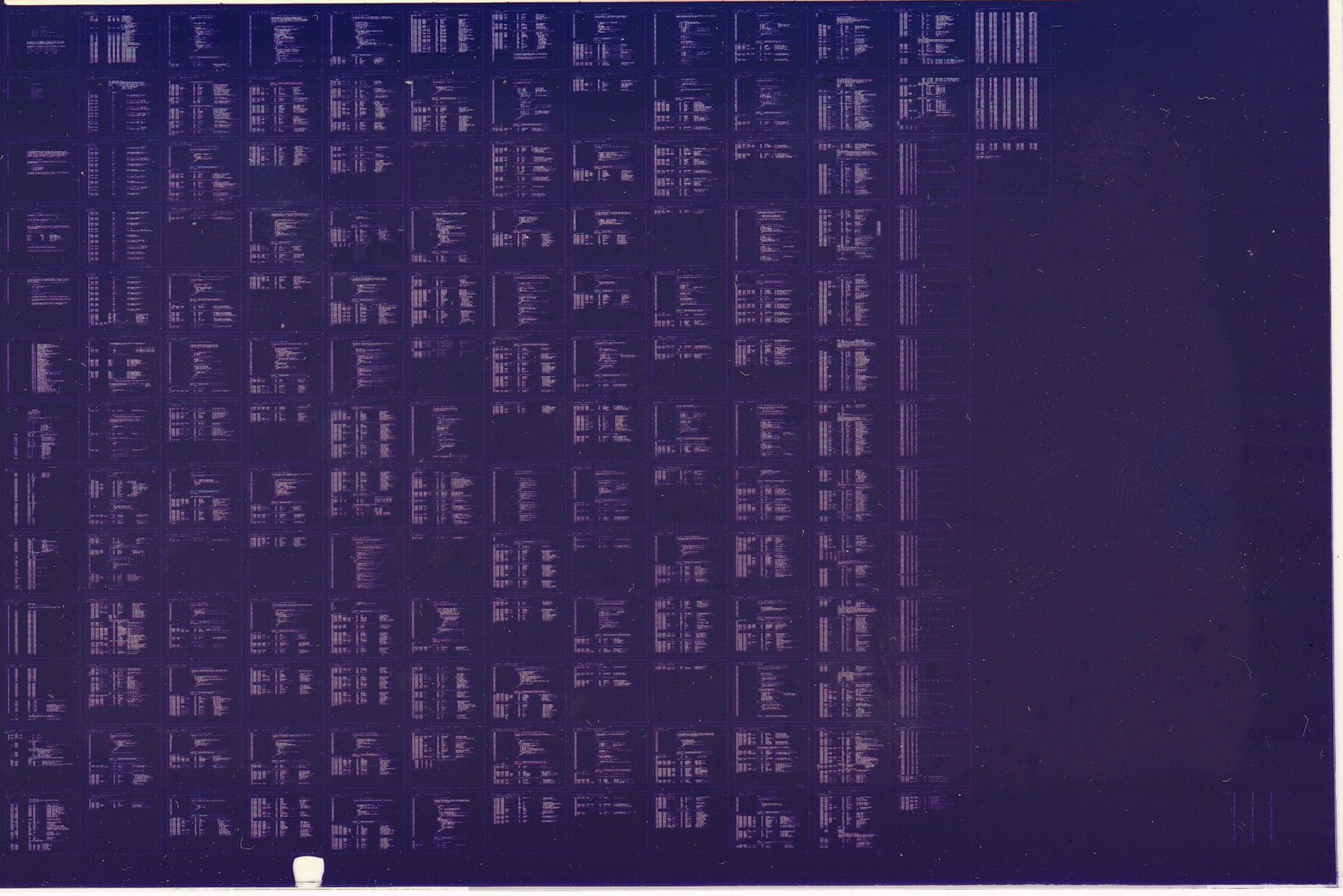
KTJ11-B KTJ11-B DIAGNOSTIC
COKTAAO

AH-T876A-MC
FICHE 01 OF 01

OCT 1984
digital
Made In USA



```
                        .REM    &
```

IDENTIFICATION

PRODUCT CODE: AC-T875A-MC

PRODUCT NAME: COKTAAO KTJ11-B DIAGNOSTIC

PRODUCT DATE: JULY, 1984

MAINTAINER: SMALL SYSTEMS DIAGNOSTIC ENGINEERING

TABLE OF CONTENTS

1. ABSTRACT

THE FOLLOWING DIAGNOSTIC TESTS THE UNIBUS ADAPTER (UBA) MODULE, KTJ11-B. THE FUNCTIONALITY OF THE MODULE IS: UNIBUS - PMI BUS ADAPTER (WHERE PMI IS A FASTER VERSION OF A Q22-BUS), M9312 COMPATIBLE BOOT FACILITY, AND THE UNIBUS MAP LOGIC. THE MODULE ALSO HAS A DMA CACHE STORE, UTILISED FOR DOING DMA TRANSFERS FROM MEMORY TO UNIBUS DEVICES. THE UBA CAN BE PROGRAMMED TO DO DIAGNOSTIC CYCLES TO VERIFY SOME OF THE FUNCTIONALITY WITHOUT REQUIRING ANY OF THE PERIPHIRALS TO BE ACTUALLY CONNECTED TO THE UNIBUS.

2. RUN-TIME REQUIREMENTS

THIS DIAGNOSTIC IS THE ONLY ONE WRITTEN SPECIFICALLY FOR THE UBA MODULE. THEREFORE, DEPENDING ON THE ENVIRONMENT IN WHICH THE PROGRAM IS RUN, DIFFERENT DEVICES CAN BE USED.

MINIMUM HARDWARE NEEDED TO RUN THE DIAGNOSTIC:

1) KDJ11-B CPU MODULE
2) AT LEAST 28K OF MEMORY
3) KTJ11-B UBA MODULE
4) CONSOLE TERMINAL
5) LOAD MEDIA

TO DO FURTHER FUNCTIONAL VERIFICATION OF THE MODULE 2 UNIBUS EXERCISERS (UBE) ARE REQUIRED. THIS SHOULD BE DONE IN MANUFACTURING OR ANY OTHER ENVIRONMENT THAT NEEDS VERIFICATION OF ALL FUNCTIONS OF THE UBA.

```
100
101                     3.      STARTING PROCEDURE
102
103                     THE DIAGNOSTIC IS A STANDART XXDP PROGRAM WITH APT INTERFACE. THEREFORE,IN
104                     STANALONE MODE, AFTER BOOTING THE SYSTEM, TYPING IN:
105
106                             R OKTAA?
107
108                     WILL START THE PROGRAM, WHICH WILL THEN PROMT THE OPERATOR FOR THE SOFTWARE
109                     REGISTER SWITCH SETTING DESRIBED BELOW.
110
111                             COKTAAO KTJ11-B DIAGNOSTIC
112
113                             SWR = XXXXXX    NEW =
114
115                     WHERE "XXXXXX" CORRESPOND TO THE OLD SETTING OF THE SOFTWARE SWITCH REGISTER.
116                     AT THIS POINT AN OPERATOR CAN EITHER TYPE IN A CARRIAGE RETURN, WHICH WOULD
117                     LEAVE THE SOFTWARE SWITCH REGISTER AS IT WAS, OR CHANGE IT, ACCORDING TO THE
118                     FOLLOWING PARAMETERS.
119
120                     OPERATIONAL SWITCH SETTINGS
121                     ---------------------------
122                     THE SWITCH SETTINGS ARE:
123                                             OCTAL           MEANING
124                                             -----           -------
125                     SW<15>=1...             100000          HALT ON ERROR
126                     SW<14>=1...              40000          LOOP ON CURRENT TEST
127                     SW<13>=1...              20000          INHIBIT ERROR TYPEOUTS
128                     SW<11>=1...               4000          INHIBIT ITERATIONS
129                     SW<10>=1...               2000          RING TTY BELL ON ERROR
130                     SW<9>=1...                1000          LOOP ON ERROR
131                     SW<8>=1...                 400          LOOP ON TEST SPECIFIED IN SW<5>
132                                                             THRU SW<0>
133
134                     FOR EXAMPLE:
135
136                             SWR = 000000    NEW = 100000
137
138                     IN THIS CASE THE OLD SOFTWARE SWITCH REGISTER DIDN'T SET ANY FLAGS. THE
139                     NEW SETTING WILL MAKE THE DIAGNOSTIC HALT ON ERROR.
140
141
142                     IF RAN FROM A UFD CHAIN FILE, THE DIAGNOSTIC IS FULLY UNDER CONTROL OF
143                     THE UFD MONITOR THAT WILL REPORT ONLY PASS/FAIL MESSAGES.
144
```

```
                        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 SETTINS IF ANYTHING DIFFERENT IS REQUIRED.

                        EXAMPLE OF ERROR PRINTOUT:

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

                        5.      EXECUTION TIME

                        THE DIAGNOSTIC RUNS A FULL PASS IN LESS THAN A MINUTE.

                        6.      UBA REGISTER DEFINITION

                        DDR     THE DIAGNOSTIC DATA REGISTER IS A BUFFER THAT PROVIDES THE ABILITY OF
                                READING AND WRITING THE DATA TO AND FROM MEMORY IN DIAGNOSTIC MODE.

                        DCSR    THE DIAGNOSTIC CONTROL AND STATUS REGISTER PROVIDES MEANS OF GOING IN
                                AND OUT OF STANDALONE MODE AND ALSO OF INITIATING DIAGNOSTIC DATI 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
                                CONTROLLLING AND DESCRIBING THE STATUS OF THE DMA CACHE.
```

```
                        7.      TESTS LIST

                        TEST 1  - UNIBUS MAP REGISTER TESTS
                        TEST 2  - UNIBUS MAP REGISTER BIT PATTERN
                        TEST 3  - UNIBUS MAP REGISTER ADDRESS UNIQUENESS
                        TEST 4  - UNIQENESS 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
```

```
                                $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.
                        ;*
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>
                        .SBTTL  BASIC DEFINITIONS
                        ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100                  STACK=  1100
104000                  ERROR=  EMT             ;;BASIC DEFINITION OF ERROR CALL
000004                  SCOPE=  IOT             ;;BASIC DEFINITION OF SCOPE CALL
                        ;*MISCELLANEOUS DEFINITIONS
000011                  HT=     11              ;;CODE FOR HORIZONTAL TAB
000012                  LF=     12              ;;CODE FOR LINE FEED
000015                  CR=     15              ;;CODE FOR CARRIAGE RETURN
000200                  CRLF=   200             ;;CODE FOR CARRIAGE RETURN-LINE FEED
177776                  PS=     177776          ;;PROCESSOR STATUS WORD
177776                  PSW=    PS
177774                  STKLMT= 177774          ;;STACK LIMIT REGISTER
177772                  PIRQ=   177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
177570                  DSWR=   177570          ;;HARDWARE SWITCH REGISTER
177570                  DDISP=  177570          ;;HARDWARE DISPLAY REGISTER
                        ;*GENERAL PURPOSE REGISTER DEFINITIONS
000000                  R0=     %0              ;;GENERAL REGISTER
000001                  R1=     %1              ;;GENERAL REGISTER
000002                  R2=     %2              ;;GENERAL REGISTER
000003                  R3=     %3              ;;GENERAL REGISTER
000004                  R4=     %4              ;;GENERAL REGISTER
000005                  R5=     %5              ;;GENERAL REGISTER
000006                  R6=     %6              ;;GENERAL REGISTER
000007                  R7=     %7              ;;GENERAL REGISTER
000006                  SP=     %6              ;;STACK POINTER
000007                  PC=     %7              ;;PROGRAM COUNTER
                        ;*PRIORITY LEVEL DEFINITIONS
000000                  PR0=    0               ;;PRIORITY LEVEL 0
000040                  PR1=    40              ;;PRIORITY LEVEL 1
000100                  PR2=    100             ;;PRIORITY LEVEL 2
```

BASIC DEFINITIONS

```
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
000004                  BIT2=   BIT02
```

BASIC DEFINITIONS

```
                        BIT1=   BIT01
                        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
  255                   .SBTTL  MEMORY MANAGEMENT DEFINITIONS
                        ;*KT11 VECTOR ADDRESS
000250                  MMVEC=  250
                        ;*KT11 STATUS REGISTER ADDRESSES
177572                  SR0=    177572
177574                  SR1=    177574
177576                  SR2=    177576
172516                  SR3=    172516
                        ;*KERNEL "I" PAGE DESCRIPTOR REGISTERS
172300                  KIPDR0= 172300
172302                  KIPDR1= 172302
172304                  KIPDR2= 172304
172306                  KIPDR3= 172306
172310                  KIPDR4= 172310
172312                  KIPDR5= 172312
172314                  KIPDR6= 172314
172316                  KIPDR7= 172316
                        ;*KERNEL "D" PAGE DESCRIPTOR REGISTERS
172320                  KDPDR0= 172320
172322                  KDPDR1= 172322
172324                  KDPDR2= 172324
172326                  KDPDR3= 172326
172330                  KDPDR4= 172330
172332                  KDPDR5= 172332
172334                  KDPDR6= 172334
172336                  KDPDR7= 172336
                        ;*KERNEL "I" PAGE ADDRESS REGISTERS
172340                  KIPAR0= 172340
172342                  KIPAR1= 172342
172344                  KIPAR2= 172344
172346                  KIPAR3= 172346
172350                  KIPAR4= 172350
172352                  KIPAR5= 172352
172354                  KIPAR6= 172354
172356                  KIPAR7= 172356
                        ;*KERNEL "D" PAGE ADDRESS REGISTERS
172360                  KDPAR0= 172360
172362                  KDPAR1= 172362
172364                  KDPAR2= 172364
172366                  KDPAR3= 172366
172370                  KDPAR4= 172370
172372                  KDPAR5= 172372
```

MEMORY MANAGEMENT DEFINITIONS

```
                172374                  KDPAR6= 172374
                172376                  KDPAR7= 172376
                                        .SBTTL  UNIBUS MAP REGISTER DEFINITIONS
    256
    257
    258                                 ;*THE LOWER 16 BITS OF THE MAP REGISTERS ARE LABELED 'MAPLXX'
    259                                 ;*THE UPPER 6 BITS OF THE MAP REGISTERS ARE LABELED 'MAPHXX'
    260
    261
    262         170200                  MAPL00 = 170200
    263         170202                  MAPH00 = 170202
    264         170204                  MAPL01 = 170204
    265         170206                  MAPH01 = 170206
    266         170210                  MAPL02 = 170210
    267         170212                  MAPH02 = 170212
    268         170214                  MAPL03 = 170214
    269         170216                  MAPH03 = 170216
    270         170220                  MAPL04 = 170220
    271         170222                  MAPH04 = 170222
    272         170224                  MAPL05 = 170224
    273         170226                  MAPH05 = 170226
    274         170230                  MAPL06 = 170230
    275         170232                  MAPH06 = 170232
    276         170234                  MAPL07 = 170234
    277         170236                  MAPH07 = 170236
    278         170240                  MAPL10 = 170240
    279         170242                  MAPH10 = 170242
    280         170244                  MAPL11 = 170244
    281         170246                  MAPH11 = 170246
    282         170250                  MAPL12 = 170250
    283         170252                  MAPH12 = 170252
    284         170254                  MAPL13 = 170254
    285         170256                  MAPH13 = 170256
    286         170260                  MAPL14 = 170260
    287         170262                  MAPH14 = 170262
    288         170264                  MAPL15 = 170264
    289         170266                  MAPH15 = 170266
    290         170270                  MAPL16 = 170270
    291         170272                  MAPH16 = 170272
    292         170274                  MAPL17 = 170274
    293         170276                  MAPH17 = 170276
    294         170300                  MAPL20 = 170300
    295         170302                  MAPH20 = 170302
    296         170304                  MAPL21 = 170304
    297         170306                  MAPH21 = 170306
    298         170310                  MAPL22 = 170310
    299         170312                  MAPH22 = 170312
    300         170314                  MAPL23 = 170314
    301         170316                  MAPH23 = 170316
    302         170320                  MAPL24 = 170320
    303         170322                  MAPH24 = 170322
    304         170324                  MAPL25 = 170324
    305         170326                  MAPH25 = 170326
    306         170330                  MAPL26 = 170330
    307         170332                  MAPH26 = 170332
    308         170334                  MAPL27 = 170334
    309         170336                  MAPH27 = 170336
    310         170340                  MAPL30 = 170340
```

UNIBUS MAP REGISTER DEFINITIONS

```
311             170342                  MAPH30 = 170342
312             170344                  MAPL31 = 170344
313             170346                  MAPH31 = 170346
314             170350                  MAPL32 = 170350
315             170352                  MAPH32 = 170352
316             170354                  MAPL33 = 170354
317             170356                  MAPH33 = 170356
318             170360                  MAPL34 = 170360
319             170362                  MAPH34 = 170362
320             170364                  MAPL35 = 170364
321             170366                  MAPH35 = 170366
322             170370                  MAPL36 = 170370
323             170372                  MAPH36 = 170372
324             170374                  MAPL37 = 170374
325             170376                  MAPH37 = 170376
326
327             170200                  MAPL0 = MAPL00
328             170202                  MAPH0 = MAPH00
329             170204                  MAPL1 = MAPL01
330             170206                  MAPH1 = MAPH01
331             170210                  MAPL2 = MAPL02
332             170212                  MAPH2 = MAPH02
333             170214                  MAPL3 = MAPL03
334             170216                  MAPH3 = MAPH03
335             170220                  MAPL4 = MAPL04
336             170222                  MAPH4 = MAPH04
337             170224                  MAPL5 = MAPL05
338             170226                  MAPH5 = MAPH05
339             170230                  MAPL6 = MAPL06
340             170232                  MAPH6 = MAPH06
341             170234                  MAPL7 = MAPL07
342             170236                  MAPH7 = MAPH07
343
344                                     .SBTTL UBA SPECIFIC REGISTERS
345
346             177572                  MMR0    = 177572                ; MEMORY MANAGEMENT REGISTER DEFINITIONS
347             177574                  MMR1    = 177574                ;
348             177576                  MMR2    = 177576                ;
349             172516                  MMR3    = 172516                ;
350             000001                  UFDSET  = 1                     ; FLAG FOR UFD MODE
351             177520                  BCSR    = 177520                ; BOOT/DIAGNOSTIC STATUS REGISTER
352             177730                  DCSR    = 177730                ; DIAGNOSTIC CONTROLLER STATUS REGISTER
353             177732                  DDR     = 177732                ; DIAGNOSTIC DATA REISTER
354             177734                  KMCR    = 177734                ; KTJ11-B MEMORY CONFIGURATION REGISTER
355             177746                  CCR     = 177746                ; CACHE CONTOL REGISTER FOR CPU
356             177522                  PCR     = 177522                ; PAGE CONTROL REGISTER
357             120001                  POLY    = 120001                ; POLYNOMIAL USED FOR CRC ROUTINES
358             170014                  SIMLGO  = 170014                ; SIMULTANEOUS GO ADDRESS FOR MULTIPLE UBE'S
359
360                                     .SBTTL  TRAP CATCHER
                000000                          .=0
                                        ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
                                        ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
                                        ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
                000174                          .=174
    000174      000000              DISPREG: .WORD  0                   ;;SOFTWARE DISPLAY REGISTER
    000176      000000              SWREG:   .WORD  0                   ;;SOFTWARE SWITCH REGISTER
```

TRAP CATCHER

```
                        000200                  .=200
000200  005037  001160                          CLR     $TMP0
000204  000137  002500                          JMP     @#START
                        000220                  .=220
000220  012737  000777  001160                  MOV     #777,$TMP0
000226  000137  002500                          JMP     @#START

                                                .SBTTL  ACT11 HOOKS
                                                ;;******************************************************************
                                                ;HOOKS REQUIRED BY ACT11
                        000232                          $SVPC=.                 ;SAVE PC
                        000046                          .=46
000046  022672                                  $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
                        000052                          .=52
000052  000000                                  .WORD   0               ;;2)SET LOC.52 TO ZERO
                        000232                          .=$SVPC                 ;; RESTORE PC
                                                .SBTTL  APT PARAMETER BLOCK
                                                ;;******************************************************************
                                                ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                                ;;******************************************************************
                        000232                          .$X=.   ;;SAVE CURRENT LOCATION
                        000024                          .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
000024  000200                                  200     ;;FOR APT START UP
                        000044                          .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
000044  000232                                  $APTHDR ;;POINT TO APT HEADER BLOCK
                        000232                          .=.$X   ;;RESET LOCATION COUNTER
                                                ;;******************************************************************
                                                ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                                                ;INTERFACE SPEC.
000232                                          $APTHD:
000232  000000                                  $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
000234  001200                                  $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
000236  000000                                  $TSTM:  .WORD           ;;RUN TIM OF LONGEST TEST
000240  000000                                  $PASTM: .WORD           ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
000242  000000                                  $UNITM: .WORD           ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
000244  000052                                          .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

```
    370                             .SBTTL  COMMON TAGS
                                    ;;**********************************************************************
                                    ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
                                    ;*USED IN THE PROGRAM.
            001100                          .=1100
    001100                          $CMTAG:                         ;;START OF COMMON TAGS
    001100  000000                          .WORD   0
    001102     000                  $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
    001103     000                  $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
    001104  000000                  $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
    001106  000000                  $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
    001110  000000                  $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
    001112  000000                  $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
    001114     000                  $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
    001115     001                  $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
    001116  000000                  $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
    001120  000000                  $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
    001122  000000                  $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
    001124  000000                  $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
    001126  000000                  $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
    001130  000000                          .WORD   0               ;;RESERVED--NOT TO BE USED
    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)
            000002                  .REPT   2
    001160  000000                  $TMPO:  .WORD   0               ;;USER DEFINED
    001162  000000                  $TMP1:  .WORD   0               ;;USER DEFINED
    001164  000000                  $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
    001166  000000                  $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
    001170     207      377     377 $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
    001173     000
    001174     077                  $QUES:  .ASCII  /?/             ;;QUESTION MARK
    001175     015                  $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
    001176     012      000         $LF:    .ASCIZ  <12>            ;;LINE FEED
                                    ;;**********************************************************************
                                    .SBTTL  APT MAILBOX-ETABLE
                                    ;;**********************************************************************
                                    .EVEN
    001200                          $MAIL:                  ;;APT MAILBOX
    001200  000000                  $MSGTY: .WORD   AMSGTY   ;;MESSAGE TYPE CODE
    001202  000000                  $FATAL: .WORD   AFATAL   ;;FATAL ERROR NUMBER
    001204  000000                  $TESTN: .WORD   ATESTN   ;;TEST NUMBER
    001206  000000                  $PASS:  .WORD   APASS    ;;PASS COUNT
    001210  000000                  $DEVCT: .WORD   ADEVCT   ;;DEVICE COUNT
    001212  000000                  $UNIT:  .WORD   AUNIT    ;;I/O UNIT NUMBER
    001214  000000                  $MSGAD: .WORD   AMSGAD   ;;MESSAGE ADDRESS
```

APT MAILBOX-ETABLE

```
001216  000000                  $MSGLG: .WORD   AMSGLG  ;;MESSAGE LENGTH
001220                          $ETABLE:                ;;APT ENVIRONMENT TABLE
001220     000                  $ENV:   .BYTE   AENV    ;;ENVIRONMENT BYTE
001221     000                  $ENVM:  .BYTE   AENVM   ;;ENVIRONMENT MODE BITS
001222  000000                  $SWREG: .WORD   ASWREG  ;;APT SWITCH REGISTER
001224  000000                  $USWR:  .WORD   AUSWR   ;;USER SWITCHES
001226  000000                  $CPUOP: .WORD   ACPUOP  ;;CPU TYPE,OPTIONS
                                ;*                      BITS 15-11=CPU TYPE
                                ;*                              11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
                                ;*                              11/70=06,PDQ=07,Q=10
                                ;*                      BIT 10=REAL TIME CLOCK
                                ;*                      BIT  9=FLOATING POINT PROCESSOR
                                ;*                      BIT  8=MEMORY MANAGEMENT
001230     000                  $MAMS1: .BYTE   AMAMS1  ;;HIGH ADDRESS,M.S. BYTE
001231     000                  $MTYP1: .BYTE   AMTYP1  ;;MEM. TYPE,BLK#1
                                ;*                      MEM.TYPE BYTE   --  (HIGH BYTE)
                                ;*                              900 NSEC CORE=001
                                ;*                              300 NSEC BIPOLAR=002
                                ;*                              500 NSEC MOS=003
001232  000000                  $MADR1: .WORD   AMADR1  ;;HIGH ADDRESS,BLK#1
                                ;*                        MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
001234     000                  $MAMS2: .BYTE   AMAMS2  ;;HIGH ADDRESS,M.S. BYTE
001235     000                  $MTYP2: .BYTE   AMTYP2  ;;MEM.TYPE,BLK#2
001236  000000                  $MADR2: .WORD   AMADR2  ;;MEM.LAST ADDRESS,BLK#2
001240     000                  $MAMS3: .BYTE   AMAMS3  ;;HIGH ADDRESS,M.S.BYTE
001241     000                  $MTYP3: .BYTE   AMTYP3  ;;MEM.TYPE,BLK#3
001242  000000                  $MADR3: .WORD   AMADR3  ;;MEM.LAST ADDRESS,BLK#3
001244     000                  $MAMS4: .BYTE   AMAMS4  ;;HIGH ADDRESS,M.S.BYTE
001245     000                  $MTYP4: .BYTE   AMTYP4  ;;MEM.TYPE,BLK#4
001246  000000                  $MADR4: .WORD   AMADR4  ;;MEM.LAST ADDRESS,BLK#4
001250  000000                  $VECT1: .WORD   AVECT1  ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
001252  000000                  $VECT2: .WORD   AVECT2  ;;INTERRUPT VECTOR#2BUS PRIORITY#2
001254  000000                  $BASE:  .WORD   ABASE   ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
001256  000000                  $DEVM:  .WORD   ADEVM   ;;DEVICE MAP
001260  000000                  $CDW1:  .WORD   ACDW1   ;;CONTROLLER DESCRIPTION WORD#1
001262  000000                  $CDW2:  .WORD   ACDW2   ;;CONTROLLER DESCRIPTION WORD#2
001264  000000                  $DDW0:  .WORD   ADDW0   ;;DEVICE DESCRIPTOR WORD#0
001266  000000                  $DDW1:  .WORD   ADDW1   ;;DEVICE DESCRIPTOR WORD#1
001270  000000                  $DDW2:  .WORD   ADDW2   ;;DEVICE DESCRIPTOR WORD#2
001272  000000                  $DDW3:  .WORD   ADDW3   ;;DEVICE DESCRIPTOR WORD#3
001274  000000                  $DDW4:  .WORD   ADDW4   ;;DEVICE DESCRIPTOR WORD#4
001276  000000                  $DDW5:  .WORD   ADDW5   ;;DEVICE DESCRIPTOR WORD#5
001300  000000                  $DDW6:  .WORD   ADDW6   ;;DEVICE DESCRIPTOR WORD#6
001302  000000                  $DDW7:  .WORD   ADDW7   ;;DEVICE DESCRIPTOR WORD#7
001304  000000                  $DDW8:  .WORD   ADDW8   ;;DEVICE DESCRIPTOR WORD#8
001306  000000                  $DDW9:  .WORD   ADDW9   ;;DEVICE DESCRIPTOR WORD#9
001310  000000                  $DDW10: .WORD   ADDW10  ;;DEVICE DESCRIPTOR WORD#10
001312  000000                  $DDW11: .WORD   ADDW11  ;;DEVICE DESCRIPTOR WORD#11
001314  000000                  $DDW12: .WORD   ADDW12  ;;DEVICE DESCRIPTOR WORD#12
001316  000000                  $DDW13: .WORD   ADDW13  ;;DEVICE DESCRIPTOR WORD#13
001320  000000                  $DDW14: .WORD   ADDW14  ;;DEVICE DESCRIPTOR WORD#14
001322  000000                  $DDW15: .WORD   ADDW15  ;;DEVICE DESCRIPTOR WORD#15
001324                          $ETEND:
```

ERROR POINTER TABLE

```
                                .SBTTL  ERROR POINTER TABLE
                                ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
                                ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
                                ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
                                ;*NOTE1:        IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
                                ;*NOTE2:        EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
                                ;*      EM              ;;POINTS TO THE ERROR MESSAGE
                                ;*      DH              ;;POINTS TO THE DATA HEADER
                                ;*      DT              ;;POINTS TO THE DATA
                                ;*      DF              ;;POINTS TO THE DATA FORMAT
    001324                      $ERRTB:
 371
 372                            .SBTTL  ERROR DEFINITIONS
 373                            ;       ERROR 1
 374
 375 001324  026514                     EM1                     ; TIMEOUT ON ACCESSING A MAP REGISTER
 376 001326  030702                     DH1                     ; TEST # ERROR PC  ERROR #  ADDRESS
 377 001330  031674                     DT1                     ; TEST,$ERRPC,ERRNUM,$BDADR
 378 001332  000000                     0
 379                            ;       ERROR 2
 380
 381 001334  026560                     EM2                     ; MAP REGISTER COULD NOT BE CLEARED
 382 001336  030751                     DH2                     ;                          GOOD  BAD
 383                                                            ; TEST # ERROR PC  ERROR # DATA  DATA ADDRESS
 384 001340  031706                     DT2                     ; TEST,$ERRPC,ERRNUM,$GDDAT,$BDDAT,$BDADR
 385 001342  000000                     0
 386                            ;       ERROR 3
 387
 388 001344  026622                     EM3                     ; MAP REGISTER COULD NOT HOLD PATTERN
 389 001346  030751                     DH2                     ;                          GOOD  BAD
 390                                                            ; TEST # ERROR PC  ERROR # DATA  DATA ADDRESS
 391 001350  031706                     DT2                     ; TEST,$ERRPC,ERRNUM,$GDDAT,$BDDAT,$BDADR
 392 001352  000000                     0
 393                            ;       ERROR 4
 394
 395 001354  026666                     EM4                     ; MAP REGISTER HAS NOT BEEN ADDRESSED CORRECTLY
 396 001356  031043                     DH3                     ;                          GOOD     BAD
 397                                                            ; TEST # ERROR PC  ERROR #  ADDRESS  ADDRESS
 398 001360  031724                     DT3                     ; TEST,$ERRPC,ERRNUM,$GDADR,$BDADR
 399 001362  000000                     0
 400                            ;       ERROR 5
 401
 402 001364  026744                     EM5                     ; THERE WAS NO DIFFERENCE FOUND BETWEEN HI AND LO MAP REGIST
ERS
 403 001366  031133                     DH4                     ;                            HI    LOW
 404                                                            ; TEST # ERROR PC  ERROR #  MAP   MAP
 405 001370  031740                     DT4                     ; TEST,$ERRPC,ERRNUM,$GDDAT,$BDDAT
 406 001372  000000                     0
 407                            ;       ERROR 6
 408
 409 001374  027042                     EM6                     ; ERROR IN BITS 3-6,9-14 IN THE DCSR
 410 001376  031211                     DH5                     ;                          GOOD  BAD
 411                                                            ; TEST # ERROR PC  ERROR #  DATA  DATA
 412 001400  031740                     DT4                     ; TEST,$ERRPC,ERRNUM,$GDDAT,$BDDAT
 413 001402  000000                     0
 414                            ;       ERROR 7
 415
 416 001404  027105                     EM7                     ; DCSR DID RESPOND PROPERLY ON RESET
```

ERROR DEFINITIONS

