BR IF INPUT IS POS.
NEG      R5      ;;MAKE THE BINARY NUMBER POS.
MOVB     #'-.1(SP)      ;;MAKE THE ASCII NUMBER NEG.
11:      CLR      R0      ;;ZERO THE CONSTANTS INDEX
MOV      #%DBLK,R3      ;;SETUP THE OUTPUT POINTER
MOVB     #'',(R3)+      ;;SET THE FIRST CHARACTER TO A BLANK
21:      CLR      R2      ;;CLEAR THE BCD NUMBER
MOV      %DTBL(R0),R1    ;;GET THE CONSTANT
31:      SUB      R1,R5      ;;FORM THIS BCD DIGIT
BLT      41:      ;;BR IF DONE
INC      R2      ;;INCREASE THE BCD DIGIT BY 1
BR       31:
41:      ADD      R1,R5      ;;ADD BACK THE CONSTANT
TST      R2      ;;CHECK IF BCD DIGIT=0
BNE      51:      ;;FALL THROUGH IF 0
TSTB     (SP)      ;;STILL DOING LEADING 0'S?
BMI      71:      ;;BR IF YES
51:      ASLB     (SP)      ;;MSD?
BCC      61:      ;;BR IF NO
61:      MOVB     1(SP),-1(R3)  ;;YES--SET THE SIGN
71:      BIS      #'0,R2      ;;MAKE THE BCD DIGIT ASCII
      BIS      #' ,R2      ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
      MOVB     R2,(R3)+      ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
      TST      (R0)+      ;;JUST INCREMENTING
024370 005720      CMP      R0,#10      ;;CHECK THE TABLE INDEX
024372 020027 000010      BLT      21:      ;;GO DO THE NEXT DIGIT
024376 002746      BGT      81:      ;;GO TO EXIT
024400 003002      MOV      R5,R2      ;;GET THE LSD
024402 010502      BR       61:      ;;GO CHANGE TO ASCII
024404 000764      81:      TSTB     (SP)+      ;;WAS THE LSD THE FIRST NON-ZERO?
024406 105726      BPL      91:      ;;BR IF NO
024410 100003

```

CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```

024412 116663 177777 177776      MOVB    -1(SP),-2(R3)    ;;YES--SET THE SIGN FOR TYPING
024420 105013                    94:  CLRB     (R3)          ;;SET THE TERMINATOR
024422 012605                    MOV     (SP)+,R5          ;;POP STACK INTO R5
024424 012603                    MOV     (SP)+,R3          ;;POP STACK INTO R3
024426 012602                    MOV     (SP)+,R2          ;;POP STACK INTO R2
024430 012601                    MOV     (SP)+,R1          ;;POP STACK INTO R1
024432 012600                    MOV     (SP)+,R0          ;;POP STACK INTO R0
024434 104401 024462              TYPE     ,#DBLK          ;;NOW TYPE THE NUMBER
024440 016666 000002 000004      MOV     2(SP),4(SP)      ;;ADJUST THE STACK
024446 012616                    MOV     (SP)+,(SP)
024450 000002                    RTI                      ;;RETURN TO USER
024452 023420                    $DTBL: 10000.
024454 001750                    1000.
024456 000144                    100.
024460 000012                    10.
024462                                $DBLK: .BLKW 4
5496                                .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.
024472 022737 000176 001140      $CKSWR: CMP     #SWREG,SWR    ;;IS THE SOFT-SWR SELECTED?
024500 001074                    BNE     154              ;;BRANCH IF NO
024502 105777 154436              TSTB     @TKS              ;;CHAR THERE?
024506 100071                    BPL     154              ;;IF NO, DON'T WAIT AROUND
024510 117746 154432              MOVB     @TKB,-(SP)        ;;SAVE THE CHAR
024514 042716 177600              BIC     @C177,(SP)        ;;STRIP-OFF THE ASCII
024520 022726 000007              CMP     #7,(SP)+          ;;IS IT A CONTROL G?
024524 001062                    BNE     154              ;;NO, RETURN TO USER
024526 123727 001134 000001      CMPB     @AUTOB,#1        ;;ARE WE RUNNING IN AUTO-MODE?
024534 001456                    BEQ     154              ;;BRANCH IF YES
024536 104401 025227              TYPE     ,#CNTLG          ;;ECHO THE CONTROL-G (+G)
024542 104401 025234              $GTSWR: TYPE    ,#MSWR    ;;TYPE CURRENT CONTENTS
024546 013746 000176              MOV     SWREG,-(SP)        ;;SAVE SWREG FOR TYPEOUT
024552 104402                    TYPOC          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
024554 104401 025245              TYPE     ,#MNEW          ;;PROMPT FOR NEW SWR
024560 005046                    194:  CLR     -(SP)          ;;CLEAR COUNTER
024562 005046                    CLR     -(SP)          ;;THE NEW SWR
024564 105777 154354              74:  TSTB     @TKS              ;;CHAR THERE?
024570 100375                    BPL     74              ;;IF NOT TRY AGAIN
024572 117746 154350              MOVB     @TKB,-(SP)        ;;PICK UP CHAR
024576 042716 177600              BIC     @C177,(SP)        ;;MAKE IT 7-BIT ASCII
024602 021627 000025              94:  CMP     (SP),#25        ;;IS IT A CONTROL-U?
024606 001005                    BNE     104              ;;BRANCH IF NOT
024610 104401 025222              TYPE     ,#CNTLU          ;;YES, ECHO CONTROL-U (+U)
024614 062706 000006              204:  ADD     #6,SP        ;;IGNORE PREVIOUS INPUT
024620 000757                    BR      194              ;;LET'S TRY IT AGAIN
024622 021627 000015              104:  CMP     (SP),#15        ;;IS IT A <CR>?
024626 001022                    BNE     164              ;;BRANCH IF NO
024630 005766 000004              TST     4(SP)            ;;YES, IS IT THE FIRST CHAR?
024634 001403                    BEQ     114              ;;BRANCH IF YES
024636 016677 000002 154274      MOV     2(SP),@SWR        ;;SAVE NEW SWR
024644 062706 000006              114:  ADD     #6,SP        ;;CLEAR UP STACK
024650 104401 001175              144:  TYPE     ,#CRLF        ;;ECHO <CR> AND <LF>

```

TTY INPUT ROUTINE

```

024654 123727 001135 000001      CMPB    #INTAG,#1      ;;RE-ENABLE TTY KBD INTERRUPTS?
024662 001003                      BNE      15$      ;;BRANCH IF NOT
024664 012777 000100 154252      MOV      #100,#ATKS      ;;RE-ENABLE TTY KBD INTERRUPTS
024672 000002                      RTI                      ;;RETURN
024674 004737 023676      15$:    JSR      PC,#TYPEC      ;;ECHO CHAR
024700 021627 000060      16$:    CMP      (SP),#60      ;;CHAR < 0?
024704 002420                      BLT      18$      ;;BRANCH IF YES
024706 021627 000067      CMP      (SP),#67      ;;CHAR > ??
024712 003015                      BGT      18$      ;;BRANCH IF YES
024714 042726 000060      BIC      #60,(SP)      ;;STRIP-OFF ASCII
024720 005766 000002      TST      2(SP)      ;;IS THIS THE FIRST CHAR
024724 001403                      BEQ      17$      ;;BRANCH IF YES
024726 006316                      ASL      (SP)      ;;NO, SHIFT PRESENT
024730 006316                      ASL      (SP)      ;;CHAR OVER 10 MAKE
024732 006316                      ASL      (SP)      ;;ROOM FOR NEW ONE.
024734 005266 000002      17$:    INC      2(SP)      ;;KEEP COUNT OF CHAR
024740 056616 177776      BIS      -2(SP),(SP)      ;;SET IN NEW CHAR
024744 000707                      BR       7$      ;;GET THE NEXT ONE
024746 104401 001174      18$:    TYPE     ,#QUES      ;;TYPE ?<CR><LF>
024752 000720                      BR       20$      ;;SIMULATE CONTROL-U
                                .DSABL    LSB
                                ;;*****
                                ;;THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
                                ;;CALL:
                                ;;*      RDCHR
                                ;;*      RETURN HERE
                                ;;*      INPUT A SINGLE CHARACTER FROM THE TTY
                                ;;*      CHARACTER IS ON THE STACK
                                ;;*      WITH PARITY BIT STRIPPED OFF
                                ;;
024754 011646      $RDCHR: MOV      (SP),-(SP)      ;;PUSH DOWN THE PC
024756 016666 000004 000002      MOV      4(SP),2(SP)      ;;SAVE THE PS
024764 105777 154154      1$:    TSTB     #ATKS      ;;WAIT FOR
024770 100375                      BPL      1$      ;;A CHARACTER
024772 117766 154150 000004      MOVB     #ATKB,4(SP)      ;;READ THE TTY
025000 042766 177600 000004      BIC      #C<177>,4(SP)      ;;GET RID OF JUNK IF ANY
025006 026627 000004 000023      CMP      4(SP),#23      ;;IS IT A CONTROL-S?
025014 001013                      BNE      3$      ;;BRANCH IF NO
025016 105777 154122      2$:    TSTB     #ATKS      ;;WAIT FOR A CHARACTER
025022 100375                      BPL      2$      ;;LOOP UNTIL ITS THERE
025024 117746 154116      MOVB     #ATKB,-(SP)      ;;GET CHARACTER
025030 042716 177600      BIC      #C177,(SP)      ;;MAKE IT 7-BIT ASCII
025034 022627 000021      CMP      (SP),#21      ;;IS IT A CONTROL-Q?
025040 001366                      BNE      2$      ;;IF NOT DISCARD IT
025042 000750                      BR       1$      ;;YES, RESUME
025044 026627 000004 000021      3$:    CMP      4(SP),#XON      ;;IS IT A RANDOM XON?
025052 001744                      BEQ      1$      ;;BRANCH IF YES
025054 026627 000004 000140      CMP      4(SP),#140      ;;IS IT UPPER CASE?
025062 002407                      BLT      4$      ;;BRANCH IF YES
025064 026627 000004 000175      CMP      4(SP),#175      ;;IS IT A SPECIAL CHAR?
025072 003003                      BGT      4$      ;;BRANCH IF YES
025074 042766 000040 000004      BIC      #40,4(SP)      ;;MAKE IT UPPER CASE
025102 000002      4$:    RTI                      ;;GO BACK TO USER
                                ;;*****
                                ;;THIS ROUTINE WILL INPUT A STRING FROM THE TTY
                                ;;CALL:
                                ;;*      RDLIN
                                ;;*      RETURN HERE
                                ;;*      INPUT A STRING FROM THE TTY
                                ;;*      ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
                                ;;*      TERMINATOR WILL BE A BYTE OF ALL 0'S

```

TTY INPUT ROUTINE

```

025104 010346      $RDLIN: MOV    R3,-(SP)      ;;SAVE R3
025106 012703 025212 1$: MOV    $TTYIN,R3      ;;GET ADDRESS
025112 022703 025222 2$: CMP    $TTYIN+8.,R3    ;;BUFFER FULL?
025116 101405      BLOS    4$                  ;;BR IF YES
025120 104410      RDOCHR      ;;GO READ ONE CHARACTER FROM THE TTY
025122 112613      MOV8    (SP)+,(R3)          ;;GET CHARACTER
025124 122713 000177 10$: CMP8    $177,(R3)      ;;IS IT A RUBOUT
025130 001003      BNE     3$                  ;;SKIP IF NOT
025132 104401 001174 4$: TYPE    , $QUES        ;;TYPE A '?'
025136 000763      BR      1$                  ;;CLEAR THE BUFFER AND LOOP
025140 111337 025210 3$: MOV8    (R3),9$         ;;ECHO THE CHARACTER
025144 104401 025210      TYPE    ,9$
025150 122723 000015      CMP8    $15,(R3)+      ;;CHECK FOR RETURN
025154 001356      BNE     2$                  ;;LOOP IF NOT RETURN
025156 105063 177777      CLRB    -1(R3)         ;;CLEAR RETURN (THE 15)
025162 104401 001176      TYPE    , $LF         ;;TYPE A LINE FEED
025166 012603      MOV     (SP)+,R3            ;;RESTORE R3
025170 011646      MOV     (SP),-(SP)          ;;ADJUST THE STACK AND PUT ADDRESS OF THE
025172 016666 000004 000002      MOV     4(SP),2(SP)      ;; FIRST ASCII CHARACTER ON IT
025200 012766 025212 000004      MOV     $TTYIN,4(SP)
025206 000002      RTI                          ;;RETURN
025210      000      9$: .BYTE    0              ;;STORAGE FOR ASCII CHAR. TO TYPE
025211      000      .BYTE    0              ;;TERMINATOR
025212      $TTYIN: .BLKB    8.                ;;RESERVE 8 BYTES FOR TTY INPUT
025222      136      125      015 $CNTLU: .ASCIZ  /↑U/<15><12>  ;;CONTROL "U"
025225      012      000
025227      136      107      015 $CNTLG: .ASCIZ  /↑G/<15><12>  ;;CONTROL "G"
025232      012      000
025234      015      012      123 $MSWR: .ASCIZ  <15><12>/SWR = /
025237      127      122      040
025242      075      040      000
025245      040      040      116 $MNEW: .ASCIZ  / NEW = /
025250      105      127      040
025253      075      040      000

5497 .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
;* RETURN HERE
;*
025256 011646      $RDOCT: MOV     (SP),-(SP)      ;;READ AN OCTAL NUMBER
025260 016666 000004 000002      MOV     4(SP),2(SP)  ;;LOW ORDER BITS ARE ON TOP OF THE STACK
025266 010046      MOV     R0,-(SP)              ;;HIGH ORDER BITS ARE IN $HIOCT
025270 010146      MOV     R1,-(SP)              ;;PROVIDE SPACE FOR THE
025272 010246      MOV     R2,-(SP)              ;;INPUT NUMBER
025274 104411      1$: RDLIN      ;;PUSH R0 ON STACK
025276 012600      MOV     (SP)+,R0              ;;PUSH R1 ON STACK
025300 005001      CLR     R1                    ;;PUSH R2 ON STACK
025302 005002      CLR     R2                    ;;READ AN ASCII LINE
025304 112046      2$: MOV8    (R0)+,-(SP)        ;;GET ADDRESS OF 1ST CHARACTER
025306 001412      BEQ     3$                    ;;CLEAR DATA WORD
025310 006301      ASL     R1                    ;;PICKUP THIS CHARACTER
025312 006102      ROL     R2                    ;;IF ZERO GET OUT
025314 006301      ASL     R1                    ;;*2
025316 006102      ROL     R2                    ;;*4

```

READ AN OCTAL NUMBER FROM THE TTY

5498

```

025320 006301      ASL      R1      ;;*8
025322 006102      ROL      R2
025324 042716 177770 BIC      #C7,(SP)  ;;STRIP THE ASCII JUNK
025330 062601      ADD      (SP)+,R1  ;;ADD IN THIS DIGIT
025332 000764      BR       2$      ;;LOOP
025334 005726      3$: TST      (SP)+  ;;CLEAN TERMINATOR FROM STACK
025336 010166 000012 MOV      R1,12(SP)  ;;SAVE THE RESULT
025342 010237 025356 MOV      R2,#HIOCT
025346 012602      MOV      (SP)+,R2  ;;POP STACK INTO R2
025350 012601      MOV      (SP)+,R1  ;;POP STACK INTO R1
025352 012600      MOV      (SP)+,R0  ;;POP STACK INTO R0
025354 000002      RTI      ;;RETURN
025356 000000      #HIOCT: .WORD 0  ;;HIGH ORDER BITS GO HERE
                    .SBITL 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:
                    ;*      RDEEC      ;;READ A DECIMAL NUMBER
                    ;*      RETURN HERE  ;;NUMBER IS ON TOP OF THE STACK
                    ;
025360 011646      $RDEEC: MOV      (SP),-(SP)  ;;PROVIDE SPACE FOR
025362 016666 000004 000002 MOV      4(SP),2(SP)  ;;THE INPUT NUMBER
025370 010046      MOV      R0,-(SP)  ;;PUSH R0 ON STACK
025372 010146      MOV      R1,-(SP)  ;;PUSH R1 ON STACK
025374 010246      MOV      R2,-(SP)  ;;PUSH R2 ON STACK
025376 104411      1$: ROLIN      ;;READ AN ASCII LINE
025400 012600      MOV      (SP)+,R0  ;;ADDRESS OF 1ST CHAR.
025402 010037 025526 MOV      R0,6$  ;;SAVE INCASE OF BAD INPUT
025406 005046      CLR      -(SP)  ;;CLEAR DATA WORD
025410 005002      CLR      R2  ;;SIGN SET POSITIVE
025412 122710 000055 CMPB      #'-(R0)  ;;SEE IF A MINUS SIGN WAS TYPED
025416 001001      BNE      2$  ;;BR IF NO MINUS SIGN
025420 112002      MOVB      (R0)+,R2  ;;SAVE FOR LATER USE
025422 112001      2$: MOVB      (R0)+,R1  ;;PICKUP THIS CHARACTER
025424 001424      BEQ      3$  ;;GET OUT IF ZERO
025426 122701 000060 CMPB      #'0,R1  ;;MAKE SURE THIS CHARACTER
025432 003032      BGT      5$  ;;IS A DIGIT BETWEEN 0 & 9
025434 122701 000071 CMPB      #'9,R1
025440 002427      BLT      5$
025442 032716 170000 BIT      #C7777,(SP)  ;;DON'T LET NUMBER GET TO BIG
025446 001024      BNE      5$  ;;BR IF NUMBER WOULD OVERFLOW
025450 006316      ASL      (SP)  ;;*2
025452 011646      MOV      (SP),-(SP)  ;;SAVE FOR LATER
025454 006316      ASL      (SP)  ;;*4
025456 006316      ASL      (SP)  ;;*8
025460 062616      ADD      (SP)+,(SP)  ;;*10
025462 102416      BVS      5$  ;;OVERFLOW ISN'T ALLOWED
025464 162701 000060 SUB      #'0,R1  ;;STRIP AWAY THE ASCII JUNK
025470 060116      ADD      R1,(SP)  ;;ADD IN THIS DIGIT
025472 102412      BVS      5$  ;;OVERFLOW ISN'T ALLOWED
025474 000752      BR       2$  ;;LOOP
025476 005702      3$: TST      R2  ;;CHECK IF NUMBER IS NEG

```

READ A DECIMAL NUMBER FROM THE TTY

```

025500 001401      BEQ      4:      ;;BR IF NO
025502 005416      NEG      (SP)      ;;YES--NEGATE THE NUMBER
025504 012666 000012 4:      MOV      (SP)+,12(SP)  ;;SAVE THE RESULT
025510 012602      MOV      (SP)+,R2    ;;POP STACK INTO R2
025512 012601      MOV      (SP)+,R1    ;;POP STACK INTO R1
025514 012600      MOV      (SP)+,R0    ;;POP STACK INTO R0
025516 000002      RTI                ;;RETURN
025520 005726      5:      TST      (SP)+    ;;CLEAN PARTIAL NUMBER FROM STACK
025522 105010      CLRB      (R0)        ;;SET A TERMINATOR
025524 104401      TYPE                ;;TYPE THE INPUT UP TO BAD CHAR.
025526 000000      6:      .WORD      0      ;;POINTER GOES HERE
025530 104401 001174  TYPE      ,#QUES    ;;?" "CR" &"LF"
025534 000720      BR        1:          ;;TRY AGAIN

;ENABLE MOB 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
$ERROR:
025536 005737 002574  TST      UQUIET      ;;TEST FOR USER-QUIET MODE
025542 001403      BEQ      9:          ;;BRANCH IF FIELD-SERVICE MODE
025544 005000      CLR      R0          ;;IN CASE R0 HAS A #3 IN IT (+C)
025546 004737 025760  JSR      PC,ABORT    ;;TEST FOR ABORT CONDITION
025552 104407      9:      CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
025554 052737 001000 177520  BIS      #BIT09,BCSR  ;;ENABLE
025562 105237 001103 7:      INCB      #ERFLG      ;;SET THE ERROR FLAG
025566 001775      BEQ      7:          ;;DON'T LET THE FLAG GO TO ZERO
025570 013777 001102 153344  MOV      #TSTNM,BDISP  ;;DISPLAY TEST NUMBER AND ERROR FLAG
025576 032777 002000 153334  BIT      #BIT10,BSWR  ;;BELL ON ERROR?
025604 001402      BEQ      1:          ;;NO - SKIP
025606 104401 001170      TYPE      ,#BELL    ;;RING BELL
025612 005237 001112      INC      #ERTTL      ;;COUNT THE NUMBER OF ERRORS
025616 011637 001116      MOV      (SP),#ERRPC  ;;GET ADDRESS OF ERROR INSTRUCTION
025622 162737 000002 001116  SUB      #2,#ERRPC
025630 117737 153262 001114  MOVB      #ERRPC,#ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
025636 032777 020000 153274  BIT      #BIT13,BSWR  ;;SKIP TYPEOUT IF SET
025644 001004      BNE      20:          ;;SKIP TYPEOUTS
025646 004737 026120      JSR      PC,ERTYPE  ;;GO TO USER ERROR ROUTINE
025652 104401 001175      TYPE      ,#CRLF
025656 122737 000001 001220 20:      CMPB      #APTENV,#ENV      ;;RUNNING IN APT MODE
025664 001007      BNE      2:          ;;NO,SKIP APT ERROR REPORT
025666 113737 001114 025700  MOVB      #ITEMB,21:  ;;SET ITEM NUMBER AS ERROR NUMBER
025674 004737 026160      JSR      PC,#ATYA    ;;REPORT FATAL ERROR TO APT
025700 000      21:      .BYTE      0
025701 000      BYTE      0

```

ERROR HANDLER ROUTINE

```

025702 000777          22: BR      22:      ;; APT ERROR LOOP
025704 005777 153230  2:  TST      BSWR      ;; HALT ON ERROR
025710 100002          3:  BPL      3:      ;; KIP IF CONTINUE
025712 000000          4:  HALT      4:      ;; HALT ON ERROR!
025714 104407          5:  CKSWR      5:      ;; TEST FOR CHANGE IN SOFT-SWR
025716 032777 001000 153214 3:  BIT      @BIT09,BSWR  ;; LOOP ON ERROR SWITCH SET?
025724 001402          4:  BEQ      4:      ;; BR IF NO
025726 013716 001110          5:  MOV      $LPERR,(SP)  ;; FUDGE RETURN FOR LOOPING
025732 005737 001166          4:  TST      $ESCAPE      ;; CHECK FOR AN ESCAPE ADDRESS
025736 001402          5:  BEQ      5:      ;; BR IF NONE
025740 013716 001166          5:  MOV      $ESCAPE,(SP)  ;; FUDGE RETURN ADDRESS FOR ESCAPE
025744          6:  CMP      @ENDAD,B#42  ;; ACT-11 AUTO-ACCEPT?
025744 022737 023142 000042 6:  BNE      6:      ;; BRANCH IF NO
025752 001001          7:  HALT      7:      ;; YES
025754 000000          8:  RTI          ;; RETURN
025756 000002          9:  .SBTTL  ABORT ROUTINE FOR LCF/ORION UFD MODE
025760 005737 002572 10:  TST      UFDLFG      ;; TEST FOR USER FRIENDLY MODE
025764 001454          11:  BEQ      NOABRT      ;; IF NOT UFD THEN CONTINUE NORMAL OPERATION
025766 020027 000032          12:  CMP      R0,#32      ;; IS IT A +? ?
025772 001443          13:  BEQ      ABORTZ      ;; JUST GO BACK TO CHAIN IF IT IS (NO ERROR)
025774 020027 000003          14:  CMP      R0,#3      ;; IS IS A +C ?
026000 001404          15:  BEQ      ABORTC      ;; BR TO LOAD +C ON XXDP+ STACK (NO ERROR)
026002 005737 002574          16:  TST      UQUIET      ;; TEST FOR USER-QUIET MODE
026006 001443          17:  BEQ      NOABRT      ;; IF FIELD-SERVICE MODE, CONTINUE NORMAL OPERATION
026010 000422          18:  BR      ABORTE      ;; BECAUSE FIELD-SERVICE MODE DOES NOT QUIT ON ERROR
026012 013737 002566 000030 19:  MOV      SAV30,30      ;; SET DRSERR THEN LEAVE
026020 013737 002570 000032 20:  MOV      SAV32,32      ;; RESTORE EMT LOCATION (30)
026026 104043          21:  EMT      +43      ;; RESTORE EMT PRIORITY LOCATION (32)
026030 005720          22:  TST      (R0)+      ;; GET XXDP STACK LOC. INTO R0 FROM MONITOR
026032 001376          23:  BNE      1:      ;; FIND END OF STACK
026034 112760 000057 177777 24:  MOVB      #' /, -1(R0)  ;; LOAD SLASH OVER ZERO
026042 112720 000136          25:  MOVB      #' +, (R0)+  ;; LOAD UPARROW
026046 112720 000103          26:  MOVB      #' C, (R0)+  ;; LOAD C
026052 105010          27:  CLRB      (R0)      ;; MAKE NEW END TO STACK
026054 000412          28:  BR      ABORTZ      ;; NOW LEAVE
026056 013737 002566 000030 29:  MOV      SAV30,30      ;; RESTORE EMT LOCATION (30)
026064 013737 002570 000032 30:  MOV      SAV32,32      ;; RESTORE EMT PRIORITY LOCATION (32)
026072 104042          31:  EMT      +42      ;; GET DCA LOCATION INTO R0 FROM MONITOR
026074 012760 177777 000042 32:  MOV      #-1,42(R0)  ;; SET A -1 INTO LOCATION DRSERR IN MONITOR
026102 013700 000042          33:  MOV      B#42,R0      ;; AND PUT THE MONITOR RETURN ADDRESS IN R0
026106 005037 000042          34:  CLR      B#42      ;; CLEAR MONITOR RETURN FLAG
026112 000137 023142          35:  JMP      $ENDAD      ;; RETURN TO MONITOR-DO NOT PUSH STACK HERE
026116 000207          36:  NOABRT: RTS      PC      ;; IF NOTUFD RETURN TO MAINLINE
5503          37:  ;* THIS ROUTINE LOADS TEST NUMBER AND ERROR NUMBER AND THEN CALLS $ERRTYP
5504 026120 113737 001102 001716 38:  ERTYPE: MOVB      $TSTNM,TEST      ;; LOAD TEST NUMBER
5505 026126 113737 001114 001714 39:  MOVB      $ITEMB,ERRNUM      ;; AND ERROR NUMBER
5506 026134 004737 026410          40:  JSR      PC,$ERRTYP      ;; GO REPORT ERRORS
5507 026140 000207          41:  RTS      PC      ;; RETURN
5508          42:  .SBTTL  APT COMMUNICATIONS ROUTINE
026142 112737 000001 026406 43:  $ATY1: MOVB      @1,$FFLG      ;; TO REPORT FATAL ERROR
026150 112737 000001 026404 44:  $ATY3: MOVB      @1,$MFLG      ;; TO TYPE A MESSAGE
026156 000403          45:  BR      $ATYC
026160 112737 000001 026406 46:  $ATY4: MOVB      @1,$FFLG      ;; TO ONLY REPORT FATAL ERROR

```

APT COMMUNICATIONS ROUTINE

```

026166          $ATYC:
026166 010046      MOV      R0,-(SP)          ;; PUSH R0 ON STACK
026170 010146      MOV      R1,-(SP)          ;; PUSH R1 ON STACK
026172 105737 026404      TSTB     $MFLG          ;; SHOULD TYPE A MESSAGE?
026176 001450      BEQ       5$                ;; IF NOT: BR
026200 122737 000001 001220      CMPB    $APTENV,$ENV          ;; OPERATING UNDER APT?
026206 001031      BNE       3$                ;; IF NOT: BR
026210 132737 000100 001221      BITB    $APTSPool,$ENVm      ;; SHOULD SPOOL MESSAGES?
026216 001425      BEQ       3$                ;; IF NOT: BR
026220 017600 000004      MOV      $4(SP),R0          ;; GET MESSAGE ADDR.
026224 062766 000002 000004      ADD     $2,4(SP)          ;; BUMP RETURN ADDR.
026232 005737 001200      1$:      TST      SGTYPE          ;; SEE IF DONE W/ LAST XMISSION?
026236 001375      BNE       1$                ;; IF NOT: WAIT
026240 010037 001214      MCV      R0,$MSGAD          ;; PUT ADDR IN MAILBOX
026244 105720      2$:      TSTB     (R0)+          ;; FIND END OF MESSAGE
026246 001376      BNE       2$                ;;
026250 163700 001214      SUB      $MSGAD,R0          ;; SUB START OF MESSAGE
026254 006200      ASR       R0                ;; GET MESSAGE LGTH IN WORDS
026256 010037 001216      MOV      R0,$MSGLGT          ;; PUT LENGTH IN MAILBOX
026262 012737 000004 001200      MOV     $4,$MSGTYPE          ;; TELL APT TO TAKE MSG.
026270 000413      BR        5$
026272 017637 000004 026316 3$:      MOV     $4(SP),4$          ;; PUT MSG ADDR IN JSR LINKAGE
026300 062766 000002 000004      ADD     $2,4(SP)          ;; BUMP RETURN ADDRESS
026306 013746 177776      MOV     177776,-(SP)          ;; PUSH 177776 ON STACK
026312 004737 023464      JSR      PC,$TYPE          ;; CALL TYPE MACRO
026316 000000      4$:      .WORD    0
026320      5$:
026320 105737 026406      10$:     TSTB     $FFLG          ;; SHOULD REPORT FATAL ERROR?
026324 001416      BEQ       12$                ;; IF NOT: BR
026326 005737 001220      TST      $ENV          ;; RUNNING UNDER APT?
026332 001413      BEQ       12$                ;; IF NOT: BR
026334 005737 001200      11$:     TST      $MSGTYPE          ;; FINISHED LAST MESSAGE?
026340 001375      BNE       11$                ;; IF NOT: WAIT
026342 017637 000004 001202      MOV     $4(SP),$FATAL          ;; GET ERROR #
026350 062766 000002 000004      ADD     $2,4(SP)          ;; BUMP RETURN ADDR.
026356 005237 001200      INC      $MSGTYPE          ;; TELL APT TO TAKE ERROR
026362 105037 026406      12$:     CLRB     $FFLG          ;; CLEAR FATAL FLAG
026366 105037 026405      CLRB     $LFLG          ;; CLEAR LOG FLAG
026372 105037 026404      CLRB     $MFLG          ;; CLEAR MESSAGE FLAG
026376 012601      MOV     (SP)+,R1          ;; POP STACK INTO R1
026400 012600      MOV     (SP)+,R0          ;; POP STACK INTO R0
026402 000207      RTS       PC                ;; RETURN
026404      000      $MFLG: .BYTE    0          ;; MESSG. FLAG
026405      000      $LFLG: .BYTE    0          ;; LOG FLAG
026406      000      $FFLG: .BYTE    0          ;; FATAL FLAG

```

000200
000001
000100
000040

APTSIZE=200
APTENV=001
APTSPool=100
APTCsup=040

5509

.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:

026410

026410 104401 001175

TYPE , \$CRLF

;; "CARRIAGE RETURN & "LINE FEED"

ERROR MESSAGE TYPEOUT ROUTINE

```

026414 010046      MOV      R0,-(SP)      ;;SAVE R0
026416 005000      CLR      R0          ;;PICKUP THE ITEM INDEX
026420 153700 001114 B1SB      B00ITEMB,R0
026424 001004      BNE      11          ;;IF ITEM NUMBER IS ZERO, JUST
                                ;;TYPE THE PC OF THE ERROR
026426 013746 001116 MOV      $ERRPC,-(SP) ;;SAVE $ERRPC FOR TYPEOUT
                                ;;ERROR ADDRESS
026432 104402      TYP0C      ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
026434 000426      BR      61          ;;GET OUT
026436 005300      11: DEC      R0          ;;ADJUST THE INDEX SO THAT IT WILL
026440 006300      ASL      R0          ;;WORK FOR THE ENRRUR TABLE
026442 006300      ASL      R0
026444 006300      ASL      R0
026446 062700 001324 ADD      0$ERRTB,R0      ;;FORM TABLE POINTER
026452 012037 026462 MOV      (R0),21      ;;PICKUP "ERROR MESSAGE" POINTER
026456 001404      BEQ      31          ;;SKIP TYPEOUT IF NO POINTER
026460 104401      TYPE      ;;TYPE THE "ERROR MESSAGE"
026462 000000      21: .WORD      0          ;;"ERROR MESSAGE" POINTER GOES HERE
026464 104401 001175 TYPE      ,1CRLF      ;;"CARRIAGE RETURN" & "LINE FEED"
026470 012037 026500 31: MOV      (R0),41      ;;PICKUP "DATA HEADER" POINTER
026474 001404      BEQ      51          ;;SKIP TYPEOUT IF 0
026476 104401      TYPE      ;;TYPE THE "DATA HEADER"
026500 000000      41: .WORD      0          ;;"DATA HEADER" POINTER GOES HERE
026502 104401 001175 TYPE      ,1CRLF      ;;"CARRIAGE RETURN" & "LINE FEED"
026506 011000      51: MOV      (R0),R0      ;;PICKUP "DATA TABLE" POINTER
026510 001004      BNE      71          ;;GO TYPE THE DATA
026512 012600      61: MOV      (SP),R0      ;;RESTORE R0
026514 104401 001175 TYPE      ,1CRLF      ;;"CARRIAGE RETURN" & "LINE FEED"
026520 000207      RTS      PC          ;;RETURN
026522          71:
026522 013046      MOV      B(R0),-(SP) ;;SAVE B(R0). FOR TYPEOUT
026524 104402      TYP0C      ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
026526 005710      TST      (R0)        ;;IS THERE ANOTHER NUMBER?
026530 001770      BEQ      61          ;;BR IF NO
026532 104401 026540 TYPE      ,81          ;;TYPE TWO(2) SPACES
026536 000771      BR      71          ;;LOOP
026540          81: .ASCIZ      / /          ;;TWO(2) SPACES
          .EVEN

```

5510

.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.

```

```

026544 010046      $TRAP: MOV      R0,-(SP)      ;;SAVE R0
026546 016600 000002 MOV      2(SP),R0      ;;GET TRAP ADDRESS
026552 005740      TST      -(R0)        ;;BACKUP BY 2
026554 111000      MOVB     (R0),R0      ;;GET RIGHT BYTE OF TRAP
026556 006300      ASL      R0          ;;POSITION FOR INDEXING
026560 016000 026600 MOV      $TRPAD(R0),R0 ;;INDEX TO TABLE
026564 000200      RTS      R0          ;;GO TO ROUTINE

;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
026566 011646      $TRAP2: MOV      (SP),-(SP) ;;MOVE THE PC DOWN
026570 016666 000004 000002 MOV      4(SP),2(SP) ;;MOVE THE PSW DOWN
026576 000002      RTI          ;;RESTORE THE PSW

```

.SBTTL TRAP TABLE