```
                                        DH5                     ;                                       GOOD  BAD
                                                                ; TEST # ERROR PC  ERROR #  DATA  DATA
                                        DT4                     ; TEST,$ERRPC,ERRNUM,$GDDAT,$BDDAT
                                        0
                                ;       ERROR 10

                                        EM10                    ; TIMEOUT HAS OCCURED ON ACCESS TO THE DCSR
                                        DH1                     ; TEST # ERROR PC  ERROR #  ADDRESS
                                        DT1                     ; TEST,$ERRPC,ERRNUM,$BDADR
                                        0
                                ;       ERROR 11

                                        EM11                    ; KMCR BITS 0-5,8 DID NOT GET SET CORRECTLY
                                        DH2                     ;                                       GOOD  BAD
                                                                ; TEST # ERROR PC  ERROR # DATA  DATA ADDRESS
                                        DT2                     ; TEST,$ERRPC,ERRNUM,$GDDAT,$BDDAT,$BDADR
                                        0
                                ;       ERROR 12

                                        EM12                    ; TIMEOUT HAS OCCURED ON ACCESS TO THE KMCR
                                        DH1                     ; TEST # ERROR PC  ERROR #  ADDRESS
                                        DT1                     ; TEST,$ERRPC,ERRNUM,$BDADR
                                        0
                                ;       ERROR 13

                                        EM13                    ; ERROR IN DATA PATH
                                        DH13                    ; TEST # ERROR PC ERROR # PATTERN DDR
                                        DT4                     ; TEST,$ERRPC,ERRNUM,$GDDAT,$BDDAT
                                        0
                                ;       ERROR 14

                                        EM14                    ; ERROR IN DATA OUT
                                        DH5                     ;                                       GOOD  BAD
                                                                ; TEST # ERROR PC  ERROR #  DATA  DATA
                                        DT4                     ; TEST,$ERRPC,ERRNUM,$GDDAT,$BDDAT
                                        0
                                ;       ERROR 15

                                        EM15                    ; ERROR IN DATA IN
                                        DH13                    ; TEST # ERROR PC  ERROR # PATTERN DDR
                                        DT4                     ; TEST,$ERRPC,ERRNUM,$GDDAT,$BDDAT
                                        0
                                ;       ERROR 16

                                        EM16                    ; DDR NOT ZERO WHEN DCSR SELECTS UNIBUS LINES
                                        DH5                     ;                                       GOOD  BAD
                                                                ; TEST # ERROR PC  ERROR #  DATA  DATA
                                        DT4                     ; TEST,$ERRPC,ERRNUM,$GDDAT,$BDDAT
                                        0
                                ;       ERROR 17

                                        EM17                    ; ERROR IN SETTING DCSR<7>
                                        DH16                    ; TEST # ERROR PC ERROR #
                                        DT16                    ; TEST,$ERRPC,ERRNUM
                                        0
                                ;       ERROR 20
```

ERROR DEFINITIONS

```
    474 001514  027615                      EM20            ; ERROR IN UNIQUE ADDRESSING OF REGISTERS
    475 001516  031405                      DH20            ; TEST # ERROR PC ERROR # KMCR PAIR FAILED
    476 001520  031764                      DT20            ; TEST,$ERRPC,ERRNUM,KMCR, $BDADR
    477 001522  000000                      0
    478                             ;       ERROR 21
    479
    480 001524  027673                      EM21            ; REG. PAIR 31. PERFORMS RELOCATION
    481 001526  031346                      DH16            ; TEST # ERROR PC ERROR #
    482 001530  031754                      DT16            ; TEST,$ERRPC,ERRNUM
    483 001532  000000                      0
    484                             ;       ERROR 22
    485
    486 001534  027740                      EM22            ; NXM CONDITION COULDN'T BE CREATED THRU ALU
    487 001536  031346                      DH16            ; TEST # ERROR PC ERROR #
    488 001540  031754                      DT16            ; TEST,$ERRPC,ERRNUM
    489 001542  000000                      0
    490                             ;       ERROR 23
    491
    492 001544  030014                      EM23            ; ALU ERROR
    493 001546  031465                      DH23            ;                                   GOOD  BAD
    494                                                     ; TEST # ERROR PC  ERROR # DATA  DATA ADDRESS
    495 001550  032010                      DT23            ; TEST,$ERRPC,ERRNUM,$GDDAT,$BDDAT,KIPAR6,$BDADR
    496 001552  000000                      0
    497                             ;       ERROR 24
    498
    499 001554  030026                      EM24            ; CPU CACHE ERROR
    500 001556  031346                      DH16            ; TEST # ERROR PC ERROR #
    501 001560  031754                      DT16            ; TEST,$ERRPC,ERRNUM
    502 001562  000000                      0
    503                             ;       ERROR 25
    504
    505 001564  030046                      EM25            ; KMCR<4-0> DOESN'T DISABLE MAIN MEMORY
    506 001566  031346                      DH16            ; TEST # ERROR PC ERROR #
    507 001570  031754                      DT16            ; TEST,$ERRPC,ERRNUM
    508 001572  000000                      0
    509                             ;       ERROR 26
    510
    511 001574  030116                      EM26            ; KMCR DOES NOT REFLECT EXPECTED STATUS OF THE CACHE/
    512 001576  031211                      DH5             ;                                   GOOD  BAD
    513                                                     ; TEST # ERROR PC  ERROR #  DATA  DATA
    514 001600  031740                      DT4             ; TEST,$ERRPC,ERRNUM,$GDDAT,$BDDAT
    515 001602  000000                      0
    516                             ;       ERROR 27
    517
    518 001604  030201                      EM27            ; ERROR IN CACHE TAG REGISTERS
    519 001606  031576                      DH27            ; TEST # ERROR PC  ERROR # ADDRESS
    520 001610  032030                      DT27            ; TEST,$ERRPC,ERRNUM,MAPH01,MAPL01
    521 001612  000000                      0
    522                             ;       ERROR 30
    523
    524 001614  030236                      EM30            ; ERROR IN THE DMA CACHE DATA RAMS
    525 001616  030702                      DH1             ; TEST # ERROR PC  ERROR # ADDRESS
    526 001620  032044                      DT30            ; TEST,$ERRPC,ERRNUM,MAPL00
    527 001622  000000                      0
    528                             ;       ERROR 31
    529
    530 001624  030277                      EM31            ; ERROR IN THE M9312 BOOT ROM SECTION
```

ERROR DEFINITIONS

```
    531 001626  031346                          DH16                    ; TEST # ERROR PC  ERROR #
    532 001630  031754                          DT16                    ; TEST,$ERRPC,ERRNUM
    533 001632  000000                          0
    534                                 ;       ERROR 32
    535
    536 001634  030343                          EM32                    ; ERROR IN ARBITRATION LOGIC THRU UBE
    537 001636  031346                          DH16                    ; TEST # ERROR PC  ERROR #
    538 001640  031754                          DT16                    ; TEST,$ERRPC,ERRNUM
    539 001642  000000                          0
    540                                 ;       ERROR 33
    541
    542 001644  030410                          EM33                    ; ERROR TRYING TO DO DMA CYCLES THRU UBE
    543 001646  031346                          DH16                    ; TEST # ERROR PC  ERROR #
    544 001650  031754                          DT16                    ; TEST,$ERRPC,ERRNUM
    545 001652  000000                          0
    546                                 ;       ERROR 34
    547
    548 001654  030464                          EM34                    ; ERROR IN THE UNIBUS MEMORY TEST
    549 001656  031346                          DH16                    ; TEST # ERROR PC  ERROR #
    550 001660  031754                          DT16                    ; TEST,$ERRPC,ERRNUM
    551 001662  000000                          0
    552                                 ;       ERROR 35
    553
    554 001664  030524                          EM35                    ; UNEXPECTED TIMEOUT HAS OCCURED
    555 001666  031346                          DH16                    ; TEST # ERROR PC  ERROR #
    556 001670  032000                          DT21                    ; TEST,$BDADR,ERRNUM
    557 001672  000000                          0
    558                                 ;       ERROR 36
    559
    560 001674  030563                          EM36                    ; UNIBUS TIMEOUT DID NOT OCCUR
    561 001676  031346                          DH16                    ; TEST # ERROR PC  ERROR #
    562 001700  031754                          DT16                    ; TEST,$ERRPC,ERRNUM
    563 001702  000000                          0
    564                                 ;       ERROR 37
    565
    566 001704  030631                          EM37                    ; DCSR<3> DIDN'T DISABLE UBA ROM'S
    567 001706  031346                          DH16                    ; TEST # ERROR PC  ERROR #
    568 001710  031754                          DT16                    ; TEST,$ERRPC,ERRNUM
    569 001712  000000                          0
    570
    571                                 .SBTTL  GLOBAL VARIABLES
    572 001714  000000                  ERRNUM: .WORD   0                       ; ERROR NUMBER FOR REPORT
    573 001716  000000                  TEST:   .WORD   0                       ; TEST NUMBER FOR REPORT
    574 001720  000000                  PMIS:   .WORD   0                       ; PMI MEMORY SIZE
    575 001722  000000                  UBECT:  .WORD   0                       ; NUMBER OF UBE'S
    576 001724  000000                  BE1INT: .WORD   0                       ; UBE #1 INTERRUPT FLAG
    577 001726  000000                  BE2INT: .WORD   0                       ; UBE #2 INTERRUPT FLAG
    578 001730  000000                  TOUT:   .WORD   0                       ; TIMEOUT FLAG
    579 001732  172100                  MCSR:   .WORD   172100                  ; POINTER TO MEMORY CSR
    580 001734                          WRTBUF: .BLKW   20                      ; WRITE BUFFER
    581 001774  000377  007417  031463  PTRN16: .WORD   377,7417,31463,52525,125252,0   ; 16-BIT BINARY DIVIDE
        002002  052525  125252  000000
    582 002010  000017  000014  000025  PTRN6:  .WORD   17,14,25,52,0                    ; 6-BIT BINARY DIVIDE
        002016  000052  000000
    583
    584                                 .SBTTL  CACHE RAM IMAGE TABLE
    585
```

CACHE RAM IMAGE TABLE

```
                                ;
                                ; THE FOLLOWING BLOCK OF DATA WILL BE USED IN THE CACHE RAM BIT TEST
                                ; THE EXACT ADDRESSES WILL BE FIGURED OUT BY THE PROGRAM DEPENDING
                                ; ON THE LOCATION OF THIS TABLE.
                                ;

002022                          CTBLE:  .BLKW   47                      ; THE LARGEST # OF LOCATIONS NEEDED

002140  000000                  OBADR:  .WORD   0                       ; FIRST ADDRESS OF A FIELD IN TABLE
002142  000000                          .WORD   0                       ; FIRST ADDRESS OF B FIELD IN TABLE
002144  000000                          .WORD   0                       ; FIRST ADDRESS OF C FIELD IN TABLE
002146  000000                          .WORD   0                       ; FIRST ADDRESS OF D FIELD IN TABLE

                                .SBTTL  UNIBUS EXERCISER REGISTER TABLES

                                ;
                                ;       UBE #1
                                ;

002150  000000                  BE1DB:  0                       ; UBE #1 DATA REGISTER
002152  000000                  BE1CC:  0                       ; UBE #1 CYCLE COUNT REGISTER
002154  000000                  BE1BA:  0                       ; UBE #1 ADDRESS REGISTER
002156  000000                  BE1CR1: 0                       ; UBE #1 CONTROL REGISTER 1
002160  000000                  BE1CLR: 0                       ; UBE #1 CLEAR ERROR REGISTER ADDRESS
002162  000000                  BE1CR2: 0                       ; UBE #1 CONTROL REGISTER 2
002164  000000                  BE1VEC: 0                       ; UBE #1 VECTOR PC
002166  000000                  BE1PSW: 0                       ; UBE #1 VECTOR PSW

                                ;
                                ;       UBE #2
                                ;

002170  000000                  BE2DB:  0                       ; UBE #2 DATA REGISTER
002172  000000                  BE2CC:  0                       ; UBE #2 CYCLE COUNT REGISTER
002174  000000                  BE2BA:  0                       ; UBE #2 ADDRESS REGISTER
002176  000000                  BE2CR1: 0                       ; UBE #2 CONTROL REGISTER 1
002200  000000                  BE2CLR: 0                       ; UBE #2 CLEAR ERROR REGISTER ADDRESS
002202  000000                  BE2CR2: 0                       ; UBE #2 CONTROL REGISTER 2
002204  000000                  BE2VEC: 0                       ; UBE #2 VECTOR PC
002206  000000                  BE2PSW: 0                       ; UBE #2 VECTOR PSW
                                .SBTTL SUBROUTINE - DIAGNOSTIC_DATA_OUT SUBROUTINE

                                ;* INPUTS: PATTERN TO BE STORED IN DDR AND THEN WRITTEN TO MEMORY $GDDAT
                                ;*         TEST_LOCATION TO BE WRITTEN TO (16 OR 22 BITS) (R1)
                                ;
                                ;* ON RETURN FROM SUBROUTINE TEST_LOCATION SHOULD HAVE THE SAME DATA
                                ;* AS DDR AND THE SAME AS PATTERN
                                ;* THE PROGRAM HAS TO BE RUNNING IN DIAGNOSTIC MODE WITH DIAGNOSTIC
                                ;* NPR REGISTER SELECTED
                                ;
                                ;  BGNROUTINE
                                ;
                                ;       MOVE PATTERN TO DDR
                                ;       DO EXTERNAL WRITE FROM TEST_LOCATION
                                ;       RETURN
                                ;
                                ;  ENDROUTINE
```

```
                                        ;
                                        ;-------------------------------------------------------------------------
002210  013737  001124  177732  DDOUT:  MOV     $GDDAT,DDR              ; STORE PATTERN IN DDR
002216  011111                  1$:     MOV     (R1),(R1)               ; EXTENAL READ TO PROVIDE ADDRESS
002220  000207                          RTS     PC

                                        .SBTTL SUBROUTINE - DIAGNOSTIC_DATA_IN SUBROUITNE

                                        ;* INPUTS: PATTERN TO BE WRITTEN TO MEMORY AND THEN TO DDR $GDDAT
                                        ;*         TEST_LOCATION TO READ FROM (16 OR 22 BITS) (R1)
                                        ;
                                        ;* ON RETURN FROM SUBROUTINE DDR SHOULD HAVE THE SAME DATA AS
                                        ;* SPECIFIED MEMORY LOCATION AND THE SAME AS THE PATTERN
                                        ;* THE PROGRAM HAS TO BE RUNNING IN DIAGNOSTIC MODE WITH DIAGNOSTIC
                                        ;* NPR REGISTER SELECTED.
                                        ;
                                        ;  BGNROUTINE
                                        ;
                                        ;       LET DCSR<0> = #1
                                        ;       DO EXTERNAL WRITE FROM TEST_LOCATION
                                        ;       RETURN
                                        ;
                                        ;  ENDROUTINE
                                        ;
                                        ;-------------------------------------------------------------------------
002222  052737  000001  177730  DDIN:   BIS     #BIT00,DCSR             ; SET GO BIT
002230  011111                  2$:     MOV     (R1),(R1)               ; PROVIDE ADDRESS FOR DMA
002232  000207                  3$:     RTS     PC

                                        .SBTTL SUBROUTINE - TIMEOUT_ROUTINE
                                        ;* THIS ROUTINE IS USED TO FLAG AN UNEXPECTED TIMEOUT.
                                        ;
                                        ;  BGNROUTINE
                                        ;
                                        ;       STORE ADDRESS THAT CAUSED TIMEOUT
                                        ;       ERROR
                                        ;       RETURN
                                        ;
                                        ;  ENDROUTINE
                                        ;
                                        ;-------------------------------------------------------------------------

002234  011637  001122          TIMOUT: MOV     (SP),$BDADR             ; STORE ADDRESS THAT TIMED OUT
002240  104035                          ERROR   +35                     ; UNEXPECTED TIMEOUT
002242  000002                          RTI

                                        .SBTTL SUBROUTINE - MAP_PROGRAM_AREA

                                        ;* THIS ROUTINE MAPS THE PROGRAM AREA TO THE FIRST 32K
                                        ;
                                        ;  BGNROUTINE
                                        ;
```

SUBROUTINE - MAP_PROGRAM_AREA

```
                                ;       MAP PROGRAM AREA THRU KIPAR'S TO FIRST 32 K
                                ;       NO CACHE BYPASS
                                ;       RETURN
                                ;
                                ;  ENDROUTINE
                                ;
                                ;-------------------------------------------------------------

002244                          MAPPR:
002244  010046                         MOV     R0,-(SP)        ;;PUSH R0 ON STACK
002246  010146                         MOV     R1,-(SP)        ;;PUSH R1 ON STACK
002250  010246                         MOV     R2,-(SP)        ;;PUSH R2 ON STACK
002252  012700  172300                 MOV     #172300,R0              ; R0 POINTS TO FIRST KIPDR
002256  012701  000010                 MOV     #8.,R1                  ; DO FOR ALL 8 REGISTERS
002262  012720  077406         1$:     MOV     #77406,(R0)+            ; . 4K PAGE, CACHE ON, READ/WRITE
002266  077103                         SOB     R1,1$                   ; . CONTINUE TILL DONE
002270  012700  172340                 MOV     #172340,R0              ; R0 POINTS TO FIRST KIPAR
002274  012701  000006                 MOV     #6.,R1                  ; DO FOR ALL 8 REGISTERS
002300  012702  000000                 MOV     #0,R2                   ; START WITH ADDRESS 0
002304  010220         2$:     MOV     R2,(R0)+                ; . LOAD WITH ADDRESS
002306  062702  000200                 ADD     #200,R2                 ; . NEXT 4K
002312  077104                         SOB     R1,2$                   ; . CONTINUE TILL DONE
002314  005020                         CLR     (R0)+                   ; CLEAR KIPAR6
002316  012710  177600                 MOV     #177600,(R0)            ; I/O PAGE TO KIPAR7
002322  012602                         MOV     (SP)+,R2        ;;POP STACK INTO R2
002324  012601                         MOV     (SP)+,R1        ;;POP STACK INTO R1
002326  012600                         MOV     (SP)+,R0        ;;POP STACK INTO R0
002330  000207                         RTS     PC

                                .SBTTL SUBROUTINE - MEMORY SIZE

                                ;* THIS ROUTINE SIZES MAIN (PMI) MEMORY IN 4K WORDS.
                                ;
                                ;  BGNROUTINE
                                ;
                                ;       REMAP TIMEOUT VECTOR AND PROGRAM AREA
                                ;       ENABLE MEMORY MANAGEMENT UNIT
                                ;       DO UNTIL TIMEOUT OR ALL 2M CHECKED
                                ;       .  CHECK LOCATION 0 IN EACH 4K PAGE THRU KIPAR6
                                ;       ENDDO
                                ;
                                ;  ENDROUTINE
                                ;
                                ;-------------------------------------------------------

002332  013702  000004         MEMSIZ: MOV     ERRVEC,R2               ; SAVE TIME OUT VECTOR
002336  012737  002412  000004         MOV     #3$,ERRVEC              ; POINT NEW TO PROGRAM
002344  012737  000340  000006         MOV     #340,ERRVEC+2           ; AT PRIORITY 7
                                ;
                                ; SIZE MEMORY IN 4K WORDS
                                ;
002352  012737  000200  172354         MOV     #200,KIPAR6             ; FIRST 4K
002360  000403                         BR      2$                      ; GO TRY TO ACCESS
002362  062737  000200  172354 1$:     ADD     #200,KIPAR6             ; . NEXT 4K
002370  005737  140000         2$:     TST     @#140000                ; . ACCESS THRU KIPAR6
002374  022737  170000  172354         CMP     #170000,KIPAR6          ; . LAST PAGE?
```

J2

SUBROUTINE - MEMORY SIZE

```
    752 002402  001367                      BNE     1$                      ; . IF NOT, BRANCH
    753 002404  012701  000040              MOV     #40,R1                  ; IF 2M PRESENT, ONLY 21ST MASKED
    754 002410  000402                      BR      4$                      ; BRANCH
    755 002412  005726              3$:     TST     (SP)+                   ; RESTORE STACK
    756 002414  005726                      TST     (SP)+                   ;
    757 002416  010237  000004      4$:     MOV     R2,ERRVEC               ; RESTORE TIMEOUT VECTOR
    758 002422  000207                      RTS     PC                      ; RETURN
    759
    760                             .SBTTL SUBROUTINE - SIZE UNIBUS MEMORY FROM KMCR
    761
    762                             ;* ON RETURN FROM THIS SUBROUTINE R2 WOULD HAVE PAR VALUE
    763                             ;* OF UNIBUS MEMORY
    764                             ;
    765 002424  013701  177734      UMSIZ:  MOV     KMCR,R1                 ; SAVE KMCR
    766 002430  042701  000040      1$:     BIC     #BIT05,R1               ; LEAVE ONLY # OF PAGES
    767 002434  012702  007600              MOV     #7600,R2                ; START W/O UNIBUS MEMORY
    768 002440  162702  000200      100$:   SUB     #200,R2                 ; . SUBTRACT 4K
    769 002444  077103                      SOB     R1,100$                 ; . FOR ALL PAGES PRESENT
    770 002446  000207                      RTS     PC
    771
    772
    773                             .SBTTL SUBROUTINE - INITIALIZE THE UBE'S
    774
    775                             ;* THIS ROUTINE IS USED TO INITIALIZE THE UBE'S
    776                             ;
    777                             ;  BGNROUTINE
    778                             ;
    779                             ;       SAVE R0,R1
    780                             ;       LET R0 := FIRST UBE ADDRESS
    781                             ;       DO FOR (R0) := BE1DB TO BE1CR1
    782                             ;        . LET (R0) := 0
    783                             ;       ENDDO
    784                             ;       RESTORE R1,R0
    785                             ;       RETURN
    786                             ;
    787                             ;  ENDROUTINE
    788                             ;
    789                             ;---------------------------------------------------------------
    790
    791 002450                      IUBE:
        002450  010046                      MOV     R0,-(SP)        ;;PUSH R0 ON STACK
        002452  010146                      MOV     R1,-(SP)        ;;PUSH R1 ON STACK
    792 002454  013700  002150              MOV     BE1DB,R0        ; POINT R0 TO UBE REGISTERS
    793 002460  012701  000010              MOV     #10,R1          ; SET UP A LOOP COUNTER
    794 002464  012720  000000      5$:     MOV     #0,(R0)+        ; CLEAR OUT A REGISTER
    795 002470  077103                      SOB     R1,5$           ; HAVE WE INITIALIZED THE UBE ?
    796 002472  012601                      MOV     (SP)+,R1        ;;POP STACK INTO R1
        002474  012600                      MOV     (SP)+,R0        ;;POP STACK INTO R0
    797 002476  000207                      RTS     PC
    798
    799
    800                             ;
    801                             ; STARTING POINT OF PROGRAM
    802                             ;
    803
    804 002500                      START:
                                    ;; LCP/ORION ROUTINE TO SAVE EMTULATOR 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,UFDFLG              ;; 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$:     EMT     42                      ;; GET ADDRESS OF XXDP DCA TABLE
002544  005060  000042                  CLR     42(R0)                  ;; CLR XXDP+ "DRSERR"
002550  013737  000030  002566          MOV     30,SAV30                ;; SAVE EMULATOR ADDRESS
002556  013737  000032  002570          MOV     32,SAV32                ;; SAVE EMULATOR PRIORITY LEVEL
002564  000404                          BR      VMKOR                   ;; GET AROUND TAG AREA
002566  000000                  SAV30:  .WORD   0                       ;; PUT EMULATOR INFO HERE
002570  000000                  SAV32:  .WORD   0                       ;; PUT PRIORITY LOCATION        HERE
002572  000000                  UFDFLG: .WORD   0                       ;; USER FRIENDLY MODE FLAG
002574  000000                  UQUIET: .WORD   0                       ;; UFD QUIET MODE FLAG
002576                          VMKOR:
                                ;*******************************************************************************
    805                         .SBTTL  INITIALIZE THE COMMON TAGS
                                ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
002576  012706  001100                  MOV     #$CMTAG,R6              ;;FIRST LOCATION TO BE CLEARED
002602  005026                          CLR     (R6)+                   ;;CLEAR MEMORY LOCATION
002604  022706  001140                  CMP     #SWR,R6 ;;DONE?
002610  001374                          BNE     .-6                     ;;LOOP BACK IF NO
002612  012706  001100                  MOV     #STACK,SP               ;;SETUP THE STACK POINTER
                                ;;INITIALIZE A FEW VECTORS
002616  012737  022712  000020          MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
002624  012737  000340  000022          MOV     #340,@#IOTVEC+2 ;;LEVEL 7
002632  012737  025252  000030          MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
002640  012737  000340  000032          MOV     #340,@#EMTVEC+2 ;;LEVEL 7
002646  012737  026252  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
002654  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
002662  012737  026336  000024          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
002670  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
002676  013737  022536  022530          MOV     $ENDCT,$EOPCT           ;;SETUP END-OF-PROGRAM COUNTER
002704  005037  001164                  CLR     $TIMES                  ;;INITIALIZE NUMBER OF ITERATIONS
002710  005037  001166                  CLR     $ESCAPE                 ;;CLEAR THE ESCAPE ON ERROR ADDRESS
002714  112737  000001  001115          MOVB    #1,$ERMAX               ;;ALLOW ONE ERROR PER TEST
002722  012737  002722  001106          MOV     #..,$LPADR              ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
002730  012737  002730  001110          MOV     #..,$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.
002736  013746  000004                  MOV     @#ERRVEC,-(SP)          ;;SAVE ERROR VECTOR
002742  012737  002776  000004          MOV     #64$,@#ERRVEC           ;;SET UP ERROR VECTOR
002750  012737  177570  001140          MOV     #DSWR,SWR               ;;SETUP FOR A HARDWARE SWICH REGISTER
002756  012737  177570  001142          MOV     #DDISP,DISPLAY          ;;AND A HARDWARE DISPLAY REGISTER
002764  022777  177777  176146          CMP     #-1,@SWR                ;;TRY TO REFERENCE HARDWARE SWR
002772  001012                          BNE     66$                     ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                        ;;AND  THE HARDWARE SWR IS NOT = -1
002774  000403                          BR      65$                     ;;BRANCH IF NO TIMEOUT
002776  012716  003004          64$:    MOV     #65$,(SP)               ;;SET UP FOR TRAP RETURN
003002  000002                          RTI
003004  012737  000176  001140  65$:    MOV     #SWREG,SWR              ;;POINT TO SOFTWARE SWR
003012  012737  000174  001142          MOV     #DISPREG,DISPLAY
003020  012637  000004          66$:    MOV     (SP)+,@#ERRVEC          ;;RESTORE ERROR VECTOR
003024  005037  001206                  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     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
        003046                          67$:
    806 003046  032737  000040  000052          BIT     #BIT5,@#52      ; ARE WE UNDER UFD ?
    807 003054  001056                          BNE     1$              ; YES,THEN SKIP THE DIAGNOSTIC TITLE PRINTOUT
    808 003056  122737  000001  001220          CMPB    #APTENV,$ENV    ; APT?
    809 003064  001452                          BEQ     1$              ; IF YES, SKIP PRINTOUT
    810                             .SBTTL  TYPE PROGRAM NAME
                                    ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
        003066  005227  177777                  INC     #-1             ;;FIRST TIME?
        003072  001047                          BNE     68$             ;;BRANCH IF NO
        003074  022737  022672  000042          CMP     #$ENDAD,@#42    ;;ACT-11?
        003102  001443                          BEQ     68$             ;;BRANCH IF YES
        003104  104401  003152                  TYPE    ,69$            ;;TYPE ASCIZ STRING
                                .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
        003110  005737  000042                  TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
        003114  001012                          BNE     70$             ;;BRANCH IF YES
        003116  123727  001220  000001          CMPB    $ENV,#1         ;;ARE WE RUNNING UNDER APT?
        003124  001406                          BEQ     70$             ;;BRANCH IF YES
        003126  023727  001140  000176          CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
        003134  001005                          BNE     71$             ;;BRANCH IF NO
        003136  104406                          GTSWR                   ;;GET SOFT-SWR SETTINGS
        003140  000403                          BR      71$
        003142  112737  000001  001134  70$:     MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
        003150                          71$:
        003150  000420                          BR      68$             ;;GET OVER THE ASCIZ
                                        ;;69$:  .ASCIZ  <CRLF>* COKTAAO KTJ11-B DIAGNOSTIC *<CRLF>
        003212                          68$:
    811 003212  012737  002234  000004  1$:      MOV     #TIMOUT,@#4     ; SETUP TIMEOUT VECTOR FOR UNEXPECTED TIMEOUT
    812 003220  012737  000340  000006          MOV     #340,@#6        ;
    813 003226  005037  001716                  CLR     TEST            ; TEST # FOR ERROR REPORTING
    814                             ; SIZE MEMORY IN 4K PAGES
    815 003232  004737  002244                  JSR     PC,MAPPR        ; REMAP PROGRAM AREA
    816 003236  052737  000060  172516          BIS     #BIT05!BIT04,MMR3       ; ENABLE RELOCATION AND 22BITS
    817 003244  005237  177572                  INC     MMR0            ; ENABLE MMU
    818 003250  004737  002332                  JSR     PC,MEMSIZ       ; GET HIGHEST 128K OF MEMORY
    819 003254  005037  177572                  CLR     MMR0            ; DISABLE MMU
    820 003260  005037  172516                  CLR     MMR3            ;
    821 003264  013737  172354  001720          MOV     KIPAR6,PMIS     ; STORE HIGHEST PAGE
    822 003272                      RESTART:
    823 003272  012706  001100                  MOV     #1100,SP        ; SET UP THE STACK POINTER
    824
    825
    826
```

```
                        .SBTTL TEST - UNIBUS MAP REGISTER TESTS

                        ;* THIS TEST WILL TRY TO ACCESS ALL THE MAP
                        ;* REGISTERS WITH BOTH MAPPING ENABLED AND
                        ;* DISABLED
                        ;
                        ;  BGNTST
                        ;
                        ;       LET 4:= ADDRESS OF ERROR ROUTINE
                        ;       LET 6:= PRIORITY OF 7
                        ;       DO FOR BOTH UNIBUS MAP ENABLED AND DISABLED
                        ;       .  DO FOR ADDRESS := 170200 TO 170376
                        ;       .  .  READ ADDRESS
                        ;       .  .  IF ADDRESS CAN'T BE READ THEN
                        ;       .  .  .  TRAP THRU VECTOR 4
                        ;       .  .  ENDIF
                        ;       .  ENDDO
                        ;       ENDDO
                        ;       EXIT TEST
                        ;
                        ;  ERROR:
                        ;       REPORT WHICH MAP REGISTER TIMED OUT
                        ;
                        ;  ENDTST
                        ;
                        ;-----------------------------------------

                        ;;*********************************************************************
                        ;*TEST 1         UNIBUS MAP REGISTER RESPONSE TEST
                        ;;*********************************************************************
003276  000004          TST1:   SCOPE