```

;THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED

```

TRAP TABLE

5511

026600 026566
 026602 023464
 026604 024044
 026606 024020
 026610 024060
 026612 024246
 026614 024542
 026616 024472
 026620 024754
 026622 025104
 026624 025256
 026626 025360

BY THE "TRAP" INSTRUCTION.
 ROUTINE

TRAPAD: .WORD \$TRAP2
 \$TYPE \$CALL=TYPE TRAP.1(104401) TTY TYPEOUT ROUTINE
 \$TYPOC \$CALL=TYPOC TRAP.2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
 \$TYPDS \$CALL=TYPDS TRAP.3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
 \$TYPON \$CALL=TYPON TRAP.4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
 \$TYPDS \$CALL=TYPDS TRAP.5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
 \$GTSMR \$CALL=GTSMR TRAP.6(104406) GET SOFT-SMR SETTING
 \$CKSMR \$CALL=CKSMR TRAP.7(104407) TEST FOR CHANGE IN SOFT-SMR
 \$RDCHR \$CALL=RDCHR TRAP.10(104410) TTY TYPEIN CHARACTER ROUTINE
 \$RDLIN \$CALL=RDLIN TRAP.11(104411) TTY TYPEIN STRING ROUTINE
 \$RDOCT \$CALL=RDOCT TRAP.12(104412) READ AN OCTAL NUMBER FROM TTY
 \$RDECD \$CALL=RDECD TRAP.13(104413) READ A DECIMAL NUMBER FROM TTY

SBTTL POWER DOWN AND UP ROUTINES

POWER DOWN ROUTINE

026630 012737 026770 000024 \$PWRDN: MOV \$ILLUP,\$PWRVEC \$SET FOR FAST UP
 026632 012737 000340 000026 MOV \$340,\$PWRVEC+2 \$PRIO:7
 026644 010046 MOV R0,-(SP) \$PUSH R0 ON STACK
 026646 010146 MOV R1,-(SP) \$PUSH R1 ON STACK
 026650 010246 MOV R2,-(SP) \$PUSH R2 ON STACK
 026652 010346 MOV R3,-(SP) \$PUSH R3 ON STACK
 026654 010446 MOV R4,-(SP) \$PUSH R4 ON STACK
 026656 010546 MOV R5,-(SP) \$PUSH R5 ON STACK
 026660 017746 152254 MOV \$SMR,-(SP) \$PUSH \$SMR ON STACK
 026664 010637 026774 MOV SP,\$SAVR6 \$SAVE SP
 026670 012737 026702 000024 MOV \$PWRUP,\$PWRVEC \$SET UP VECTOR
 026676 000000 HALT
 026700 000776 BR -2 \$HANG UP

POWER UP ROUTINE

026702 012737 026770 000024 \$PWRUP: MOV \$ILLUP,\$PWRVEC \$SET FOR FAST DOWN
 026710 013706 026774 MOV \$SAVR6,SP \$GET SP
 026714 005037 026774 CLR \$SAVR6 \$WAIT LOOP FOR THE TTY
 026720 005237 026774 \$INC \$SAVR6 \$WAIT FOR THE INC
 026724 001375 \$BNE \$1 \$OF WORD
 026726 012677 152206 MOV (SP), \$SMR \$POP STACK INTO \$SMR
 026732 012605 MOV (SP), R5 \$POP STACK INTO R5
 026734 012604 MOV (SP), R4 \$POP STACK INTO R4
 026736 012603 MOV (SP), R3 \$POP STACK INTO R3
 026740 012602 MOV (SP), R2 \$POP STACK INTO R2
 026742 012601 MOV (SP), R1 \$POP STACK INTO R1
 026744 012600 MOV (SP), R0 \$POP STACK INTO R0
 026746 012737 026630 000024 MOV \$PWRDN,\$PWRVEC \$SET UP THE POWER DOWN VECTOR
 026754 012737 000340 000026 MOV \$340,\$PWRVEC+2 \$PRIO:7
 026762 104401 TYPE \$REPORT THE POWER FAILURE
 026764 026776 \$PWRMG: .WORD \$POWER \$POWER FAIL MESSAGE POINTER
 026766 000002 RTI
 026770 000000 \$ILLUP: HALT \$THE POWER UP SEQUENCE WAS STARTED
 026772 000776 BR -2 \$BEFORE THE POWER DOWN WAS COMPLETE
 026774 000000 \$SAVR6: 0 \$PUT THE SP HERE
 026776 015 012 120 \$POWER: .ASCIZ <15><12>"POWER"
 027001 117 127 105
 027004 122 000

.EVEN

POWER DOWN AND UP ROUTINES