                        ;
                        ; SET UP MEMORY TIMEOUT VECTOR TO ERROR ROUTINE
                        ;
003300  013705  000004          MOV     @#4,R5                  ; SAVE THE TIMEOUT VECTOR
003304  012737  003376  000004  MOV     #20$,@#4                ; SET UP TIMEOUT VECTOR TO ERROR ROUTINE
003312  012737  000340  000006  MOV     #340,@#6                ; SET UP TIMEOUT PRIORITY
                        ;
                        ; DISABLE UNIBUS MAPPING
                        ;
003320  042737  000040  172516  BIC     #BIT5,@#SR3             ; DISABLE UNIBUS MAPPING
                        ;
                        ; TRY TO ACCESS ALL THE UNIBUS MAP REGISTERS
                        ;
003326  005001          CLR     R1                              ; INITIALIZE LOOP COUNT
003330  012700  170200  5$:     MOV     #MAPL00,R0              ; . R0 POINTS TO MAP REGISTER #0
003334  005720          10$:    TST     (R0)+                   ; . . TRY TO READ MAP REGISTER
003336  020027  170400  12$:    CMP     R0,#170400              ; . . HAVE WE READ ALL THE MAP REGISTERS
003342  001374          BNE     10$                             ; . . NO THEN GO READ THE NEXT ONE
003344  005701          TST     R1                              ; . HAVE WE DONE THIS TWICE ?
003346  001406          BEQ     15$                             ; . GO TURN ON UNIBUS MAP AND TRY AGAIN
003350  042737  000040  172516  BIC     #BIT5,SR3               ; . TURN OFF UNIBUS MAPPING
003356  010537  000004          MOV     R5,@#4                  ; RESTORE THE TIMEOUT VECTOR
003362  000417          BR      TST2            ;; . GO TO NEXT TEST
003364  005201  15$:    INC     R1                              ; . INCREMENT LOOP COUNT
003366  052737  000040  172516  BIS     #BIT5,SR3               ; . TURN ON UNIBUS MAP
```

N2

T1 UNIBUS MAP REGISTER RESPONSE TEST

```
    882 003374  000755                        BR      5$                      ; . GO TRY TO ACCESS THE REGISTERS
    883                                     ;
    884                                     ; MAP REGISTER TIMEOUT ROUTINE
    885                                     ;
    886 003376  005726                  20$:    TST     (SP)+                   ; CLEAN UP THE STACK
    887 003400  005726                          TST     (SP)+                   ;
    888 003402  162700  000002                  SUB     #2,R0                   ; GET ADDRESS THAT TIMED OUT
    889 003406  010037  001122                  MOV     R0,$BDADR               ; PUT IT IN $BDADR
    890 003412  062700  000002                  ADD     #2,R0                   ; GET NEXT ADDRESS
    891 003416  104001                          ERROR   +1                      ; REPORT THAT ACCESS HAS TIMED OUT
    892 003420  000746                          BR      12$                     ;
    893
```

```
                                .SBTTL  TEST - UNIBUS MAP REGISTER BIT PATTERN
                                ;
                                ;* THIS TEST IS DESIGNED TO DETECT ANY STUCK
                                ;* AT BITS OR SHORTED BITS IN THE UNIBUS MAP
                                ;* REGISTERS
                                ;
                                ;  BGNTST
                                ;
                                ;       CLEAR ALL MAP REGISTERS
                                ;       DO FOR ADDRESS := 170200 TO 170376
                                ;*
                                ;*  CHECK LOW MAPPING REGISTERS
                                ;*
                                ;         .  READ ADDRESS
                                ;         .  IF ADDRESS NE 0 THEN
                                ;         .  .  ERROR MAP REGISTER DID NOT CLEAR
                                ;         .  ENDIF
                                ;         .  DO FOR PATTERNS := 377,7417,1463,52525,125252
                                ;         .  .  WRITE PATTERN INTO ADDRESS
                                ;         .  .  READ PATTERN
                                ;         .  .  IF CONTENTS OF ADDRESS NE PATTERN THEN
                                ;         .  .  .  ERROR IN BITS OF ADDRESS
                                ;         .  .  ENDIF
                                ;         .  ENDDO
                                ;         .  LET ADDRESS := ADDRESS + 2
                                ;*
                                ;*  CHECK HIGH MAPPING REGISTERS
                                ;*
                                ;         .  READ ADDRESS
                                ;         .  IF ADDRESS NE 0 THEN
                                ;         .  .  ERROR MAP REGISTER DID NOT CLEAR
                                ;         .  ENDIF
                                ;         .  DO FOR PATTERNS := 17,14,25,52
                                ;         .  .  WRITE PATTERN INTO ADDRESS
                                ;         .  .  READ PATTERN
                                ;         .  .  IF CONTENTS OF ADDRESS NE PATTERN THEN
                                ;         .  .  .  ERROR IN BITS OF ADDRESS
                                ;         .  .  ENDIF
                                ;         .  ENDDO
                                ;       ENDDO
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------
                                ;
                                ;;*****************************************************************
                                ;*TEST 2        UNIBUS MAP REGISTER BIT PATTERN TEST
                                ;;*****************************************************************
003422  000004                  TST2:   SCOPE
                                ;
                                ; CLEAR ALL MAP REGISTERS
                                ;
003424  012700  170200                  MOV     #MAPL00,R0              ; GET ADDRESS TO 1ST MAP REGISTER
003430  005020                  5$:     CLR     (R0)+                   ; . CLEAR THE MAP REGISTER
003432  020027  170400                  CMP     R0,#170400              ; . HAVE WE CLEARED ALL THE MAP REGISTERS ?
003436  001374                          BNE     5$                      ; . NO THEN GO CLEAR THE NEXT ONE
```

T2 UNIBUS MAP REGISTER BIT PATTERN TEST

```
                                        ;
                                        ; DO A READ,WRITE,READ ON THE REGISTERS TO
                                        ; VERIFY ALL THE BITS OF THE  LOW MAP REGISTERS
                                        ;
003440  012700  170200          MOV     @MAPLOO,R0              ; GET ADDRESS TO 1ST MAP REGISTER
003444  005710          10$:    TST     (R0)                    ; . IS THE MAP REGISTER CLEAR ?
003446  001410                  BEQ     15$                     ; . YES,THEN GO CHECK THE BITS
003450  011037  001126          MOV     (R0),$BDDAT             ; . NO,THEN GET THE BAD DATA
003454  012737  000000  001124  MOV     @0,$GDDAT               ; .  GET THE GOOD DATA
003462  010037  001122          MOV     R0,$BDADR               ; .  GET THE ADDRESS THAT DID NOT CLEAR
003466  104002                  ERROR   +2                      ; .  ERROR - REGISTER DID NOT CLEAR
003470  012701  001774  15$:    MOV     @PTRN16,R1              ; . GET POINTER TO PATTERN TABLE
003474  011110          20$:    MOV     (R1),(R0)               ; . . MOVE PATTERN TO MAP REGISTER
003476  012102                  MOV     (R1)+,R2                ; . . MOVE COPY OF PATTERN TO R2
003500  042702  000001          BIC     @BIT0,R2                ; . . CLEAR BIT0 OF THE PATTERN
003504  020210                  CMP     R2,(R0)                 ; . . DID PATTERN GET WRITTEN CORRECTLY ?
003506  001410                  BEQ     25$                     ; . . YES,THEN GO SEE IF ALL PATTERNS HAVE BEEN WRIT
TEN
003510  011037  001126          MOV     (R0),$BDDAT             ; . . NO,GET THE BAD DATA
003514  005741                  TST     -(R1)                   ; . .  POINT TO THE GOOD DATA
003516  012137  001124          MOV     (R1)+,$GDDAT            ; . .  GET THE GOOD DATA,AND POINT R1 TO THE NEXT DA
TA PATTERN
003522  010037  001122          MOV     R0,$BDADR               ; . .  GET THE ADDRESS OF DATA FAULT
003526  104003                  ERROR   +3                      ; . .  ERROR - REGISTER COULD NOT HOLD PATTERN
003530  005711          25$:    TST     (R1)                    ; . . ARE WE THROUGH THE TABLE ?
003532  001360                  BNE     20$                     ; . . NO THEN GO WRITE NEXT PATTERN
                                        ;
                                        ; DO A READ,WRITE,READ ON THE REGISTERS TO
                                        ; VERIFY ALL THE BITS OF THE HIGH MAP REGISTERS
                                        ;
003534  062700  000002          ADD     @2,R0                   ; . POINT TO THE HIGH MAPPING REGISTER
003540  005710          30$:    TST     (R0)                    ; . IS THE MAP REGISTER CLEAR ?
003542  001410                  BEQ     35$                     ; . YES,THEN GO CHECK THE BITS
003544  011037  001126          MOV     (R0),$BDDAT             ; . NO,THEN GET THE BAD DATA
003550  012737  000000  001124  MOV     @0,$GDDAT               ; .  GET THE GOOD DATA
003556  010037  001122          MOV     R0,$BDADR               ; .  GET THE ADDRESS THAT DID NOT CLEAR
003562  104002                  ERROR   +2                      ; .  ERROR - REGISTER DID NOT CLEAR
003564  012701  002010  35$:    MOV     @PTRN6,R1               ; . GET POINTER TO PATTERN TABLE
003570  011110          40$:    MOV     (R1),(R0)               ; . . MOVE PATTERN TO MAP REGISTER
003572  012102                  MOV     (R1)+,R2                ; . . MOVE COPY OF THE PATTERN TO R2
003574  042702  177700          BIC     @177700,R2              ; . . MASK OUT THE UPPER 10 BITS
003600  020210                  CMP     R2,(R0)                 ; . . DID PATTERN GET WRITTEN CORRECTLY ?
003602  001410                  BEQ     45$                     ; . . YES,THEN GO SEE IF ALL PATTERNS HAVE BEEN WRIT
TEN
003604  011037  001126          MOV     (R0),$BDDAT             ; . . NO,GET THE BAD DATA
003610  005741                  TST     -(R1)                   ; . .  POINT TO THE GOOD DATA
003612  012137  001124          MOV     (R1)+,$GDDAT            ; . .  GET THE GOOD DATA,AND POINT R1 BACK TO THE NE
XT PATTERN
003616  010037  001122          MOV     R0,$BDADR               ; . .  GET THE ADDRESS OF DATA FAULT
003622  104003                  ERROR   +3                      ; . .  ERROR - REGISTER COULD NOT HOLD PATTERN
003624  005711          45$:    TST     (R1)                    ; . . ARE WE THROUGH THE TABLE ?
003626  001360                  BNE     40$                     ; . . NO THEN GO WRITE NEXT PATTERN
                                        ;
                                        ; SEE IF WE HAVE CHECKED ALL THE MAP REGISTERS
                                        ;
003630  062700  000002          ADD     @2,R0                   ; . POINT TO NEXT REGISTER
003634  020027  170400          CMP     R0,@170400              ; . ARE WE THROUGH ALL THE MAP REGISTERS ?
003640  001301                  BNE     10$                     ; . GO CHECK THE NEXT REGISTER
```

```
                                .SBTTL  TEST - UNIBUS MAP REGISTER ADDRESS UNIQUENESS
                                ;
                                ;* THIS TEST IS DESIGNED TO DETECT ANY DUAL ADDRESSING
                                ;* IN THE UNIBUS MAP ADDRESSING LOGIC
                                ;
                                ;  BGNTST
                                ;
                                ;       DO FOR ADDRESS := 170200 TO 170376
                                ;       .  WRITE #ADDRESS TO ADDRESS
                                ;       ENDDO
                                ;       DO FOR ADDRESS := 170200 TO 170376
                                ;       .  READ ADDRESS
                                ;       .  IF CONTENTS OF ADDRESS NE ADDRESS THEN
                                ;       .  .  REPORT ERROR IN ADDRESSING
                                ;       .  ENDIF
                                ;       ENDDO
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------

                                ;;*********************************************************************
                                ;*TEST 3        UNIBUS MAP REGISTER UNIQUENESS TEST
                                ;;*********************************************************************
003642  000004                  TST3:   SCOPE

                                ;
                                ; WRITE ADDRESS INTO ALL MAP REGISTERS
                                ;
003644  012700  170200                  MOV     @MAPLOO,R0              ; GET ADDRESS OF 1ST MAP REGISTER
003650  012701  170200                  MOV     @MAPLOO,R1              ; GET ADDRESS OF 1ST MAP REGISTER
003654  010021                  5$:     MOV     R0,(R1)+                ; . WRITE ADDRESS TO ITSELF
003656  062700  000002                  ADD     #2,R0                   ; . GET THE NEXT ADDRESS
003662  022700  170400                  CMP     #170400,R0              ; . HAVE WE WRITTEN TO ALL MAP REGISTERS ?
003666  001372                          BNE     5$                      ; . NO,THEN GO WRITE TO NEXT ADDRESS
                                ;
                                ; MAKE SURE EACH LOW MAP REGISTER CONTAINS IT'S ADDRESS
                                ;
003670  012700  170200                  MOV     @MAPLOO,R0              ; GET ADDRESS TO 1ST MAP REGISTER
003674  012701  170200                  MOV     @MAPLOO,R1              ; GET ADDRESS TO 1ST MAP REGISTER
003700  020021                  10$:    CMP     R0,(R1)+                ; . WAS THE ADDRESS WRITTEN TO THE MAP REGISTER ?
003702  001406                          BEQ     15$                     ; . YES,THEN GO CHECK THE HIGH MAP REGISTER
003704  005740                          TST     -(R0)                   ; . NO GET THE ADDRESS THAT WAS WRITTEN
003706  010037  001120                  MOV     R0,$GDADR               ; .  GET THE GOOD ADDRESS
003712  012037  001122                  MOV     (R0)+,$BDADR            ; .  GET THE BAD ADDRESS,AND POINT BACK TO NEXT ADDR
ESS
003716  104004                          ERROR   +4                      ; .  ERROR IN ADDRESSING OF MAP REGISTERS
                                ;
                                ; MAKE SURE EACH HIGH MAP REGISTER CONTAINS IT'S ADDRESS
                                ;
003720  062700  000002          15$:    ADD     #2,R0                   ; . POINT R0 TO HIGH MAP REGISTER
003724  010002                          MOV     R0,R2                   ; . PUT A COPY OF IT INTO R2
003726  042702  177700                  BIC     #177700,R2              ; . MASK OUT THE UPPER 10 BITS
003732  020221                          CMP     R2,(R1)+                ; . WAS THE ADDRESS WRITTEN TO THE HIGH MAP REGISTER
 ?
003734  001406                          BEQ     20$                     ; . YES,THEN GO CHECK IF WE ARE THROUGH ALL THE MAP
REGISTERS
003736  005740                          TST     -(R0)                   ; . NO GET THE ADDRESS THAT WAS WRITTEN
003740  010037  001120                  MOV     R0,$GDADR               ; .  GET THE GOOD ADDRESS
```

```
T3      UNIBUS MAP REGISTER UNIQUENESS TEST

   1060 003744  012037  001122                  MOV     (R0)+,$BDADR            ; .  GET THE BAD ADDRESS,AND POINT BACK TO NEXT ADDR
ESS
   1061 003750  104004                          ERROR   +4                      ; .  ERROR IN ADDRESSING OF MAP REGISTERS
   1062                                 ;
   1063                                 ; CHECK TO SEE IF WE HAVE CHECKED ALL MAP REGISTERS
   1064                                 ;
   1065 003752  062700  000002          20$:    ADD     #2,R0                   ; . GET THE NEXT REGISTER
   1066 003756  020027  170400                  CMP     R0,#170400              ; . HAVE WE CHECKED ALL THE MAPPING REGISTERS
   1067 003762  001346                          BNE     10$                     ; . NO,THEN GO CHECK THE NEXT REGISTER
   1068
   1069
   1070
```

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

                                ;;*****************************************************************
                                ;*TEST 4         HIGH AND LOW MAP REGISTER UNIQUENESS TEST
                                ;;*****************************************************************
     003764  000004             TST4:   SCOPE

                                ;
                                ; WRITE #77777 TO ALL ADDRESSES
                                ;
     003766  012700  170200             MOV     #MAPLOO,R0              ; GET ADDRESS OF FIRST MAP REGISTER
     003772  012720  077777     5$:     MOV     #77777,(R0)+            ; . WRITE 77777 TO MAP REGISTER
     003776  020027  170400             CMP     R0,#170400              ; . HAVE WE WRITTEN TO ALL REGISTERS ?
     004002  001373                     BNE     5$                      ; . NO,THEN GO WRITE TO THE NEXT REGISTER ?
                                ;
                                ; MAKE SURE THAT HIGH MAP REGISTER BITS 6-15 ARE CLEARED
                                ;
     004004  012700  170200             MOV     #MAPLOO,R0              ; GET ADDRESS OF FIRST MAP REGISTER
     004010  012002             10$:    MOV     (R0)+,R2                ; . GET CONTENTS OF LOW MAP REGISTER
     004012  012003                     MOV     (R0)+,R3                ; . GET CONTENTS OF HIGH MAP REGISTER
     004014  160302                     SUB     R3,R2                   ; . DID THE REGISTERS GET ADDRESSED CORRECTLY ?
     004016  003007                     BGT     15$                     ; . YES,THEN GO SEE IF WE HAVE CHECKED ALL THE MAP R
EGISTERS
     004020  005740                     TST     -(R0)                   ; . NO,
     004022  005740                     TST     -(R0)                   ; .  POINT TO THE LOW MAP REGISTER
     004024  012037  001124             MOV     (R0)+,$GDDAT            ; .  GET CONTENTS OF LOW MAP REGISTER
     004030  012037  001126             MOV     (R0)+,$BDDAT            ; .  GET CONTENTS OF HIGH MAP REGISTER
     004034  104005                     ERROR   +5                      ; .  ERROR - MAP REGISTERS WERE NOT ADDRESSED CORREC
TLY
                                ;
                                ; SEE IF WE HAVE CHECKED ALL THE MAP REGISTER PAIRS
                                ;
     004036  020027  170400     15$:    CMP     R0,#170400              ; . HAVE WE CHECKED ALL THE REGISTERS ?
     004042  001362                     BNE     10$                     ; . NO,THEN GO CHECK THE NEXT PAIR !!!
```

```
                                .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 INSRUCTION
                                ;       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
                                ;;*****************************************************************
004044  000004                  TST5:   SCOPE
                                ;
                                ; SET UP MEMORY TIMEOUT VECTOR
                                ;
004046  012737  004220  000004          MOV     #20$,@#4                ; SET UP VECTOR PC TO ERROR ROUTINE
004054  012737  000340  000006          MOV     #340,@#6                ; SET UP VECTOR PSW TO PRIORITY 7
                                ;
                                ; READ ADDRESS OF DCSR TO SEE IF IT TIMES OUT
```

H3

T5 DCSR REGISTER TEST

```
1179                                    ;
1180 004062  005737  177730             TST     @#DCSR                  ; WILL THE DCSR TIMEOUT ?
1181 004066  012737  002234  000004     MOV     #TIMOUT,@#4             ; RESTORE TIMEOUT VECTOR
1182                                    ;
1183                                    ; TRY TO WRITE ONES TO BITS 3-6 AND 9-14 ( BIT 03 SHOULD STAY WRITTEN)
1184                                    ;
1185 004074  052737  077170  177730     BIS     #77170,@#DCSR           ; TRY TO WRITE TO READ ONLY BITS
1186 004102  032737  077160  177730     BIT     #77160,@#DCSR           ; DID THEY GET WRITTEN TO ?
1187 004110  001412                     BEQ     5$                      ; NO,THEN GO CHECK THE EFFECT OF A RESET ON THE DCSR
1188 004112  013701  177730             MOV     @#DCSR,R1               ; GET COPY OF DCSR INTO R1
1189 004116  042701  100607             BIC     #100607,R1              ; MASK OUT UNTESTED BITS
1190 004122  010137  001126             MOV     R1,$BDDAT               ; PUT THEM INTO BAD DATA
1191 004126  012737  000010  001124     MOV     #10,$GDDAT              ; GOOD DATA
1192 004134  104006                     ERROR   +6                      ; ERROR - IN BITS 3-6,9-14 OF DCSR
1193                                    ;
1194                                    ; CACHE THE RESET INSTRUCTION
1195                                    ;
1196 004136  012737  000200  001124  5$: MOV     #BIT7,$GDDAT            ; ONLY 7 SHOULD BE SET AFTER RESET
1197 004144  053737  177730  001124     BIS     DCSR,$GDDAT             ; SET BOOT BITS
1198 004152  005737  004164             TST     10$                     ; CACHE THE RESET
1199                                    ;
1200                                    ; SET UP DCSR,DO A RESET,MAKE SURE PROPER RESPONSE OCCURS
1201                                    ;
1202 004156  052737  000006  177730     BIS     #6,@#DCSR               ; SET BIT 7 AND CLEAR BIT 1,2
1203 004164  000005                 10$: RESET                           ; DO A BUS INIT
1204 004166  032737  000200  177730     BIT     #BIT7,@#DCSR            ; DID BIT 7 GET SET ?
1205 004174  001001                     BNE     15$                     ; YES,THEN GO CHECK IF BITS 1,2 GOT CLEAR
1206 004176  104007                     ERROR   +7                      ; NO,THEN ERROR - BIT 7 DIDN'T GET SET
1207 004200  032737  000006  177730 15$: BIT     #6,@#DCSR               ; DID BIT 1,2 GET CLEARED ?
1208 004206  001410                     BEQ     TST6            ;; YES,THEN GO TO NEXT TEST
1209 004210  013737  177730  001126     MOV     @#DCSR,$BDDAT           ; NO,THEN GET CONTENTS OF THE DCSR
1210 004216  104007                     ERROR   +7                      ;  ERROR - BIT 1,2 DID NOT GET CLEAR
1211                                    ;
1212                                    ; ERROR ROUTINE
1213                                    ;
1214 004220  012737  177730  001122 20$: MOV     #DCSR,$BDADR            ; GET ADDRESS OF THE DCSR
1215 004226  104010                     ERROR   +10                     ; ERROR - DCSR REGISTER ACCESS HAS TIMED OUT
1216
1217
```

```
                                .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> DISABELD
                                ;       IF NOT THEN
                                ;       .  ERROR
                                ;       ENDIF
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------------
                                ;;*************************************************************
                                ;*TEST 6        KMCR MEMORY CONFIGURATION BITS TEST
                                ;;*************************************************************
004230  000004                  TST6:   SCOPE
                                ;
                                ; SETUP TIMEOUT VECTOR,TO ERROR ROUTINE
                                ; VERIFY THAT AT LEAST SOME PMI MEMORY PRESENT
                                ;
004232  012737  004376  000004          MOV     #10$,@#4                ; SET VECTOR PC TO ERROR ROUTINE
004240  012737  000340  000006          MOV     #340,@#6                ; SET VECTOR PSW TO ERROR ROUTINE
004246  013737  177734  001160          MOV     KMCR,$TMP0              ; STORE KMCR
004254  012737  002234  000004          MOV     #TIMOUT,@#4             ; RESTORE TIMEOUT VECTOR
004262  042737  177700  001160          BIC     #177700,$TMP0           ; CLEAR ALL BUT U.MEMORY
004270  022737  000077  001160          CMP     #77,$TMP0               ; NO PMI MEMORY?
004276  001003                          BNE     2$                      ; IF SOME, GO TO THE NEXT TEST
004300  005037  001720                  CLR     PMIS                    ; OTHERWISE FLAG NO PMI MEMORY
004304  000440                          BR      TST7            ;; AND EXIT TEST
                                ;
                                ; CHECK THAT NOT IN DIAGNOSTIC MODE WRITE ACCESS TO KMCR<5-0> IS DISABLED
                                ;
004306  013701  177734          2$:     MOV     KMCR,R1                 ; STORE PATTERN FROM KMCR TO R1
004312  005101                          COM     R1                      ; COMPLEMENT THE PATTERN
004314  042701  177700                  BIC     #177700,R1              ; CLEAR BITS <15-6>
004320  042737  000400  177730          BIC     #BIT08,DCSR             ; MAKE SURE THAT NOT IN DIAGN. MODE
004326  042701  000010                  BIC     #BIT03,R1               ; ENABLE 32K, IN CASE OF ERROR
004332  050137  177734                  BIS     R1,KMCR                 ; TRY TO WRITE TO KMCR<5 0>
004336  020137  177734                  CMP     R1,KMCR                 ; WRITTEN OK?
004342  001014                          BNE     1$                      ; IF DIDN'T WRITE, BRANCH
004344  013737  177734  001124          MOV     KMCR,$GDDAT             ; NO,THEN GET THE GOOD DATA
004352  010137  001126                  MOV     R1,$BDDAT               ;  GET THE BAD DATA
004356  012737  177734  001122          MOV     #KMCR,$BDADR            ;  GET THE ADDRESS OF THE KMCR
004364  013737  001160  177734          MOV     $TMP0,KMCR              ; RESTORE KMCR
004372  104011                          ERROR   +11                     ; NO,THEN ERROR - BITS DID NOT GET SET CORRECTLY IN
THE KMCR
004374                          1$:
004374  000404                          BR      TST7            ;;  GO TO THE NEXT TEST
                                ;
```

T6 KMCR MEMORY CONFIGURATION BITS TEST

```
                                        ; TIMEOUT ROUTINE
                                        ;
004376  012737  177734  001122  10$:    MOV     #KMCR,$BDADR        ; GET THE ADDRESS OF THE KMCR
004404  104012                          ERROR   +12                 ; ERROR - MEMORY TIMEOUT HAS OCCURED
```

TEST - UNIBUS TIMEOUT TEST

```
                                .SBTTL TEST - UNIBUS TIMEOUT TEST

                                ;* THIS TEST CHECKS OUT THE UNIBUS TIMEOUT LOGIC
                                ;* BEFORE IT IS USED IN ANY TESTS
                                ;
                                ;  BGNTST
                                ;
                                ;       LET 4 := THE ADDRESS 10$
                                ;       LET 6 := #340
                                ;       MAKE SURE THAT UBA IS IN DIAGNOSTIC MODE,DCSR<8> IS SET
                                ;       READ UNIBUS LOCATION
                                ;       IF NO TIMEOUT THEN
                                ;          ERROR - UBA DIDN'T TIMEOUT ON UNIBUS ACCESS IN DIAGNOSTIC MODE
                                ;       ENDIF
                                ;  10$: CLEAN UP THE STACK
                                ;
                                ;  ENDTST
                                ;
                                ;-----------------------------------------

                                ;;*********************************************************************
                                ;*TEST 7          UNIBUS TIMEOUT TEST
                                ;;*********************************************************************
004406  000004                  TST7:   SCOPE

                                ;
                                ; SET UP MEMORY TIMEOUT VECTOR AND PUT UBA INTO DIAGNOSTIC MODE
                                ;
004410  005737  001720                  TST     PMIS                    ; ANY PMI MEMORY?
004414  001422                          BEQ     TST10           ;; IF NONE, EXIT TEST
004416  012737  004450  000004          MOV     #10$,@#4                ; SET UP TIMEOUT VECTOR TO END OF TEST
004424  012737  000340  000006          MOV     #340,@#6                ;
004432  052737  000400  177730          BIS     #BIT8,@#DCSR            ; PUT UBA INTO DIAGNOSTIC MODE

                                ;
                                ; READ A UNIBUS LOCATION
                                ;
004440  005737  170002                  TST     @#170002                ; TRY TO READ UBE REGISTER ?
004444  104036                          ERROR   +36                     ; ERROR - SHOULD TIMEOUT HERE
004446  000402                          BR      11$                     ; GO TO THE NEXT TEST

                                ;
                                ; TIMEOUT ROUTINE
                                ;
004450  005726                  10$:    TST     (SP)+                   ; CLEAN UP THE STACK
004452  005726                          TST     (SP)+                   ;
004454  012737  002234  000004  11$:    MOV     #TIMOUT,@#4             ; RESTORE TIMEOUT VECTOR
```

```
                        .SBTTL TEST - DATA OUT WITHOUT RELOCATION

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

                        ;;*******************************************************************
                        ;*TEST 10       DATA OUT WITHOUT RELOCATION
                        ;;*******************************************************************
     004462  000004     TST10:  SCOPE

     004464  005737  001720         TST     PMIS                    ; ANY PMI MEMORY?
     004470  001450                 BEQ     TST11           ;; IF NONE, EXIT TEST
     004472  052737  000400  177730 BIS     #BIT08,DCSR             ; MAKE SURE DIAGNOSTIC MODE IS ON
     004500  042737  000006  177730 BIC     #BIT02!BIT01,DCSR       ; MAKE SURE NPR REGISTER IS SELECTED
     004506  005037  001160         CLR     $TMP0                   ; CLEAR TEST LOCATION
     004512  012702  001774         MOV     #PTRN16,R2              ; POINT TO BINARY DIVIDE PATTERN
     004516  012701  001160         MOV     #$TMP0,R1               ; STORE TEST LOCATION
     004522  032737  000200  177730 BIT     #BIT07,DCSR             ; READY BIT SET?
     004530  001001                 BNE     1$                      ; IF SO, BRANCH
     004532  104017                 ERROR   +17                     ; DCSR<7> NOT SET
     004534  012237  001124 1$:     MOV     (R2)+,$GDDAT            ; . STORE PATTERN
     004540  004737  002210         JSR     PC,DDOUT                ; . PERFORM DIAGNOSTIC DATA OUT
     004544  023737  177732  001124 CMP     DDR,$GDDAT              ; . DDR HAS PROPER PATTERN
     004552  001404                 BEQ     2$                      ; . IF YES, BRANCH
     004554  013737  177732  001126 MOV     DDR,$BDDAT              ; . STORE DDR FOR ERROR REPORTS
     004562  104013                 ERROR   +13                     ; . ERROR IN DATA PATH
     004564  023737  001160  001124 2$:     CMP     $TMP0,$GDDAT    ; . DATA OUT OK?
     004572  001404                 BEQ     3$                      ; . BRANCH, IF YES
     004574  013737  001160  001126 MOV     $TMP0,$BDDAT            ; . STORE MEMORY FOR ERROR REPORTS
     004602  104014                 ERROR   +14                     ; . ERROR IN DATA OUT
     004604  022712  000000 3$:     CMP     #0,(R2)                 ; . LAST PATTERN?
     004610  001351                 BNE     1$                      ; . IF NOT, BRANCH
```

TEST - DATA IN WITHOUT RELOCATION

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

                                ;;*********************************************************************
                                ;*TEST 11       DATA IN WITHOUT RELOCATION
                                ;;*********************************************************************
004612  000004                  TST11:  SCOPE

                                        TST     PMIS                    ; ANY PMI MEMORY?
004614  005737  001720                  BEQ     TST12           ;; IF NONE, EXIT TEST
004620  001427                  BIS     #BIT08,DCSR             ; MAKE SURE DIAGNOSTIC MODE IS ON
004622  052737  000400  177730  BIC     #BIT02!BIT01,DCSR       ; MAKE SURE NPR REGISTER IS SELECTED
004630  042737  000006  177730  MOV     #52525,$GDDAT           ; USE ALTERNATING 1'S AND 0'S
004636  012737  052525  001124  BIS     #BIT00,DCSR             ; SET GO BIT
004644  052737  000001  177730  MOV     $GDDAT,$TMP0            ; STORE PATTERN IN TEST LOCATON
004652  013737  001124  001160  ;
                                ; NOW COMPARE THE RESULTS
                                ;
                                        CMP     DDR,$GDDAT              ; DDR GOT DATA OK?
004660  023737  177732  001124  BEQ     TST12           ;; IF YES, EXIT
004666  001404                  MOV     DDR,$BDDAT              ; STORE DDR FOR ERROR REPORTS
004670  013737  177732  001126  ERROR   +15                    ; ERROR IN DATA IN
004676  104015
```