```

5512
5513
5514      ; ERROR MESSAGES
5515      ;
5516
5517 027006      124      111      .15  EM1:      ASCIIZ /TIMEOUT ON ACCESSING A MAP REGISTER/
      027011      105      117      125
      027014      124      040      117
      027017      116      040      101
      027022      103      103      105
      027025      123      123      111
      027030      116      107      040
      027033      101      040      115
      027036      101      120      040
      027041      122      105      107
      027044      111      123      124
      027047      105      122      000
5518 027052      115      101      120  EM2:      .ASCIIZ /MAP REGISTER COULD NOT BE CLEARED/
      027055      040      122      105
      027060      107      111      123
      027063      124      105      122
      027066      040      103      117
      027071      125      114      104
      027074      040      116      117
      027077      124      040      102
      027102      105      040      103
      027105      114      105      101
      027110      122      105      104
      027113      000
5519 027114      115      101      120  EM3:      .ASCIIZ /MAP REGISTER COULD NOT HOLD PATTERN/
      027117      040      122      105
      027122      107      111      123
      027125      124      105      122
      027130      040      103      117
      027133      125      114      104
      027136      040      116      117
      027141      124      040      110
      027144      117      114      104
      027147      040      120      101
      027152      124      124      105
      027155      122      116      000
5520 027160      115      101      120  EM4:      .ASCIIZ /MAP REGISTER HAS NOT BEEN ADDRESSED CORRECTLY/
      027163      040      122      105
      027166      107      111      123
      027171      124      105      122
      027174      040      110      101
      027177      123      040      116
      027202      117      124      040
      027205      102      105      105
      027210      116      040      101
      027213      104      104      122
      027216      105      123      123
      027221      105      104      040
      027224      103      117      122
      027227      122      105      103
      027232      124      114      131
      027235      000

```

POWER DOWN AND UP ROUTINES

5521	027236	124	110	105	EMS:	.ASCIZ /THERE WAS NO DIFFERENCE FOUND BETWEEN HI AND LO MAP REGISTERS/
	027241	122	105	040		
	027244	127	101	123		
	027247	040	116	117		
	027252	040	104	111		
	027255	106	106	105		
	027260	122	105	116		
	027263	103	105	040		
	027266	106	117	125		
	027271	116	104	040		
	027274	102	105	124		
	027277	127	105	105		
	027302	116	040	110		
	027305	111	040	101		
	027310	116	104	040		
	027313	114	117	040		
	027316	115	101	120		
	027321	040	122	105		
	027324	107	111	123		
	027327	124	105	122		
	027332	123	000			
5522	027334	105	122	122	EM6:	.ASCIZ /ERROR IN BITS 3-6,9-14 IN THE DCSR/
	027337	117	122	040		
	027342	111	116	040		
	027345	102	111	124		
	027350	123	040	063		
	027353	055	066	054		
	027356	071	055	061		
	027361	064	040	111		
	027364	116	040	124		
	027367	110	105	040		
	027372	104	103	123		
	027375	122	000			
5523	027377	104	103	123	EM7:	.ASCIZ /DCSR DID NOT RESPOND PROPERLY ON RESET/
	027402	122	040	104		
	027405	111	104	040		
	027410	116	117	124		
	027413	040	122	105		
	027416	123	120	117		
	027421	116	104	040		
	027424	120	122	117		
	027427	120	117	122		
	027432	114	131	040		
	027435	117	116	040		
	027440	122	105	123		
	027443	105	124	000		
5524	027446	124	111	115	EM10:	.ASCIZ /TIMEOUT HAS OCCURED ON ACCESS TO THE DCSR/
	027451	105	117	125		
	027454	124	040	110		
	027457	101	123	040		
	027462	117	103	103		
	027465	125	122	105		
	027470	104	040	117		
	027473	116	040	101		
	027476	103	103	105		
	027501	123	123	040		
	027504	124	117	040		

POWER DOWN AND UP ROUTINES

	027507	124	110	105	
	027512	040	104	103	
	027515	123	122	000	
5525	027520	113	115	103	EM11: .ASCIZ /KMCR BITS 0-5,8 DID NOT GET SET CORRECTLY/
	027523	122	040	102	
	027526	111	124	123	
	027531	040	060	055	
	027534	065	054	070	
	027537	040	104	111	
	027542	104	040	116	
	027545	117	124	040	
	027550	107	105	124	
	027553	040	123	105	
	027556	124	040	103	
	027561	117	122	122	
	027564	105	103	124	
	027567	114	131	000	
5526	027572	124	111	115	EM12: .ASCIZ /TIMEOUT HAS OCCURED ON ACCESS TO THE KMCR/
	027575	105	117	125	
	027600	124	040	110	
	027603	101	123	040	
	027606	117	103	103	
	027611	125	122	105	
	027614	104	040	117	
	027617	116	040	101	
	027622	103	103	105	
	027625	123	123	040	
	027630	124	117	040	
	027633	124	110	105	
	027636	040	113	115	
	027641	103	122	000	
5527	027644	105	122	122	EM13: .ASCIZ /ERROR IN DATA PATH PERFORMING DATA OUT/
	027647	117	122	040	
	027652	111	116	040	
	027655	104	101	124	
	027660	101	040	120	
	027663	101	124	110	
	027666	040	120	105	
	027671	122	106	117	
	027674	122	115	111	
	027677	116	107	040	
	027702	104	101	124	
	027705	101	040	117	
	027710	125	124	000	
5528	027713	105	122	122	EM14: .ASCIZ /ERROR IN DATA OUT CYCLE/
	027716	117	122	040	
	027721	111	116	040	
	027724	104	101	124	
	027727	101	040	117	
	027732	125	124	040	
	027735	103	131	103	
	027740	114	105	000	
5529	027743	105	122	122	EM15: .ASCIZ /ERROR IN DATA IN CYCLE/
	027746	117	122	040	
	027751	111	116	040	
	027754	104	101	124	
	027757	101	040	111	

POWER DOWN AND UP ROUTINES

	027762	116	040	103	
	027765	131	103	114	
	027770	105	000		
5: 30	027772	104	104	122	EM16: .ASCIZ /DDR IS NOT 0, WHEN DCRS<2-1> SELECTS UNIBUS LINES/
	027775	040	111	123	
	030000	040	116	117	
	030003	124	040	060	
	030006	054	040	127	
	030011	110	105	116	
	030014	040	104	103	
	030017	122	123	074	
	030022	062	055	061	
	030025	076	040	123	
	030030	105	114	105	
	030033	103	124	123	
	030036	040	125	116	
	030041	111	102	125	
	030044	123	040	114	
	030047	111	116	105	
	030052	123	000		
5531	030054	104	103	123	EM17: .ASCIZ /DCSR<07> DOES NOT BECOME 1/
	030057	122	074	060	
	030062	067	076	040	
	030065	104	117	105	
	030070	123	040	116	
	030073	117	124	040	
	030076	102	105	103	
	030101	117	115	105	
	030104	040	061	000	
5532	030107	111	116	104	EM20: .ASCIZ /INDIRECT ADDRESSING OF MAPPING REGISTER PAIRS/
	030112	111	122	105	
	030115	103	124	040	
	030120	101	104	104	
	030123	122	105	123	
	030126	123	111	116	
	030131	107	040	117	
	030134	106	040	115	
	030137	101	120	120	
	030142	111	116	107	
	030145	040	122	105	
	030150	107	111	123	
	030153	124	105	122	
	030156	040	120	101	
	030161	111	122	123	
	030164	000			
5533	030165	122	105	107	EM21: .ASCIZ /REGISTER PAIR 40 PERFORMS RELOCATION/
	030170	111	123	124	
	030173	105	122	040	
	030176	120	101	111	
	030201	122	040	064	
	030204	060	040	120	
	030207	105	122	106	
	030212	117	122	115	
	030215	123	040	122	
	030220	105	114	117	
	030223	103	101	124	
	030226	111	117	116	

POWER DOWN AND UP ROUTINES

	030231	000			
5534	030232	116	130	115	EM22: .ASCIZ /NXM CONDITION COULD NOT BE CREATED THRU ALU/
	030235	040	103	117	
	030240	116	104	111	
	030243	124	111	117	
	030246	116	040	103	
	030251	117	125	114	
	030254	104	040	116	
	030257	117	124	040	
	030262	102	105	040	
	030265	103	122	105	
	030270	101	124	105	
	030273	104	040	124	
	030276	110	122	125	
	030301	040	101	114	
	030304	125	000		
5535	030306	101	114	125	EM23: .ASCIZ /ALU ERROR/
	030311	040	105	122	
	030314	122	117	122	
	030317	000			
5536	030320	103	120	125	EM24: .ASCIZ /CPU CACHE ERROR/
	030323	040	103	101	
	030326	103	110	105	
	030331	040	105	122	
	030334	122	117	122	
	030337	000			
5537	030340	113	115	103	EM25: .ASCIZ /KMCR<4-0> DON'T DISABLE MEMORY RESPONSE/
	030343	122	074	064	
	030346	055	060	076	
	030351	040	104	117	
	030354	116	047	124	
	030357	040	104	111	
	030362	123	101	102	
	030365	114	105	040	
	030370	115	105	115	
	030373	117	122	131	
	030376	040	122	105	
	030401	123	120	117	
	030404	116	123	105	
	030407	000			
5538	030410	113	115	103	EM26: .ASCIZ /KMCR DOES NOT REFLECT EXPECTED STATUS OF THE CACHE/
	030413	122	040	104	
	030416	117	105	123	
	030421	040	116	117	
	030424	124	040	122	
	030427	105	106	114	
	030432	105	103	124	
	030435	040	105	130	
	030440	120	105	103	
	030443	124	105	104	
	030446	040	123	124	
	030451	101	124	125	
	030454	123	040	117	
	030457	106	040	124	
	030462	110	105	040	
	030465	103	101	103	
	030470	110	105	000	

POWER DOWN AND UP ROUTINES

5539	030473	105	122	122	EM27: .ASCIZ /ERROR IN CACHE TAG REGISTERS/
	030476	117	122	040	
	030501	111	116	040	
	030504	103	101	103	
	030507	110	105	040	
	030512	124	101	107	
	030515	040	122	105	
	030520	107	111	123	
	030523	124	105	122	
	030526	123	000		
5540	030530	105	122	122	EM30: .ASCIZ /ERROR IN THE DMA CACHE DATA RAMS/
	030533	117	122	040	
	030536	111	116	040	
	030541	124	110	105	
	030544	040	104	115	
	030547	101	040	103	
	030552	101	103	110	
	030555	105	040	104	
	030560	101	124	101	
	030563	040	122	101	
	030566	115	123	000	
5541	030571	105	122	122	EM31: .ASCIZ /ERROR IN THE M9312 BOOT ROM SECTION/
	030574	117	122	040	
	030577	111	116	040	
	030602	124	110	105	
	030605	040	115	071	
	030610	063	061	062	
	030613	040	102	117	
	030616	117	124	040	
	030621	122	117	115	
	030624	040	123	105	
	030627	103	124	111	
	030632	117	116	000	
5542	030635	105	122	122	EM32: .ASCIZ /ERROR IN ARBITRATION LOGIC USING UBE/
	030640	117	122	040	
	030643	111	116	040	
	030646	101	122	102	
	030651	111	124	122	
	030654	101	124	111	
	030657	117	116	040	
	030662	114	117	107	
	030665	111	103	040	
	030670	125	123	111	
	030673	116	107	040	
	030676	125	102	105	
	030701	000			
5543	030702	105	122	122	EM33: .ASCIZ /ERROR TRYING TO EXECUTE DMA CYCLES THRU UBE/
	030705	117	122	040	
	030710	124	122	131	
	030713	111	116	107	
	030716	040	124	117	
	030721	040	105	130	
	030724	105	103	125	
	030727	124	105	040	
	030732	104	115	101	
	030735	040	103	131	
	030740	103	114	105	

POWER DOWN AND UP ROUTINES

	030743	123	040	124	
	030746	110	122	125	
	030751	040	125	102	
	030754	105	000		
5544	030756	105	122	122	EM34: .ASCIZ /ERROR IN THE UNIBUS MEMORY TEST/
	030761	117	122	040	
	030764	111	116	040	
	030767	124	110	105	
	030772	040	125	116	
	030775	111	102	125	
	031000	123	040	115	
	031003	105	115	117	
	031006	122	131	040	
	031011	124	105	123	
	031014	124	000		
5545	031016	125	116	105	EM35: .ASCIZ /UNEXPECTED TIMEOUT HAS OCCURED/
	031021	130	120	105	
	031024	103	124	105	
	031027	104	040	124	
	031032	111	115	105	
	031035	117	125	124	
	031040	040	110	101	
	031043	123	040	117	
	031046	103	103	125	
	031051	122	105	104	
	031054	000			
5546	031055	125	116	111	EM36: .ASCIZ /UNIBUS TIMEOUT DID NOT WORK CORRECTLY/
	031060	102	125	123	
	031063	040	124	111	
	031066	115	105	117	
	031071	125	124	040	
	031074	104	111	104	
	031077	040	116	117	
	031102	124	040	127	
	031105	117	122	113	
	031110	040	103	117	
	031113	122	122	105	
	031116	103	124	114	
	031121	131	000		
5547	031123	104	103	123	EM37: .ASCIZ /DCSR<3> DID NOT DISABLE UBA ROM RESPONSE/
	031126	122	074	063	
	031131	076	040	104	
	031134	111	104	040	
	031137	116	117	124	
	031142	040	104	111	
	031145	123	101	102	
	031150	114	105	040	
	031153	125	102	101	
	031156	040	122	117	
	031161	115	040	122	
	031164	105	123	120	
	031167	117	116	123	
	031172	105	000		
5548					
5549	031174	124	105	123	DM1: .ASCII /TEST ERROR ERROR ADDRESS/<12><15>
	031177	124	011	105	
	031202	122	122	117	

POWER DOWN AND UP ROUTINES

Address	Octal	Hex	ASCII	Label	Field 1	Field 2	Field 3	Field 4	Field 5	Field 6
031205	122	011	105							
031210	122	122	117							
031213	122	011	101							
031216	104	104	122							
031221	105	123	123							
031224	012	015								
5550 031226	040	040	043		.ASCIZ	/	#	PC	#/	
031231	011	040	040							
031234	120	103	011							
031237	040	040	043							
031242	000									
5551 031243	124	105	123	DM2:	.ASCII	/TEST	ERROR	ERROR	GOOD	BAD/<12><15>
031246	124	011	105							
031251	122	122	117							
031254	122	011	105							
031257	122	122	117							
031262	122	011	107							
031265	117	117	104							
031270	011	102	101							
031273	104	012	015							
5552 031276	040	040	043		.ASCIZ	/	#	PC	#	DATA DATA ADDRESS/
031301	011	040	040							
031304	120	103	011							
031307	040	040	043							
031312	011	104	101							
031315	124	101	011							
031320	104	101	124							
031323	101	011	101							
031326	104	104	122							
031331	105	123	123							
031334	000									
5553 031335	124	105	123	DM3:	.ASCII	/TEST	ERROR	ERROR	GOOD	BAD/<12><15>
031340	124	011	105							
031343	122	122	117							
031346	122	011	105							
031351	122	122	117							
031354	122	011	107							
031357	117	117	104							
031362	011	102	101							
031365	104	012	015							
5554 031370	040	040	043		.ASCIZ	/	#	PC	#	ADDRESS ADDRESS/
031373	011	040	040							
031376	120	103	011							
031401	040	040	043							
031404	011	101	104							
031407	104	122	105							
031412	123	123	011							
031415	101	104	104							
031420	122	105	123							
031423	123	000								
5555 031425	124	105	123	DM4:	.ASCII	/TEST	ERROR	ERROR	HI	LOW/<12><15>
031430	124	011	105							
031433	122	122	117							
031436	122	011	105							
031441	122	122	117							
031444	122	011	110							
031447	111	011	114							

POWER DOWN AND UP ROUTINES

	031452	117	127	012					
	031455	015							
5556	031456	040	040	043		.ASCIZ	/	PC	MAP
	031461	011	040	040					MAP/
	031464	120	103	011					
	031467	040	040	043					
	031472	011	115	101					
	031475	120	011	115					
	031500	101	120	000					
5557	031503	124	105	123	DM5:	.ASCII	/TEST	ERROR	GOOD
	031506	124	011	105					BA0/<12><15>
	031511	122	122	117					
	031514	122	011	105					
	031517	122	122	117					
	031522	122	011	107					
	031525	117	117	104					
	031530	011	102	101					
	031533	104	012	015					
5558	031536	040	040	043		.ASCIZ	/	PC	DATA
	031541	011	040	040					DATA/
	031544	120	103	011					
	031547	040	040	043					
	031552	011	104	101					
	031555	124	101	011					
	031560	104	101	124					
	031563	101	000						
5559	031565	124	105	123	DM13:	.ASCII	/TEST	ERROR	DDR
	031570	124	011	105					PATTERN/<12><15>
	031573	122	122	117					
	031576	122	011	105					
	031601	122	122	117					
	031604	122	011	104					
	031607	104	122	011					
	031612	120	101	124					
	031615	124	105	122					
	031620	116	012	015					
5560	031623	040	040	043		.ASCIZ	/	PC	/
	031626	011	040	040					
	031631	120	103	011					
	031634	040	040	043					
	031637	000							
5561	031640	124	105	123	DM16:	.ASCII	/TEST	ERROR	ERROR/<12><15>
	031643	124	011	105					
	031646	122	122	117					
	031651	122	011	105					
	031654	122	122	117					
	031657	122	012	015					
5562	031662	040	040	043		.ASCIZ	/	PC	/
	031665	011	040	040					
	031670	120	103	011					
	031673	040	040	043					
	031676	000							
5563	031677	124	105	123	DM20:	.ASCII	/TEST	ERROR	KMCR PAIR/<12><15>
	031702	124	011	105					
	031705	122	122	117					
	031710	122	011	105					
	031713	122	122	117					

POWER DOWN AND UP ROUTINES

	031716	122	011	113									
	031721	115	103	122									
	031724	040	120	101									
	031727	111	122	012									
	031732	015											
5564	031733	040	040	043		.ASCIZ / *	PC	*		FAILED/			
	031736	011	040	040									
	031741	120	103	011									
	031744	040	040	043									
	031747	011	106	101									
	031752	111	114	105									
	031755	104	000										
5565	031757	124	105	123	DH23:	.ASCII /TEST	ERROR	ERROR	GOOD	BAD	ADDRESS/<12><15>		
	031762	124	011	105									
	031765	122	122	117									
	031770	122	011	105									
	031773	122	122	117									
	031776	122	011	107									
	032001	117	117	104									
	032004	011	102	101									
	032007	104	011	101									
	032012	104	104	122									
	032015	105	123	123									
	032020	012	015										
5566	032022	040	040	043		.ASCIZ / *	PC	*		DATA	DATA	<22-16> <13-0>/	
	032025	011	040	040									
	032030	120	103	011									
	032033	040	040	043									
	032036	011	104	101									
	032041	124	101	011									
	032044	104	101	124									
	032047	101	011	074									
	032052	062	062	055									
	032055	061	066	076									
	032060	011	074	061									
	032063	063	055	060									
	032066	076	000										
5567	032070	124	105	123	DH27:	.ASCII /TEST	ERROR	ERROR	ADDRESS	ADDRESS/<12><15>			
	032073	124	011	105									
	032076	122	122	117									
	032101	122	011	105									
	032104	122	122	117									
	032107	122	011	101									
	032112	104	104	122									
	032115	105	123	123									
	032120	011	101	104									
	032123	104	122	105									
	032126	123	123	012									
	032131	015											
5568	032132	040	040	043		.ASCIZ / *	PC	*		<22			

POWER DOWN AND UP ROUTINES

	032162	055	060	076			
	032165	000					
5569					.EVEN		
5570	032166	001716	001116	001714	DT1:	.WORD	TEST, \$ERRPC, ERRNUM, \$BDADR, 0
	032174	001122	000000				
5571	032200	001716	001116	001714	DT2:	.WORD	TEST, \$ERRPC, ERRNUM, \$GDDAT, \$BDDAT, \$BDADR, 0
	032206	001124	001126	001122			
	032214	000000					
5572	032216	001716	001116	001714	DT3:	.WORD	TEST, \$ERRPC, ERRNUM, \$GDADR, \$BDADR, 0
	032224	001120	001122	000000			
5573	032232	001716	001116	001714	DT4:	.WORD	TEST, \$ERRPC, ERRNUM, \$GDDAT, \$BDDAT, 0
	032240	001124	001126	000000			
5574	032246	001716	001116	001714	DT16:	.WORD	TEST, \$ERRPC, ERRNUM, 0
	032254	000000					
5575	032256	001716	001116	001714	DT20:	.WORD	TEST, \$ERRPC, ERRNUM, KMCR, \$BDADR, 0
	032264	177734	001122	000000			
5576	032272	001716	001122	001714	DT21:	.WORD	TEST, \$BDADR, ERRNUM, 0
	032300	000000					
5577	032302	001716	001116	001714	DT23:	.WORD	TEST, \$ERRPC, ERRNUM, \$GDDAT, \$BDDAT, KIPAF6, \$BDADR, 0
	032310	001124	001126	172354			
	032316	001122	000000				
5578	032322	001716	001116	001714	DT27:	.WORD	TEST, \$ERRPC, ERRNUM, MAPH01, MAPL01, 0
	032330	170206	170204	000000			
5579	032336	001716	001116	001714	DT30:	.WORD	TEST, \$ERRPC, ERRNUM, MAPL00, 0
	032344	170200	000000				
5580		000001				.END	

SYMBOL TABLE

ABASE -	000000	BE1BA	002154	DM1	031174	ERROR -	104000	MAPH16-	170272
ABORT	025760	BE1CC	002152	DM13	031565	ERRVEC-	000004	MAPH17-	170276
ABORTC	026012	BE1CLR	002160	DM16	031640	ERTYPE	026120	MAPH2 -	170212
ABORTE	026056	BE1CR1	002156	DM2	031243	GTSWR -	104406	MAPH20-	170302
ABORTZ	026102	BE1CR2	002162	DM20	031677	HT -	000011	MAPH21-	170306
ACDW1 -	000000	BE1DB	002150	DM23	031757	IOTVEC-	000020	MAPH22-	170312
ACDW2 -	000000	BE1INT	001724	DM27	032070	IUBE	002450	MAPH23-	170316
ACPUOP-	000000	BE1PSM	002166	DM3	031335	KDPA0-	172360	MAPH24-	170322
ADDW0 -	000000	BE1SV	015476	DM4	031425	KDPA1-	172362	MAPH25-	170326
ADDW1 -	000000	BE1VEC	002164	DM5	031503	KDPA2-	172364	MAPH26-	170332
ADDW10-	000000	SE2BA	002174	DISPLA	001142	KDPA3-	172366	MAPH27-	170336
ADDW11-	000000	BE2CC	002172	DISPRE	000174	KDPA4-	172370	MAPH3 -	170216
ADDW12-	000000	BE2CLR	002200	USWR -	177570	KDPA5-	172372	MAPH30-	170342
ADDW13-	000000	BE2CR1	002176	DT1	032166	KDPA6-	172374	MAPH31-	170346
ADDW14-	000000	BE2CR2	002202	DT16	032246	KDPA7-	172376	MAPH32-	170352
ADDW15-	000000	BE2DB	002170	DT2	032200	KDPDR0-	172320	MAPH33-	170356
ADDW2 -	000000	BE2INT	001726	DT20	032256	KDPDR1-	172322	MAPH34-	170362
ADDW3 -	000000	BE2PSM	002206	DT21	032272	KDPDR2-	172324	MAPH35-	170366
ADDW4 -	000000	BE2SV	015516	DT23	032302	KDPDR3-	172326	MAPH36-	170372
ADDW5 -	000000	BE2VEC	002204	DT27	032322	KDPDR4-	172330	MAPH37-	170376
ADDW6 -	000000	BIT0 -	000001	DT3	032216	KDPDR5-	172332	MAPH4 -	170222
ADDW7 -	000000	BIT00 -	000001	DT30	032336	KDPDR6-	172334	MAPH5 -	170226
ADDW8 -	000000	BIT01 -	000002	DT4	032232	KDPDR7-	172336	MAPH6 -	170232
ADDW9 -	000000	BIT02 -	000004	EMTSV	002500	KIPAR0-	172340	MAPH7 -	170236
ADEVCT-	000000	BIT03 -	000010	EMTVEC-	000030	KIPAR1-	172342	MAPL0 -	170200
ADEVN -	000000	BIT04 -	000020	EM1	027006	KIPAR2-	172344	MAPL00-	170200
AENV -	000000	BIT05 -	000040	EM10	027446	KIPAR3-	172346	MAPL01-	170204
AENVN -	000000	BIT06 -	000100	EM11	027520	KIPAR4-	172350	MAPL02-	170210
AFATAL-	000000	BIT07 -	000200	EM12	027572	KIPAR5-	172352	MAPL03-	170214
AMADR1-	000000	BIT08 -	000400	EM13	027644	KIPAR6-	172354	MAPL04-	170220
AMADR2-	000000	BIT09 -	001000	EM14	027713	KIPAR7-	172356	MAPL05-	170224
AMACV3-	000000	BIT1 -	000002	EM15	027743	KIPDR0-	172300	MAPL06-	170230
AMADH4-	000000	BIT10 -	002000	EM16	027772	KIPDR1-	172302	MAPL07-	170234
AMMS1-	000000	BIT11 -	004000	EM17	030054	KIPDR2-	172304	MAPL1 -	170204
AMMS2-	000000	BIT12 -	010000	EM2	027052	KIPDR3-	172306	MAPL10-	170240
AMMS3-	000000	BIT13 -	020000	EM20	030107	KIPDR4-	172310	MAPL11-	170244
AMMS4-	000000	BIT14 -	040000	EM21	030165	KIPDR5-	172312	MAPL12-	170250
AMSGAD-	000000	BIT15 -	100000	EM22	030232	KIPDR6-	172314	MAPL13-	170254
AMSLG-	000000	BIT2 -	000004	EM23	030306	KIPDR7-	172316	MAPL14-	170260
AMSGTY-	000000	BIT3 -	000010	EM24	030320	KMCR -	177734	MAPL15-	170264
ANTYP1-	000000	BIT4 -	000020	EM25	030340	LF -	000012	MAPL16-	170270
ANTYP2-	000000	BIT5 -	000040	EM26	030410	MAPH0 -	170202	MAPL17-	170274
ANTYP3-	000000	BIT6 -	000100	EM27	030473	MAPH00-	170202	MAPL2 -	170210
ANTYP4-	000000	BIT7 -	000200	EM3	027114	MAPH01-	170206	MAPL20-	170300
APASS -	000000	BIT8 -	000400	EM30	030530	MAPH02-	170212	MAPL21-	170304
APRIOR-	000000	BIT9 -	001000	EM31	030571	MAPH03-	170216	MAPL22-	170310
APTCBU-	000040	BPTVEC-	000014	EM32	030635	MAPH04-	170222	MAPL23-	170314
APTENV-	000001	CCR -	177746	EM33	030702	MAPH05-	170226	MAPL24-	170320
APTSIZ-	000200	CKSWR -	104407	EM34	030756	MAPH06-	170232	MAPL25-	170324
APTSPO-	000100	CR -	000015	EM35	031016	MAPH07-	170236	MAPL26-	170330
ASAREG-	000000	CRLF -	000200	EM36	031055	MAPH1 -	170206	MAPL27-	170334
ATESTN-	000000	CTBLE	002022	EM37	031123	MAPH10-	170242	MAPL3 -	170214
AUNIT -	000000	DCSR -	177730	EM4	027160	MAPH11-	170246	MAPL30-	170340
AUSWR -	000000	DDIN	002222	EM5	027236	MAPH12-	170252	MAPL31-	170344
AVECT1-	000000	DDISP -	177570	EM6	027334	MAPH13-	170256	MAPL32-	170350
AVECT2-	000000	DDOUT	002210	EM7	027377	MAPH14-	170262	MAPL33-	170354
BCSR -	177520	DDR -	177732	ERRNUM	001714	MAPH15-	170266	MAPL34-	170360

SYMBOL TABLE

MAPL35 =	170364	SWREG	000176	TST30	012120	IBASE	001254	IGDADR	001120
MAPL36 =	170370	SW0	000001	TST31	012334	IBDADR	001122	IGDAT	001124
MAPL37 =	170374	SW00	000001	TST32	012756	IBDAT	001126	IGET42	023132
MAPL4 =	170220	SW01	000002	TST33	013210	IBELL	001170	IGTSMR	024542
MAPL5 =	170224	SW02	000004	TST34	013634	ICDW1	001260	IHD	000001
MAPL6 =	170230	SW03	000010	TST35	014034	ICDW2	001262	IHIBTS	000232
MAPL7 =	170234	SW04	000020	TST36	014222	ICHARC	024014	IHIOCT	025356
MAPPK	002244	SW05	000040	TST37	014324	ICKSMR	024472	ICNT	001104
MCSR	001732	SW06	000100	TST4	003762	ICHTAG	001100	ILLUP	026770
MENSIZ	002332	SW07	000200	TST40	014604	ICM3	000000	INTAG	001135
MHR0	177572	SW08	000400	TST41	015144	ICM4	000002	ITEMB	001114
MHR1	177574	SW09	001000	TST42	015300	ICNTLG	025227	LF	001176
MHR2	177576	SW1	000002	TST43	015536	ICNTLU	025222	LF LG	026405
MHR3	172516	SW10	002000	TST44	015752	ICPUOP	001226	LPADR	001106
MHVEC	000250	SW11	004000	TST45	016044	ICRLF	001175	LPERR	001110
NOABRT	026116	SW12	010000	TST46	016124	IDBLK	024462	MAOR1	001232
NOCAM	007664	SW13	020000	TST47	016224	IDDW0	001264	MAOR2	001236
NROM	013576	SW14	040000	TST5	004042	IDDW1	001266	MAOR3	001242
OBADR	002140	SW15	100000	TST50	016440	IDDW10	001310	MAOR4	001246
PCR	177522	SW2	000004	TST51	016532	IDDW11	001312	MAIL	001200
PIRQ	177772	SW3	000010	TST52	016674	IDDW12	001314	MAIS1	001230
PIRQVE	000240	SW4	000020	TST53	017002	IDDW13	001316	MAIS2	001234
PHIS	001720	SW5	000040	TST54	017432	IDDW14	001320	MAIS3	001240
POLY	120001	SW6	000100	TST55	017576	IDDW15	001322	MAIS4	001244
PRC	000000	SW7	000200	TST56	017746	IDDW2	001270	MBADR	000234
PR1	000040	SW8	000400	TST57	020374	IDDW3	001272	MFLG	026404
PR2	000100	SW9	001000	TST6	004226	IDDW4	001274	MNEW	025245
PR3	000140	TBITVE	000014	TST60	020746	IDDW5	001276	MSGAD	001214
PR4	000200	TBLPH	012716	TST61	021030	IDDW6	001300	MSGLG	001216
PR5	000240	TBLPL	012656	TST62	021134	IDDW7	001302	MSGTY	001200
PR6	000300	TEST	001716	TST63	021476	IDDW8	001304	MSMR	025234
PR7	000340	TIMOUT	002234	TST64	022152	IDDW9	001306	MTYP1	001231
PS	177776	TKVEC	000060	TST65	022336	IDEVCT	001210	MTYP2	001235
PSW	177776	TOPTBL	010542	TST66	022560	IDEVH	001256	MTYP3	001241
PTRN16	001774	TOUT	001730	TST67	022740	IDOAGN	023152	MTYP4	001245
PTRN6	002010	TPVEC	000064	TST7	004404	IDTBL	024452	MXCNT	023462
PURVEC	000024	TRAPVE	000034	TYPDS	104405	IDENDAD	023142	NULL	001154
RDCHR	104410	TRTVEC	000014	TYPE	104401	IDENDCT	023022	NMTST	000001
RDEEC	104413	TST1	003274	TYPOC	104402	IDENV	001220	OCNT	024242
RDLIN	104411	TST10	004466	TYPON	104404	IDENVH	001221	OMODE	024244
RDOCT	104412	TST11	004624	TYPOS	104403	IDOP	022766	OVER	023440
RESTAR	003270	TST12	004722	UBECT	001722	IDOPCT	023014	PASS	001206
RESVEC	000010	TST13	005044	UBEM	022740	IDRFLG	001103	PASTH	000240
R6	0000006	TST14	005450	UBETST	013210	IDRMAX	001115	POWER	026776
R7	0000007	TST15	005646	UFDLFG	002572	IDRROR	025536	PWRON	026630
SAV30	002566	TST16	006052	UFDSET	000001	IDRRPC	001116	PWRMG	026764
SAV32	002570	TST17	006242	UMSIZ	002424	IDRTTB	001324	PWRUP	026702
SCOPE	000004	TST2	003420	UQUIET	002574	IDRTTY	026410	QUES	001174
SIMLGO	170014	TST20	006514	VALTBL	010526	IDRTTL	001112	RDCHR	024754
SRO	177572	TST21	007114	VHKOR	002576	IDESCAP	001166	RDEEC	025360
SR1	177574	TST22	007574	WRTBUF	001734	IDETABL	001220	RDLIN	025104
SR2	177576	TST23	007704	WPTH0	000232	IDETEND	001324	RDOCT	025256
SR3	172516	TST24	010076	WATYC	026166	IDFATAL	001202	RDSZ	000010
STACK	001100	TST25	010556	WATY1	026142	IDFFLG	026406	RTNAD	023154
START	002500	TST26	011266	WATY3	026150	IDFILLC	001156	SAVR6	026774
STKLMT	177774	TST27	011436	WATY4	026160	IDFILLS	001155	SCOPE	023162
SWR	001140	TST3	003640	WATOB	001134			SETUP	000137

KTJ11-B DIAGNOSTIC

MACRO M1200 05 OCT-84 09:24 PAGE 64-29

SEQ 0159

SYMBOL TABLE

\$STUP = 177777	\$TKS 001144	\$TRAP2 026566	\$TYPEC 023676	\$VECT1 001250
\$SVLAD 023404	\$TMP0 001160	\$TRP = 000014	\$TYPEX 024016	\$VECT2 001252
\$SVPC = 000232	\$TMP1 001162	\$TRPAD 026600	\$TYPOC 024044	\$XOFF = 000023
\$SMK = 167400	\$TN = 000070	\$TSTM 000236	\$TYPON 024060	\$XON = 000021
\$SMREG 001222	\$TPB 001152	\$TSTNM 001102	\$TYPOS 024020	\$XTSTR 023202
\$SMRMK = 000300	\$TPFLG 001157	\$TTYIN 025212	\$UNIT 001212	\$GET4 = 000000
\$TESTN 001204	\$TPS 001150	\$TYPDS 024246	\$UNITM 000242	\$OFILL 024243
\$TIMES 001164	\$TRAP 026544	\$TYPE 023464	\$USMR 001224	.\$X = 000232
\$TKB 001146				

. ABS. 032350 000
000000 001

ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 56592 WORDS (222 PAGES)

DYNAMIC MEMORY: 19748 WORDS (75 PAGES)

ELAPSED TIME: 00:02:40

OKTABO.BIC.COKTABO/CR/ML:TOC/-SP=ORION.MLB/ML.COKTABO

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES
ABASE	= 000000	9-381 9-381
ABORT	025760	64-5502 #64-5502
ABORTC	026012	64-5502 #64-5502
ABORTE	026056	64-5502 #64-5502
ABORTZ	026102	64-5502 64-5502 #64-5502
ACDW1	= 000000	9-381 9-381
ACDW2	= 000000	9-381 9-381
ACPUOP	= 000000	9-381 9-381
ADDW0	= 000000	9-381 9-381
ADDW1	= 000000	9-381 9-381
ADDW10	= 000000	9-381 9-381
ADDW11	= 000000	9-381 9-381
ADDW12	= 000000	9-381 9-381
ADDW13	= 000000	9-381 9-381
ADDW14	= 000000	9-381 9-381
ADDW15	= 000000	9-381 9-381
ADL12	= 000000	9-381 9-381
ADDW3	= 000000	9-381 9-381
ADDW4	= 000000	9-381 9-381
ADDW5	= 000000	9-381 9-381
ADDW6	= 000000	9-381 9-381
ADDW7	= 000000	9-381 9-381
ADDW8	= 000000	9-381 9-381
ADDW9	= 000000	9-381 9-381
ADEVCT	= 000000	9-381 9-381
ADEVN	= 000000	9-381 9-381
RENV	= 000000	9-381 9-381
RENV1	= 000000	9-381 9-381
AFATAL	= 000000	9-381 9-381
AMADR1	= 000000	9-381 9-381
AMADR2	= 000000	9-381 9-381
AMADR3	= 000000	9-381 9-381
AMADR4	= 000000	9-381 9-381
AMMS1	= 000000	9-381 9-381
AMMS2	= 000000	9-381 9-381
AMMS3	= 000000	9-381 9-381
AMMS4	= 000000	9-381 9-381
AMSGAD	= 000000	9-381 9-381
AMSGLG	= 000000	9-381 9-381
AMSGTY	= 000000	9-381 9-381
AMTYP1	= 000000	9-381 9-381
AMTYP2	= 000000	9-381 9-381
AMTYP3	= 000000	9-381 9-381
AMTYP4	= 000000	9-381 9-381
APASS	= 000000	9-381 9-381