```
                                .SBTTL TEST - CONTENT OF DDR

                                ;* THIS TEST VERIFIES THAT IF DCSR<2-1> ARE NOT ZERO, DDR IS ALWAYS
                                ;* READ AS 0, MEANING THAT NONE OF THE UNIBUS LINES ARE STUCK.
                                ;
                                ;  BGNTST
                                ;
                                ;       DO FOR DCSR<2-1> FROM <0,1> TO <1,1>
                                ;       .  DO DATI
                                ;       .  IF DDR NE 0 THEN
                                ;       .  .  ERROR IN UNIBUS LINES
                                ;       .  ENDIF
                                ;       ENDDO
                                ;
                                ;  ENDTST
                                ;
                                ;---------------------------------------------------------------

                                ;;*************************************************************
                                ;*TEST 12       CONTENT OF DDR
                                ;;*************************************************************
 004700  000004                 TST12:  SCOPE
1433 004702  005737  001720             TST     PMIS                    ; ANY PMI MEMORY?
1434 004706  001442             BEQ     TST13           ;; IF NONE, EXIT TEST
1435 004710  052737  000400  177730     BIS     #BIT08,DCSR             ; MAKE SURE IN DIAG. MODE
1436 004716  052737  000002  177730     BIS     #BIT01,DCSR             ; START WITH 1 IN DCSR
1437 004724  012703  000003             MOV     #3,R3                   ; DO 3 TIMES
1438 004730  005037  001124             CLR     $GDDAT                  ; RECIEVED DATA 0
1439 004734  005001                 1$: CLR     R1                      ; . ADDRESS 0
1440 004736  004737  002222             JSR     PC,DDIN                 ; . DO DIAGNOSTIC DATI
1441 004742  022703  000001             CMP     #1,R3                   ; . CONTROL LINES SELECTED?
1442 004746  001403                     BEQ     2$                      ; . IF YES, CHECK IT
1443 004750  005737  177732             TST     DDR                     ; . ALL ZEROES?
1444 004754  001413                     BEQ     3$                      ; . IF YES, BRANCH
1445 004756  012737  177400  001124 2$: MOV     #177400,$GDDAT          ; . EXPECTED PATTERN
1446 004764  032737  000373  177732     BIT     #373,DDR                ; . SELECT ONLY USED
1447 004772  001404                     BEQ     3$                      ; . IF ALL ZEROES, BRANCH
1448 004774  013737  177732  001126     MOV     DDR,$BDDAT              ; . STORE RECIEVED DATA
1449 005002  104016                     ERROR   +16                     ; . ERROR IN UNIBUS LINES
1450 005004  062737  000002  177730 3$: ADD     #BIT01,DCSR             ; . GET NEXT SET OF UNIBUS LINES
1451 005012  077330                     SOB     R3,1$                   ; . DO FOR ALL COMBINATIONS
1452
```

```
                        .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 NXM 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 DIGNOSTIC MODE IS ON DCSR<8>=1
                        ;       SELECT DIAGNOSTIC NPR THRU DDR BY DCSR<2-1>=<0,0>
                        ;       DO FOR 32. MAPPING REGISTERS
                        ;       .  LET LO REGISTER = #160000
                        ;       .  LET HI REGISTER = #77
                        ;       ENDDO (ALL REGISTERS POINT TO NXM 17760000)
                        ;       POINT TIMEOUT_ROUTINE TO VECTOR 4
                        ;       CALL MAP_PROGRAM_AREA
                        ;       LET MMR0<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 @#0 (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 @#0 NE PATTERN THEN
                        ;       .  .  ERROR PERFORMING DATA OUT
                        ;       .  ENDIF
                        ;       .  LET (R0)+ = #160000
                        ;       .  LET (R0)+ = #77 TO MAP JUST USED PAIR TO NXM
                        ;       .  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

```
                                        ;-------------------------------------------------------------
                                        ;;*************************************************************
                                        ;*TEST 13       INDIRECT ACCESSING OF UNIBUS MAP REGISTERS
                                        ;;*************************************************************
     005014  000004                     TST13:  SCOPE

1514
1515 005016  005737  001720                     TST     PMIS                    ; ANY PMI MEMORY?
1516 005022  001567                             BEQ     TST14           ;; IF NONE, EXIT TEST
1517 005024  032737  000040  000052             BIT     #BIT05,@#52             ; UFD MODE ?
1518 005032  001406                             BEQ     1$                      ; BRANCH, IF NOT
1519 005034  032737  000040  177734             BIT     #BIT05,KMCR             ; 18 BIT MODE ?
1520 005042  001402                             BEQ     1$                      ; BRANCH, IF NOT
1521 005044  000137  005362                     JMP     10$                     ; EXIT TEST
1522 005050  052737  000400  177730  1$:       BIS     #BIT08,DCSR             ; SET DIAGNOSTIC MODE
1523 005056  042737  000006  177730             BIC     #BIT02!BIT01,DCSR       ; SELECT NPR THRU DDR
1524                                    ;
1525                                    ; POINT EACH OF MAPPING REGISTER PAIRS TO 17760000
1526                                    ;
1527 005064  012701  000040                     MOV     #32.,R1                 ; DO FOR 32. REGISTER PAIRS
1528 005070  012702  170200                     MOV     #MAPLOO,R2              ; POINT TO ADDRESS OF THE FIRST REGISTER
1529 005074  012722  160000             2$:     MOV     #160000,(R2)+           ; . 160000 -> LO REGISTER
1530 005100  012722  000077                     MOV     #77,(R2)+               ; . 77 -> HI REGISTER
1531 005104  077105                             SOB     R1,2$                   ; . CONTINUE UNTILL ALL DONE
1532                                    ;
1533                                    ; INITIALISE TO DO RELOCATION IN 22 BIT MODE
1534                                    ;
1535 005106  012737  002234  000004             MOV     #TIMOUT,@#4             ; POINT TIMEOUT ROUTINE
1536 005114  004737  002244                     JSR     PC,MAPPR                ; REMAP PROGRAM AREA
1537 005120  005237  177572                     INC     MMR0                    ; TURN ON MMU
1538 005124  052737  000040  172516             BIS     #BIT05,MMR3             ; ENABLE RELOCATION
1539 005132  012700  170200                     MOV     #MAPLOO,R0              ; POINT R0 TO FIRST REGISTER
1540 005136  012737  000000  172354             MOV     #0, KIPAR6              ; LET KIPAR6 SELECT MAP PAIR 0
1541 005144  012710  000000                     MOV     #0,(R0)                 ; MOVE 0 TO LOW ADDRESS
1542 005150  012760  000000  000002             MOV     #0,2(R0)                ; MOVE 0 TO HI ADDRESS
1543 005156  012701  140000                     MOV     #140000,R1              ; TO ACCESS THRU KIPAR6
1544                                    ;
1545                                    ; STORE # OF REGISTERS TO TEST IN R4
1546                                    ;
1547 005162  032737  000040  000052             BIT     #BIT05,@#52             ; UFD MODE ?
1548 005170  001010                             BNE     3$                      ; IF YES, BRANCH
1549 005172  013737  177734  001162             MOV     KMCR,$TMP1              ; SAVE KMCR
1550 005200  005037  177734                     CLR     KMCR                    ; DO NOT DISABLE ANY REGISTERS
1551 005204  012704  000036                     MOV     #30.,R4                 ; DO FOR ALL REGISTERS
1552 005210  000407                             BR      4$                      ; BRANCH AROUND
1553 005212  113703  177734             3$:     MOVB    KMCR,R3                 ; STORE # OF DISABLED REGISTERS
1554 005216  042703  000300                     BIC     #BIT07!BIT06,R3         ; LEAVE JUST BITS <4-0>
1555 005222  012704  000036                     MOV     #30.,R4                 ; STORE MAX. # OF REGISTERS
1556 005226  160304                             SUB     R3,R4                   ; GET # OF ACCESSABLE REGISTERS
1557                                    ;
1558                                    ; DO DATA OUT FOR EACH REGISTER PAIR UNDER TEST
1559                                    ;
1560 005230  005003                     4$:     CLR     R3                      ; START WITH REGISTER PAIR 0
1561 005232  005037  000000             5$:     CLR     @#0                     ; . CLEAR LOCATION UNDER TEST
1562 005236  005037  001730                     CLR     TOUT                    ; . CLEAR TIMEOUT FLAG
1563 005242  010037  001124                     MOV     R0,$GDDAT               ; . PATTERN IS ADDRESS OF LO REGISTER
1564 005246  004737  002210                     JSR     PC,DDOUT                ; . DO DIAGN. DATA OUT
```

T13 INDIRECT ACCESSING OF UNIBUS MAP REGISTERS

```
1565 005252  005737  001730                    TST     TOUT            ; . TIMEOUT FLAG SET?
1566 005256  001411                            BEQ     6$              ; . IF NOT BRANCH
1567 005260  013737  172354  001122            MOV     KIPAR6,$BDADR   ; . STORE # OF PAIR FOR ERRORS
1568 005266  012705  000007                    MOV     #7,R5           ; . PREPARE TO SHIFT TO GET TO LOWER BITS
1569 005272  006237  001122          11$:      ASR     $BDADR          ; . . SHIFT ONCE
1570 005276  077503                            SOB     R5,11$          ; . . SHIFT 7 TIMES
1571 005300  104020                            ERROR   +20             ; . IN UNIQUE ADDRESSING OF REGISTERS
1572 005302  023737  000000  001124  6$:       CMP     @#0,$GDDAT      ; . DATA OUT OK?
1573 005310  001404                            BEQ     7$              ; . IF YES, BRANCH
1574 005312  013737  000000  001126            MOV     @#0,$BDDAT      ; . STORE FOR ERROR REPORTS
1575 005320  104013                            ERROR   +13             ; . PERFORMING DATA OUT
1576 005322  012720  160000          7$:       MOV     #160000,(R0)+   ; . POINT JUST USED REGISTER
1577 005326  012720  000077                    MOV     #77,(R0)+       ; . TO NXM
1578 005332  012710  000000                    MOV     #0,(R0)         ; . POINT NEXT REGISTER PAIR
1579 005336  012760  000000  000002            MOV     #0,2(R0)        ; . TO LOCATION 0
1580 005344  062737  000200  172354            ADD     #200,KIPAR6     ; . ACCESS NEXT REGISTER PAIR
1581 005352  062703  000001                    ADD     #1,R3           ; . INCREMENT COUNT
1582 005356  020304                            CMP     R3,R4           ; . ALL AVAILABLE PAIRS DONE?
1583 005360  003724                            BLE     5$              ; . BRANCH IF NOT
1584 005362  013737  001162  177734  10$:       MOV     $TMP1,KMCR      ; RESTORE KMCR
1585 005370  005037  177572                    CLR     MMR0            ; DISABLE MMU
1586 005374  042737  000040  172516            BIC     #BIT05,MMR3     ; DISABLE MAPPING
```

```
                                .SBTTL TEST - DISABLING OF THE MAPPING REGISTERS

                                ;* THIS TEST WILL CHECK THAT NONE OF THE REGISTERS DISABLED BY KMCR<4-0>
                                ;* PERFORM RELOCATION.  IN UFD MODE ONLY THE BITS ACTUALLY SET IN THE KMCR WILL
                                ;* BE CHECKED, OTHREWISE ALL OF THEM EXCEPT THE FIRST 4 ARE CHECKED.
                                ;* IN UFD MODE THIS TEST WILL RUN ONLY IF KMCR<5>=0. (IN 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 DIGNOSTIC MODE IS ON DCSR<8>=1
                                ;       SELECT DIAGNOSTIC NPR THRU DDR BY DCSR<2-1>=<0,0>
                                ;       CALL MAP_PROGRAM_AREA
                                ;       LET MMR3<5> = #1 TO ENABLE RELOCATION
                                ;       LET MMR0<0> = #1 TO ENABLE MMU
                                ;       LET R1 = #140000 TO ACCESS THRU KIPAR6
                                ;       LET KIPAR6 = #7600 TO ACCESS MAP REG. 32.
                                ;       SET 4 TO IGNORE TIMEOUT
                                ;       LET PATTERN = #125252
                                ;       LET @#0 = #0
                                ;       LET MAPL37 = #0
                                ;       LET MAPH37 = #0
                                ;       CALL DIAGNOSTIC_DATA_OUT<(R0),PATTERN>
                                ;       IF @#0 EQ PATTERN THEN
                                ;       .  ERROR REG. PAIR 37 PERFORMS RELOCATION
                                ;       ENDIF
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------------

                                ;;*****************************************************************
                                ;*TEST 14       DISABLING REGISTERS
                                ;;*****************************************************************
    005402  000004              TST14:  SCOPE

                                        TST     PMIS                    ; ANY PMI MEMORY?
                                        BEQ     TST15           ;; IF NONE, EXIT TEST
    005404  005737  001720              BIT     #BIT05,@#52             ; UFD MODE ?
    005410  001470                      BEQ     1$                      ; BRANCH, IF NOT
    005412  032737  000040  000052      BIT     #BIT05,KMCR             ; 18 BIT MODE ?
    005420  001404                      BNE     TST15           ;;EXIT TEST IF 18 BIT MODE
    005422  032737  000040  177734
    005430  001060
                                ;
                                ; INITIALISE MMU AND DIAGNOSTIC REGISTERS
                                ;
    005432  052737  000400  177730  1$:     BIS     #BIT08,DCSR             ; SET DIAGNOSTIC MODE
    005440  042737  000006  177730          BIC     #BIT02!BIT01,DCSR       ; SELECT NPR THRU DDR
    005446  004737  002244                  JSR     PC,MAPPR                ; REMAP PROGRAM AREA
    005452  005237  177572                  INC     MMR0                    ; TURN ON MMU
    005456  052737  000040  172516          BIS     #BIT05,MMR3             ; ENABLE RELOCATION
                                ;
                                ; TRY TO DO RELOCATION THRU REGISTER PAIR 31.
                                ;
    005464  012737  005546  000004          MOV     #2$,@#4                 ; INITIALISE TIMEOUT ROUTINE
    005472  012701  140000                  MOV     #140000,R1              ; ACCESS THRU KIPAR6
```

F4

T14 DISABLING REGISTERS

```
 1642 005476  012737  007600  172354          MOV     #7600,KIPAR6            ; ACCESS REG. PAIR 31.
 1643 005504  012737  125252  001124          MOV     #125252,$GDDAT          ; STORE PATTERN
 1644 005512  005037  000000                  CLR     @#0                     ; CLEAR LOCATION UNDER TEST
 1645 005516  012737  000000  170374          MOV     #0,MAPL37               ; LET MAP PAIR 31.
 1646 005524  012737  000000  170376          MOV     #0,MAPH37               ; POINT TO 0
 1647 005532  013737  001124  177732          MOV     $GDDAT,DDR              ; STORE PATTERN IN DDR
 1648 005540  005011                          CLR     (R1)                    ; EXTENAL READ TO PROVIDE ADDRESS
 1649 005542  104021                          ERROR   +21                     ; RELOCATION PERFORMED THRU REG. 31.
 1650 005544  000402                          BR      3$                      ; EXIT TEST
 1651 005546  005726                  2$:     TST     (SP)+                   ; RESTORE STACK
 1652 005550  005726                          TST     (SP)+
 1653 005552  005037  177572          3$:     CLR     MMR0                    ; DISABLE MMU
 1654 005556  042737  000040  172516          BIC     #BIT05,MMR3             ; DISABLE MAPPING
 1655 005564  012737  002234  000004          MOV     #TIMOUT,@#4             ; RESTORE TIMEOUT
```

TEST - NXM MEMORY TIMEOUT

```
                                .SBTTL TEST - NXM MEMORY TIMEOUT

                                ;* THIS TEST WILL VERIFY THAT WHENEVER LOCATION 17760000 IS ACCESSED
                                ;* DIAGNOSTIC NPR CYCLES RESULT IN A TIMEOUT SETTING BIT <15> IN DCSR.
                                ;
                                ;  BGNTST
                                ;
                                ;       MAKE SURE THAT DIGNOSTIC MODE IS ON DCSR<8>=1
                                ;       SELECT DIAGNOSTIC NPR THRU DDR BY DCSR<2-1>=<0,0>
                                ;       CALL MAP_PROGRAM_AREA
                                ;       LET MMR0<0> = 1 TO ENABLE MMU
                                ;       IF NOT UFD MODE OR KMCR<5> EQ #0
                                ;       .  LET TEST_LOCATION = #140000 TO ACCESS THRU KIPAR6
                                ;       .  LET KIPAR6 = #0 TO ACCESS REGISTER PAIR 0
                                ;       .  LET LO REGISTER = #160000 TO SET REG. 0 TO NXM
                                ;       .  LET HI REGISTER = #77
                                ;       .  LET MMR3<5> = 1 TO ENABLE RELOCATION
                                ;       ELSE
                                ;       .  IF KMCR<5> EQ #1 THEN(18 BIT MODE)
                                ;       .  .  LET TEST_LOCATION = #140000 TO ACCESS THRU KIPAR6
                                ;       .  .  LET KIPAR6 = #177600 TO ACCESS NXM
                                ;       .  ENDIF
                                ;       ENDIF
                                ;       START DIAGNOSTIC_DATA_OUT TO @#0
                                ;       IF DCSR<15> NE 0 THEN
                                ;       .  ERROR NXM BIT SET
                                ;       ENDIF
                                ;       PUT ACTUAL EXTERNAL ADDRESS ON THE BUS
                                ;       IF DCSR<15> NE 1 THEN
                                ;       .  ERROR NXM MEMORY DOES NOT SET DCSR<15>
                                ;       ENDIF
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------------------------

                                ;;*********************************************************************
                                ;*TEST 15       NXM MEMORY TIMEOUT
                                ;;*********************************************************************
     005572  000004             TST15:  SCOPE

                                        TST     PMIS                            ; ANY PMI MEMORY?
 005574  005737  001720                 BEQ     TST16           ;; IF NONE, EXIT TEST
 005600  001470                         JSR     PC,MAPPR                        ; REMAP PROGRAM AREA
 005602  004737  002244                 INC     MMR0                            ; ENABLE MMU
 005606  005237  177572
                                ;
                                ; DECIDE WHETHER TO  ACCESS WITH RELOCATION OR NOT
                                ;
                                        MOV     #140000,R1                      ; ACCESS THRU KIPAR6
 005612  012701  140000                 BIT     #BIT05,@#52                     ; UFD MODE?
 005616  032737  000040  000052         BEQ     1$.                             ; IF NOT, BRANCH
 005624  001410                         BIT     #BIT05,KMCR                     ; 18 BIT MODE?
 005626  032737  000040  177734         BEQ     2$                              ; IF NOT BRANCH
 005634  001412                         MOV     #177600,KIPAR6                  ; KIPAR6 HAS NXM
 005636  012737  177600  172354         BR      3$                              ; GO DO IT
 005644  000422                 1$:     MOV     KMCR,$TMP1                      ; SAVE KMCR
 005646  013737  177734  001162         BIC     #BIT05,KMCR                     ; IN NOT UFD, DO IN 22 BITS
 005654  042737  000040  177734
```

T15 NXM MEMORY TIMEOUT

```
 1711 005662  012737  000000  172354  2$:     MOV     #0,KIPAR6                       ; KIPAR6 ACCESS MAP REG. 0
 1712 005670  012737  160000  170200          MOV     #160000,MAPL00                  ; MAP PAIR HAS NXM
 1713 005676  012737  000077  170202          MOV     #77,MAPH00
 1714 005704  052737  000040  172516          BIS     #BIT05,MMR3                     ; ENABLE RELOCATION
 1715                                 ;
 1716                                 ; DO ACTUAL DATA OUT CYCLE
 1717                                 ;
 1718 005712  005037  177732          3$:     CLR     DDR                             ; PROMT DIAG. DATO
 1719 005716  005011                          CLR     (R1)                            ; WRITE TO NXM
 1720 005720  032737  100000  177730          BIT     #BIT15,DCSR                     ; NXM SET?
 1721 005726  001001                          BNE     5$                              ; IF YES, BRANCH
 1722 005730  104022                          ERROR   +22                             ; NXM ACCESS DOES NOT SET DCSR<15>
 1723 005732  005037  177572          5$:     CLR     MMR0                            ; DISABLE MMU
 1724 005736  042737  000040  172516          BIC     #BIT05,MMR3                     ; AND MAPPING TOO
 1725 005744  032737  000040  000052          BIT     #BIT05,@#52                     ; UFD MODE
 1726 005752  001003                          BNE     TST16           ;;IF NOT UFD, GO TO NEXT TEST
 1727 005754  013737  001162  177734          MOV     $TMP1,KMCR                      ; OTHERWISE, RESTORE KMCR
 1728
```

```
1730                                    .SBTTL TEST - CARRY PROPOGATION TEST
1731
1732                                    ;* THIS TEST CHECKS THE CARRY PROPOGATION THRU THE ALU. MAPPING REGISTER PAIR 0
1733                                    ;* IS LOADED WITH ALL 1'S.  LOCATION 1 IS ACCESSED. THE RESULT OF THE DATO
1734                                    ;* CYCLE IS SUPPOSED TO BE LOADED IN LOCATION 0.
1735                                    ;
1736                                    ;  BGNTST
1737                                    ;
1738                                    ;       IF UFD MODE AND KMCR<5> EQ #1 THEN
1739                                    ;       .  EXIT TEST (RELOCATION DOES NOT HAVE TO BE PERFORMED IN 18 BIT)
1740                                    ;       ENDIF
1741                                    ;       MAKE SURE THAT DIGNOSTIC MODE IS ON DCSR<8>=1
1742                                    ;       SELECT DIAGNOSTIC NPR THRU DDR BY DCSR<2-1>=<0,0>
1743                                    ;       POINT TIMEOUT_ROUTINE TO VECTOR 4
1744                                    ;       CALL MAP_PROGRAM_AREA
1745                                    ;       LET MMR0<0> = 1 TO ENABLE MMU
1746                                    ;       LET MMR3<5> = 1 TO ENABLE RELOCATION
1747                                    ;       LET KIPAR6 = #0 TO ACCESS THRU MAP REG.0
1748                                    ;       LET R1 = #140001 TO ACCESS THRU KIPA6
1749                                    ;       LET LO MAP REG. = #177777
1750                                    ;       LET HI MAP REG. = #77
1751                                    ;       LET PATTERN = #125252
1752                                    ;       CALL DIAGNOSTIC_DATA_OUT<(R1),PATTERN>
1753                                    ;       LET MMR0<0> = #0 TO DISABLE MMU
1754                                    ;       IF @0 NE #125252 THEN
1755                                    ;       .  ERROR IN CARRY PROPOGATION
1756                                    ;       ENDIF
1757                                    ;
1758                                    ;  ENDTST
1759                                    ;
1760                                    ;-------------------------------------------------------------------------
1761
1762                                    ;;*************************************************************************
                                        ;*TEST 16       CARRY PROPOGATION TEST
                                        ;;*************************************************************************
     005762  000004                     TST16:  SCOPE
1763
1764 005764  005737  001720             TST     PMIS                    ; ANY PMI MEMORY?
1765 005770  001465                     BEQ     TST17           ;; IF NONE, EXIT TEST
1766 005772  032737  000040  000052     BIT     #BIT05,@#52             ; UFD MODE ?
1767 006000  001404                     BEQ     1$                      ; BRANCH, IF NOT
1768 006002  032737  000040  177734     BIT     #BIT05,KMCR             ; 18 BIT MODE ?
1769 006010  001055                     BNE     TST17           ;; GO TO NEXT TEST, IF YES
1770                                    ;
1771                                    ; INITIALISE MMU AND DIAGNOSTIC REGISTERS
1772                                    ;
1773 006012  052737  000400  177730     1$:     BIS     #BIT08,DCSR             ; SET DIAGNOSTIC MODE
1774 006020  042737  000006  177730             BIC     #BIT02!BIT01,DCSR       ; SELECT NPR THRU DDR
1775 006026  004737  002244             JSR     PC,MAPPR                ; REMAP PROGRAM AREA
1776 006032  005237  177572             INC     MMR0                    ; TURN ON MMU
1777 006036  052737  000040  172516     BIS     #BIT05,MMR3             ; ENABLE RELOCATION
1778                                    ;
1779                                    ; ACCESS LOCATION 0 THRU RELOCATION REGISTERS
1780                                    ;
1781 006044  005037  172354             CLR     KIPAR6                  ; CLEAR KIPAR6
1782 006050  012701  140002             MOV     #140002,P1              ; ACCESS THRU KIPAR6
1783 006054  012737  177777  170200     MOV     #177777,MAPLO0          ; ALL ONE'S TO LO REG
```