APRIOR	= 000000	9-381
APTCSU	= 000040	64-5493 #64-5508
APTENV	= 000001	10-819 28-2175 37-3075 45-3845 64-5493 64-5502 64-5508 #64-5508
APTSIZ	= 000200	10-816 #64-5508
APTSPO	= 000100	64-5493 64-5508 #64-5508
ASWREG	= 000000	9-381 9-381
ATESTN	= 000000	9-381 9-381

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES
AUNIT	000000	9-381 9-381
AUSMR	000000	9-381 9-381
AVECT1	000000	9-381 9-381
AVECT2	000000	9-381 9-381
BCSR	177520	08-362 *37-3057 *45-3849 *45-3850 *60-5067 *64-5492 *64-5492 *64-5502
BE1BA	002154	010-618 40-3400 46-3910 48-3975 49-4047 50-4137 51-4210 52-4289 53-4411
		53-4444 54-4539 55-4611 56-4694 56-4740 57-4807 58-4883 59-4952 60-5043
		60-5058 60-5074 60-5093 61-5189 61-5208 61-5227 61-5251 62-5296 62-5307
		63-5404 64-5456
BE1CC	002152	010-617 40-3399 40-3404 46-3909 48-3974 49-4049 50-4139 51-4213 52-4292
		53-4414 53-4447 54-4542 55-4614 56-4698 57-4808 58-4882 59-4954 60-5045
		60-5060 60-5076 60-5095 61-5192 61-5211 61-5228 61-5254 62-5297 62-5308
		62-5315 63-5406 64-5458
BE1CLR	002160	010-620
BE1CR1	002156	010-619 40-3401 40-3402 41-3482 41-3485 41-3493 41-3497 41-3508 41-3512
		41-3523 41-3527 42-3608 42-3611 42-3625 42-3628 42-3642 42-3645 42-3659
		42-3662 43-3700 43-3711 44-3776 44-3779 44-3805 46-3912 47-3946 48-3978
		48-3979 49-4050 49-4056 50-4140 50-4146 51-4222 51-4223 52-4301 52-4302
		53-4423 53-4424 53-4456 53-4457 54-4551 54-4552 55-4621 55-4622 56-4707
		56-4708 56-4746 56-4747 57-4809 57-4810 58-4887 58-4888 59-4955 59-4956
		60-5046 60-5047 60-5061 60-5062 60-5077 60-5078 60-5096 60-5098 61-5193
		61-5194 61-5212 61-5213 61-5231 61-5233 61-5256 61-5258 62-5300 62-5301
		62-5309 62-5310 62-5316 62-5317 63-5410 63-5411 64-5462 64-5463
BE1CR2	002162	010-621 45-3864 45-3869 46-3911 46-3915 47-3944 47-3949 47-3952 49-4048
		50-4138 51-4211 52-4290 53-4412 53-4445 54-4540 55-4612 55-4624 56-4695
		56-4696 56-4710 56-4741 57-4812 59-4953 60-5044 60-5059 60-5075 60-5094
		60-5106 61-5190 61-5209 61-5240 61-5252 61-5266 62-5299 63-5405 64-5457
BE1DB	002150	010-616 10-803 39-3321 48-3982 49-4046 50-4136 50-4153 51-4212 52-4291
		52-4310 53-4413 53-4435 53-4437 53-4446 53-4470 54-4541 54-4560
		54-4562 55-4613 56-4697 56-4745 56-4749 58-4881 60-5052 60-5068 60-5083
		61-5191 61-5210 61-5253 62-5298 62-5312 62-5319 63-5423
BE1INT	001724	010-587 44-3785 *44-3792 *44-3801 *44-3807
BE1PSW	002166	010-623 41-3477 42-3601 43-3701 44-3769 55-4610 56-4702 57-4801 60-5090
		61-5230 61-5250
BE1SV	015476	44-3768 *44-3800
BE1VEC	002164	010-622 41-3481 41-3491 41-3506 41-3521 42-3605 42-3622 42-3639 42-3656
		44-3768 48-3976 55-4609 56-4701 57-4800 60-5089 61-5229 61-5249
BE2BA	002174	010-631
BE2CC	002172	010-630
BE2CLR	002200	010-633 44-3797
BE2CR1	002176	010-632 44-3778 44-3780 44-3800
BE2CR2	002202	010-634
BE2DB	002170	010-629
BE2INT	001726	010-588 44-3789 *44-3793 *44-3802 *44-3806
BE2PSW	002206	010-636 44-3771
BE2SV	015516	44-3770 *44-3805
BE2VEC	002204	010-635 44-3770
BIT0	000001	08-265 12-974
BIT00	000001	08-265 8-265 10-682 19-1417 44-3779 44-3780 57-4786 57-4789 64-5447
		64-5470
BIT01	000002	08-265 8-265 18-1369 19-1415 20-1451 20-1465 21-1539 22-1652 24-1796
		25-1862 26-1947 49-4044 49-4067 49-4070

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES
BIT02	= 000004	08-265 8-265 18-1369 19-1415 21-1539 22-1652 24-1796 25-1862 26-1947 64-5447 64-5450
BIT03	= 000010	08-265 8-265 16-1272 33-2687 33-2700 37-3140 37-3147 45-3847 47-3944
BIT04	= 000020	08-265 8-265 10-827 25-1864 26-1949 27-2107 33-2700 35-2821 38-3224 45-3864 45-3869 57-4787
BIT05	= 000040	08-265 8-265 10-777 10-827 21-1533 21-1535 21-1554 21-1563 21-1600 21-1604 22-1644 22-1646 22-1655 22-1673 23-1724 23-1726 23-1731 23-1735 23-1745 23-1746 24-1788 24-1790 24-1799 24-1811 25-1854 25-1856 25-1864 26-1941 26-1949 27-2062 27-2084 27-2111 27-2140 28-2177 29-2216 29-2218 29-2235 30-2307 30-2372 31-2475 32-2589 33-2640 33-2704 34-2736 34-2763 35-2816 35-2841 37-3077 38-3220 39-3313 45-3843 45-3850 53-4396 56-4729 57-4836 57-4840 64-5451
BIT06	= 000100	08-265 8-265 28-2170 28-2171 29-2219 30-2304 30-2373 31-2476 32-2587 32-2604 33-2639 33-2668 33-2669 33-2703 34-2737 34-2762 35-2838 35-2839 45-3849 45-3851 45-3855 63-5422 64-5452 64-5453
BIT07	= 000200	08-265 8-265 18-1373 27-2092 37-3057 45-3853 60-5067
BIT08	= 000400	08-265 8-265 16-1271 18-1368 19-1414 19-1424 20-1450 21-1538 22-1651 24-1795 25-1861 26-1946 27-2054 29-2220 29-2227 30-2317 30-2324 30-2338 30-2346 30-2354 30-2362 31-2492 31-2499 31-2509 31-2516 31-2526 31-2533 31-2543 31-2550 31-2552 32-2588 32-2590 32-2591 32-2598 32-2605 33-2638 33-2643 33-2645 33-2653 33-2659 33-2683 33-2705 34-2735 34-2745 34-2747 34-2751 34-2756 34-2764 35-2818 35-2840 35-2875 36-2953 36-3008 37-3056 37-3132 37-3138 37-3147 39-3312 55-4624 56-4678 56-4680 56-4710 56-4731 56-4733 56-4752 56-4754 57-4795 57-4797 57-4812 57-4834 57-4838 57-4841 57-4843 60-5106 61-5240 61-5266 63-5393 63-5394 63-5395 64-5492
BIT09	= 001000	08-265 8-265 64-5492 64-5492 64-5492 64-5502 64-5502
BIT1	= 000002	08-265
BIT10	= 002000	08-265 29-2221 29-2225 32-2594 64-5502
BIT11	= 004000	08-265 29-2221 29-2225 32-2594 33-2697 64-5492
BIT12	= 010000	08-265 29-2221 29-2225 32-2594 33-2684 46-3911
BIT13	= 020000	08-265 25-1888 26-1972 26-1981 64-5502
BIT14	= 040000	08-265 25-1888 26-1972 26-1981 61-5191 61-5199 64-5492
BIT15	= 100000	08-265 23-1741 25-1888 26-1972 26-1981 27-2128 33-2677 33-2693 35-2869 36-2999
BIT2	= 000004	08-265
BIT3	= 000010	08-265
BIT4	= 000020	08-265
BIT5	= 000040	08-265 10-815 11-877 11-888 11-892 36-2969 49-4036 50-4123 51-4216 51-4230 52-4295 52-4304 53-4417 53-4426 53-4450 53-4459 54-4545 54-4554 55-4617 55-4637 56-4684 57-4793 61-5255 63-5392
BIT6	= 000100	08-265 10-815 36-2972 63-5384 63-5385
BIT7	= 000200	08-265 15-1207 15-1215 49-4056 50-4146 51-4223 52-4302 53-4424 53-4457 54-4552 55-4622 56-4708 57-4810 58-4888 59-4956 60-5047 60-5062 60-5078 60-5098 61-5194 61-5213 61-5258 63-5411 64-5463
BIT8	= 000400	08-265 17-1317 17-1334 18-1388 20-1467 21-1605 22-1675 23-1717 23-1749 24-1812 25-1904 26-2005 27-2148 29-2215 29-2237 30-2303 30-2374 31-2474 36-2971
BIT9	= 001000	08-265 29-2221 29-2225 32-2594
BPTVEC	= 000014	08-265
CCR	= 177746	08-366
CKSMR	= 104407	64-5492 64-5502 64-5502 64-5510
CR	= 000015	08-265 64-5493 64-5493

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES
CRLF	000200	#8-265 10-821 10-821 64-5493 64-5493
CTBLE	002022	#10-603 36-2954
DCSR	177730	#8-363 *10-682 15-1191 *15-1196 15-1197 15-1199 15-1208 *15-1213 15-1215 15-1218 15-1220 15-1225 *16-1271 *17-1317 *17-1334 *18-1368 *18-1369 18-1373 *18-1388 *19-1414 *19-1415 *19-1417 *19-1424 *20-1450 *20-1451 *20-1465 *20-1467 *21-1538 *21-1539 *21-1605 *22-1651 *22-1652 *22-1675 *23-1717 23-1741 *23-1749 *24-1795 *24-1796 *24-1812 *25-1861 *25-1062 *25-1904 *26-1946 *26-1947 *26-2005 *27-2054 27-2128 *27-2148 *29-2215 *29-2237 *30-2303 *30-2374 *31-2474 *31-2552 *32-2588 *32-2605 *33-2638 *33-2705 *34-2735 *34-2764 *35-2818 *35-2875 *36-2953 *36-2971 *36-3008 *37-3056 37-3132 *37-3138 *37-3140 *37-3147 *39-3312 *49-4044 *49-4067 49-4068 *49-4070 *56-4678 *56-4680 *56-4731 *56-4733 *56-4752 *56-4754 *57-4795 *57-4797 *57-4834 *57-4838 *57-4841 *57-4843 *63-5393 *63-5395
DDIN	002222	#10-682 20-1455 25-1881 26-1966 27-2127 30-2316 30-2339 30-2355 31-2484 31-2493 31-2510 31-2527 31-2544 33-2671 33-2673 34-2741 34-2750 35-2851 35-2868 36-2988 36-2998
DDISP	177570	#8-265 9-381 10-816
DDOUT	002210	#10-657 18-1377 21-1580 24-1809 26-1976 33-2652 33-2676
DOR	177732	#8-364 *10-657 18-1378 18-1380 19-1422 19-1426 20-1458 20-1461 20-1463 *22-1666 *23-1739 25-1884 25-1887 26-1968 26-1970 36-3002 49-4058 49-4062 49-4064
DH1	031174	10-387 10-435 10-448 10-536 #64-5549
DH13	031565	10-454 10-467 #64-5559
DH16	031640	10-480 10-492 10-498 10-511 10-517 10-542 10-548 10-554 10-560 10-566 10-572 10-578 #64-5561 10-393 10-400 10-441 #64-5551
DH2	031243	10-393 10-400 10-441 #64-5551
DH20	031677	10-486 #64-5563
DH3	031757	10-504 #64-5565
D27	032070	10-530 #64-5567
D13	031335	10-407 #64-5553
D4	031425	10-414 #64-5555
DH5	031503	10-421 10-428 10-460 10-473 10-523 #64-5557
DISPLA	001142	#9-381 *10-816 *10-816 64-5492 64-5502
DISPRE	000174	#8-371 10-816
DSMR	177570	#8-265 9-381 10-816
DT1	032166	10-388 10-436 10-449 #64-5570
DT16	032246	10-481 10-493 10-499 10-512 10-518 10-543 10-549 10-555 10-561 10-573 10-579 #64-5574 10-395 10-402 10-443 #64-5571
DT2	032200	10-395 10-402 10-443 #64-5571
DT20	032256	10-487 #64-5575
DT21	032272	10-567 #64-5576
DT23	032302	10-506 #64-5577
DT27	032322	10-531 #64-5578
DT3	032216	10-409 #64-5572
DT30	032336	10-537 #64-5579
DT4	032232	10-416 10-423 10-430 10-455 10-462 10-468 10-475 10-525 #64-5573
EMTSV	002500	#10-815
EMTVEC	000030	#8-265 *10-816 *10-816
EM1	027006	10-386 #64-5517
EM10	027446	10-434 #64-5524
EM11	027520	10-440 #64-5525
EM12	027572	10-447 #64-5526
EM13	027644	10-453 #64-5527

SYMBOL CROSS REFERENCE

CRF V02

SYMBOL	VALUE	REFERENCES							
EM14	027713	10-459	064-5528						
EM15	027743	10-466	064-5529						
EM16	027772	10-472	064-5530						
EM17	030054	10-479	064-5531						
EM2	027052	10-392	064-5518						
EM20	030107	10-485	064-5532						
EM21	030165	10-491	064-5533						
EM22	030232	10-497	064-5534						
EM23	030306	10-503	064-5535						
EM24	030320	10-510	064-5536						
EM25	030340	10-516	064-5537						
EM26	030410	10-522	064-5538						
EM27	030473	10-529	064-5539						
EM3	027114	10-399	064-5519						
EM30	030530	10-535	064-5540						
EM31	030571	10-541	064-5541						
EM32	030635	10-547	064-5542						
EM33	030702	10-553	064-5543						
EM34	030756	10-559	064-5544						
EM35	031016	10-565	064-5545						
EM36	031055	10-571	064-5546						
EM37	031123	10-577	064-5547						
EM4	027160	10-406	064-5520						
EM5	027236	10-413	064-5521						
EM6	027334	10-420	064-5522						
EM7	027377	10-427	064-5523						
ERRNUM	001714	010-583	064-5505	64-5570	64-5571	64-5572	64-5573	64-5574	64-5575
		64-5577	64-5578	64-5579					
ERROR	= 104000	08-265	10-701	11-902	12-970	12-981	12-994	12-1005	13-1000
		14-1128	15-1203	15-1217	15-1221	15-1226	16-1280	16-1286	17-1324
		18-1381	18-1385	19-1427	20-1464	21-1587	21-1591	22-1668	23-1743
		25-1889	26-1973	26-1982	27-2074	27-2099	27-2120	27-2130	29-2226
		30-2323	30-2330	30-2345	30-2352	30-2361	30-2368	31-2498	31-2504
		31-2521	31-2532	31-2538	31-2549	31-2556	32-2597	32-2603	33-2658
		33-2682	33-2689	33-2696	33-2702	34-2755	34-2761	35-2857	35-2871
		36-3004	37-3095	37-3120	37-3128	37-3143	38-3235	38-3239	40-3406
		41-3499	41-3514	41-3529	42-3612	42-3629	42-3646	42-3663	43-3716
		44-3791	45-3867	45-3874	46-3914	47-3951	48-3984	48-3987	49-4066
		50-4155	51-4235	52-4312	53-4440	53-4473	54-4565	55-4626	55-4630
		56-4717	56-4751	57-4814	57-4819	58-4895	59-4967	60-5054	60-5070
		60-5102	60-5108	61-5201	61-5220	61-5223	61-5236	61-5242	61-5262
		62-5314	62-5321	63-5420	63-5425	64-5467			
ERRVEC	= 000004	08-265	10-752	010-753	010-754	010-768	10-816	010-816	010-816
		037-3126	037-3141	037-3148	64-5492	064-5492	064-5492	064-5492	
ERTYPE	026120	64-5502	064-5504						
GNS	= *****	8-371	8-371	10-821	64-5482	64-5482	64-5510	64-5510	64-5510
		64-5510	64-5510	64-5510	64-5510	64-5510	64-5510	64-5510	64-5510
		64-5510	64-5510	64-5510	64-5510	64-5510	64-5510	64-5510	64-5510
GTSNR	= 104406	10-821	064-5510						
HT	= 000011	08-265	64-5493	64-5493					
IOTVEC	= 000020	08-265	010-816	010-816					
IUBE	002450	010-802	40-3397	41-3475	42-3600	42-3621	42-3638	42-3655	43-3699
									44-3762

J13

SEQ 0165

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES
		45-3857 46-3905 47-3943 48-3973 49-4031 50-4122 51-4204 52-4283 53-4400
		54-4532 55-4604 56-4688 57-4806 58-4878 59-4937 60-5038 60-5110 61-5184
		61-5246 61-5270 62-5292 63-5383 64-5446
KDPA0	172360	08-266
KDPA1	172362	08-266
KDPA2	172364	08-266
KDPA3	172366	08-266
KDPA4	172370	08-266
KDPA5	172372	08-266
KDPA6	172374	08-266
KDPA7	172376	08-266
KDPA0	172320	08-266
KDPA1	172322	08-266
KDPA2	172324	08-266
KDPA3	172326	08-266
KDPA4	172330	08-266
KDPA5	172332	08-266
KDPA6	172334	08-266
KDPA7	172336	08-266
KIPAR0	172340	08-266
KIPAR1	172342	08-266
KIPAR2	172344	08-266
KIPAR3	172346	08-266
KIPAR4	172350	08-266
KIPAR5	172352	08-266
KIPAR6	172354	08-266
		*10-758 *10-760 10-762 10-832 *21-1556 21-1583 *21-1596 *22-1661
		*23-1728 *23-1732 *24-1803 *25-1877 25-1893 *25-1898 *25-1900 25-1901 *26-1960
		*26-1967 *26-1977 *26-1986 *27-2064 *27-2066 *27-2078 *27-2114 27-2134 *27-2142
		*38-3230 *38-3243 38-3244 *53-4399 *54-4531 *57-4802 57-4826 *57-4833 64-5577
KIPAR7	172356	08-266
KIPAR0	172300	08-266
KIPAR1	172302	08-266
KIPAR2	172304	08-266
KIPAR3	172306	08-266
KIPAR4	172310	08-266
KIPAR5	172312	08-266
KIPAR6	172314	08-266
KIPAR7	172316	08-266
KMCR	177734	08-365
		10-776 16-1258 16-1268 *16-1273 16-1274 16-1276 16-1278 *16-1279
		16-1285 21-1535 21-1565 *21-1566 21-1569 *21-1602 22-1646 23-1726 23-1730
		*23-1731 *23-1748 24-1790 25-1856 26-1941 27-2057 27-2062 27-2086 *27-2088
		27-2089 27-2093 27-2096 27-2097 *27-2098 *27-2115 *27-2139 27-2140 *27-2145
		*28-2170 28-2171 *29-2219 *29-2220 29-2221 29-2223 29-2224 *29-2227 29-2228
		29-2231 29-2232 *30-2304 30-2310 *30-2317 30-2318 *30-2324 30-2325 *30-2338
		30-2340 *30-2346 30-2347 *30-2354 30-2356 *30-2362 30-2363 *30-2373 *31-2476
		31-2479 *31-2492 31-2494 *31-2499 31-2500 *31-2509 31-2511 *31-2516 31-2517
		*31-2526 31-2528 *31-2533 31-2534 *31-2543 31-2545 *31-2550 31-2551 *32-2587
		*32-2590 *32-2591 32-2592 32-2594 32-2596 *32-2598 32-2599 *32-2604 *33-2639
		*33-2643 33-2644 *33-2645 33-2646 *33-2653 33-2654 33-2657 *33-2659 33-2660
		33-2663 *33-2668 *33-2669 33-2677 33-2679 33-2681 *33-2683 33-2684 33-2686
		33-2688 33-2690 33-2697 33-2695 33-2697 33-2699 33-2701 *33-2703 *34-2737
		*34-2745 34-2746 *34-2747 34-2748 *34-2751 34-2752 34-2754 *34-2756 34-2757