J4

T16 CARRY PROPOGATION TEST

```
1784 006062  012737  000077  170202          MOV     #77,MAPH00              ; ALL ONE'S TO HI REG
1785 006070  012737  125252  001124          MOV     #125252,$GDDAT          ; THE PATTERN TO BE STORED
1786 006076  005037  000000                  CLR     @#0                     ; CLEAR TEST LOCATION
1787 006102  004737  002210                  JSR     PC,DDOUT                ; DO DATA OUT
1788 006106  005037  177572                  CLR     MMR0                    ; DISABLE MMU
1789 006112  042737  000040  172516          BIC     #BIT05,MMR3             ; AND MAPPING
1790 006120  022737  125252  000000          CMP     #125252,@#0             ; DATA OUT OK?
1791 006126  001406                          BEQ     TST17           ;; IF YES, GO TO NEXT TEST
1792 006130  013737  000000  001126          MOV     @#0,$BDDAT              ; STORE FOR ERROR REPORTS
1793 006136  005037  001122                  CLR     $BDADR                  ; ADDRESS 0
1794 006142  104023                          ERROR   +23                     ; IN CARRY PROPOGATION
1795
```

```
                                .SBTTL TEST - EXTENSIVE CARRY PROPOGATION TEST

                                ;* THIS TEST PERFORMS EXTENSIVE CHECKING OF CARRY PROPOGATION. FIRST 1K IS
                                ;* CHECKED IN WORD INCREMENTS, AFTER THAT EACH 1K UP TO 4K. DIAGNOSTIC DATI
                                ;* CYCLES ARE PERFORMED AND A MAPPING REGISTER PAIR USED IS ALWAYS AT ALL 1'S.
                                ;
                                ;  BGNTST
                                ;
                                ;       IF UFD MODE AND KMCR<5> EQ #1 THEN
                                ;       .  EXIT TEST (RELOCATION DOES NOT HAVE TO BE PERFORMED IN 18 BIT)
                                ;       ENDIF
                                ;       CALL MAP_PROGRAM_AREA
                                ;       ENABLE MAPPING
                                ;       INITIALISE ALL MAPPING REGISTERS TO POINT TO ALL 1'S
                                ;       DO FOR 4K OF MEMORY
                                ;       .  IF MEMORY ACCESSED LESS THEN 1K
                                ;       .  .  DO WORD INCREMENTS
                                ;       .  ELSE
                                ;       .  .  DO 1K INCREMENTS
                                ;       .  ENDIF
                                ;       .  DO DATI INCREMENTS
                                ;       .  IF DDR NE TO LOCATION IN MEMORY THEN
                                ;       .  .  ALU ERROR
                                ;       .  ENDIF
                                ;       ENDDO
                                ;
                                ;       ENDTST
                                ;
                                ;-------------------------------------------------------------------------

                                ;;*********************************************************************
                                ;*TEST 17       EXTENSIVE CARRY PROPOGATION TEST
                                ;;*********************************************************************
     006144  000004             TST17:  SCOPE

                                        TST     PMIS                            ; ANY PMI MEMORY?
     006146  005737  001720             BEQ     TST20           ;; IF NONE, EXIT TEST
     006152  001516             BIT     #BIT05,@#52                             ; UFD MODE
     006154  032737  000040  000052     BEQ     1$                              ; IF NOT, BRANCH
     006162  001404             BIT     #BIT05,KMCR                             ; 18 BIT MODE
     006164  032737  000040  177734     BNE     TST20           ;; IF 18 BIT MODE, DON'T DO THIS TEST
     006172  001106
                                ;
                                ; INITIALISE MMU AND RELOCATION
                                ;
     006174  052737  000400  177730 1$:     BIS     #BIT08,DCSR                     ; SET DIAGNOSTIC MODE
     006202  042737  000006  177730     BIC     #BIT02!BIT01,DCSR               ; SELECT NPR THRU DDR
     006210  004737  002244             JSR     PC,MAPPR                        ; REMAP PROGRAM AREA
     006214  052737  000060  172516     BIS     #BIT05!BIT04,MMR3               ; ENABLE RELOCATION AND 22BITS
     006222  005237  177572             INC     MMR0                            ; ENABLE MMU
                                ;
                                ; POINT ALL MAPPING REGISTERS TO ALL 1'S
                                ;
     006226  012701  170200             MOV     #MAPL00,R1                      ; START WITH 0
     006232  012703  000036             MOV     #36,R3                          ; DO FOR ALL REGISTERS
     006236  012721  177776 2$:     MOV     #177776,(R1)+                   ; LOW MAP REGISTER
     006242  012721  000077             MOV     #77,(R1)+                       ; HIGH MAP REGISTER
     006246  077305                     SOB     R3,2$                           ; DO FOR ALL
```

```
1851                                    ;
1852                                    ; START WITH DATI ON A WORD BOUNDARY
1853                                    ;
1854 006250  005037  172354                CLR     KIPAR6                  ; ACCESS REGISTER PAIR 0
1855 006254  012701  140002                MOV     #140002,R1              ; START WITH 0 THRU KIPAR6
1856 006260  000401                        BR      4$                      ;
1857 006262  005721                3$:     TST     (R1)+                   ; . DO IN WORD INCREMENTS
1858 006264  004737  002222        4$:     JSR     PC,DDIN                 ; . DO DATI
1859 006270  010137  001122                MOV     R1,$BDADR               ; . STORE JUST ACCESSED ADDRESS
1860 006274  162737  000002  001122        SUB     #2,$BDADR               ; . GET RID OF OVERFLOW
1861 006302  027737  172614  177732        CMP     @$BDADR,DDR             ; . PROPER DATA?
1862 006310  001412                        BEQ     7$                      ; . IF EQUAL, BRANCH
1863 006312  017737  172604  001124        MOV     @$BDADR,$GDDAT          ; . STORE RECEIVED DATA
1864 006320  013737  177732  001126        MOV     DDR,$BDDAT              ; . STORE EXPECTED DATA
1865 006326  042737  160000  001122        BIC     #BIT15!BIT14!BIT13,$BDADR ; . STRIP OF PAR BITS
1866 006334  104023                        ERROR   +23                     ; . ERROR IN RELOCATION
1867                                    ;
1868                                    ; DECIDE WHICH CYCLE SHOULD OCCUR NEXT
1869                                    ;
1870 006336  005737  172354        7$:     TST     KIPAR6                  ; . FIRST TIME 1K BOUNDARY?
1871 006342  001011                        BNE     10$                     ; . IF NOT, BRANCH
1872 006344  022701  144000                CMP     #144000,R1              ; . LESS THAN 1K DONE?
1873 006350  103744                        BLO     3$                      ; . DO IN WORD INCREMENTS
1874 006352  012701  140002                MOV     #140002,R1              ; . NOW DO INCREMENTING THRU PAR
1875 006356  012737  000040  172354        MOV     #40,KIPAR6              ; . FIRST 1K
1876 006364  000737                        BR      4$                      ; . GO DO COMPARE
1877 006366  062737  000040  172354 10$:    ADD     #40,KIPAR6              ; . ACCESS NEXT 1K
1878 006374  022737  000200  172354        CMP     #200,KIPAR6             ; . OVER 4K BOUNDARY?
1879 006402  003330                        BGT     4$                      ; . IF NOT, GO DO DATI
1880 006404  005037  177572                CLR     MMR0                    ; DISABLE MMU
1681
```

```
                        .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)
                        ;       ENDIF
                        ;       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
                        ;       .  ENDIF
                        ;       .  IF RESULT OF THE CYCLE IS NOT PROPER THEN
                        ;       .  .  ERROR IN ALU
                        ;       .  ENDIF
                        ;       ENDDO
                        ;
                        ;  ENDTST
                        ;
                        ;-------------------------------------------------------------
                        ;
                        ;;*****************************************************************
                        ;*TEST 20       ALU TEST
                        ;;*****************************************************************
 006410  000004         TST20:  SCOPE

                                TST     PMIS                            ; ANY 22-BIT PMI MEMORY?
                                BEQ     TST21           ;; IF NONE, EXIT TEST
 006420  032737  000040 177734  BIT     #BIT05,KMCR                     ; 18 BIT MODE
                                BNE     TST21           ;; IF 18 BIT MODE, DON'T DO THIS TEST
                        ;
                        ; INITIALISE MMU AND RELOCATION
                        ;
 006430  052737  000400 177730  1$:     BIS     #BIT08,DCSR             ; SET DIAGNOSTIC MODE
 006436  042737  000006 177730          BIC     #BIT02!BIT01,DCSR       ; SELECT NPR THRU DDR
 006444  004737  002244                 JSR     PC,MAPPR                ; REMAP PROGRAM AREA
 006450  052737  000060 172516          BIS     #BIT05!BIT04,MMR3       ; ENABLE RELOCATION AND 22BITS
 006456  005237  177572                 INC     MMR0                    ; ENABLE MMU
 006462  013702  001720                 MOV     PMIS,R2                 ; STORE SIZE OF PMI MEMORY
 006466  010237  001160                 MOV     R2,$TMP0                ; CREATE A PATTERN
 006472  012703  004200                 MOV     #4200,R3                ; CONSTANT FOR PAR
 006476  040203                         BIC     R2,R3                   ; MASK OUT NXM
 006500  072227  177766                 ASH     #-10.,R2                ; FOR MAPH00
                        ;
                        ; DO FOR ALL COMBINATIONS POSSIBLE, 4 BITS AT A TIME
                        ;
 006504  012701  140000                 MOV     #140000,R1              ; ACCESS THRU KIPAR6
 006510  005037  172354                 CLR     KIPAR6                  ; ACCESS REGISTER PAIR 0
```

```
 1937 006514  005037  170200                 CLR     MAPLOO                  ; CLEAR MAP REGISTERS
 1938 006520  005037  170202                 CLR     MAPHOO                  ;
 1939 006524  005037  001162                 CLR     $TMP1                   ; CLEAR STORAGE FOR PAR
 1940 006530  005037  001122                 CLR     $BDADR                  ; FIRST ADDRESS USED
 1941 006534  005037  001124                 CLR     $GDDAT                  ; CLEAR PATTERN
 1942 006540  004737  002222          2$:    JSR     PC,DDIN                 ; . DO DATI
 1943 006544  013737  001162  172354        MOV     $TMP1,KIPAR6            ; . GET ADDED PAR
 1944 006552  023777  177732  172342        CMP     DDR,@$BDADR             ; . DATI OK?
 1945 006560  001435                        BEQ     10$                     ; . IF SO, BRANCH
 1946 006562  013737  177732  001126        MOV     DDR,$BDDAT              ; . STORE RECIEVED DATA
 1947 006570  017737  172326  001124        MOV     @$BDADR,$GDDAT          ; . STORE EXPECTED DATA
 1948 006576  042737  160000  001122        BIC     #BIT15!BIT14!BIT13,$BDADR ; . STRIP OF PAR BITS
 1949 006604  104023                        ERROR   +23                     ;
 1950 006606  005037  001124                CLR     $GDDAT                  ;
 1951 006612  000420                        BR      10$                     ;
 1952 006614  004737  002210          5$:    JSR     PC,DDOUT                ; . DO DATO
 1953 006620  013737  001162  172354        MOV     $TMP1,KIPAR6            ; . GET ADDED PAR
 1954 006626  023777  001124  172266        CMP     $GDDAT,@$BDADR          ; . DATI OK?
 1955 006634  001407                        BEQ     10$                     ; . IF SO, BRANCH
 1956 006636  017737  172260  001126        MOV     @$BDADR,$BDDAT          ; . STORE EXPECTED DATA
 1957 006644  042737  160000  001122        BIC     #BIT15!BIT14!BIT13,$BDADR ; . STRIP OF PAR BITS
 1958 006652  104023                        ERROR   +23                     ;
 1959                               ;
 1960                               ; DECIDE WHAT ACCESS NEXT
 1961                               ;
 1962 006654  005037  172354        10$:    CLR     KIPAR6                  ; . CLEAR PAR
 1963 006660  062701  001042                ADD     #1042,R1                ; . START INCREMENTING
 1964 006664  062737  021042  170200        ADD     #21042,MAPLOO           ; . EVERY 4TH BIT
 1965 006672  005537  170202                ADC     MAPHOO                  ; . DON'T FORGET CARRY
 1966 006676  062737  000002  170202        ADD     #2,MAPHOO               ; . FOR ALL REGISTERS
 1967 006704  010137  001122                MOV     R1,$BDADR               ; . SAVE ADDRESS USED
 1968 006710  063737  170200  001122        ADD     MAPLOO,$BDADR           ; . ADD UNIBUS MAP REG.
 1969 006716  052737  140000  001122        BIS     #140000,$BDADR          ; . MAKE SURE KIPAR6 SELECTED
 1970 006724  042737  020000  001122        BIC     #20000,$BDADR           ;
 1971 006732  022701  150420                CMP     #150420,R1              ; . OVERFLOW FROM ADDITION?
 1972 006736  001003                        BNE     15$                     ; . IF NO, BRANCH
 1973 006740  062737  000200  001162        ADD     #200,$TMP1              ; . ADD OVERFLOW FOR PAR
 1974 006746  060337  001162          15$:  ADD     R3,$TMP1                ; . ADD INCR. TO PAR
 1975 006752  022737  002000  001162        CMP     #2000,$TMP1             ; . LESS THAN 32K?
 1976 006760  003267                        BGT     2$                      ; . IF YES, DO DATI
 1977 006762  005237  001124                INC     $GDDAT                  ; . INCREMENT PATTERN
 1978 006766  023737  001160  001162        CMP     $TMP0,$TMP1             ; . OVER AVAILABLE MEMORY?
 1979 006774  003307                        BGT     5$                      ; . IF NOT, DO ANOTHER DATO
 1980 006776  005037  177572          25$:  CLR     MMR0                    ; DISABLE MMU
```

```
                                .SBTTL TEST - MAIN MEMORY DISABLE

                                ;* THIS TEST WILL CHECK THAT A MAIN MEMORY RESPONSE IS DISABLED WHENEVER
                                ;* THE APPROPRIATE BITS IN THE KMCR REGISTER ARE SET. IN UFD MODE ONLY BITS SET
                                ;* IN KMCR WILL BE TESTED. IN ALL OTHER ENVIRONMENTS ALL BITS WILL BE TESTED.
                                ;
                                ;  BGNTST
                                ;
                                ;       CALL MAP_PROGRAM_AREA
                                ;       LET MMR0<0> = 1 TO ENABLE MMU
                                ;       LET R1 = KMCR_LOW_BYTE
                                ;       CLEAR BITS <5,6,7> IN R1
                                ;       IF KMCR<5> EQ #1 THEN (18 BIT MODE)
                                ;       .  LET KIPAR6 = #7400
                                ;       ELSE (22 BIT MODE)
                                ;       .  LET KIPAR6 = #177400
                                ;       ENDIF
                                ;       LET TEST_LOCATION = #140000 TO ACCESS THRU KIPAR6
                                ;       DO FOR R0 FROM #1 TO R1 ALL PAGES SPECIFIED IN KMCR
                                ;       .  LET R2 = @TEST_LOCATION
                                ;       .  IF NO TIMEOUT THEN
                                ;       .  .  ERROR KMCR<4-5> DOESNOT DISABLE MAIN MEMORY
                                ;       .  ENDIF
                                ;       .  LET KIPAR6 = KIPAR6 - #200
                                ;       ENDDO
                                ;       IF NOT UFD MODE THEN
                                ;       .  SAVE KMCR
                                ;       .  CHECK KMCR<5-0> TO BE READ-WRITE
                                ;       .  LET KMCR<5-0> = #0 (ALL NON-UNIBUS MEMORY)
                                ;       .  LET KIPAR6 = #171600
                                ;       .  IF @#140000 DOES NOT TIMEOUT THEN
                                ;       .  .  LET KMCR<4-0> = <1000> TO DISABLE HIGHER THAN 28K
                                ;       .  .  IF @#140000 DOESNOT TIMEOUT THEN
                                ;       .  .  .  ERROR SETTING KMCR<5-0> DOESNOT DISABLE MEMORY
                                ;       .  .  ENDIF
                                ;       .  ENDIF
                                ;       .  RESTORE KMCR
                                ;       ENDIF
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------------------------

                                ;;*********************************************************************
                                ;*TEST 21        MAIN MEMORY DISABLE
                                ;;*********************************************************************
 007002  000004                 TST21:  SCOPE
2026 007004  005737  001720             TST     PMIS                    ; ANY PMI MEMORY?
2027 007010  001002             .       BNE     20$                     ; IF YES, DO THE TEST
2028 007012  000137  007454     .       JMP     100$                    ; OTHERWISE EXIT TEST
2029 007016  052737  000400  177730  20$:  BIS     #BIT08,DCSR             ; STANDALONE MODE
2030 007024  004737  002244             JSR     PC,MAPPR                ; REMAP PROGRAM AREA
2031 007030  005237  177572             INC     MMR0                    ; ENABLE MMU
2032 007034  113701  177734             MOVB    KMCR,R1                 ; SAVE KMCR
2033 007040  042701  177740             BIC     #177740,R1              ; R1 HAS # OF PAGES DISABLED
2034                            ;
2035                            ; GET THE HIGHEST NXM LOCATION
```

T21 MAIN MEMORY DISABLE

```
2036                                        ;
2037 007044  032737  000040  177734         BIT     #BIT05,KMCR             ; 18 BIT MODE?
2038 007052  001404                         BEQ     1$                      ; IF NOT, BRANCH
2039 007054  012737  007400  172354         MOV     #7400,KIPAR6            ; NXM FOR 18 BITS
2040 007062  000403                         BR      2$                      ; GO DO IT
2041 007064  012737  177400  172354  1$:    MOV     #177400,KIPAR6          ; NXM FOR 22 BITS
2042                                        ;
2043                                        ; GO THRU ALL PAGES OF MEMORY DISABLED
2044                                        ;
2045 007072  105701                 2$:     TSTB    R1                      ; ANY UNIBUS MEMORY?
2046 007074  001420                         BEQ     6$                      ; IF NO, BRANCH
2047 007076  012737  007114  000004         MOV     #4$,@#4                 ; POINT TIMEOUT VECTOR TO PROGRAM
2048 007104  005737  140000         3$:     TST     @#140000                ; . ACCESS THRU KIPAR6
2049 007110  104025                         ERROR   ^25                     ; . KMCR<4-0> DOES NOT DISABLE MAIN MEMORY
2050 007112  000402                         BR      5$                      ; .
2051 007114  005726                 4$:     TST     (SP)+                   ; . ADJUST STACK
2052 007116  005726                         TST     (SP)+                   ; .
2053 007120  162737  000200  172354  5$:     SUB     #200,KIPAR6             ; . ACCESS NEXT LOWER PAGE
2054 007126  077112                         SOB     R1,3$                   ; . DO FOR ALL PAGES IN KMCR
2055 007130  012737  002234  000004         MOV     #TIMOUT,@#4             ; RESTORE TIMEOUT VECTOR
2056                                        ;
2057                                        ; IN NON-UFD MODE, CHECK KMCR<5-0>
2058                                        ;
2059 007136  032737  000040  000052  6$:     BIT     #BIT05,@#52             ; UFD MODE?
2060 007144  001143                         BNE     TST22           ;; IF YES, GO TO NEXT TEST
2061 007146  013737  177734  001162         MOV     KMCR,$TMP1              ; SAVE KMCR
2062 007154  012701  000067                 MOV     #67,R1                  ; HIGHEST VALUE (32K OF MEMORY)
2063 007160  110137  177734         7$:     MOVB    R1,KMCR                 ; . WRITE TO KMCR
2064 007164  120137  177734                 CMPB    R1,KMCR                 ; . WRITTEN OK?
2065 007170  001426                         BEQ     8$                      ; . IF SO, BRANCH
2066 007172  010137  001124                 MOV     R1,$GDDAT               ; . NO,THEN GET THE GOOD DATA
2067 007176  052737  000200  001124         BIS     #BIT07,$GDDAT           ; . IN CASE REBOOT BIT SET
2068 007204  123737  001124  177734         CMPB    $GDDAT,KMCR             ; . WAS THAT THE ONLY WRONG?
2069 007212  001415                         BEQ     8$                      ; . IF YES, BRANCH
2070 007214  010137  001124                 MOV     R1,$GDDAT               ; . IF NOT, RESTORE PATTERN WRITTEN
2071 007220  013737  177734  001126         MOV     KMCR,$BDDAT             ; . GET THE BAD DATA
2072 007226  012737  177734  001122         MOV     #KMCR,$BDADR            ; . GET THE ADDRESS OF THE KMCR
2073 007234  013737  001160  177734         MOV     $TMP0,KMCR              ; . RESTORE RREGISTER
2074 007242  104011                         ERROR   ^11                     ; . ERROR IN KMCR<5-0>
2075 007244  000446                         BR      12$                     ; . EXIT TEST ON ERROR
2076 007246  005301                 8$:     DEC     R1                      ; . LAST LOCATION (0)?
2077 007250  002343                         BGE     7$                      ; . IF NOT, BRANCH
2078                                        ;
2079                                        ; IN NON-UFD MODE, GO THRU THE PAGE JUST ABOVE 32K
2080                                        ;
2081
2082 007252  052737  000020  172516         BIS     #BIT04,MMR3             ; ENABLE 22-BIT
2083 007260  013702  001720                 MOV     PMIS,R2                 ; STORE HIGHEST PAGE
2084 007264  005037  170200                 CLR     MAPL00                  ; CLEAR MAP REG. 0
2085 007270  012737  000001  170202         MOV     #1,MAPH00               ; POINTS TO 32K
2086 007276  052737  000040  172516         BIS     #BIT05,MMR3             ; ENABLE MAPPING
2087 007304  005001                         CLR     R1                      ; ADDRESS FOR DMA CYCLES
2088 007306  012737  007340  000004         MOV     #10$,@#4                ; TIMEOUT TO PROGRAM
2089 007314  012737  002000  172354         MOV     #2000,KIPAR6            ; PAGE JUST ABOVE 32K
2090 007322  012737  000067  177734         MOV     #67,KMCR                ; DISABLE HIGHER THEN 32K
2091                                        ;
2092                                        ; DO A CPU CYCLE TO DISABLED MEMORY
```

T21 MAIN MEMORY DISABLE

```
                                        ;
007330  005737  140000          9$:   TST     @#140000          ; . ACCESS DISABLED MEMORY
007334  104025                  ERROR   .25                     ; . IN DISABLING MEMORY THRU KMCR<4-0>
007336  000411                  BR      12$                     ;
007340  005726                10$:   TST     (SP)+             ; . RESTORE STACK
007342  005726                  TST     (SP)+
                                        ;
                                        ; DO A DMA DATI CYCLE TO DISABLED MEMORY
                                        ;
007344  004737  002222          11$:   JSR     PC,DDIN           ; . DIAGNOSTIC DATI
007350  032737  100000  177730        BIT     @BIT15,DCSR       ; . NXM SET?
007356  001001                  BNE     12$                     ; . IF SET, BRANCH
007360  104025                  ERROR   .25                     ; . IN DMA MEMORY NOT DISABLED
                                        ;
                                        ; CHECK WHETHER ALL PAGES IN MEMORY VERIFIED
                                        ;
007362  023702  172354          12$:   CMP     KIPAR6,R2         ; . IS IT IN MEMORY?
007366  103020                  BHIS    14$                     ; . IF NOT, BRANCH
007370  062737  020000  170200        ADD     @20000,MAPL00     ; . ACCESS NEXT 4K
007376  103002                  BCC     13$                     ; . IF DIDN'T EXCEED 32K, BRANCH
007400  005237  170202            INC     MAPH00            ; . OTHERWISE, INCREMENT MAP REGISTER
007404  005337  177734          13$:   DEC     KMCR              ; . ENABLE NEXT 4K OF MEMORY
007410  032737  000040  177734        BIT     @BIT05,KMCR       ; . ALL 128K DONE?
007416  001404                  BEQ     14$                     ; . IF SO, BRANCH
007420  062737  000200  172354        ADD     @200,KIPAR6       ; . ACCESS NEXT 4K
007426  000740                  BR      9$                      ; . TRY TO DO IT
007430  012737  002234  000004  14$:   MOV     @TIMOUT,@#4       ; RESTORE TIMEOUT VECTOR
007436  013737  001162  177734        MOV     $TMP1,KMCR        ; RESTORE KMCR
007444  005037  172516              CLR     MMR3              ; DISABLE 22-BIT MODE
007450  005037  177572              CLR     MMR0              ; DISABLE MMU
007454                          100$:
```

```
                                .SBTTL  TEST  - CACHE PRESENCE

                                ;* THIS TEST WILL FIND OUT WHETHER THE CACHE IS PRESENT
                                ;
                                ;  BGNTST
                                ;
                                ;       LET KMCR<6>=#1
                                ;       IF KMCR<6> NE #1 THEN
                                ;       .  CACHE NOT PRESENT OR NOT SEEN
                                ;       ENDIF
                                ;
                                ;  ENDTST
                                ;
                                ;-----------------------------------------------

                                ;;*****************************************************************
                                ;*TEST 22       CACHE PRESENCE
                                ;;*****************************************************************
 007454  000004                 TST22:  SCOPE

                                        TST     PMIS                            ; ANY PMI MEMORY?
 007456  005737  001720                 BEQ     TST23           ;; IF NONE, EXIT TEST
 007462  001440                         BIS     #BIT06,KMCR                     ; SET CACHE ENABLE BIT
 007464  052737  000100  177734         BIT     #BIT06,KMCR                     ; DID IT GET SET?
 007472  032737  000100  177734         BNE     TST23           ;;      IF CACHE PRESENT, EXIT TEST
 007500  001031                         TST     $PASS                           ; FIRST PASS?
 007502  005737  001206                 BNE     1$                              ; IF YES, SKIP PRINTOUT
 007506  001014                         CMPB    #APTENV,$ENV                    ; IN APT MODE?
 007510  122737  000001  001220         BEQ     1$                              ; IF SO, SKIP PRINTOUT
 007516  001410                         BIT     #BIT05,@#52                     ; UFD MODE ?
 007520  032737  000040  000052         BNE     1$                              ; IF YES, BRANCH
 007526  001004                         TYPE    .NOCAH                          ; TYPE NO CACHE MESSAGE
 007530  104401  007544                 TYPE    .$CRLF                          ;
 007534  104401  001175                 JMP     UBETST                          ; IF NO CACHE, SKIP ALL CACHE TSTS
 007540  000137  012724         1$:
                                
 007544      040     116     117 NOCAH:  .ASCIZ  / NO CACHE SEEN/
 007547      040     103     101
 007552      103     110     105
 007555      040     123     105
 007560      105     116     000
                                .EVEN
```

TEST - CACHE DISABLED AND KMCR

```
                                .SBTTL  TEST  - CACHE DISABLED AND KMCR

                                ;* THIS TEST WILL CHECK THAT WHENEVER THE DMA CACHE IS DISABLED
                                ;* THE STATUS BITS IN KMCR REGISTER ARE INITIALISED TO POWER UP
                                ;* CONDITIONS: THE VALID BITS ARE CLEARED, A IS THE NEXT AVAILABLE
                                ;* SET, FOLLOWED BY B, C, AND D.
                                ;
                                ;  BGNTST
                                ;
                                ;       DO FOR MMR3<5>=0,1 WITH RELOCATION ENABLED AND DISABLED
                                ;       .  LET KMCR<6> = #0 TO DISABLE CACHE
                                ;       .  LET KMCR<8>=0
                                ;       .  IF KMCR<12!11!10!9> NE #0 THEN
                                ;       .  .  ERROR IN VALID BITS WITH CACHE DISABLED
                                ;       .  ENDIF
                                ;       .  LET KMCR<8>=1
                                ;       .  IF KMCR<14!13!12!11!10!9!> NE #1 THEN
                                ;       .  .  ERROR IN LRU BITS SELECTION
                                ;       .  ENDIF
                                ;       ENDDO
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------------

                                ;;*****************************************************************
                                ;*TEST 23       CACHE DISABLED AND KMCR
                                ;;*****************************************************************
007564  000004                  TST23:  SCOPE

007566  005737  001720                  TST     PMIS                            ; ANY PMI MEMORY?
007572  001463                          BEQ     TST24           ;; IF NONE, EXIT TEST
007574  042737  000040  172516          BIC     #BIT05,MMR3                     ; FIRST DO WITH RELOCATION DISABLED
007602  000403                          BR      2$                              ; GO DO IT
007604  052737  000040  172516  1$:     BIS     #BIT05,MMR3                     ; NOW DO WITH RELOCATION ENABLED
007612  042737  000100  177734  2$:     BIC     #BIT06,KMCR                     ; MAKE SURE THAT CACHE IS DISABLED
007620  042737  000400  177734          BIC     #BIT08,KMCR                     ; READ VALID BITS FIRST
007626  032737  017000  177734          BIT     #BIT12!BIT11!BIT10!BIT9,KMCR    ; ALL VALID BITS CLEAR?
007634  001412                          BEQ     3$                              ; IF YES, BRANCH
007636  013737  177734  001126          MOV     KMCR,$BDDAT                     ; STORE KMCR
007644  013737  177734  001124          MOV     KMCR,$GDDAT                     ; STORE TO PRESERVE LOW BYTE
007652  042737  017000  001124          BIC     #BIT12!BIT11!BIT10!BIT9,$GDDAT  ; HIGH BYTE CLEAR
007660  104026                          ERROR   +26                             ; VALID BITS NOT CLEAR WITH KMCR<6>=0
007662  052737  000400  177734  3$:     BIS     #BIT08,KMCR                     ; NOW READ AVAILABILITY BITS
007670  013737  177734  001160          MOV     KMCR,$TMP0                      ; STORE KMCR
007676  122737  000177  001161          CMPB    #177,$TMP0+1                    ; ALL SET?
007704  001412                          BEQ     4$                              ; IF YES, BRANCH
007706  013737  177734  001126          MOV     KMCR,$BDDAT                     ; STORE KMCR
007714  013737  177734  001124          MOV     KMCR,$GDDAT                     ; STORE TO PRESERVE LOW BYTE
007722  152737  000177  001124          BISB    #177,$GDDAT                     ; HIGH BYTE SET
007730  104026                          ERROR   +26                             ; AVAIL. BITS NOT SET WITH KMCR<6>=0
007732  032737  000040  172516  4$:     BIT     #BIT05,MMR3                     ; DONE WITH RELOCATION ENABLED?
007740  001721                          BEQ     1$                              ; IF NOT, BRANCH
```

```
                        .SBTTL  TEST - AVAILABILITY OF SETS

                        ;* THIS TEST WILL CHECK KMCR<15-8>. FIRST, EACH SET IN THE CACHE IS ALLOCATED,
                        ;* THEN EACH SET IS MADE MOST RECENTLY USED (MRU), AND THEN EACH SET IS
                        ;* INVALIDATED BY READING THE 8TH REGISTER OF THE SET.
                        ;
                        ;  BGNTST
                        ;
                        ;       ENABLE CACHE
                        ;       POINT R2 TO VALTBL FOR VALID BITS (KMCR<8>=0)
                        ;       POINT R3 TO TOPTBL FOR AVAILABILITY (KMCR<8>=1)
                        ;       ENABLE RELOCATION
                        ;*
                        ;* ALLOCATE SETS IN CACHE MAKING EACH OF THEM VALID AND GETTING MISSES
                        ;*
                        ;       DO FOR ALL 4 SETS ALLOCATING THEM IN CACHE
                        ;       .  LET KMCR<8>=0
                        ;       .  DO DMA READ ON OCTAL BOUNDARY USING DIAGNOSTIC_DATA_IN
                        ;       .  IFB (R2)+ NE IN KMCR+1 THEN
                        ;       .  .  ERROR IN READING VALID BITS ON MISS
                        ;       .  ENDIF
                        ;       .  LET KMCR<8>=1
                        ;       .  IFB KMCR+1 NE (R3)+ THEN
                        ;       .  .  ERROR READING AVAILABILITY BITS ON MISS
                        ;       .  ENDIF
                        ;       ENDDO
                        ;*
                        ;* READ A REGISTER FROM A VALID SET MAKING EACH OF THE SETS MOST RECENTLY USED
                        ;*
                        ;       DO FOR ALL 4 SETS MAKING THEM MRU AND THEN MOST AVAILABLE
                        ;       .  LET KMCR<8>=0
                        ;       .  DO DMA READ FROM 3RD LOCATION USING DIAGNOSTIC_DATA_IN
                        ;       .  IFB KMCR+1 NE (R2)+ THEN
                        ;       .  .  ERROR IN READING VALID BITS ON HIT
                        ;       .  ENDIF
                        ;       .  LET KMCR<8>=1
                        ;       .  IFB KMCR+1 NE (R3)+ THEN
                        ;       .  .  ERROR READING AVAILABILITY BITS ON HIT
                        ;       .  ENDIF
                        ;*
                        ;* READ THE LAST REGISTER FROM A SET MAKING EACH SET INVALID AND MOST AVAILABLE
                        ;*
                        ;       .  LET KMCR<8>=0
                        ;       .  DO DMA READ FROM 8TH LOCATION USING DIAGNOSTIC_DATA_IN
                        ;       .  IFB KMCR+1 NE (R2)+ THEN
                        ;       .  .  ERROR IN READING VALID BITS INVALIDATING A SET
                        ;       .  ENDIF
                        ;       .  LET KMCR<8>=1
                        ;       .  IFB KMCR+1 NE (R3)+ THEN
                        ;       .  .  ERROR READING AVAILABILITY BITS INVALIDATING A SET
                        ;       .  ENDIF
                        ;       ENDDO
                        ;
                        ;  ENDTST
                        ;
                        ;-------------------------------------------------------------------
```

T24 AVAILABILITY OF SETS

```
2270                                ;;*****************************************************************
                                    ;*TEST 24       AVAILABILITY OF SETS
                                    ;;*****************************************************************
     007742  000004                 TST24:  SCOPE
2271
2272 007744  005737  001720                 TST     PMIS                    ; ANY PMI MEMORY?
2273 007750  001002                         BNE     20$                     ; IF SOME, DO THE TEST
2274 007752  000137  010354                 JMP     100$                    ; OTHERWISE EXIT
2275 007756  052737  000100  177734  20$:   BIS     #BIT06,KMCR             ; ENABLE CACHE
2276 007764  012702  010356                 MOV     #VALTBL,R2              ; POINT TO VALID BITS
2277 007770  012703  010372                 MOV     #TOPTBL,R3              ; POINT TO AVAILABILITY BITS
2278 007774  052737  000040  172516         BIS     #BIT05,MMR3             ; ENABLE RELOCATION
2279 010002  012737  000200  170200         MOV     #200,MAPL00             ; POINT MAP00 TO FIRST 8KB
2280 010010  012737  000000  170202         MOV     #0,MAPH00               ; IN HIGH AND LOW MAP REGISTERS
2281 010016  013737  177734  001124         MOV     KMCR,$GDDAT             ; PRESERVE LOW BYTE
2282 010024  105037  001125                 CLRB    $GDDAT+1                ; CLEAR TO REPORT ERRORS, IF ANY
2283                                ;
2284                                ; ALLOCATE 4 SETS BY READING THEM THRU DIAGNOSTIC_DATA_IN
2285                                ;
2286 010030  012701  000000                 MOV     #0,R1                   ; START READING MEMORY FROM 0
2287 010034  004737  002222         1$:     JSR     PC,DDIN                 ; . ALLOCATE IN CACHE NEXT 8 WORDS
2288 010040  042737  000400  177734         BIC     #BIT08,KMCR             ; . READ VALID BITS
2289 010046  013737  177734  001126         MOV     KMCR,$BDDAT             ; . STORE KMCR
2290 010054  122237  001127                 CMPB    (R2)+,$BDDAT+1          ; . VALID BITS OK?
2291 010060  001404                         BEQ     2$                      ; . IF YES, BRANCH
2292 010062  105742                         TSTB    -(R2)                   ; . GET THE PATTERN JUST WRITTEN
2293 010064  112237  001125                 MOVB    (R2)+,$GDDAT+1          ; . STORE HIGH BYTE
2294 010070  104026                         ERROR   +26                     ; . ERROR IN VALID BITS
2295 010072  052737  000400  177734  2$:    BIS     #BIT08,KMCR             ; . READ AVAILABILITY BIT
2296 010100  013737  177734  001126         MOV     KMCR,$BDDAT             ; . STORE KMCR
2297 010106  122337  001127                 CMPB    (R3)+,$BDDAT+1          ; . AVAILABILITY BITS OK?
2298 010112  001404                         BEQ     3$                      ; . IF YES, BRANCH
2299 010114  105743                         TSTB    -(R3)                   ; . GET THE PATTERN JUST WRITTEN
2300 010116  112337  001125                 MOVB    (R3)+,$GDDAT+1          ; . STORE HIGH BYTE
2301 010122  104026                         ERROR   +26                     ; . ERROR IN AVAILABILITY BITS
2302 010124  062701  000020         3$:     ADD     #20,R1                  ; . DO FOR NEXT OCTAL BOUNDARY
2303 010130  022701  000100                 CMP     #100,R1                 ; . ALL 4 DONE?
2304 010134  003337                         BGT     1$                      ; . IF NOT, BRANCH
2305                                ;
2306                                ; NOW READ 3RD AND THEN 8TH LOCATION FROM CACHE
2307                                ;
2308 010136  012701  000004                 MOV     #4,R1                   ; START WITH 3RD LOCATION A SET
2309 010142  042737  000400  177734  4$:     BIC     #BIT08,KMCR             ; . READ VALID BITS
2310 010150  004737  002222                 JSR     PC,DDIN                 ; . READ FROM CACHE
2311 010154  013737  177734  001126         MOV     KMCR,$BDDAT             ; . STORE KMCR
2312 010162  122237  001127                 CMPB    (R2)+,$BDDAT+1          ; . VALID BITS OK?
2313 010166  001404                         BEQ     5$                      ; . IF YES, BRANCH
2314 010170  105742                         TSTB    -(R2)                   ; . GET THE PATTERN JUST WRITTEN
2315 010172  112237  001125                 MOVB    (R2)+,$GDDAT+1          ; . STORE HIGH BYTE
2316 010176  104026                         ERROR   +26                     ; . ERROR IN VALID BITS
2317 010200  052737  000400  177734  5$:    BIS     #BIT08,KMCR             ; . READ AVAILABILITY BIT
2318 010206  013737  177734  001126         MOV     KMCR,$BDDAT             ; . STORE KMCR
2319 010214  122337  001127                 CMPB    (R3)+,$BDDAT+1          ; . AVAILABILITY BITS OK?
2320 010220  001404                         BEQ     6$                      ; . IF YES, BRANCH
2321 010222  105743                         TSTB    -(R3)                   ; . GET THE PATTERN JUST WRITTEN
2322 010224  112337  001125                 MOVB    (R3)+,$GDDAT+1          ; . STORE HIGH BYTE
2323 010230  104026                         ERROR   +26                     ; . ERROR IN AVAILABILITY BITS
```

T24 AVAILABILITY OF SETS

```
2324 010232  062701  000012          6$:   ADD   #12,R1                    ; . READ THE 8TH WORD
2325 010236  052737  000400  177734        BIS   #BIT08,KMCR               ; . READ AVAILABILITY BIT
2326 010244  004737  002222                JSR   PC,DDIN                   ; . READ FROM CACHE
2327 010250  013737  177734  001126        MOV   KMCR,$BDDAT               ; . STORE KMCR
2328 010256  122337  001127                CMPB  (R3)+,$BDDAT+1            ; . AVAILABILITY BITS OK?
2329 010262  001404                        BEQ   7$                        ; . IF YES, BRANCH
2330 010264  105743                        TSTB  -(R3)                     ; . GET THE PATTERN JUST WRITTEN
2331 010266  112337  001125                MOVB  (R3)+,$GDDAT+1            ; . STORE HIGH BYTE
2332 010272  104026                        ERROR +26                       ; . ERROR IN AVAILABILITY BITS
2333 010274  042737  000400  177734  7$:   BIC   #BIT08,KMCR               ; . READ VALID BITS
2334 010302  013737  177734  001126        MOV   KMCR,$BDDAT               ; . STORE KMCR
2335 010310  122237  001127                CMPB  (R2)+,$BDDAT+1            ; . VALID BITS OK?
2336 010314  001404                        BEQ   8$                        ; . IF YES, BRANCH
2337 010316  105742                        TSTB  -(R2)                     ; . GET THE PATTERN JUST WRITTEN
2338 010320  112237  001125                MOVB  (R2)+,$GDDAT+1            ; . STORE HIGH BYTE
2339 010324  104026                        ERROR +26                       ; . ERROR IN VALID BITS
2340 010326  062701  000006          8$:   ADD   #6,R1                     ; . DO FOR NEXT OCTAL BOUNDARY
2341 010332  022701  000100                CMP   #100,R1                   ; . ALL 4 DONE?
2342 010336  003301                        BGT   4$                        ; . IF NOT, BRANCH
2343 010340  042737  000040  172516        BIC   #BIT05,MMR3               ; DISABLE RELOCATION
2344 010346  042737  000100  177734        BIC   #BIT06,KMCR               ; DISABLE CACHE
2345 010354                          100$:
     010354  000414                        BR    TST25           ;; EXIT TEST
2346
2347                                ;* THIS TABLE HAS BITS KMCR<15-8> WHEN KMCR<8>=0
2348                                ;* KMCR<15>=0 FLAGGING CACHE MISS
2349
2350 010356      020     030     034  VALTBL: .BYTE  20,30,34,36             ; ORDER FOR VALID: A, AB, ABC, ABCD
     010361      036
2351                                                                        ; NOW ALL VALID, HITS-MISS(WRITE TO I/O)
2352 010362      236     016                .BYTE  236,16                   ; HIT-MISS, A: VALID - NOT VALID
2353 010364      216     006                .BYTE  216,6                    ; HIT-MISS, B: VALID - NOT VALID
2354 010366      206     002                .BYTE  206,2                    ; HIT-MISS, C: VALID - NOT VALID
2355 010370      202     000                .BYTE  202,0                    ; HIT-MISS, D: VALID - NOT VALID
2356
2357                                ;* THIS TABLE HAS BITS KMCR<15-8> WHEN KMCR<8>=1
2358                                ;* AT FIRST NO HITS ARE RECORDED, THE COMMENT FIELD SHOWS SETS
2359                                ;* STARTING WITH LEAST AVAILABLE
2360
2361 010372      017             TOPTBL: .BYTE  17                       ; ADCB - 000111
2362 010373      103                     .BYTE  103                      ; BADC - 100001
2363 010374      151                     .BYTE  151                      ; CBAD - 110100
2364 010375      177                     .BYTE  177                      ; DCBA - 111111
2365                                                                        ; NOW MISS-HITS
2366 010376      017     377             .BYTE  17,377                   ; ->ADCB->DCBA - 000111,111111
2367 010400      163     277             .BYTE  163,277                  ; ->BDCA->DCAB - 111001,011111
2368 010402      075     227             .BYTE  75,227                   ; ->CDAB->DABC - 011110,001011
2369 010404      027     201             .BYTE  27,201                   ; ->DABC->ABCD - 001011,000000
2370
2371                                .EVEN
```

```
                        .SBTTL TEST - DEALLOCATION OF SETS

                        ;* THIS TEST CHECKS THAT EACH SET CAN BE MADE MOST AVAILABLE AND
                        ;* DEALLOCATED ON THE NEXT READ ON THE OCTAL BOUNDARY.
                        ;
                        ;       DISABLE CACHE TO MAKE A MOST AVAILABLE
                        ;       ENABLE CACHE
                        ;*
                        ;* ALLOCATE SETS IN CACHE MAKING EACH OF THEM VALID AND GETTING MISSES
                        ;*
                        ;       DO FOR ALL 4 SETS ALLOCATING THEM IN CACHE
                        ;       .  DO DMA READ ON OCTAL BOUNDARY USING DIAGNOSTIC_DATA_IN
                        ;       ENDDO - THIS LEAVES A AS THE MOST AVAILABLE SET
                        ;*
                        ;* TRY TO BRING A NEW LOCATIONS TO SET A
                        ;*
                        ;       DO DMA READ ON OCTAL BOUNDARY FROM A FIFTH LOCATION TO SET A AGAIN
                        ;       LET KMCR<8> = #0
                        ;       IF KMCR<15!12!11!10!9!> NE <0,1,1,1,1> THEN
                        ;       .  ERROR DEALLOCATIONG LRU SET (A)
                        ;       ENDIF
                        ;       LET KMCR<8> = #1
                        ;       IF KMCR<15-9> NE <0,0,0,0,1,1,1> THEN
                        ;       .  ERROR DEALLOCATING LRU SET (A)
                        ;       ENDIF
                        ;*
                        ;* TRY TO BRING A NEW LOCATIONS TO SET B
                        ;*
                        ;       DO DMA READ ON OCTAL BOUNDARY FROM 1ST LOCATION TO SET B
                        ;       LET KMCR<8> = #0
                        ;       IF KMCR<15!12!11!10!9!> NE <0,1,1,1,1> THEN
                        ;       .  ERROR DEALLOCATIONG LRU SET (B)
                        ;       ENDIF
                        ;       LET KMCR<8> = #1
                        ;       IF KMCR<15-9> NE <0,1,0,0,0,0,1> THEN
                        ;       .  ERROR DEALLOCATING LRU SET (B)
                        ;       ENDIF
                        ;*
                        ;* TRY TO BRING A NEW LOCATIONS TO SET C
                        ;*
                        ;       DO DMA READ ON OCTAL BOUNDARY FROM 1ST LOCATION TO SET C
                        ;       LET KMCR<8> = #0
                        ;       IF KMCR<15!12!11!10!9!> NE <0,1,1,1,1> THEN
                        ;       .  ERROR DEALLOCATIONG LRU SET (C)
                        ;       ENDIF
                        ;       LET KMCR<8> = #1
                        ;       IF KMCR<15-9> NE <0,1,1,0,1,0,0> THEN
                        ;       .  ERROR DEALLOCATING LRU SET (C)
                        ;       ENDIF
                        ;*
                        ;* TRY TO BRING A NEW LOCATIONS TO SET D
                        ;*
                        ;       DO DMA READ ON OCTAL BOUNDARY FROM 1ST LOCATION TO SET D
                        ;       LET KMCR<8> = #0
                        ;       IF KMCR<15!12!11!10!9!> NE <0,1,1,1,1> THEN
                        ;       .  ERROR DEALLOCATIONG LRU SET (D)
                        ;       ENDIF
```

TEST - DEALLOCATION OF SETS

```
                                ;       LET KMCR<8> = #1
                                ;       IF KMCR<15-9> NE <0,1,1,1,1,1,1> THEN
                                ;       . ERROR DEALLOCATING LRU SET (D)
                                ;       ENDIF
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------------

                                ;;=============================================================
                                ;*TEST 25       DEALLOCATION OF SETS
                                ;;=============================================================
010406  000004                  TST25:  SCOPE

                                        TST     PMIS                    ; ANY PMI MEMORY?
010410  005737  001720                  BNE     20$                     ; IF SOME, DO THE TEST
010414  001002                          JMP     100$                    ; OTHERWISE, EXIT
010416  000137  011102
010422  052737  000040  172516  20$:    BIS     #BIT05,MMR3             ; ENABLE RELOCATION
010430  052737  000100  177734          BIS     #BIT06,KMCR             ; ENABLE CACHE
010436  012737  000200  170200          MOV     #200,MAPL00             ; POINT MAP00 TO FIRST 8KB
010444  012737  000000  170202          MOV     #0,MAPH00               ; IN HIGH AND LOW MAP REGISTERS
010452  013737  177734  001124          MOV     KMCR,$GDDAT             ; STORE LOW BYTE OF KMCR
                                ;
                                ; ALLOCATE ALL SETS MAKING A MOST AVAILABLE
                                ;
010460  012701  000000                  MOV     #0,R1                   ; START WITH 0 IN SET A
010464  004737  002222          1$:     JSR     PC,DDIN                 ; . ALLOCATE A SET
010470  062701  000020                  ADD     #20,R1                  ; . GET READY FOR NEXT SET
010474  022701  000100                  CMP     #100,R1                 ; . ALL 4 DONE?
010500  003371                          BGT     1$                      ; . IF NOT, BRANCH
                                ;
                                ; TRY TO BRING A NEW SET TO A
                                ;
010502  012701  000100                  MOV     #100,R1                 ; TRY TO DO DATI
010506  042737  000400  177734          BIC     #BIT08,KMCR             ; READ VALID BITS
010514  004737  002222                  JSR     PC,DDIN                 ; ALLOCATE IN CACHE
010520  013737  177734  001126          MOV     KMCR,$BDDAT             ; STORE KMCR
010526  122737  000036  001127          CMPB    #36,$BDDAT+1            ; ALL SETS VALID?
010534  001404                          BEQ     2$                      ; IF OK, BRANCH
010536  112737  000036  001125          MOVB    #36,$GDDAT+1            ; STORE EXPECTED PATTERN
010544  104026                          ERROR   +26                     ; DEALLOCATING A
010546  052737  000400  177734  2$:     BIS     #BIT08,KMCR             ; READ AVAILABILITY BITS
010554  013737  177734  001126          MOV     KMCR,$BDDAT             ; STORE KMCR
010562  122737  000017  001127          CMPB    #17,$BDDAT+1            ; MRU:A,D,C,B
010570  001404                          BEQ     3$                      ; IF OK, BRANCH
010572  112737  000017  001125          MOVB    #17,$GDDAT+1            ; STORE EXPECTED PATTERN
010600  104026                          ERROR   +26                     ; DEALLOCATING A
                                ;
                                ; TRY TO BRING A NEW SET TO B
                                ;
010602  012701  000120          3$:     MOV     #120,R1                 ; PREPARE FOR NEXT READ
010606  042737  000400  177734          BIC     #BIT08,KMCR             ; READ VALID BITS
010614  004737  002222                  JSR     PC,DDIN                 ; READ INTO CACHE
010620  013737  177734  001126          MOV     KMCR,$BDDAT             ; STORE KMCR
010626  122737  000036  001127          CMPB    #36,$BDDAT+1            ; ALL VALID?
010634  001404                          BEQ     4$                      ; IF YES, BRANCH
010636  112737  000036  001125          MOVB    #36,$GDDAT+1            ; STORE EXPECTED PATTERN
```

T25 DEALLOCATION OF SETS

```
2484 010644  104026                          ERROR   ;26                    ; DEALLOCATING B
2485 010646  052737  000400  177734  4$:     BIS     @#BIT08,KMCR           ; READ AVAILABILITY BITS
2486 010654  013737  177734  001126          MOV     KMCR,$BDDAT            ; STORE KMCR
2487 010662  122737  000103  001127          CMPB    #103,$BDDAT+1          ; MRU:B,A,D,C
2488 010670  001404                          BEQ     5$                     ; IF OK, BRANCH
2489 010672  112737  000103  001125          MOVB    #103,$GDDAT+1          ; STORE EXPECTED PATTERN
2490 010700  104026                          ERROR   ;26                    ; DEALLOCATING B
2491                                 ;
2492                                 ; TRY TO BRING A NEW SET TO C
2493                                 ;
2494 010702  012701  000140          5$:     MOV     #140,R1                ; PREPARE FOR NEXT READ
2495 010706  042737  000400  177734          BIC     @#BIT08,KMCR           ; READ VALID BITS
2496 010714  004737  002222                  JSR     PC,DDIN                ; READ INTO CACHE
2497 010720  013737  177734  001126          MOV     KMCR,$BDDAT            ; STORE KMCR
2498 010726  122737  000036  001127          CMPB    #36,$BDDAT+1           ; ALL VALID?
2499 010734  001404                          BEQ     6$                     ; IF YES, BRANCH
2500 010736  112737  000036  001125          MOVB    #36,$GDDAT+1           ; STORE EXPECTED PATTERN
2501 010744  104026                          ERROR   ;26                    ; DEALLOCATING C
2502 010746  052737  000400  177734  6$:     BIS     @#BIT08,KMCR           ; READ AVAILABILITY BITS
2503 010754  013737  177734  001126          MOV     KMCR,$BDDAT            ; STORE KMCR
2504 010762  122737  000151  001127          CMPB    #151,$BDDAT+1          ; MRU:C,B,A,D
2505 010770  001404                          BEQ     7$                     ; IF OK, BRANCH
2506 010772  112737  000151  001125          MOVB    #151,$GDDAT+1          ; STORE EXPECTED PATTERN
2507 011000  104026                          ERROR   ;26                    ; DEALLOCATING C
2508                                 ;
2509                                 ; TRY TO BRING A NEW SET TO D
2510                                 ;
2511 011002  012701  000160          7$:     MOV     #160,R1                ; PREPARE FOR NEXT READ
2512 011006  042737  000400  177734          BIC     @#BIT08,KMCR           ; READ VALID BITS
2513 011014  004737  002222                  JSR     PC,DDIN                ; READ INTO CACHE
2514 011020  013737  177734  001126          MOV     KMCR,$BDDAT            ; STORE KMCR
2515 011026  122737  000036  001127          CMPB    #36,$BDDAT+1           ; ALL VALID?
2516 011034  001404                          BEQ     8$                     ; IF YES, BRANCH
2517 011036  112737  000036  001125          MOVB    #36,$GDDAT+1           ; STORE EXPECTED PATTERN
2518 011044  104026                          ERROR   ;26                    ; DEALLOCATING D
2519 011046  052737  000400  177734  8$:     BIS     @#BIT08,KMCR           ; READ AVAILABILITY BITS
2520 011054  013737  177734  001126          MOV     KMCR,$BDDAT            ; STORE KMCR
2521 011062  122737  000177  001127          CMPB    #177,$BDDAT+1          ; MRU:D,C,B,A
2522 011070  001404                          BEQ     TST26           ;;     IF OK, EXIT TEST
2523 011072  112737  000177  001125          MOVB    #177,$GDDAT+1          ; STORE EXPECTED PATTERN
2524 011100  104026                          ERROR   ;26                    ; DEALLOCATING D
2525 011102                          100$:
2526
```

```
                                .SBTTL  TEST - CACHE WITH RELOCATION DISABLED

                                ;* THIS TEST CHECKS THAT CACHE IS NOT OPERATIONAL WHEN RELOCATION IS
                                ;* DISABLED AND KMCR<15-9> ARE NOT INITIALIZED.
                                ;
                                ;  BGNTST
                                ;
                                ;       SAVE KMCR<15-9> FOR KMCR<8>=0,1
                                ;       LET MMR3<5>=#0 TO DISABLE RELOCATION
                                ;       DO DMA READ USING DIAGNOSTIC_DATA_IN
                                ;       LET KMCR<8>=0
                                ;       IFB KMCR+1 NE SAVED THEN
                                ;       . ERROR RELOCATION DISABLED STILL AFFECTS THE CACHE
                                ;       ENDIF
                                ;       LET KMCR<8>=1
                                ;       IFB KMCR+1 NE SAVED THEN
                                ;       . ERROR RELOCATION DISABLED STILL AFFECTS THE CACHE
                                ;       ENDIF
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------------------

                                ;;*****************************************************************
                                ;*TEST 26       CACHE WITH RELOCATION DISABLED
                                ;;*****************************************************************
011102  000004                  TST26:  SCOPE

011104  005737  001720                  TST     PMIS                            ; ANY PMI MEMORY?
011110  001452                          BEQ     TST27           ;; IF NONE, EXIT TEST
011112  052737  000100  177734          BIS     #BIT06,KMCR                     ; MAKE SURE THAT CACHE IS ENABLED
011120  042737  000040  172516          BIC     #BIT05,MMR3                     ; DISABLE RELOCATION
011126  042737  000400  177734          BIC     #BIT08,KMCR                     ; SELECT VALID BITS
011134  042737  000400  177734          BIC     #BIT08,KMCR                     ; READ VALID BITS FIRST
011142  013737  177734  001124          MOV     KMCR,$GDDAT                     ; STORE LOW BYTE
011150  105037  001125                  CLRB    $GDDAT+1                        ; CLEAR HIGH BYTE
011154  032737  017000  177734          BIT     #BIT12!BIT11!BIT10!BIT9,KMCR    ; ALL VALID BITS CLEAR?
011162  001404                          BEQ     3$                              ; IF YES, BRANCH
011164  013737  177734  001126          MOV     KMCR,$BDDAT                     ; STORE KMCR
011172  104026                          ERROR   +26                             ; VALID BITS NOT CLEAR WITH KMCR<6>=0
011174  052737  000400  177734  3$:     BIS     #BIT08,KMCR                     ; NOW READ AVAILABILITY BITS
011202  013737  177734  001126          MOV     KMCR,$BDDAT                     ; STORE KMCR
011210  122737  000177  001127          CMPB    #177,$BDDAT+1                   ; ALL SET?
011216  001404                          BEQ     4$                              ; IF YES, EXIT TEST
011220  112737  000177  001125          MOVB    #177,$GDDAT+1                   ; EXPECTED PATTERN
011226  104026                          ERROR   +26                             ; AVAIL. BITS NOT SET WITH KMCR<6>=0
011230  042737  000100  177734  4$:     BIC     #BIT06,KMCR                     ; DISABLE CACHE
```

```
                                .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
                                ;       .  ENDIF
                                ;       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
                                ;       ENDIF
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------------------

                                ;;*****************************************************************
                                ;*TEST 27       WRITE CYCLES AND CACHE
                                ;;*****************************************************************
    011236  000004              TST27:  SCOPE

2601 011240  005737  001720             TST     PMIS                    ; ANY PMI MEMORY?
2602 011244  001002             BNE     20$                             ; IF SOME, DO THE TEST
2603 011246  000137  011704             JMP     100$                    ; OTHERWISE EXIT
2604 011252  052737  000100  177734 20$:  BIS     #BIT06,KMCR             ; CACHE STILL ENABLED
2605 011260  052737  000040  172516     BIS     #BIT05,MMR3             ; ENABLE RELOCATION
2606 011266  012737  000200  170200     MOV     #200,MAPL00             ; POINT MAP00 TO FIRST 8KB
2607 011274  012737  000000  170202     MOV     #0,MAPH00               ; IN HIGH AND LOW MAP REGISTERS
2608 011302  042737  000400  177734     BIC     #BIT08,KMCR             ; SELECT VALID BITS
2609 011310  013702  177734             MOV     KMCR,R2                 ; SAVE VALID BITS OF KMCR
2610 011314  052737  000400  177734     BIS     #BIT08,KMCR             ; SELECT AVAILABILITY BITS
2611 011322  013703  177734             MOV     KMCR,R3                 ; SAVE AVAILABILITY BITS
2612                            ;
2613                            ; ACCESS LOCATION NOT IN CACHE
2614                            ;
2615 011326  012701  000200             MOV     #200,R1                 ; ACCESS ADDRESS
2616 011332  013737  000400  001124     MOV     @#400,$GDDAT            ; THE PATTERN TO BE WRITTEN
2617 011340  004737  002210             JSR     PC,DDOUT                ; DO DIAGNOSTIC DATA OUT
2618 011344  042737  000400  177734     BIC     #BIT08,KMCR             ; SELECT VALID BITS
2619 011352  020237  177734             CMP     R2,KMCR                 ; KMCR CHANGED?
2620 011356  001406             BEQ     1$                              ; IF NOT, BRANCH
2621 011360  010237  001124             MOV     R2,$GDDAT               ; EXPECTED PATTERN
2622 011364  013737  177734  001126     MOV     KMCR,$BDDAT             ; RECIEVED PATTERN
2623 011372  104026             ERROR   #26                             ; WRITE MISSES AFFETS CACHE
2624 011374  052737  000400  177734 1$:  BIS     #BIT08,KMCR             ; SELECT AVAILABILITY BITS
2625 011402  020337  177734             CMP     R3,KMCR                 ; KMCR CHANGED?
2626 011406  001406             BEQ     2$                              ; IF OK, BRANCH
2627 011410  010337  001124             MOV     R3,$GDDAT               ; EXPECTED PATTERN
```

T27 WRITE CYCLES AND CACHE

```
2628 011414  013737  177734  001126          MOV     KMCR,$BDDAT                     ; RECIEVED PATTERN
2629 011422  104026                          ERROR   +26                             ; WRITE MISSES AFFETS CACHE
2630                                         ;
2631                                         ; CHECK THAT WRITE HIT INVALIDATES CACHE
2632                                         ;
2633 011424  042737  000100  177734  2$:     BIC     @BIT06,KMCR                     ; DISABLE CACHE TO GET TO KNOW STATE
2634 011432  052737  000100  177734          BIS     @BIT06,KMCR                     ; ENABLE CACHE
2635 011440  012701  000000                  MOV     @0,R1                           ; ADDRESS 0
2636 011444  004737  002222                  JSR     PC,DDIN                         ; ALLOCATE CACHE SET A
2637 011450  012701  000020                  MOV     @20,R1                          ; ADDRESS 20
2638 011454  004737  002222                  JSR     PC,DDIN                         ; AND ALLOCATE SET B
2639 011460  005001                          CLR     R1                              ; ACCESS SET A
2640 011462  013737  000200  001124          MOV     @@200,$GDDAT                    ; THE PATTERN TO BE WRITTEN
2641 011470  004737  002210                  JSR     PC,DDOUT                        ; INVALIDATE CACHE
2642 011474  032737  100000  177734          BIT     @BIT15,KMCR                     ; HIT?
2643 011502  001012                          BNE     3$                              ; IF YES, BRANCH
2644 011504  013737  177734  001124          MOV     KMCR,$GDDAT                     ; PRESERVE LOW BYTE IF ERROR
2645 011512  112737  000363  001124          MOVB    @363,$GDDAT                     ; HIT, B LEAST AVAILABLE
2646 011520  013737  177734  001126          MOV     KMCR,$BDDAT                     ; RECIEVED PATTERN
2647 011526  104026                          ERROR   +26                             ; WRITE HIT WAS NOT RECORDED
2648 011530  042737  000400  177734  3$:     BIC     @BIT08,KMCR                     ; READ VALID BITS
2649 011536  032737  010000  177734          BIT     @BIT12,KMCR                     ; A VALID?
2650 011544  001412                          BEQ     4$                              ; IF INVALIDATED, BRANCH
2651 011546  013737  177734  001124          MOV     KMCR,$GDDAT                     ; PRESERVE LOW BYTE IF ERROR
2652 011554  112737  000010  001125          MOVB    @BIT03,$GDDAT+1                 ; MISS, B VALID
2653 011562  013737  177734  001126          MOV     KMCR,$BDDAT                     ; RECIEVED PATTERN
2654 011570  104026                          ERROR   +26                             ; WRITE HIT DOES NOT INVALIDATE
2655 011572  013737  177734  001124  4$:     MOV     KMCR,$GDDAT                     ; PRESERVE LOW BYTE IF ERROR
2656 011600  112737  000200  001125          MOVB    @200,$GDDAT+1                   ; HIT RECIEVED
2657 011606  013737  000220  000220          MOV     @@220,@@220                     ; CPU WRITE TO SET B
2658 011614  032737  100000  177734          BIT     @BIT15,KMCR                     ; HIT?
2659 011622  001004                          BNE     5$                              ; IF YES, BRANCH
2660 011624  013737  177734  001126          MOV     KMCR,$BDDAT                     ; STORE KMCR
2661 011632  104026                          ERROR   +26                             ; HIT NOT RECORDED
2662 011634  032737  004000  177734  5$:     BIT     @BIT11,KMCR                     ; B STILL VALID?
2663 011642  001412                          BEQ     6$                              ; IF NOT, EXIT TEST
2664 011644  013737  177734  001124          MOV     KMCR,$GDDAT                     ; IN CASE HIT WENT AWAY
2665 011652  142737  000030  001125          BICB    @BIT04!BIT03,$GDDAT+1           ; CLEAR ALL EXTRA
2666 011660  013737  177734  001126          MOV     KMCR,$BDDAT                     ; RECIEVED PATTERN
2667 011666  104026                          ERROR   +26                             ; CPU WRITE DOES NOT INVALIDATE
2668 011670  042737  000100  177734  6$:     BIC     @BIT06,KMCR                     ; DISABLE CACHE
2669 011676  042737  000040  172516          BIC     @BIT05,MMR3                     ; AND MAPPING
2670 011704                          100$:
```

TEST - DMA READ WITH INDEX NOT ZERO

```
                                .SBTTL TEST - DMA READ WITH INDEX NOT ZERO

                                ;* THIS TEST CHECKS THAT A DMA READ MISS DOES NOT AFFECT THE CACHE IF THE INDEX
                                ;* FIELD IS NOT ZERO.
                                ;
                                ;  BGNTST
                                ;
                                ;       SAVE KMCR<15-9> FOR KMCR<8>=0,1
                                ;       DO DIAGNOSTIC_DATA_IN FOR A LOCATION WITH NON-ZERO INDEX
                                ;       DO FOR KMCR<8>=0,1
                                ;       .  IF KMCR<15-9> NE SAVED THEN
                                ;       .  .  ERROR IN COMPARING INDEXES
                                ;       .  ENDIF
                                ;       ENDDO
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------------------------

                                ;;*********************************************************************
                                ;*TEST 30       DMA READ WITH INDEX NOT ZERO
                                ;;*********************************************************************
011704  000004                  TST30:  SCOPE

011706  005737  001720                  TST     PMIS                    ; ANY PMI MEMORY?
011712  001474                  BEQ     TST31           ;; IF NONE, EXIT TEST

                                ;
                                ; ENABLE CACHE AND ALLOCATE SET A
                                ;
011714  052737  000040  172516          BIS     @BIT05,MMR3             ; ENABLE RELOCATION
011722  052737  000100  177734          BIS     @BIT06,KMCR             ; ENABLE CACHE
011730  012737  000200  170200          MOV     @200,MAPL00             ; POINT MAP00 TO FIRST 8KB
011736  012737  000000  170202          MOV     @0,MAPH00               ; IN HIGH AND LOW MAP REGISTERS
011744  012701  000000                  MOV     @0,R1                   ; ALLOCATE 200-216
011750  004737  002222                  JSR     PC,DDIN                 ; IN CACHE
                                ;
                                ; TRY TO CHECK WHETHER NON-ZERO INDEX EFFECT CACHE
                                ;
011754  042737  000400  177734          BIC     @BIT08,KMCR             ; SELECT VALID BITS
011762  013737  177734  001124          MOV     KMCR,$GDDAT             ; SAVE VALID BITS OF KMCR
011770  052737  000400  177734          BIS     @BIT08,KMCR             ; SELECT AVAILABILITY BITS
011776  013703  177734                  MOV     KMCR,R3                 ; SAVE AVAILABILITY BITS
012002  012701  000102                  MOV     @102,R1                 ; TRY CACHING 102
012006  004737  002222                  JSR     PC,DDIN                 ; DO DIAGNOSTIC DATA IN
012012  042737  000400  177734          BIC     @BIT08,KMCR             ; SELECT VALID BITS
012020  023737  001124  177734          CMP     $GDDAT,KMCR             ; KMCR CHANGED?
012026  001404                  BEQ     1$                      ; IF NOT, BRANCH
012030  013737  177734  001126          MOV     KMCR,$BDDAT             ; RECIEVED PATTERN
012036  104026                  ERROR   +26                     ; NON-ZERO INDEX AFFECTS CACHE
012040  052737  000400  177734  1$:     BIS     @BIT08,KMCR             ; SELECT AVAILABILITY BITS
012046  020337  177734                  CMP     R3,KMCR                 ; KMCR CHANGED?
012052  001406                  BEQ     2$                      ; IF NOT, EXIT TEST
012054  010337  001124                  MOV     R3,$GDDAT               ; EXPECTED DATA
012060  013737  177734  001126          MOV     KMCR,$BDDAT             ; RECIEVEED DATE
012066  104026                  ERROR   +26                     ; NON-ZERO INDEX AFFECTS CACHE
012070  042737  000100  177734  2$:     BIC     @BIT06,KMCR             ; DISABLE CACH
```