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES
LF	= 000012	34-2760 *34-2762 35-2816 *35-2838 *35-2839 *35-2840 35-2853 35-2869 *36-2972
MAPH0	= 170202	36-2999 37-3134 38-3202 38-3207 38-3220 51-4199 53-4396 56-4670 56-4671
MAPH00	= 170202	*56-4679 56-4729 *56-4732 *56-4753 57-4794 *57-4796 *57-4835 57-4836 *57-4842
		*63-5384 63-5385 *63-5394 63-5416 *63-5422 *64-5452 *64-5453 64-5465 64-5575
		*8-265 64-5493 64-5493
		*8-339
		*8-274 8-339 *23-1734 *24-1806 *26-1962 *26-1989 *26-1990 *27-2110 *27-2138
		*30-2309 *31-2478 *33-2642 *34-2739 *36-2970 *51-4215 *52-4294 *54-4544 *55-4616
		*57-4799 *57-4830 *61-5207 61-5221 *63-5391
MAPH01	= 170206	*8-276 8-341 *35-2849 *35-2850 *35-2866 *35-2867 *63-5403 *64-5455 64-5578
MAPH02	= 170212	*8-278 8-343
MAPH03	= 170216	*8-280 8-345
MAPH04	= 170222	*8-282 8-347
MAPH05	= 170226	*8-284 8-349
MAPH06	= 170232	*8-286 8-351
MAPH07	= 170236	*8-288 8-353
MAPH1	= 170206	*8-341
MAPH10	= 170242	*8-290
MAPH11	= 170246	*8-292
MAPH12	= 170252	*8-294
MAPH13	= 170256	*8-296
MAPH14	= 170262	*8-298
MAPH15	= 170266	*8-300
MAPH16	= 170272	*8-302
MAPH17	= 170276	*8-304
MAPH2	= 170212	*8-343
MAPH20	= 170302	*8-306
MAPH21	= 170306	*8-308
MAPH22	= 170312	*8-310
MAPH23	= 170316	*8-312
MAPH24	= 170322	*8-314
MAPH25	= 170326	*8-316
MAPH26	= 170332	*8-318
MAPH27	= 170336	*8-320
MAPH3	= 170216	*8-345
MAPH30	= 170342	*8-322
MAPH31	= 170346	*8-324
MAPH32	= 170352	*8-326
MAPH33	= 170356	*8-328
MAPH34	= 170362	*8-330
MAPH35	= 170366	*8-332
MAPH36	= 170372	*8-334 *56-4743
MAPH37	= 170376	*8-336 *22-1665 62-5324 *63-5389
MAPH4	= 170222	*8-347
MAPH5	= 170226	*8-349
MAPH6	= 170232	*8-351
MAPH7	= 170236	*8-353
MAPL0	= 170200	*8-338
MAPL00	= 170200	*8-273 8-338 11-882 12-956 12-964 13-1044 13-1045 13-1053 13-1054
		14-1112 14-1119 21-1544 21-1555 *23-1733 *24-1805 25-1869 *26-1961 *26-1988
		26-1992 *27-2109 *27-2136 *30-2308 *31-2477 *33-2641 *34-2738 *36-2987 *36-2997
		*51-4214 *52-4293 53-4403 *54-4543 *55-4615 *57-4798 *57-4828 57-4831 *61-5206

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES
MAPL01	170204	61-5208 61-5218 62-5296 62-5306 *63-5390 64-5579
MAPL02	170210	*8-275 8-340 *35-2848 *35-2865 *63-5402 *64-5454 64-5578
MAPL03	170214	*8-277 8-342
MAPL04	170220	*8-279 8-344
MAPL05	170224	*8-281 3-346
MAPL06	170230	*8-283 8-348
MAPL07	170234	*8-285 8-350
MAPL1	170204	*8-287 8-352
MAPL10	170240	*8-340
MAPL11	170244	*8-289 56-4685
MAPL12	170250	*8-291
MAPL13	170254	*8-293
MAPL14	170260	*8-295
MAPL15	170264	*8-297
MAPL16	170270	*8-299
MAPL17	170274	*8-301
MAPL2	170210	*8-303
MAPL20	170300	*8-342
MAPL21	170304	*8-305
MAPL22	170310	*8-307
MAPL23	170314	*8-309
MAPL24	170320	*8-311
MAPL25	170324	*8-313
MAPL26	170330	*8-315
MAPL27	170334	*8-317
MAPL3	170214	*8-319
MAPL30	170340	*8-344
MAPL31	170344	*8-321
MAPL32	170350	*8-323
MAPL33	170354	*8-325
MAPL34	170360	*8-327
MAPL35	170364	*8-329
MAPL36	170370	*8-331
MAPL37	170374	*8-333 *56-4742 *22-1664 56-4673 56-4738 *63-5388
MAPL4	170220	*8-335
MAPL5	170224	*8-346
MAPL6	170230	*8-348
MAPL7	170234	*8-350
MAPP	002244	*8-352
MCSR	001732	*10-719 10-826 21-1552 22-1653 23-1718 24-1797 25-1863 26-1948 27-2055
MEMSIZ	002332	35-2819 38-3228 57-4785
MHRO	177572	*10-590 60-5043 60-5052 61-5188 61-5189 61-5199 61-5205
		*10-752 10-829
		*8-357 *10-828 *10-830 *21-1553 *21-1603 *22-1654 *22-1672 *23-1719 *23-1744
		*24-1798 *24-1810 *25-1865 *25-1903 *26-1950 *26-2004 *27-2056 *27-2147 *35-2820
		*35-2874 *38-3229 *38-3246 *57-4786 *57-4789
MHR1	177574	*8-358
MHR2	177576	*8-359
MHR3	172516	*8-360 *10-827 *10-831 *21-1554 *21-1604 *22-1655 *22-1673 *23-1735 *23-1745
		*24-1799 *24-1811 *25-1864 *26-1949 *27-2107 *27-2111 *27-2146 *29-2216 *29-2218
		29-2235 *30-2307 *30-2372 *31-2475 *32-2589 *33-2640 *33-2704 *34-2736 *34-2763
		*35-2821 *35-2841 *36-2969 *38-3224 *39-3313 *49-4036 *50-4123 *51-4216 *51-4230

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES
		*52-4295 *52-4304 *53-4417 *53-4426 *53-4450 *53-4459 *54-4545 *54-4554 *55-4617
		*55-4637 *56-4684 *57-4767 *57-4793 *57-4840 *61-5255 *63-5392 *63-5421 *64-5451
		*64-5469
MPVEC	* 000250	*08-266
NOABF	026116	64-5502
NOCAM	007664	64-5502 *64-5502
NROM	013576	28-2179 *28-2183
OBADR	002140	37-3080 *37-3151
PCR	* 177522	*10-605 *36-2961 *36-2974 *36-2984 *36-2994
PIRQ	* 177772	*08-367 *37-3059
PIRQVE	* 000240	*08-265 *43-3712 *43-3724
PMIS	001720	*08-265 *43-3702
		*10-585 *10-832 *16-1263 17-1313 18-1366 19-1412 20-1448 21-1531 22-1642
		23-1715 24-1786 25-1852 26-1939 26-1951 27-2051 27-2108 28-2168 29-2213
		30-2300 31-2471 32-2585 33-2635 34-2729 35-2813 35-2822 35-2828 35-2830
		36-2948 53-4477 57-4780 57-4788
POLY	* 120001	*08-368
PRO	* 000000	*08-265
PR1	* 000040	*08-265
PR2	* 000100	*08-265
PR3	* 000140	*08-265
PR4	* 000200	*08-265
PR5	* 000240	*08-265
PR6	* 000300	*08-265
PR7	* 000340	*08-265
PS	* 177776	*08-265
PSW	* 177776	*08-265 8-265
		*40-3398 *40-3407 40-3408 *40-3410 *41-3476 *41-3494 *41-3509 *41-3524
		*42-3607 *42-3609 *42-3624 *42-3626 *42-3641 *42-3643 *42-3658 *42-3660 *43-3710
		*43-3713 *44-3775 *44-3781 *47-3945 *47-3953 *48-3977 *48-3981 *61-5257
PTRN16	001774	*10-592 12-971 18-1371 36-2973 49-4038 49-4079 50-4129 52-4293
PTRN6	002010	*10-593 12-995
PURVEC	* 000024	*08-265 *10-816 *10-816 45-3861 *45-3862 45-3863 *45-3871 *45-3877 *64-5511
		*64-5511 *64-5511 *64-5511 *64-5511
RDCHR	* 104410	64-5496
RDDEC	* 104413	*64-5510
RDLIN	* 104411	64-5497 64-5498 *64-5510
RDOCT	* 104412	*64-5510
RESTAR	003270	*10-833 64-5482
RESVEC	* 000010	*08-265
R6	*0000006	*08-265 *10-816 *10-816 10-816
R7	*0000007	*08-265
SAV30	002566	10-815 *10-815 *10-815 64-5502 64-5502
SAV32	002570	*10-815 *10-815 64-5502 64-5502
SCOPE	* 000004	*08-265 11-866 12-951 13-1039 14-1107 15-1181 16-1250 17-1308 18-1364
		19-1410 20-1447 21-1529 22-1640 23-1713 24-1784 25-1850 26-1937 27-2050
		28-2166 29-2211 30-2298 31-2469 32-2583 33-2633 34-2727 35-2811 36-2947
		37-3051 38-3197 39-3310 40-3395 41-3473 42-3598 43-3698 44-3758 45-3837
		46-3903 47-3941 48-3971 49-4028 50-4120 51-4197 52-4281 53-4394 54-4529
		55-4603 56-4668 57-4778 58-4876 59-4935 60-5036 61-5182 62-5291 63-5381
		64-5444 64-5473 64-5482
SIMLGO	* 170014	*08-369
SRO	* 177572	*08-266
SR1	* 177574	*08-266

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES
SR2	= 177576	#8-266
SR3	= 172516	#8-266 *11-877 *11-888 *11-892
STACK	= 001100	#8-265 10-816
START	002500	8-374 8-377 #10-815
STKLMT	= 177774	#8-265
SWR	001140	#9-381 10-816 *10-816 10-816 *10-816 *10-816 10-821 64-5492 64-5492 64-5492 64-5492 64-5492 64-5496 64-5496 64-5502 64-5502 64-5502 64-5502 64-5511 64-5511
SWREG	000176	#8-371 10-816 10-821 64-5496 64-5496
SW0	= 000001	#8-265
SW00	= 000001	#8-265 8-265
SW01	= 000002	#8-265 8-265
SW02	= 000004	#8-265 8-265
SW03	= 000010	#8-265 8-265
SW04	= 000020	#8-265 8-265
SW05	= 000040	#8-265 8-265
SW06	= 000100	#8-265 8-265
SW07	= 000200	#8-265 8-265
SW08	= 000400	#8-265 8-265
SW09	= 001000	#8-265 8-265
SW1	= 000002	#8-265
SW10	= 002000	#8-265
SW11	= 004000	#8-265
SW12	= 010000	#8-265
SW13	= 020000	#8-265
SW14	= 040000	#8-265
SW15	= 100000	#8-265
SW2	= 000004	#8-265
SW3	= 000010	#8-265
SW4	= 000020	#8-265
SW5	= 000040	#8-265
SW6	= 000100	#8-265
SW7	= 000200	#8-265
SW8	= 000400	#8-265
SW9	= 001000	#8-265
TBITVE	= 000014	#8-265
TBLMH	012716	35-2843 #35-2888
TBLML	012656	35-2842 #35-2881
TEST	001716	#10-584 *10-824 *64-5504 64-5570 64-5571 64-5572 64-5573 64-5574 64-5575 64-5576 64-5577 64-5578 64-5579
TIMOUT	002234	#10-700 10-822 15-1192 16-1259 17-1333 21-1551 22-1674 27-2080 27-2144 39-3369
TKVEC	= 000060	#8-265
TOPTBL	010542	30-2306 #30-2391
TOUT	001730	#10-589 *21-1578 21-1581
TPVEC	= 000064	#8-265
TRAPVE	= 000034	#8-265 *10-816 *10-816
TRTVEC	= 000014	#8-265
TST1	003274	#11-866
TST10	004466	17-1314 #18-1364
TST11	004624	18-1367 #19-1410
TST12	004722	19-1413 19-1425 #20-1447

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES
TST13	005044	20-1449 021-1529
TST14	005450	21-1532 022-1640
TST15	005646	22-1643 22-1647 023-1713
TST16	006052	23-1716 23-1747 024-1784
TST17	006242	24-1787 24-1791 24-1814 025-1850
TST2	003420	11-890 012-951
TST20	006514	25-1853 25-1857 026-1937
TST21	007114	26-1940 26-1942 027-2050
TST22	007574	27-2085 028-2166
TST23	007704	28-2169 28-2172 029-2211
TST24	010076	29-2214 030-2298
TST25	010556	30-2375 031-2469
TST26	011266	31-2554 032-2583
TST27	011436	32-2586 033-2633
TST3	003640	013-1039
TST30	012120	034-2727
TST31	012334	34-2730 035-2811
TST32	012756	35-2817 35-2876 036-2947
TST33	013210	36-2949 037-3051
TST34	013634	37-3149 038-3197
TST35	014034	38-3203 039-3310
TST36	014222	39-3371 040-3395
TST37	014324	041-3473
TST4	003762	014-1107
TST40	014604	042-3598
TST41	015144	043-3698
TST42	015300	044-3758
TST43	015536	44-3761 44-3798 045-3837
TST44	015752	45-3842 45-3844 45-3846 45-3848 45-3852 45-3854 45-3856 046-3903
TST45	016044	047-3941
TST46	016124	048-3971
TST47	016224	48-3983 48-3985 049-4028
TST5	004042	015-1181
TST50	016440	050-4120
TST51	016532	50-4154 051-4197
TST52	016674	052-4281
TST53	017002	52-4311 053-4394
TST54	017432	53-4478 054-4529
TST55	017576	54-4561 055-4603
TST56	017746	056-4668
TST57	020374	057-4778
TST6	004226	15-1219 016-1250
TST60	020746	57-4781 058-4876
TST61	021030	58-4894 059-4935
TST62	021134	060-5036
TST63	021476	061-5182
TST64	022152	062-5291
TST65	022336	063-5381
TST66	022560	63-5424 064-5444
TST67	022740	064-5473
TST7	004404	16-1264 16-1281 017-1308
TPDS	104405	64-5482 64-5482 064-5510

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES									
TYPE	• 104401	10-821	28-2179	28-2180	37-3080	37-3084	64-5482	64-5482	64-5482	64-5493	
		64-5494	64-5495	64-5496	64-5496	64-5496	64-5496	64-5496	64-5496	64-5496	
		64-5496	64-5496	64-5498	64-5498	64-5502	64-5502	64-5509	64-5509	64-5509	
		64-5509	64-5509	64-5509	64-5509	64-5510	64-5511				
		64-5496	64-5509	64-5509	64-5510						
TYPOC	• 104402	64-5510									
TYPON	• 104404	37-3081	64-5510								
TYPOS	• 104403	10-586	•39-3355	39-3361	39-3370	44-3760					
UBECT	001722	39-3372	64-5473								
UBEM	022740	28-2181	•37-3051								
UBETST	013210	•10-815	•10-815	64-5502							
UFDLFG	002572	•8-361	10-815	64-5502	64-5502						
UFDSET	• 000001	•10-776	38-3214								
UMSIZ	002424	•10-815	•10-815	10-817	64-5482	64-5502	64-5502				
UQUIET	002574	30-2305	•30-2380								
VALTBL	010526	10-815	10-815	10-815	•10-815						
VPROR	002576	•10-591	49-4037	49-4078	51-4214	51-4231	58-4879	59-4943	59-4952	59-4963	
WRTBUF	001734	8-380	•8-380								
\$APTHD	000232	64-5508	64-5508								
\$ASTAT	• *****	64-5508	64-5508								
\$ATYC	026166	64-5508	64-5508								
\$ATY1	026142	64-5508	64-5508								
\$ATY3	026150	64-5493	64-5508								
\$ATY4	026160	64-5502	64-5508								
\$AUTOB	001134	•9-381	•10-821	64-5496	64-5496	64-5496					
\$BASE	001254	•9-381									
\$BDADR	001122	•9-381	•10-700	•11-900	•12-969	•12-980	•12-993	•12-1004	•13-1059	•13-1071	
		•15-1225	•16-1278	•16-1285	•21-1583	•21-1585	•24-1816	•25-1882	•25-1883	•25-1884	
		25-1886	•25-1888	•26-1964	26-1968	26-1971	•26-1972	26-1978	26-1980	•26-1981	
		•26-1991	•26-1992	•26-1993	•26-1994	•27-2097	•53-4439	•53-4472	•54-4564	64-5570	
		64-5571	64-5572	64-5575	64-5576	64-5577					
\$BDDAT	001126	•9-381	•12-967	•12-977	•12-991	•12-1001	•14-1127	•15-1201	•15-1220	•16-1277	
		•18-1380	•18-1384	•19-1426	•20-1463	•21-1590	•24-1815	•25-1887	•26-1970	•26-1980	
		•27-2096	•29-2223	•29-2231	•30-2318	30-2319	•30-2325	30-2326	•30-2340	30-2341	
		•30-2347	30-2348	•30-2356	30-2357	•30-2363	30-2364	•31-2494	31-2495	•31-2500	
		31-2501	•31-2511	31-2512	•31-2517	31-2518	•31-2528	31-2529	•31-2534	31-2535	
		•31-2545	1-2546	•31-2551	31-2553	•32-2596	•32-2599	32-2600	•33-2657	•33-2663	
		•33-2681	33-2688	•33-2695	•33-2701	•34-2754	•34-2760	•49-4064	•53-4437	•53-4470	
		•54-4562	64-5571	64-5573	64-5577						
\$BELL	001170	•9-381	64-5502	64-5502	64-5502						
\$CDW1	001260	•9-381									
\$CDW2	001262	•9-381									
\$CHARC	024014	•64-5493	•64-5493	64-5493	•64-5493	•64-5493					
\$CKSWR	024472	•64-5496	64-5510	64-5510							
\$CHTAG	001100	•9-381	10-816	10-816	10-816	10-816	10-816	10-816			
\$CM3	• 000000	•9-381	9-381								
\$CM4	• 000002	•9-381	9-381	9-381	•9-381	9-381	9-381	•9-381			
\$CNTLG	025227	64-5496	•64-5496								
\$CNTLU	025222	64-5496	•64-5496								
\$CPUOP	001226	•9-381									
\$CRLF	001175	•9-381	28-2180	37-3084	64-5482	64-5493	64-5493	64-5493	64-5496	64-5496	
		64-5496	64-5498	64-5498	64-5502	64-5502	64-5502	64-5509	64-5509	64-5509	
		64-5509									

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES
\$DBLK	024462	64-5495 64-5495 664-5495
\$DDW0	001264	09-381
\$DDW1	001266	09-381
\$DDW10	001310	09-381
\$DDW11	001312	09-381
\$DDW12	001314	09-381
\$DDW13	001316	09-381
\$DDW14	001320	09-381
\$DDW15	001322	09-381
\$DDW2	001270	09-381
\$DDW3	001272	09-381
\$DDW4	001274	09-381
\$DDW5	001276	09-381
\$DDW6	001300	09-381
\$DDW7	001302	09-381
\$DDW8	001304	09-381
\$DDW9	001306	09-381
\$DEVCT	001210	09-381
\$DEVH	001256	09-381
\$DOAGN	023152	64-5482 64-5482 664-5482
\$DTBL	024452	64-5495 664-5495
\$ENDAO	023142	8-379 10-821 664-5482 64-5502 64-5502
\$ENDCT	023022	10-816 664-5477 664-5482
\$ENULL	023156	664-5482
\$ENV	001220	09-381 10-819 10-821 28-2175 37-3075 45-3815 64-5493 64-5502 64-5508
\$ENVH	001221	09-381 10-816 64-5493 64-5493 64-5508
\$EOP	022766	51-4203 63-5387 64-5475 664-5482
\$EOPCT	023014	*10-816 *64-5476 664-5482 64-5482
\$ERFLG	001103	09-381 64-5492 64-5492 64-5492 *64-5492 64-5492 64-5492 *64-5502 64-5502
\$ERMAX	001115	09-381 *10-816 64-5492 *64-5492 64-5492 64-5492
\$ERROR	025536	10-816 664-5502
\$ERRPC	001116	09-381 *64-5502 *64-5502 64-5502 64-5502 64-5502 64-5509 64-5570 64-5571
\$ERRTB	001324	64-5572 64-5573 64-5574 64-5575 64-5577 64-5578 64-5579
\$ERRTY	026410	*10-381 64-5509
\$ERTTL	001112	64-5506 664-5509
\$ESCAP	001166	09-381 64-5482 *64-5502 64-5502 64-5502 64-5502 64-5502