T30 DMA READ WITH INDEX NOT ZERO

```
 2726 012076  042737  000040  172516          BIC     #BIT05,MMR3                     ; AND MAPPING
 2727
```

```
                                .SBTTL TEST - TAG REGISTERS

                                ;* THIS TEST WILL AUTOSIZE MEMORY IN 128K WORDS. THEN DEPENDING
                                ;* ON THE AMOUNT OF MEMORY AVAILABLE, A PATTERN OF ALTERNATING 0'S
                                ;* AND 1'S WILL BE CONSTRUCTED TO VERIFY THE TAG REGISTERS.
                                ;
                                ;  BGNTST
                                ;
                                ;       SAVE TIMEOUT VECTOR
                                ;       ENABLE MMU
                                ;       LET R1 = KIPAR6
                                ;       REPEAT TO CONSTRUCT A MASK FOR NXM FOR MAPHO1
                                ;       .  LET R1 = R1 SHIFT.LEFT BY #1
                                ;       .  LET R1 = R1 + KIPAR6
                                ;       UNTIL CARRY SET
                                ;       SWAP R1
                                ;       LET R1 = R1 SHIFT.RIGHT BY #3 TO GET BITS<21-16> FROM <21-13>
                                ;       LET KMCR<6> = #1 TO ENABLE CACHE
                                ;       LET R2 = #TBLML FOR PATTERNS FOR LOW REGISTER
                                ;       LET R3 = #TBLMH FOR PATTERNS FOR HIGH REGISTER
                                ;       DO FOR ALL 16 PATTERNS
                                ;       .  DO FOR 4 PATTERNS
                                ;       .  .  LET MAPLO1 = (R2)+
                                ;       .  .  LET MAPHO1 = (R3)+
                                ;       .  .  LET MAPHO1 = MAPHO1 CLEAR.BY R1 NOT TO GET NXM
                                ;       .  .  DO DIAGNOSTIC_DATA_IN TO ALLOCATE TAG PATTERN
                                ;       .  ENDDO
                                ;       .  MOVE POINTERS THRU PATTERN TABLE TO PREVIOUS 4 WORDS
                                ;       .  ENABLE 22-BIT MMU
                                ;       .  DO FOR 4 PATTERNS
                                ;       .  .  LET MAPLO1 = (R2)+
                                ;       .  .  LET MAPHO1 = (R3)+
                                ;       .  .  LET MAPHO1 = MAPHO1 CLEAR.BY R1 NOT TO GET NXM
                                ;       .  .  DO DIAGNOTIC_DATA_IN FROM IN CACHE
                                ;       .  .  IF NOT A HIT THEN
                                ;       .  .  .  ERROR IN TAG REGISTERS
                                ;       .  ENDDO
                                ;       .  DISABLE 22-BIT MMU
                                ;       ENDDO
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------------------------------


                                ;;*****************************************************************************
                                ;*TEST 31       TAG REGISTERS
                                ;;*****************************************************************************
    012104  000004              TST31:  SCOPE

                                        TST     PMIS                            ; ANY PMI MEMORY?
012106  005737  001720                  BEQ     TST32           ;; IF NONE, EXIT TEST
012112  001575                          BIT     #BIT05,KMCR                     ; 18-BIT MODE?
012114  032737  000040  177734          BNE     TST32           ;; IF YES, EXIT TEST
012122  001171                          JSR     PC,MAPPR                        ; REMAP PROGRAM AREA
012124  004737  002244                  INC     MMR0                            ; ENABLE MMU
012130  005237  177572                  BIS     #BIT04,MMR3                     ; ENABLE 22-BIT
012134  052737  000020  172516          CMP     #40,PMIS                        ; IF 2M PRESENT, ONLY 21ST MASKED
012142  022737  000040  001720
```

```
                                        BEQ     5$                      ; DON'T CONSTRUCT MASK
                                ;
                                ; IF LESS THEN 2M OF MEMORY, MAKE UP A MASK FOR MAP HIGH REGISTERS
                                ;
                                        CLC                             ; CLEAR CARRY
                                        MOV     PMIS,R1                 ; SAVE KIPAR6
                                4$:     ROL     R1                      ; . ROTATE LEFT R1
                                        BIS     PMIS,R1                 ; . AND ADD BIT AT FIRST POSITION
                                        BMIS    4$                      ; . IF CARRY NOT SET, BRANCH
                                        SWAB    R1                      ; <15-8><--->&lt;7-0&gt;
                                        ROR     R1                      ; NOW GET ADDRESS BITS <21-16>
                                        ROR     R1                      ; FROM <21-14>
                                ;
                                ; ALLOCATE DIFFERENT PATTERNS IN TAG REGISTERS
                                ;
                                5$:     BIC     #BIT06,KMCR             ; DISABLE CACHE
                                        BIS     #BIT06,KMCR             ; ENABLE CACHE
                                        BIC     #BIT08,KMCR             ; MAKE SURE THAT VALID READ
                                        BIS     #BIT05,MMR3             ; ENABLE MAPPING
                                        MOV     #TBLML,R2               ; POINTER FOR LOW MAP PATTERNS
                                        MOV     #TBLMH,R3               ; POINTER FOR HIGH MAP PATTERNS
                                        MOV     R1,$TMP0                ; SAVE THE MASK
                                        MOV     #4,R4                   ; COUNTER FOR 16 PATTERNS
                                6$:     MOV     #20000,R1               ; . TO ACCESS THRU MAP REG. 1
                                        MOV     #4,R5                   ; . DO EACH FOR AT A TIME
                                7$:     MOV     (R2)+,MAPL01            ; . . LOAD LOW MAP
                                        MOV     (R3)+,MAPH01            ; . . LOAD HIGH MAP
                                        BIC     $TMP0,MAPH01            ; . . MASK OUT NXM
                                        JSR     PC,DDIN                 ; . . DO DIAGNOSTIC DATI
                                        SOB     R5,7$                   ; . . DO FOR ALL 4 PATTERNS
                                        MOV     KMCR,R1                 ; . STORE SETS VALID?
                                        COM     R1                      ; . COMPLEMENT VALID BITS
                                        BIT     #17000,R1               ; . THAT'S ALL VALID BITS
                                        BEQ     8$                      ; . IF ALL SET BEFORE COM, BRANCH
                                        ERROR   +27                     ; . NOT ALL VALID BITS SET
                                ;
                                ; NOW INVALIDATE BY WRITING TO THE SAME LOCATIONS
                                ;
                                8$:     SUB     #10,R2                  ; . MOVE POINTER TO -4
                                        SUB     #10,R3                  ; . MOVE HIGH MAP POINTER TOO
                                        MOV     #4,R5                   ; . VALIDATE ALLOCATED PATTERNS
                                        MOV     #20000,R1               ; . POINTER TO MAP 1
                                9$:     MOV     (R2)+,MAPL01            ; . . LOAD LOW MAP
                                        MOV     (R3)+,MAPH01            ; . . LOAD HIGH MAP
                                        BIC     $TMP0,MAPH01            ; . . MASK OUT NXM
                                        JSR     PC,DDIN                 ; . . DO DAIGN. DATA IN
                                        BIT     #BIT15,KMCR             ; . . HIT?
                                        BNE     10$                     ; . . IF HIT, BRANCH
                                        ERROR   +27                     ; . . IN TAG REGISTERS
                                10$:    SOB     R5,9$                   ; . . DO FOR ALL 4 PATTERNS
                                        SOB     R4,6$                   ; . REPEAT FOR ALL 16 PATTERNS
                                        CLR     MMR0                    ; DISABLE MMU
                                        BR      TST32           ;;      EXIT TEST

                                ;
                                ; PATTERNS FOR LOW MAP REGISTER
                                ;
```

T31 TAG REGISTERS

```
2840 012406 125240 052520 000000 TBLML:  .WORD   125240,52520,0,177760          ; FIRST PATTERN THRU 4 TAGS
     012414 177760
2841 012416 052520 000000 177760         .WORD   52520,0,177760,125240          ; SECOND PATTERN
     012424 125240
2842 012426 000000 177760 125240         .WORD   0,177760,125240,52520          ; THIRD
     012434 052520
2843 012436 177760 125240 052520         .WORD   177760,125240,52520,0          ; FOURTH
     012444 000000
2844                                     ;
2845                                     ; PATTERNS FOR HIGH MAP REGISTER
2846                                     ;
2847 012446 000052 000025 000000 TBLMH:  .WORD   52,25,0,77                     ; FIRST FOR HIGH MAP REGISTER
     012454 000077
2848 012456 000025 000000 000077         .WORD   25,0,77,52                     ; SECOND
     012464 000052
2849 012466 000000 000077 000052         .WORD   0,77,52,25                     ; THIRD
     012474 000025
2850 012476 000077 000052 000025         .WORD   77,52,25,0                     ; FOURTH
     012504 000000
```

```
                        .SBTTL  TEST - CACHE RAM BIT PATTERN TEST

                        ;* THIS TEST IS DONE TO CHECK THE OPERATION OF
                        ;* THE CACHE RAM . BIT PATTERNS 0,177777,52525
                        ;* AND 125252 ARE WRITTEN TO ALL OF THE MEMORY
                        ;* USED AND THEN VERIFIED
                        ;
                        ;  BGNTST
                        ;
                        ;*
                        ;* GO SET UP THE CACHE IMAGE TABLE
                        ;*
                        ;       LET R0 := #CTBLE
                        ;       REPEAT
                        ;       .  IF R0 POINTS TO OCTAL BOUNDARY ADDRESS THEN
                        ;       .  .  WE HAVE FOUND THE BEGINNING OF THE TABLE
                        ;       .  ELSE
                        ;       .  .  POINT R0 TO THE NEXT ADDRESS
                        ;       .  ENDIF
                        ;       UNTIL WE HAVE FOUND BEGINNING OF THE TABLE
                        ;*
                        ;* SET UP BOARD FOR THE TRANSFERS
                        ;*
                        ;       ENABLE UNIBUS MAPPING
                        ;       ENABLE DIAGNOSTIC MODE
                        ;       ENABLE CACHE
                        ;       DO FOR PATTERN := 0,177777,52525,125252
                        ;*
                        ;* INITIALIZE THE CACHE TABLE
                        ;*
                        ;       .  DO FOR ALL OF THE CACHE TABLE
                        ;       .  .  WRITE PATTERN TO TABLE
                        ;       .  ENDDO
                        ;*
                        ;* ALLOCATE ALL THE CACHE
                        ;*
                        ;       .  DO UNTIL ALL CACHE ALLOCATED
                        ;       .  .  DO A DIAGNOSTIC DATI NPR CYCLE (ON OCTAL BOUNDARY)
                        ;       .  ENDDO
                        ;*
                        ;* CHECK THAT THE DATA GOT CACHED CORRECTLY
                        ;*
                        ;       .  DO FOR ALL OF THE CACHE TABLE
                        ;       .  .  DO A DIAGNOSTIC DATI NPR CYCLE
                        ;       .  .  IF DDR NEQ PATTERN THEN
                        ;       .  .  .  ERROR IN CACHE RAM
                        ;       .  .  ENDIF
                        ;       .  ENDDO
                        ;       ENDDO
                        ;
                        ;  ENDTST
                        ;
                        ;-------------------------------------------------------------

                        ;;*****************************************************************
                        ;*TEST 32       CACHE RAM BIT TEST
                        ;;*****************************************************************
```

T32 CACHE RAM BIT TEST

```
        012506  000004                  TST32:  SCOPE
   2907 012510  005737  001720                  TST     PMIS            ; ANY PMI MEMORY?
   2908 012514  001503                          BEQ     TST33           ;; IF NONE, EXIT TEST
   2909                                         ;
   2910                                         ; FIND THE FIRST OCTAL BOUNDARY ADDRESS IN THE TABLE
   2911                                         ;
   2912 012516  012700  002022                  MOV     #CTBLE,R0       ; GET POINTER TO THE START OF THE SPACE ALLOCATED FOR THE CA
CHE TABLE
   2913 012522  010001                  5$:     MOV     R0,R1           ; SAVE IT IN R1 FOR WORKING
   2914 012524  042701  177760                  BIC     #177760,R1      ; MASK IN THE LEAST SIGNIFICANT DIGIT
   2915 012530  005701                          TST     R1              ; IS THE ADDRESS ON AN OCTAL BOUNDARY ?
   2916 012532  001402                          BEQ     10$             ; YES,THEN GO SET UP THE TABLE ADDRESSES
   2917 012534  005720                          TST     (R0)+           ; NO,THEN GET THE NEXT ADDRESS
   2918 012536  000771                          BR      5$              ; . GO CHECK IF IT FALLS ON AN OCTAL BOUNDARY
   2919 012540  012702  002140          10$:    MOV     #OBADR,R2       ; GET POINTER TO THE OCTAL BOUNDARY TABLE
   2920 012544  012703  000004                  MOV     #4,R3           ; SET UP THE LOOP COUNTER
   2921 012550  010022                  12$:    MOV     R0,(R2)+        ; . GO PUT ADDRESS INTO THE TABLE
   2922 012552  062700  000020                  ADD     #20,R0          ; . GET NEXT OCTAL BOUNDARY
   2923 012556  077304                          SOB     R3,12$          ; . FILL UP THE TABLE ?
   2924                                         ;
   2925                                         ; INITIALIZE THE BOARD FOR THE TRANSFERS
   2926                                         ;
   2927 012560  052737  000040  172516          BIS     #BIT5,@#MMR3    ; ENABLE UNIBUS MAPPING
   2928 012566  005037  170202                  CLR     MAPH00          ; CLEAR HIGH MAP REGISTER
   2929 012572  052737  000400  177730          BIS     #BIT8,@#DCSR    ; ENABLE DIAGNOSTIC MODE
   2930 012600  052737  000100  177734          BIS     #BIT6,KMCR      ; ENABLE THE CACHE
   2931 012606  012700  001774                  MOV     #PTRN16,R0      ; GET POINTER TO THE CACHE PATTERN TABLE
   2932 012612  013701  002140          15$:    MOV     OBADR,R1        ; GET POINTER TO THE CACHE IMAGE TABLE
   2933                                         ;
   2934                                         ; INITIALIZE THE CACHE IMAGE TABLE TO CURRENT PATTERN
   2935                                         ;
   2936 012616  012702  000040                  MOV     #40,R2          ; . SET UP LOOP COUNTER
   2937 012622  011021                  20$:    MOV     (R0),(R1)+      ; . . MOVE PATTERN TO TABLE
   2938 012624  077202                          SOB     R2,20$          ; . . HAVE WE DONE IT 40 TIMES ?
   2939                                         ;
   2940                                         ; ALLOCATE ALL THE CACHE MEMORY AVAILABLE
   2941                                         ;
   2942 012626  012702  002140                  MOV     #OBADR,R2       ; . GET ADDRESS OF OCTAL BOUNDARIRES WITHIN THE TAB;E
   2943 012632  012703  000004                  MOV     #4,R3           ; . SET UP LOOP COUNTER
   2944 012636  005001                          CLR     R1              ; . NON-MAPPED ADDRRESS=0
   2945 012640  012237  170200          25$:    MOV     (R2)+,MAPL00    ; . . SET UP MAP FOR DIAG NPR DATI CALL
   2946 012644  004737  002222                  JSR     PC,DDIN         ; . . GO DO A DIAGNOSTIC DATA IN CYCLE
   2947 012650  077305                          SOB     R3,25$          ; . . HAVE WE DONE IT 4 TIMES ?
   2948                                         ;
   2949                                         ; CHECK THE CACHE RAM IMAGE PATTERN
   2950                                         ;
   2951 012652  012703  000040                  MOV     #40,R3          ; . SET UP LOOP COUNTER
   2952 012656  013702  002140                  MOV     OBADR,R2        ; . GET ADDRESS OF CACHE IMAGE TABLE
   2953                                         ;   (THIS WHOLE TABLE SHOULD BE CACHED)
   2954 012662  005001                          CLR     R1              ; . USE ADDRESS 0
   2955 012664  010237  170200          30$:    MOV     R2,MAPL00       ; . . GET ADDRESS FPR DIAG. DATA IN CYCLE
   2956 012670  004737  002222                  JSR     PC,DDIN         ; . . GO DO A DIAGNOSTIC DATA IN
   2957 012674  032737  100000  177734          BIT     #BIT15,KMCR     ; . . HIT?
   2958 012702  001001                          BNE     35$             ; . . IF YES, BRANCH
   2959 012704  104030                          ERROR   +30             ; . . OTHERWISE, ERROR
   2960 012706  023722  177732          35$:    CMP     DDR,(R2)+       ; . . DID IT GET STORED CORRECTLY ?
   2961 012712  001401                          BEQ     40$             ; . . YES,THEN GO CHECK THE NEXT LOCATION
   2962 012714  104030                          ERROR   +30             ; . . NO,THEN ERROR IN CACHE RAM
```

J6

T32 CACHE RAM BIT TEST

```
2963 012716  077316                40$:    SOB     R3,30$          ; . . HAVE WE CHECKED ALL THE RAM ?
2964 012720  005720                        TST     (R0)+           ; . ALL PATTERNS DONE?
2965 012722  001333                        BNE     15$             ; . IF NOT HAVE DONE, BRANCH
2966
```

```
                        .SBTTL TEST - BOOT ROMS TEST

                        ;* THIS TEST CHECKS FOR THE PRESENCE OF BOOT ROMS.  IF THEY ARE FOUND,
                        ;* A CRC TEST IS PERFORMED FOR ALL THE ROM'S.
                        ;
                        ;  BGNTST
                        ;
                        ;       LET BCSR = BCSR SET.BY #BIT05 TO ENABLE ACCESSING OF ROM'S
                        ;       LET PCR = #0 TO ACCESS FIRST PAGE
                        ;       LET R4 = #173000 FOR STARTING ADDRESS
                        ;       LET $TMP0 = #0 TO CLEAR COUNT FOR EMPTY ROM'S
                        ;       DO FOR ALL 4 POSSIBLE ROMS
                        ;       .  IF (R4) EQ #161777 (EMPTY SOCKET) THEN
                        ;       .  .  INCREMENT $TMP0 TO COUNT EMPTY ROM'S
                        ;       .  .  IDENTIFY POSITION OF EMPTY SOCKET
                        ;       .  ELSE
                        ;       .  .  LET R1 = #0 TO COUNT BYTES IN ROMS
                        ;       .  .  DO FOR EACH LOCATION IN A ROM UNTILL R1=128
                        ;       .  .  .  IF LOCATION 24 ACCESSED THEN
                        ;       .  .  .  .  IF (R4) NE #173000 THEN
                        ;       .  .  .  .  .  ERROR IN LOCATION 24
                        ;       .  .  .  .  ENDIF
                        ;       .  .  .  .  INCREMENT R1 NOT TO DO BYTE 25
                        ;       .  .  .  ELSE
                        ;       .  .  .  .  CALCULATE CRC FOR THE BYTE
                        ;       .  .  .  ENDIF
                        ;       .  .  .  INCREMENT R1 FOR THE NEXT BYTE
                        ;       .  .  ENDDO
                        ;       ENDDO
                        ;       IF $TMP0 NE #4 NOT ALL EMPTY
                        ;       .  TRY TO WRITE TO @#173000 IN DIAGNOSTIC MODE AND OUT
                        ;       .  IF NO TIMEOUT THEN
                        ;       .  .  ERROR WRITE ACCESS TO ROMS DOES NOT TIMEOUT
                        ;       .  ENDIF
                        ;       ENDIF
                        ;
                        ;  ENDTST
                        ;
                        ;-------------------------------------------------------------------------
                        UBETST:
                        ;;*************************************************************************
                        ;*TEST 33       BOOT ROMS TEST
                        ;;*************************************************************************
    012724  000004      TST33:  SCOPE
                        ;
                        ; PREPARE TO DO CRC
                        ;
012726  042737  000400  177730          BIC     #BIT08,DCSR             ; LEAVE STANDALONE MODE
012734  052737  000200  177520          BIS     #BIT07,BCSR             ; TO ENABLE BOOT ROMS
012742  005037  177522                  CLR     PCR                     ; TO READ PAGE 0
012746  012704  173000                  MOV     #173000,R4              ; R4 PONTS TO FIRST ADDRESS
012752  005037  001160                  CLR     $TMP0                   ; CLEAR EMPTY SOCKET COUNT
                        ;
                        ; DO FOR EACH POSSIBLE ROM
                        ;
```

T33 BOOT ROMS TEST

```
3021 012756  022704  173776          1$:     CMP     #173776,R4              ; . ALL ROM'S DONE?
3022 012762  103475                          BLO     11$                     ; . IF SO, EXIT
3023 012764  005001                          CLR     R1                      ; . CLEAR COUNTER THRU A ROM
3024 012766  005005                          CLR     R5                      ; . INITIALIZE PARTIAL CRC
3025                                 ;
3026                                 ; CHECK FOR EMPTY SOCKETS AND EXIT IF SO
3027                                 ;
3028 012770  022714  161777                  CMP     #161777,(R4)            ; . EMPTY SOCKET?
3029 012774  001027                          BNE     3$                      ; . IF NOT, BRANCH TO DO CRC
3030 012776  005737  001206                  TST     $PASS                   ; . FIRST PASS?
3031 013002  001017                          BNE     2$                      ; . IF NOT, BRANCH
3032 013004  122737  000001  001220          CMPB    #APTENV,$ENV            ; IN APT MODE?
3033 013012  001413                          BEQ     2$                      ; IF SO, SKIP PRINTOUT
3034 013014  032737  000040  000052          BIT     #BIT05,@#52             ; . IN UFD MODE?
3035 013022  001007                          BNE     2$                      ; . IF IN UFD, BRANCH
3036 013024  010446                          MOV     R4,-(SP)                ; . PUT NUMBER TO TYPE OUT
3037 013026  104401  013312                  TYPE    ,NROM                   ; . TYPE ASCII MESSAGE
3038 013032  104403                          TYPOS                           ; . CALL TYPE OUT ROUTINE
3039 013034     006                          .BYTE   6                       ; . TYPE 6 DIGITS
3040 013035     000                          .BYTE   0                       ; . SUPRESS LEADING 0'S
3041 013036  104401  001175                  TYPE    ,$CRLF                  ; . CARRIAGE RETURN
3042 013042  005237  001160          2$:     INC     $TMP0                   ; . INCREMENT EMPTY SOCKET COUNT
3043 013046  062704  000200                  ADD     #200,R4                 ; . PREPARE TO DO NEXT ROM
3044 013052  000741                          BR      1$                      ; . BRANCH TO DO NEXT ROM
3045                                 ;
3046                                 ; IN EACH ROM CHECK LOCATION 24 AND DON'T DO CRC ON IT
3047                                 ;
3048 013054  022701  000024          3$:     CMP     #24,R1                  ; . LOCATION 24 ACCESSED?
3049 013060  001007                          BNE     5$                      ; . IF NO, GO DO CRC
3050 013062  022724  173000                  CMP     #173000,(R4)+           ; . PROPER DATA AT 24?
3051 013066  001401                          BEQ     4$                      ; . IF YES, BRANCH
3052 013070  104031                          ERROR   +31                     ; . WRONG DATA AT 24
3053 013072  005201                  4$:     INC     R1                      ; . INCREMENT FOR AN EXTRA BYTE
3054 013074  000137  013134                  JMP     8$                      ; . DO NOT DO CRC FOR 24
3055                                 ;
3056                                 ; DO CRC FOR EACH BYTE
3057                                 ;
3058 013100  112403                  5$:     MOVB    (R4)+,R3                ; . STORE CORRECT DATA IN R3
3059 013102  012702  000010                  MOV     #8.,R2                  ; . NUMBER OF BITS PER BYTE
3060 013106  000241                  6$:     CLC                             ; . . CLEAR CARRY
3061 013110  006005                          ROR     R5                      ; . . LOW BIT PARTIAL TO CARRY
3062 013112  006003                          ROR     R3                      ; . . CARRY TO BYTE AND BYTE TO CARRY
3063 013114  102006                          BVC     7$                      ; . . XOR OF PARIAL AND BYTE LOW BITS
3064 013116  012746  120001                  MOV     #POLY,-(SP)             ; . . XOR POLY TO PARTIAL (4 INSTRUCTIONS)
3065 013122  040516                          BIC     R5,(SP)                 ; . . NOT PARTIAL AND POLY
3066 013124  042705  120001                  BIC     #POLY,R5                ; . . NOT POLY AND PARTIAL
3067 013130  052605                          BIS     (SP)+,R5                ; . . POLY XOR PARTIAL
3068 013132  077213                  7$:     SOB     R2,6$                   ; . . DECREMENT BIT COUNT AND CONTINUE
3069 013134  005201                  8$:     INC     R1                      ; . COUNT BYTES
3070                                 ;
3071                                 ; IF A ROM DONE, CHECK FOR 0 IN R5
3072                                 ;
3073 013136  022701  000200                  CMP     #128.,R1                ; . ALL 64 WORDS DONE?
3074 013142  003344                          BGT     3$                      ; . IF NOT, BRANCH
3075 013144  005705                          TST     R5                      ; . IF YES, CRC 0?
3076 013146  001401                          BEQ     10$                     ; . IF CRC = 0, BRANCH
3077 013150  104031                          ERROR   +31                     ; . CRC FOR A ROM NOT EQUAL TO 0
```

```
3078 013152  000137  012756          10$:   JMP     1$                      ; . DO FOR NEXT ROM
3079                                 ;
3080                                 ; TRY TO WRITE TO GET A TIMEOUT
3081                                 ;
3082 013156  013737  000004  001160  11$:   MOV     ERRVEC,$TMP0            ; SAVE TIMEOUT VECTOR
3083 013164  012737  013202  000004        MOV     #15$,ERRVEC             ; POINT NEW TO PROGRAM
3084 013172  005037  173000          14$:   CLR     @#173000                ; TRY TO WRITE TO 1ST PAGE OF ROM
3085 013176  104031                         ERROR   +31                     ; WRITE ACCESS TO ROM'S DIDN'T TIMEOUT
3086 013200  000402                         BR      16$                     ;
3087 013202  005726                  15$:   TST     (SP)+                   ; RESTORE STACK
3088 013204  005726                         TST     (SP)+                   ;
3089 013206  032737  000400  177730  16$:   BIT     #BIT08,DCSR             ; OUT OF STANDALONE MODE?
3090 013214  001013                         BNE     20$                     ; IF NOT, EXIT TEST
3091 013216  013702  177734                 MOV     KMCR,R2                 ; MAKE SURE THAT
3092 013222  042702  177700                 BIC     #177700,R2              ; AT LEAST SOME OF
3093 013226  022702  000077                 CMP     #77,R2                  ; MEMORY PMI
3094 013232  001404                         BEQ     20$                     ; IF NOT, BRANCH
3095 013234  052737  000400  177730        BIS     #BIT08,DCSR             ; DO 1 MORE TIME IN STANDALONE MODE
3096 013242  000753                         BR      14$                     ;
3097 013244  052737  000010  177730  20$:   BIS     #BIT03,DCSR             ; DISABLE UBA ROM RESPONSE
3098 013252  012737  013270  000004        MOV     #25$,ERRVEC             ; POINT NEW TIMEOUT VECTOR
3099 013260  005737  173000                 TST     @#173000                ; READ BOOT ROM
3100 013264  104037                         ERROR   +37                     ; DSCR<3> DIDN'T DISABLE UBA ROM
3101 013266  000402                         BR      26$                     ;
3102 013270  005726                  25$:   TST     (SP)+                   ; RESTORE STACK
3103 013272  005726                         TST     (SP)+
3104 013274  042737  000410  177730  26$:   BIC     #BIT08!BIT03,DCSR       ; LEAVE STANDALONE MODE
3105 013302  013737  001160  000004        MOV     $TMP0,ERRVEC            ; RESTORE STACK
3106 013310  000417                         BR      TST34           ;; EXIT TEST
3107
3108 013312      101      104      104  NROM:  .ASCIZ  /ADDRESS OF EMPTY UBA SOCKET /
     013315      122      105      123
     013320      123      040      117
     013323      106      040      105
     013326      115      120      124
     013331      131      040      125
     013334      102      101      040
     013337      123      117      103
     013342      113      105      124
     013345      040      000
3109                                 .EVEN
```

```
                                .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
                                ;       ENDIF
                                ;       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
                                ;       .  ENDIF
                                ;       ENDIF
                                ;       LET MMR0<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
                                ;       .  .  ENDIF
                                ;       .  .  LET (R4) = COMPLEMENT <(R4)>
                                ;       .  .  IF (R4) NE #52525 THEN
                                ;       .  .  .  ERROR IN UNIBUS MEMORY
                                ;       .  .  ENDIF
                                ;       .  ENDDO
                                ;       ENDDO
                                ;       DISABLE MMU
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------------------------

                                ;;*********************************************************************
                                ;*TEST 34        UNIBUS MEMORY TEST
                                ;;*********************************************************************
013350  000004                  TST34:  SCOPE
                                ;
                                ; CHECK IF ALL NON-UNIBUS MEMORY
                                ;
013352  032737  000037  177734          BIT     #37,KMCR                ; BITS <5-0> CLEAR?
013360  001473                        BEQ     TST35           ;;      IF YES, SKIP TEST
                                ;
                                ; CHECK IF ALL UNIBUS MEMORY
                                ;
013362  013701  177734                MOV     KMCR,R1                 ; SAVE KMCR
```

T34 UNIBUS MEMORY TEST

```
3165 013366  042701  177700          BIC     #177700,R1          ; LEAVE ONLY <5-0>
3166 013372  022701  000077          CMP     #77,R1              ; BITS <5-0> ALL SET?
3167 013376  001005                  BNE     1$                  ; IF NOT, BRANCH
3168 013400  012701  001600          MOV     #1600,R1            ; DON'T TEST FIRST 32K
3169 013404  012702  007600          MOV     #7600,R2            ; 248KB MAXIMUM CONFIGURATION
3170 013410  000420                  BR      2$                  ; GO DO TEST
3171 013412  004737  002424  1$:     JSR     PC,UMSIZ            ; SIZE UNIBUS MEMORY
3172                                 ;
3173                                 ; ON RETURN R2 HAS PAR VALUE FOR UNIBUS MEMORY
3174                                 ;
3175 013416  010201                  MOV     R2,R1               ; STORE LOWER BOUNDARY
3176 013420  012702  007600          MOV     #7600,R2            ; HIGHER BOUNDARY
3177 013424  032737  000040  177734  BIT     #BIT05,KMCR         ; 18 BIT MODE?
3178 013432  001007                  BNE     2$                  ; IF YES, GO DO TEST
3179 013434  052701  170000          BIS     #170000,R1          ; EXTEND TO 22 BITS
3180 013440  052702  170000          BIS     #170000,R2          ; FOR HIGHER BOUNDARY TOO
3181 013444  052737  000020  172516  BIS     #BIT04,MMR3         ; ENABLE 22 BIT MAPPING
3182                                 ;
3183                                 ; NOW WRITE MEMORY WITH 01 PATTERN, VERIFY IT AND DO THE SAME FOR 10 PATTERN
3184                                 ;
3185 013452  004737  002244  2$:     JSR     PC,MAPPR            ; REMAP PROGRAM TO FIRST 32K
3186 013456  005237  177572          INC     MMR0                ; ENABLE MMU
3187 013462  010137  172354          MOV     R1,KIPAR6           ; START AT LOWER BOUNDARY
3188 013466  012704  140000  3$:     MOV     #140000,R4          ; . START 4K PAGE
3189 013472  012714  125252  4$:     MOV     #125252,(R4)        ; . . WRITE FIRST PATTERN
3190 013476  022714  125252          CMP     #125252,(R4)        ; . . WRITEN OK?
3191 013502  001401                  BEQ     5$                  ; . . IF OK, BRANCH
3192 013504  104034                  ERROR   +34                 ; . . IN UNIBUS MEMORY
3193 013506  005114  5$:     COM     (R4)                        ; . . COMPLEMENT INITIAL PATTERN
3194 013510  022714  052525          CMP     #052525,(R4)        ; . . NEW PATTERN OK?
3195 013514  001401                  BEQ     6$                  ; . . IF OK, BRANCH
3196 013516  104034                  ERROR   +34                 ; . . IN UNIBUS MEMORY
3197 013520  005724  6$:     TST     (R4)+                       ; . . GET NEXT MEMORY LOCATION
3198 013522  022704  160000          CMP     #160000,R4          ; . . LAST ONE IN 4K PAGE?
3199 013526  101361                  BHI     4$                  ; . . IF NOT, BRANCH
3200 013530  062737  000200  172354  ADD     #200,KIPAR6         ; . GET NEXT PAGE
3201 013536  020237  172354          CMP     R2,KIPAR6           ; . LAST PAGE DONE?
3202 013542  101351                  BHI     3$                  ; . IF NOT, BRANCH
3203 013544  005037  177572          CLR     MMR0                ; DISABLE MMU
3204
3205                                 .SBTTL UBE TESTS FOR THE UNIBUS ADAPTER BOARD
3206
3207                                 ;*
3208                                 ;* THE FOLLOWING TESTS REQUIRE THE USE OF THE UNIBUS EXERCISER
3209                                 ;* TO TEST OUT THE UNIBUS ADAPTER BOARD.SOME TESTS REQUIRE ONE
3210                                 ;* UBE OTHER TESTS REQUIRE TWO UBE'S.
3211                                 ;*
3212                                 ;* THE PROGRAM WILL AUTOSIZE TO SEE HOW MANY UBE'S ARE IN THE
3213                                 ;* SYSTEM. DEPENDING ON THE RESULTS OF THE AUTOSIZING CERTAIN
3214                                 ;* TESTS WILL BE SELECTED OR DESELECTED.
3215                                 ;*
3216                                 ;-----------------------------------------------------------------
3217
```

```
                                .SBTTL TEST - UBE AUTOSIZING ROUTINE
                                ;*
                                ;* THE FOLLOWING ROUTINE IS USED TO AUTOSIZE THE NUMBER OF
                                ;* UBE'S PRESENT IN THE SYSTEM.
                                ;*
                                ;
                                ;  BGNTST
                                ;
                                ;       LET R0 := #170000               ;1ST UBE ADDRESS
                                ;       LET R2 := #510                  ;1ST UBE VECTOR
                                ;       LET R1 := #BE1DB                ;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 PSW LOCATION
                                ;       .  .  LET (R3)+ := R2           ;GET POINTER TO VECTOR PSW
                                ;       .  .  LET R2 := R2 + 2          ;GET NEXT VECTOR ADDRESS
                                ;       .  .  IF WE HAVE FOUND TWO UBE'S THEN
                                ;       .  .  .  EXIT TEST
                                ;       .  .  ENDIF
                                ;       .  ENDIF
                                ;       ENDDO
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------------------------


                                ;;*********************************************************************
                                ;*TEST 35       UNIBUS EXERCISER AUTOSIZING ROUTINE
                                ;;*********************************************************************
    013550  000004              TST35:  SCOPE

                                        BIC     #BIT08,DCSR     ; MAKE SURE THAT OUT OF DIAGN. MODE
                                        BIC     #BIT05,MMR3     ; MAPPING DISABLED
                                        TST     @#$PASS         ; IS THIS THE FIRST PASS ?
                                        BNE     10$             ; NO,THEN NO NEED TO SIZE AGAIN
```