\$ETABL	001220	09-381 *10-816 *64-5492 64-5502 64-5502 64-5502 64-5502
\$ETEND	001324	09-381
\$FATAL	001202	8-380 09-381 *64-5508
\$FFLG	026406	09-381 *64-5508 *64-5508 64-5508 *64-5508 664-5508
\$FILLC	001156	09-381 64-5493 64-5493
\$FILLS	001155	09-381 64-5493 64-5493
\$GDAOR	001120	09-381 *13-1058 *13-1070 64-5572
\$GDOAT	001124	09-381 10-657 *12-968 *12-979 *12-992 *12-1003 *14-1126 *15-1202 *15-1207
		*15-1208 *16-1276 *18-1376 18-1378 18-1382 *19-1416 19-1418 19-1422 *20-1453
		*20-1460 *21-1579 21-1588 *22-1662 22-1666 *24-1807 *25-1886 *26-1965 *26-1971
		*26-1974 26-1978 *26-2001 *27-2091 *27-2092 27-2093 *27-2095 *29-2224 *29-2225
		*29-2232 *29-2233 *30-2310 *30-2311 *30-2322 *30-2329 *30-2344 *30-2351 *30-2360
		*30-2367 *31-2479 *31-2497 *31-2503 *31-2514 *31-2520 *31-2531 *31-2537 *31-2548

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES
		*31-2555 *32-2592 *32-2593 *32-2602 *33-2651 *33-2656 *33-2662 *33-2675 *33-2679
		*33-2680 *33-2686 *33-2687 *33-2690 *33-2691 *33-2699 *33-2700 *34-2746 34-2752
		*34-2759 *49-4065 *53-4438 *53-4471 *54-4563 64-5571 64-5573 64-5577
\$GET42	023132	*64-5482
\$GTSMR	024542	*64-5496 64-5510 64-5510
\$HD	000001	8-262 8-262 8-262
\$HIBTS	000232	*8-380
\$HIOCT	025356	*. -5497 *64-5497
\$ICNT	001104	*9-381 *64-5492 64-5492 *64-5492 64-5492 64-5492
\$ILLUP	026770	64-5511 64-5511 *64-5511
\$INTAG	001135	*9-381 64-5496 64-5496 64-5496
\$ITEMB	001114	*9-381 *64-5502 64-5502 64-5502 64-5502 64-5505 64-5509
\$LF	001176	*9-381 64-5493 64-5493 64-5496 64-5496 64-5496 64-5498 64-5498 64-5502
		64-5502
\$LFLG	026405	*64-5508 *64-5508
\$LPADR	001106	*9-381 *10-816 *64-5492 *64-5492 64-5492 64-5492 64-5492
\$LPERR	001110	*9-381 *10-816 64-5492 *64-5492 64-5492 64-5492 64-5502
\$MADR1	001232	*9-381
\$MADR2	001236	*9-381
\$MADR3	001242	*9-381
\$MADR4	001246	*9-381
\$MAIL	001200	8-380 8-380 *9-381 10-816 10-821 64-5492 64-5493 64-5502
\$MAPS1	001230	*9-381
\$MAPS2	001234	*9-381
\$MAPS3	001240	*9-381
\$MAPS4	001244	*9-381
\$MBADR	000234	*8-380
\$MFLG	026404	*64-5508 64-5508 *64-5508 *64-5508
\$MNEW	025245	64-5496 *64-5496
\$MSGAD	001214	*9-381 *64-5508 64-5508
\$MSGLG	001216	*9-381 *64-5508
\$MSGTY	001200	*9-381 64-5508 *64-5508 64-5508 *64-5508
\$MSMR	025234	64-5496 *64-5496
\$MTYP1	001231	*9-381
\$MTYP2	001235	*9-381
\$MTYP3	001241	*9-381
\$MTYP4	001245	*9-381
\$MXCNT	023462	64-5492 64-5492 64-5492 *64-5492
\$NULL	001154	*9-381 64-5493 64-5493 64-5493
\$NMTS1	000001	*11-866 11-866 *11-866 *12-951 12-951 *12-951 *13-1039 13-1039 *13-1039
		*14-1107 14-1107 *14-1107 *15-1181 15-1181 *15-1181 *16-1250 16-1250 *16-1250
		*17-1308 17-1308 *17-1308 *18-1364 18-1364 *18-1364 *19-1410 19-1410 *19-1410
		*20-1447 20-1447 *20-1447 *21-1529 21-1529 *21-1529 *22-1640 22-1640 *22-1640
		*23-1713 23-1713 *23-1713 *24-1784 24-1784 *24-1784 *25-1850 25-1850 *25-1850
		*26-1937 26-1937 *26-1937 *27-2050 27-2050 *27-2050 *28-2166 28-2166 *28-2166
		*29-2211 29-2211 *29-2211 *30-2298 30-2298 *30-2298 *31-2469 31-2469 *31-2469
		*32-2583 32-2583 *32-2583 *33-2633 33-2633 *33-2633 *34-2727 34-2727 *34-2727
		*35-2811 35-2811 *35-2811 *36-2947 36-2947 *36-2947 *37-3051 37-3051 *37-3051
		*38-3197 38-3197 *38-3197 *39-3310 39-3310 *39-3310 *40-3395 40-3395 *40-3395
		*41-3473 41-3473 *41-3473 *42-3598 42-3598 *42-3598 *43-3698 43-3698 *43-3698
		*44-3758 44-3758 *44-3758 *45-3837 45-3837 *45-3837 *46-3903 46-3903 *46-3903
		*47-3941 47-3941 *47-3941 *48-3971 48-3971 *48-3971 *49-4028 49-4028 *49-4028

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES
		#50-4120 50-4120 #50-4120 #51-4197 51-4197 #51-4197 #52-4281 52-4281 #52-4281
		#53-4394 53-4394 #53-4394 #54-4529 54-4529 #54-4529 #55-4603 55-4603 #55-4603
		#56-4668 56-4668 #56-4668 #57-4778 57-4778 #57-4778 #58-4876 58-4876 #58-4876
		#59-4935 59-4935 #59-4935 #60-5036 60-5036 #60-5036 #61-5182 61-5182 #61-5182
		#62-5291 62-5291 #62-5291 #63-5381 63-5381 #63-5381 #64-5444 64-5444 #64-5444
		#64-5473 64-5473 #64-5473
\$OCNT	024242	#64-5494 #64-5494 #64-5494
\$OMODE	024244	#64-5494 #64-5494 64-5494 #64-5494 #64-5494 #64-5494
\$OVER	023440	64-5492 64-5492 64-5492 64-5492 #64-5492
\$PASS	001206	#9-381 #10-816 28-2173 37-3073 39-3314 45-3841 64-5474 #64-5482 #64-5482
		64-5482 64-5482 64-5482 64-5492 64-5492 64-5492
\$PASTH	000240	#8-380
\$POWER	026776	64-5511 #64-5511
\$PWRDN	026630	10-816 #64-5511 64-5511
\$PWRMG	026764	#64-5511
\$PWRUP	026702	64-5511 #64-5511
\$QUES	001174	#9-381 64-5493 64-5493 64-5496 64-5496 64-5496 64-5496 64-5496 64-5498 64-5498
		64-5498 64-5502 64-5502
\$RDCHR	024754	#64-5496 64-5510 64-5510
\$RDEEC	025360	#64-5498 64-5510 64-5510
\$ROLIN	025104	#64-5496 64-5510 64-5510
\$ROCT	025256	#64-5497 64-5510 64-5510
\$ROSZ	= 000010	#64-5496 64-5496
\$RTNAD	023154	#64-5482
\$R2A	= *****	64-5510
\$SAVRE	= *****	64-5510
\$SAVR6	026774	#64-5511 64-5511 #64-5511 #64-5511 #64-5511
\$SCOPE	023162	10-816 #64-5492
\$SETUP	= 000137	#10-815 10-815 #10-815 10-815 #10-815 10-815 #10-815 10-815 #10-815 10-815 #10-815
		10-815 #10-815 10-815 #10-815 10-816 10-816 10-816 10-816 10-816 10-821 10-821
		10-816 10-816 10-816 10-816 10-816 10-816 10-816 10-821 10-821
		10-821 64-5482 64-5482 64-5492 64-5496 64-5496 64-5502 64-5502 64-5502
\$STUP	= 177777	64-5502 #10-815 #10-815 10-815 #10-815 #10-815 10-815 #10-815 10-815 #10-815 10-815
		#10-815 #10-815 10-815 #10-815 #10-815 10-815 #10-815 10-815 #10-815 10-815
\$SVLAD	023404	64-5492 #64-5492
\$SVPC	= 000232	#8-379 8-379
\$SWR	= 167400	#8-260 8-262 8-263 8-263 8-263 8-263 8-263 8-263 8-263 8-263
		8-263 9-381 9-381 9-381 10-816 10-816 10-816 10-816 10-816 10-816
		11-866 12-951 13-1039 14-1107 15-1181 16-1250 17-1308 18-1364 19-1410
		20-1447 21-1529 22-1640 23-1713 24-1784 25-1850 26-1937 27-2050 28-2166
		29-2211 30-2298 31-2469 32-2583 33-2633 34-2727 35-2811 36-2947 37-3051
		38-3197 39-3310 40-3395 41-3473 42-3598 43-3698 44-3758 45-3837 46-3903
		47-3941 48-3971 49-4028 50-4120 51-4197 52-4281 53-4394 54-4529 55-4603
		56-4668 57-4778 58-4876 59-4935 60-5036 61-5182 62-5291 63-5381 64-5444
		64-5473 64-5482 64-5482 64-5482 64-5482 64-5482 64-5492 64-5492 64-5492
		64-5492 64-5492 64-5492 64-5492 64-5492 64-5492 64-5492 64-5492 64-5502
		64-5502 64-5502 64-5502 64-5502 64-5502 64-5502 64-5502 64-5502 64-5502
		64-5502 64-5511
\$SWREG	001222	#9-381 10-816
\$SWRMK	= 000300	#8-261 8-263 8-263 8-263 8-263 8-263 8-263 8-263 8-263

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES								
		8-263	64-5492	64-5492	64-5492	64-5492	64-5492	64-5492	64-5492	64-5492
		64-5492	64-5492	64-5492						
\$TESTN	001204	09-381	*64-5492							
\$TIMES	001164	09-381	*10-816	*64-5482	*64-5492	64-5492	*64-5492	64-5492	64-5492	
\$TKB	001146	09-381	64-5493	64-5493	64-5493	64-5493	64-5496	64-5496	64-5496	64-5496
		64-5496	64-5496							
\$TKS	001144	09-381	64-5493	64-5493	64-5493	64-5493	64-5496	64-5496	64-5496	64-5496
		64-5496	64-5496	64-5496						
\$TMP0	001160	*8-373	*8-376	09-381	*16-1258	*16-1260	16-1261	16-1279	*18-1370	18-1372
		18-1382	18-1384	*19-1418	*26-1952	26-2002	27-2098	*29-2228	29-2229	*35-2844
		35-2850	35-2867	*37-3060	*37-3085	*37-3125	37-3148	40-3400	*45-3861	45-3877
		46-3910	48-3975	48-3982	*53-4432	*53-4433	*53-4434	53-4435	53-4438	53-4439
		*53-4465	*53-4466	*53-4467	53-4468	53-4471	53-4472	*56-4673	*56-4676	56-4724
		56-4738	*57-4794	57-4842	*61-5188	61-5205				
\$TMP1	001162	09-381	*21-1565	21-1602	*23-1730	23-1748	*26-1963	26-1967	26-1977	*26-1997
		*26-1998	26-1999	26-2002	*27-2086	27-2145	*53-4406	*53-4484	53-4485	53-4486
		53-4487	*56-4670	56-4753						
\$TN	= 000070	8-262	*8-262	11-866	11-866	*11-866	11-890	12-951	12-951	*12-951
		13-1039	13-1039	*13-1039	14-1107	14-1107	*14-1107	15-1181	15-1181	*15-1181
		15-1219	16-1250	16-1250	*16-1250	16-1264	16-1281	17-1308	17-1308	*17-1308
		17-1314	18-1364	18-1364	*18-1364	18-1367	19-1410	19-1410	*19-1410	19-1413
		19-1425	20-1447	20-1447	*20-1447	20-1449	21-1529	21-1529	*21-1529	21-1532
		22-1640	22-1640	*22-1640	22-1643	22-1647	23-1713	23-1713	*23-1713	23-1716
		23-1747	24-1784	24-1784	*24-1784	24-1787	24-1791	24-1814	25-1850	25-1850
		*25-1850	25-1853	25-1857	26-1937	26-1937	*26-1937	26-1940	26-1942	27-2050
		27-2050	*27-2050	27-2085	28-2166	28-2166	*28-2166	28-2169	28-2172	29-2211
		29-2211	*29-2211	29-2214	30-2298	30-2298	*30-2298	30-2375	31-2469	31-2469
		*31-2469	31-2554	32-2583	32-2583	*32-2583	32-2586	33-2633	33-2633	*33-2633
		34-2727	34-2727	*34-2727	34-2730	35-2811	35-2811	*35-2811	35-2817	35-2876
		36-2947	36-2947	*36-2947	36-2949	37-3051	37-3051	*37-3051	37-3149	38-3197
		38-3197	*38-3197	38-3203	39-3310	39-3310	*39-3310	39-3371	40-3395	40-3395
		*40-3395	41-3473	41-3473	*41-3473	42-3598	42-3598	*42-3598	43-3698	43-3698
		*43-3698	44-3758	44-3758	*44-3758	44-3761	44-3798	45-3837	45-3837	*45-3837
		45-3842	45-3844	45-3846	45-3848	45-3852	45-3854	45-3856	46-3903	46-3903
		*46-3903	47-3941	47-3941	*47-3941	48-3971	48-3971	*48-3971	48-3983	48-3985
		49-4026	49-4028	*49-4028	50-4120	50-4120	*50-4120	50-4154	51-4197	51-4197
		*51-4197	52-4281	52-4281	*52-4281	52-4311	53-4394	53-4394	*53-4394	53-4478
		54-4529	54-4529	*54-4529	54-4561	55-4603	55-4603	*55-4603	56-4668	56-4668
		*56-4668	57-4778	57-4778	*57-4778	57-4781	58-4876	58-4876	*58-4876	58-4894
		59-4935	59-4935	*59-4935	60-5036	60-5036	*60-5036	61-5182	61-5182	*61-5182
		62-5291	62-5291	*62-5291	63-5381	63-5381	*63-5381	63-5424	64-5444	64-5444
		*64-5444	64-5473	64-5473	*64-5473					
\$TPB	001152	09-381	64-5493	64-5493	64-5493					
\$TPFLG	001157	09-381	64-5493	64-5493	64-5493					
\$TPS	001150	09-381	64-5493	64-5493	64-5493					
\$TRAP	026544	10-816	*64-5510							
\$TRAP2	026566	*64-5510	64-5510							
\$TRP	= 000014	*64-5510	64-5510	64-5510	64-5510	64-5510	*64-5510	64-5510	64-5510	64-5510
		64-5510	*64-5510	64-5510	64-5510	64-5510	64-5510	*64-5510	64-5510	64-5510
		64-5510	64-5510	*64-5510	64-5510	64-5510	64-5510	64-5510	*64-5510	64-5510
		64-5510	64-5510	64-5510	*64-5510	64-5510	64-5510	64-5510	64-5510	*64-5510
		64-5510	64-5510	64-5510	64-5510	*64-5510	64-5510	64-5510	64-5510	64-5510

SYMBOL CROSS REFERENCE

CREF V02

SYMBOL	VALUE	REFERENCES
		64-5510 64-5510 64-5510 64-5510 64-5510 64-5510 64-5510 64-5510 64-5510
\$TRPAD	026600	64-5510 64-5510 64-5510 64-5510
\$TSTM	000236	08-380
\$TSTM	001102	09-381 64-5482 64-5492 64-5492 64-5492 64-5492 64-5492 64-5492 64-5492
		64-5502 64-5502 64-5502 64-5504
\$TTYIN	025212	64-5496 64-5496 64-5496
\$TYPBN	*****	64-5510
\$TYPDS	024246	64-5495 64-5510 64-5510
\$TYPE	023464	64-5493 64-5508 64-5510 64-5510
\$YPEC	023676	64-5493 64-5493 64-5493 64-5493 64-5496
\$YPEX	024016	64-5493 64-5493 64-5493
\$YPOC	024044	64-5494 64-5510 64-5510
\$YPON	024060	64-5494 64-5494 64-5510
\$YPOS	024020	64-5494 64-5510
\$UNIT	001212	09-381
\$UNITM	000242	08-380
\$USMR	001224	09-381
\$VECT1	001250	09-381
\$VECT2	001252	09-381
\$XOFF	000023	64-5493 64-5493
\$XON	000021	64-5493 64-5493 64-5496
\$XTSTR	023202	64-5492
\$GET4	000000	64-5482 64-5482
\$OFILL	024243	64-5494 64-5494 64-5494 64-5494
\$OCAT	***	64-5492 64-5502
.\$ASTA	*****	64-5508 64-5508
.\$X	000232	08-380 8-380

MACRO CROSS REFERENCE

CREF V02

MACRO NAME	REFERENCES
COMEN	#8-265
ENDCOM	#8-265
ENDPAS	#8-249 64-5482
ESCAPE	#8-265
GETPRI	#8-265
GETSWR	#8-265 #10-821 10-821
MULT	#8-265
NEWST	#8-265 11-866 12-951 13-1039 14-1107 15-1181 16-1250 17-1308 18-1364 19-1410
	20-1447 21-1529 22-1640 23-1713 24-1784 25-1850 26-1937 27-2050 28-2166 29-2211
	30-2298 31-2469 32-2583 33-2633 34-2727 35-2811 36-2947 37-3051 38-3197 39-3310
	40-3395 41-3473 42-3598 43-3698 44-3758 45-3837 46-3903 47-3941 48-3971 49-4028
	50-4120 51-4197 52-4281 53-4394 54-4529 55-4603 56-4668 57-4778 58-4876 59-4935
	60-5036 61-5182 62-5291 63-5381 64-5444 64-5473 64-5495 64-5497 64-5498 64-5508 64-5508 64-5511
POP	#8-247 #8-265 10-732 10-807 64-5495 64-5497 64-5498 64-5508 64-5508 64-5511
PUSH	#8-247 #8-265 10-719 10-802 64-5495 64-5497 64-5498 64-5508 64-5508 64-5508
	64-5511 64-5511
REPORT	#8-265
SAVENT	#10-815 10-815
SETPRI	#8-265
SETTRA	#64-5510 64-5510 64-5510 64-5510 64-5510 64-5510 64-5510 64-5510 64-5510 64-5510
	64-5510 64-5510
SETUP	#8-265 10-816
SKIP	#8-247 #8-265 11-890 15-1219 16-1264 16-1281 17-1314 18-1367 19-1413 19-1425
	20-1449 21-1532 22-1643 22-1647 23-1716 23-1747 24-1787 24-1791 24-1814 25-1853
	25-1857 26-1940 26-1942 27-2085 28-2169 28-2172 29-2214 30-2375 31-2554 32-2586
	34-2730 35-2817 35-2876 36-2949 37-3149 38-3203 39-3371 44-3761 44-3798 45-3842
	45-3844 45-3846 45-3848 45-3852 45-3854 45-3856 48-3983 48-3985 50-4154 52-4311
	53-4478 54-4561 57-4781 58-4894 63-5424
SLASH	#8-265
STARS	#8-265 8-379 8-380 8-380 8-380 9-381 9-381 9-381 11-866 11-866
	12-951 12-951 13-1039 13-1039 14-1107 14-1107 15-1181 15-1181 16-1250 16-1250
	17-1308 17-1308 18-1364 18-1364 19-1410 19-1410 20-1447 20-1447 21-1529 21-1529
	22-1640 22-1640 23-1713 23-1713 24-1784 24-1784 25-1850 25-1850 26-1937 26-1937
	27-2050 27-2050 28-2166 28-2166 29-2211 29-2211 30-2298 30-2298 31-2469 31-2469
	32-2583 32-2583 33-2633 33-2633 34-2727 34-2727 35-2811 35-2811 36-2947 36-2947
	37-3051 37-3051 38-3197 38-3197 39-3310 39-3310 40-3395 40-3395 41-3473 41-3473
	42-3598 42-3598 43-3698 43-3698 44-3758 44-3758 45-3837 45-3837 46-3903 46-3903
	47-3941 47-3941 48-3971 48-3971 49-4028 49-4028 50-4120 50-4120 51-4197 51-4197
	52-4281 52-4281 53-4394 53-4394 54-4529 54-4529 55-4603 55-4603 56-4668 56-4668
	57-4778 57-4778 58-4876 58-4876 59-4935 59-4935 60-5036 60-5036 61-5182 61-5182
	62-5291 62-5291 63-5381 63-5381 64-5444 64-5444 64-5473 64-5473 64-5482 64-5492
	64-5493 64-5494 64-5495 64-5496 64-5496 64-5496 64-5497 64-5498 64-5502
	64-5508 64-5509 64-5510 64-5511 64-5511
SWRSU	#8-265 #10-816 10-816
TRMTRP	#64-5510
TYPBIN	#8-265
TYPDEC	#8-265 64-5482 64-5482
TYPNAM	#8-265 10-821
TYPNUM	#8-265
TYPOCS	#8-265
TYPOCT	#8-265 64-5496 64-5509 64-5509