T35 UNIBUS EXERCISER AUTOSIZING ROUTINE

```
                                        ;
                                        ; INITIALIZE POINTERS FOR AUTO-SIZING
                                        ;
013574  012700  170000                  MOV     #170000,R0      ; GET ADDRESS OF FIRST POSSIBLE UBE
013600  012702  000510                  MOV     #510,R2         ; GET VECTOR OF FIRST POSSIBLE UBE
013604  012701  002150                  MOV     #8E1DB,R1       ; TABLE HEADER FOR 1ST UBE
013610  012737  013622  000004          MOV     #1$,@#4         ; SET UP TIMEOUT VECTOR
013616  005710                  100$:   TST     (R0)            ; . IS THERE A UBE HERE ?
013620  000412                          BR      2$              ; . . YES,THEN BRANCH AROUND TOMEOUT ROUTINE
                                        ;
                                        ; TIMEOUT ROUTINE
                                        ;
013622  062706  000004          1$:     ADD     #4,SP           ; . . READJUST THE STACK
013626  020027  170160                  CMP     R0,#170160      ; . . HAVE WE CHECKED ALL THE ADDRESSES
013632  001431                          BEQ     10$             ; . . YES THEN GO SEE IF WE FOUND ANY UBE'S
013634  062700  000020                  ADD     #20,R0          ; . . NO,THEN GET THE NEXT UBE ADDRESS
013640  062702  000004                  ADD     #4,R2           ; . . GET THE NEXT UBE VECTOR
013644  000764                          BR      100$            ; . . GO SEE IF THE NEXT UBE IS THERE
                                        ;
                                        ; ASSIGN UBE ADDRESSES TO THE CURRENT UBE TABLE
                                        ;
013646  012704  000005          2$:     MOV     #5,R4           ; . SET UP LOOP COUNTER
013652  010021                  3$:     MOV     R0,(R1)+        ; . . ASSIGN AN ADDRESS TO THE POINTER
013654  005720                          TST     (R0)+           ; . . GET THE NEXT ADDRESS
013656  077403                          SOB     R4,3$           ; . . ARE WE DONE ? NO THEN GO GET NEXT ADDRESS
013660  062700  000004                  ADD     #4,R0           ; . POINT R0 TO LAST UBE ADDRESS
013664  010021                          MOV     R0,(R1)+        ; . PUT THE ADDRESS INTO THE POINTER TABLE
013666  005720                          TST     (R0)+           ; . POINT R0 TO NEXT UBE ADDRESS

                                        ;
                                        ; ASSIGN UBE VECTORS TO CURRENT UBE TABLE
                                        ;

013670  010221                          MOV     R2,(R1)+        ; . GET POINTER TO VECTOR ADDRESS
013672  005722                          TST     (R2)+           ; . GET THE VECTOR PSW LOCATION
013674  010221                          MOV     R2,(R1)+        ; . GET POINTER TO VECTOR PSW
013676  005722                          TST     (R2)+           ; . GET THE NEXT UBE'S VECTOR ADDRESS
013700  005237  001722                  INC     UBECT           ; . FLAG WE HAVE FOUND ANOTHER UBE

                                        ;
                                        ; SEE IF WE HAVE FOUND TWO UBE'S
                                        ;

013704  022737  000002  001722          CMP     #2,UBECT        ; . HAVE WE FOUND TWO UBE'S
013712  001401                          BEQ     10$             ; . GO DO THE UBE TESTS
013714  000740                          BR      100$

                                        ;
                                        ;  PROGRAM WILL CHECK IF ANY UBE'S WERE FOUND
                                        ;

013716  012737  002234  000004  10$:     MOV     #TIMOUT,@#4     ; RESTORE TIMEOUT VECTOR
013724  005737  001722                  TST     UBECT           ; HAVE ANY UBE'S BEEN FOUND ?
013730  001002                          BNE     TST36           ;; GO DO THE SELECTED TESTS FOR 1 UBE
013732  000137  022454                  JMP     UBEM            ; SKIP ALL THE UBE TESTS
```

```
                                .SBTTL TEST - NPG ARBITRATION
                                ;
                                ;* THIS TEST CHECKS THAT A NPG CAN BE GRANTED AT ANY PRIORITY OF THE CPU.
                                ;*
                                ;
                                ;  BGNTST
                                ;
                                ;       DO FOR PSW FROM #340 DOWNTO #0 BY #40
                                ;       .  LET @BE1CC = -1 TO DO 1 CYCLE
                                ;       .  LET @BE1BA = #$TMP0 FOR THE ADDRESS
                                ;       .  LET @BE1CR1 = #2041 TO DO 1 DATI
                                ;       .  WAIT FOR @BE1CR1<7>=1 READY
                                ;       .  IF @BE1CC EQ -1 THEN
                                ;       .  .  ERROR NPG DIDN'T HAPPEN
                                ;       .  ENDIF
                                ;       ENDDO
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------------

                                ;;*****************************************************************
                                ;*TEST 36       NPG ARBITRATION
                                ;;*****************************************************************
013736  000004                  TST36:  SCOPE

013740  004737  002450                  JSR     PC,IUBE                 ; INITIALIZE THE UBE
013744  012737  000340  177776          MOV     #340,PSW                ; STORE PRIORITY 7
013752  012777  177777  166172  1$:     MOV     #-1,@BE1CC              ; . DO FOR 1 CYCLE
013760  012777  001160  166166          MOV     #$TMP0,@BE1BA           ; . STORE ADDRESS FOR DATI
013766  012777  002041  166162          MOV     #2041,@BE1CR1           ; . DO 1 DATI AT NPG
013774  105777  166156          2$:     TSTB    @BE1CR1                 ; . . READY ON?
014000  100375                          BPL     2$                      ; . . WAIT FOR READY
014002  022777  177777  166142          CMP     #-1,@BE1CC              ; . CYCLE COUNT INCREMENTED?
014010  001001                          BNE     3$                      ; . IF YES, BRANCH
014012  104032                          ERROR   +32                     ; . IF NO, ERROR IN NPG
014014  162737  000040  177776  3$:     SUB     #40,PSW                 ; . DECREMENT PRIORITY
014022  022737  000000  177776          CMP     #0,PSW                  ; . LAST ONE?
014030  003750                          BLE     1$                      ; . IF NOT, BRANCH
014032  012737  000340  177776          MOV     #340,PSW                ; RESTORE PRIORITY
```

```
3370                            .SBTTL TEST - NO BUS GRANTS WITH PROCESSOR AT HIGHER PRIORITY
3371                            ;
3372                            ;* THIS TEST CHECKS THAT NO BUS REQUESTS ARE GOING TO BE HONORED
3373                            ;* WHEN THE PROCESSOR HAS HIGHER PRIORITY THAN THE REQUESTING DEVICE.
3374                            ;*
3375                            ;
3376                            ;  BGNTST
3377                            ;
3378                            ;       SET UP @BE1VEC TO POINT TO ERROR_CHECK_ROUTINE
3379                            ;       SET UP @BE1PSW TO #340
3380                            ;*
3381                            ;* TRY TO DO BR7 WITH CPU PRIORITY AT 7
3382                            ;*
3383                            ;       LET @BE1CR1 = #21 TO DO 1 DATI
3384                            ;       LET PSW = #340 TO SET PRIORITY AT 7
3385                            ;       DO "NOP"
3386                            ;       IF INTERUPT THEN
3387                            ;       .  ERROR BG7 GRANTED WITH PROCESSOR AT HIGHER PRIORITY
3388                            ;       ENDIF
3389                            ;
3390                            ;*
3391                            ;* TRY TO DO BR6 WITH CPU PRIORITY AT 7 - 6
3392                            ;*
3393                            ;       DO FOR R3 FROM #340 DOWNTO #300 BY #40
3394                            ;       .  LET @BE1CR1 = #11 TO DO 1 DATI
3395                            ;       .  LET PSW = R3 TO CHANGE PRIORITY
3396                            ;       .  DO "NOP"
3397                            ;       .  IF INTERRUPT  THEN
3398                            ;       .  .  ERROR BG6 GRANTED WITH PROCESSOR AT HIGHER PRIORITY
3399                            ;       .  ENDIF
3400                            ;       ENDDO
3401                            ;       LET PSW = #340 FOR THE NEXT PART
3402                            ;*
3403                            ;* TRY TO DO BR5 WITH CPU PRIORITY AT 7 - 5
3404                            ;*
3405                            ;       DO FOR R3 FROM #340 DOWNTO #240 BY #40
3406                            ;       .  LET @BE1CR1 = #5 TO DO 1 DATI
3407                            ;       .  LET PSW = R3 TO CHANGE PRIORITY
3408                            ;       .  DO "NOP"
3409                            ;       .  IF INTERRUPT THEN
3410                            ;       .  .  ERROR BG5 GRANTED WITH PROCESSOR AT HIGHER PRIORITY
3411                            ;       .  ENDIF
3412                            ;       ENDDO
3413                            ;       LET PSW = #340
3414                            ;*
3415                            ;* TRY TO DO BR4 WITH CPU PRIORITY AT 7 - 4
3416                            ;*
3417                            ;       DO FOR R3 FROM #340 DOWNTO #200 BY #40
3418                            ;       .  LET @BE1CR1 = #2003 TO DO 1 DATI
3419                            ;       .  LET PSW = R3 TO CHANGE PRIORITY
3420                            ;       .  DO "NOP"
3421                            ;       .  IF INTERRUPT THEN
3422                            ;       .  .  ERROR BR4 GRANTED WITH PROCESSOR AT HIGHER PRIORITY
3423                            ;       .  ENDIF
3424                            ;       ENDDO
3425                            ;
3426                            ;  ENDTST
```

TEST - NO BUS GRANTS WITH PROCESSOR AT HIGHER PRIORITY

```
                                        ;
                                        ;-------------------------------------------------------------
                                        ;
                                        ;;*************************************************************
                                        ;*TEST 37       NO BUS GRANTS WITH PROCESSOR AT HIGHER PRIORITY
                                        ;;*************************************************************
014040  000004                          TST37:  SCOPE
                                        ;
014042  004737  002450                          JSR     PC,IUBE                 ; INITIALIZE THE UBE
014046  012737  000340  177776            MOV     #340,PSW                ; SET CPU PRIORITY AT 7
014054  012777  000340  166104            MOV     #340,@#BE1PSW           ; AT PRIORITY 7
                                        ;
                                        ; TRY TO DO BR7 WITH CPU PRIORITY AT 7
                                        ;
014062  012777  014102  166074            MOV     #1$,@#BE1VEC            ; POINT UBE VECTOR TO PROGRAM
014070  012777  000021  166060            MOV     #21,@#BE1CR1            ; BR7 WITH FUNO
014076  000240                            NOP                             ; JUST IN CASE
014100  000405                            BR      2$                      ; BRANCH AROUND IF NO INTERRUPT
014102  005077  166050            1$:     CLR     @#BE1CR1                ; CLEAR ANY OTHER REQUESTS
014106  062706  000004                    ADD     #4,SP                   ; ADJUST STACK POINTER
014112  104032                            ERROR   +32                     ; BG7 GRANTED WITH CPU AT 7
                                        ;
                                        ; TRY TO DO BR6 WITH CPU PRIORITY AT 7-6
                                        ;
014114  012777  014144  166042    2$:     MOV     #4$,@#BE1VEC            ; POINT UBE VECTOR TO PROGRAM
014122  012703  000340                    MOV     #340,R3                 ; START WITH PRIORITY AT 7
014126  012777  000011  166022    3$:     MOV     #11,@#BE1CR1            ; . BR6 WITH FUNO
014134  010337  177776                    MOV     R3,PSW                  ; . CHANGE PRIORITY
014140  000240                            NOP                             ; . JUST IN CASE OF INTERRUPS
014142  000405                            BR      5$                      ; . IF NO INTERRUPS, BRANCH
014144  005077  166006            4$:     CLR     @#BE1CR1                ; . CLEAR ANY OTHER REQUESTS
014150  062706  000004                    ADD     #4,SP                   ; . ADJUST STACK POINTER
014154  104032                            ERROR   +32                     ; . BG6 GRANTED WITH CPU AT 6-7
014156  162703  000040            5$:     SUB     #40,R3                  ; . LOWER PRIORITY
014162  020327  000300                    CMP     R3,#300                 ; . LAST ONE TO CHECK?
014166  002357                            BGE     3$                      ; . IF NOT, BRANCH
                                        ;
                                        ; TRY TO DO BR5 WITH CPU PRIORITY AT 7-5
                                        ;
014170  012777  014220  165766            MOV     #7$,@#BE1VEC            ; POINT UBE VECTOR TO PROGRAM
014176  012703  000340                    MOV     #340,R3                 ; START WITH PRIORITY AT 7
014202  012777  000005  165746    6$:     MOV     #5,@#BE1CR1             ; . BR5 WITH FUNO
014210  010337  177776                    MOV     R3,PSW                  ; . CHANGE PRIORITY
014214  000240                            NOP                             ; . JUST IN CASE OF INTERRUPS
014216  000405                            BR      8$                      ; . IF NO INTERRUPS, BRANCH
014220  005077  165732            7$:     CLR     @#BE1CR1                ; . CLEAR ANY OTHER REQUESTS
014224  062706  000004                    ADD     #4,SP                   ; . ADJUST STACK POINTER
014230  104032                            ERROR   +32                     ; . BG5 GRANTED WITH CPU AT 5-7
014232  162703  000040            8$:     SUB     #40,R3                  ; . LOWER PRIORITY
014236  020327  000240                    CMP     R3,#240                 ; . LAST ONE TO CHECK?
014242  002357                            BGE     6$                      ; . IF NOT, BRANCH
                                        ;
                                        ; TRY TO DO BR4 WITH CPU PRIORITY AT 7-4
                                        ;
014244  012777  014274  165712            MOV     #10$,@#BE1VEC           ; POINT UBE VECTOR TO PROGRAM
014252  012703  000340                    MOV     #340,R3                 ; START WITH PRIORITY AT 7
014256  012777  000003  165672    9$:     MOV     #3,@#BE1CR1             ; . BR6 WITH FUNO
```

T37 NO BUS GRANTS WITH PROCESSOR AT HIGHER PRIORITY

```
 3481 014264  010337  177776              MOV     R3,PSW          ; . CHANGE PRIORITY
 3482 014270  000240                      NOP                     ; . JUST IN CASE OF INTERRUPS
 3483 014272  000405                      BR      11$             ; . IF NO INTERRUPS, BRANCH
 3484 014274  005077  165656        10$:  CLR     @BE1CR1         ; . CLEAR ANY OTHER REQUESTS
 3485 014300  062706  000004              ADD     #4,SP           ; . ADJUST STACK POINTER
 3486 014304  104032                      ERROR   +32             ; . BG4 GRANTED WITH CPU AT 4-7
 3487 014306  162703  000040        11$:  SUB     #40,R3          ; . LOWER PRIORITY
 3488 014312  020327  000200              CMP     R3,#200         ; . LAST ONE TO CHECK?
 3489 014316  002357                      BGE     9$              ; . IF NOT, BRANCH
 3490
 3491
```

```
3493                            .SBTTL TEST - BR7-BR4 ARBITRATION
3494
3495                            ;* THIS TEST CHECKS THAT BR7-BR4 INTERRUPTS CAN OCCUR WITH A PROCESSOR
3496                            ;* PRIORITY AT A LOWER LEVEL
3497                            ;*
3498                            ;
3499                            ;  BGNTST
3500                            ;
3501                            ;       SET UP @BE1PSW TO #340
3502                            ;*
3503                            ;* TRY TO DO BR7 WITH CPU PRIORITY AT 6 - 0
3504                            ;*
3505                            ;       DO FOR R3 FROM #300 DOWNTO #0 BY #40
3506                            ;       .  LET PSW = #340
3507                            ;       .  LET @BE1CR1 = #21 TO DO 1 DATI
3508                            ;       .  LET PSW = R3 TO CHANGE PRIORITY
3509                            ;       .  DO "NOP"
3510                            ;       .  IF NO INTERRUPT THEN
3511                            ;       .  .  ERROR BG7 DOESN'T  OCCUR
3512                            ;       .  ENDIF
3513                            ;       ENDDO
3514                            ;*
3515                            ;* TRY TO DO BR6 WITH CPU PRIORITY AT 5 - 0
3516                            ;*
3517                            ;       DO FOR R3 FROM #240 DOWNTO #0 BY #40
3518                            ;       .  LET PSW = #340
3519                            ;       .  LET @BE1CR1 = #11 TO DO 1 DATI
3520                            ;       .  LET PSW = R3 TO CHANGE PRIORITY
3521                            ;       .  DO "NOP"
3522                            ;       .  IF NO INTERUPT THEN
3523                            ;       .  .  ERROR BG6 DOESN'T  OCCUR
3524                            ;       .  ENDIF
3525                            ;       ENDDO
3526                            ;*
3527                            ;* TRY TO DO BR5 WITH CPU PRIORITY AT 4 - 0
3528                            ;*
3529                            ;       DO FOR R3 FROM #200 DOWNTO #0 BY #40
3530                            ;       .  LET PSW = #340
3531                            ;       .  LET @BE1CR1 = #5 TO DO 1 DATI
3532                            ;       .  LET PSW = R3 TO CHANGE PRIORITY
3533                            ;       .  DO "NOP"
3534                            ;       .  IF NO INTERRUPT THEN
3535                            ;       .  .  ERROR BG5 DOESN'T  OCCUR
3536                            ;       .  ENDIF
3537                            ;       ENDDO
3538                            ;*
3539                            ;* TRY TO DO BR4 WITH CPU PRIORITY AT 3 - 0
3540                            ;*
3541                            ;       DO FOR R3 FROM #140 DOWNTO #0 BY #40
3542                            ;       .  LET PSW = #340
3543                            ;       .  LET @BE1CR1 = #3 TO DO 1 DATI
3544                            ;       .  LET PSW = R3 TO CHANGE PRIORITY
3545                            ;       .  DO "NOP"
3546                            ;       .  IF NO INTERRUPT THEN
3547                            ;       .  .  ERROR BG4 DOESN'T  OCCUR
3548                            ;       .  ENDIF
3549                            ;       ENDDO
```

TEST - BR7-BR4 ARBITRATION

```
                                        ;
                                        ;  ENDTST
                                        ;
                                        ;-------------------------------------------------------------
                                        ;
                                        ;;*************************************************************
                                        ;*TEST 40       BR7-BR4 ARBITRATION
                                        ;;*************************************************************
     014320  000004                     TST40:  SCOPE
                                        ;
014322  004737  002450                          JSR     PC,IUBE                 ; INITIALIZE THE UBE
014326  012777  000340  165632                  MOV     #340,@BE1PSW            ; AT PRIORITY 7
                                        ;
                                        ; TRY TO DO BR7 WITH CPU PRIORITY AT 6-0
                                        ;
014334  012777  014400  165622                  MOV     #2$,@BE1VEC             ; POINT UBE VECTOR TO PROGRAM
014342  012703  000300                          MOV     #300,R3                 ; START WITH PRIORITY AT 6
014346  012737  000340  177776  1$:     MOV     #340,PSW                ; . RAISE PRIORITY TO 7
014354  012777  000021  165574                  MOV     #21,@BE1CR1             ; . BR7 WITH FUNO
014362  010337  177776                          MOV     R3,PSW                  ; . LOWER PRIORITY
014366  000240                                  NOP                             ; . JUST IN CASE OF INTERRUPS
014370  005077  165562                          CLR     @BE1CR1                 ; . CLEAR ANY OTHER REQUESTS
014374  104032                                  ERROR   +32                     ; . BR7 NOT GRANTED WITH CPU AT 6-0
014376  000402                                  BR      3$                      ; . DON'T ADJUST STACK IF NO INTERRUPT
014400  062706  000004          2$:     ADD     #4,SP                   ; . ADJUST STACK POINTER
014404  162703  000040          3$:     SUB     #40,R3                  ; . LOWER PRIORITY
014410  022703  000000                          CMP     #0,R3                   ; . LAST ONE TO CHECK?
014414  002354                                  BGE     1$                      ; . IF NOT, BRANCH
                                        ;
                                        ; TRY TO DO BR6 WITH CPU PRIORITY AT 5-0
                                        ;
014416  004737  002450                          JSR     PC,IUBE                 ; INITIALIZE THE UBE
014422  012777  014466  165534                  MOV     #5$,@BE1VEC             ; POINT UBE VECTOR TO PROGRAM
014430  012703  000240                          MOV     #240,R3                 ; START WITH PRIORITY AT 5
014434  012737  000340  177776  4$:     MOV     #340,PSW                ; . RAISE PRIORITY TO 7
014442  012777  000011  165506                  MOV     #11,@BE1CR1             ; . BR6 WITH FUNO
014450  010337  177776                          MOV     R3,PSW                  ; . LOWER PRIORITY
014454  000240                                  NOP                             ; . JUST IN CASE OF INTERRUPS
014456  005077  165474                          CLR     @BE1CR1                 ; . CLEAR ANY OTHER REQUESTS
014462  104032                                  ERROR   +32                     ; . BR6 NO1 GRANTED WITH CPU AT 5-0
014464  000402                                  BR      6$                      ; . DON'T ADJUST STACK
014466  062706  000004          5$:     ADD     #4,SP                   ; . ADJUST STACK POINTER
014472  162703  000040          6$:     SUB     #40,R3                  ; . LOWER PRIORITY
014476  022703  000000                          CMP     #0,R3                   ; . LAST ONE TO CHECK?
014502  002354                                  BGE     4$                      ; . IF NOT, BRANCH
                                        ;
                                        ; TRY TO DO BR5 WITH CPU PRIORITY AT 4-0
                                        ;
014504  004737  002450                          JSR     PC,IUBE                 ; INITIALIZE THE UBE
014510  012777  014554  165446                  MOV     #8$,@BE1VEC             ; POINT UBE VECTOR TO PROGRAM
014516  012703  000200                          MOV     #200,R3                 ; START WITH PRIORITY AT 4
014522  012737  000340  177776  7$:     MOV     #340,PSW                ; . RAISE PRIORITY TO 7
014530  012777  000005  165420                  MOV     #5,@BE1CR1              ; . BR5 WITH FUNO
014536  010337  177776                          MOV     R3,PSW                  ; . LOWER PRIORITY
014542  000240                                  NOP                             ; . JUST IN CASE OF INTERRUPS
014544  005077  165406                          CLR     @BE1CR1                 ; . CLEAR ANY OTHER REQUESTS
014550  104032                                  ERROR   +32                     ; . BR5 NOT GRANTED WITH CPU AT 4-0
```

T40 BR7-BR4 ARBITRATION

```
3604 014552  000402                        BR      9$                      ; . DON'T TOUCH STACK
3605 014554  062706  000004          8$:   ADD     #4,SP                   ; . ADJUST STACK POINTER
3606 014560  162703  000040          9$:   SUB     #40,R3                  ; . LOWER PRIORITY
3607 014564  022703  000000                CMP     #0,R3                   ; . LAST ONE TO CHECK?
3608 014570  002354                        BGE     7$                      ; . IF NOT, BRANCH
3609                                     ;
3610                                     ; TRY TO DO BR4 WITH CPU PRIORITY AT 3-0
3611                                     ;
3612 014572  004737  002450                JSR     PC,IUBE                 ; INITIALIZE THE UBE
3613 014576  012777  014642  165360        MOV     #11$,@BE1VEC            ; POINT UBE VECTOR TO PROGRAM
3614 014604  012703  000140                MOV     #140,R3                 ; START WITH PRIORITY AT 3
3615 014610  012737  000340  177776  10$:   MOV     #340,PSW                ; . RAISE PRIORITY TO 7
3616 014616  012777  000003  165332        MOV     #3,@BE1CR1              ; . BR4 WITH FUNO
3617 014624  010337  177776                MOV     R3,PSW                  ; . LOWER PRIORITY
3618 014630  000240                        NOP                             ; . JUST IN CASE OF INTERRUPS
3619 014632  005077  165320                CLR     @BE1CR1                 ; . CLEAR ANY OTHER REQUESTS
3620 014636  104032                        ERROR   +32                     ; . BR4 NOT GRANTED WITH CPU AT 3-0
3621 014640  000402                        BR      12$                     ; . DON'T ADJUST STACK
3622 014642  062706  000004          11$:  ADD     #4,SP                   ; . ADJUST STACK POINTER
3623 014646  162703  000040          12$:  SUB     #40,R3                  ; . LOWER PRIORITY
3624 014652  022703  000000                CMP     #0,R3                   ; . LAST ONE TO CHECK?
3625 014656  002354                        BGE     10$                     ; . IF NOT, BRANCH
3626
3627
```

TEST - ARBITRATION BETWEEN INTERRUPTS AND PIRQ'S

```
                                .SBTTL TEST - ARBITRATION BETWEEN INTERRUPTS AND PIRQ'S

                                ;* THIS TEST CHECKS THAT PIRQ REQUESTS OVERRIDE INTERRUPT REQUESTS
                                ;* OF THE SAME LEVEL.
                                ;*
                                ;       SET UP @BE1VEC AND PIRQVEC AND PRIORITY FOR BOTH
                                ;       LET R1 = @20 FOR BG7 FROM UBE
                                ;       LET R2 = @100000 FOR PIRQ7
                                ;       DO FOR R3 FROM @300 DOWNTO @140 BY @40
                                ;       . LET PSW = @340
                                ;       . LET @BE1CR1 = R1 FOR INTERRUPT LEVEL
                                ;       . LET @BE1CR1 = @BE1CR1 SET.BY @1 TO DO 1 DATI
                                ;       . LET PIRQ = PIRQ SET.BY R2 FOR PIRQ LEVEL
                                ;       . LET PSW = R3 TO CHANGE PRIORITY
                                ;       . WAIT FOR INTERRUPT
                                ;       . IF PIRQ INTERRUPT DIDN'T HAPPEN OR @BE1CC NE -1 THEN
                                ;       . . ERROR IN ARBITRATION BETWEEN PIRQ'S AND INTERRUPTS
                                ;       . ENDIF
                                ;       . LET R1 = R1 SHIFT RIGHT 1
                                ;       . LET R2 = R2 SHIFT RIGHT 1
                                ;       ENDDO
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------------

                                ;;*****************************************************************
                                ;*TEST 41       ARBITRATION BETWEEN INTERRUPTS AND PIRQ'S
                                ;;*****************************************************************
    014660  000004              TST41:  SCOPE
3656 014662  004737  002450             JSR     PC,IUBE                 ; INITIALIZE THE UBE
3657 014666  012777  014756  165262     MOV     @2$,@BE1CR1             ; POINT UBE VECTOR TO PROGRAM
3658 014674  012777  000340  165264     MOV     @340,@BE1PSW            ; AT PRIORITY 7
3659 014702  012737  014766  000240     MOV     @3$,PIRQVEC             ; POINT PIRQ VECTOR TO PROGRAM
3660 014710  012737  000340  000242     MOV     @340,@@242              ; AT PRIORITY 7
3661 014716  012701  000020             MOV     @20,R1                  ; STORE FOR BR7
3662 014722  012702  100000             MOV     @100000,R2              ; STORE FOR PIRQ 7
3663 014726  012703  000300             MOV     @300,R3                 ; START WITH PRIORITY 6
3664                                    ;
3665                                    ; DO ARBITRATION FOR LEVEL 7-4
3666                                    ;
3667 014732  012737  000340  177776 1$:        MOV     @340,PSW                ; . START WITH PRIORITY 7
3668 014740  010177  165212             MOV     R1,@BE1CR1              ; . CHANGE UBE PRIORITY
3669 014744  010237  177772             MOV     R2,PIRQ                 ; . CHANGE PIRQ PRIORITY
3670 014750  010337  177776             MOV     R3,PSW                  ; . LOWER CPU PRIORITY
3671 014754  000001             WAIT                    ; . WAIT FOR INTERRUPT
3672 014756  062706  000004     2$:     ADD     @4,SP                   ; . CLEAN UP STACK
3673 014762  104032             ERROR   +32             ; . PIRQ'S DON'T TAKE PRIORITY
3674 014764  000402             BR      4$              ; . DON'T CLEAN UP STACK TWICE
3675 014766  062706  000004     3$:     ADD     @4,SP                   ; . CLEAN UP STACK
3676 014772  006001             4$:     ROR     R1                      ; . ADJUST BR FOR UBE
3677 014774  006002             ROR     R2                      ; . ADJUST PIRQ'S
3678 014776  162703  000040             SUB     @40,R3                  ; . LOWER FOR CPU PRIORITY
3679 015002  022703  000140             CMP     @140,R3                 ; . PRIORITY 3?
3680 015006  001351             BNE     1$                      ; . BRANCH IF NOT YET
3681 015010  005037  177772             CLR     PIRQ                    ; CLEAR ANY REQUESTS
```

```
                                .SBTTL TEST - ARBITRATION BETWEEN INTERRUPTS FROM 2 UBE

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


                                ;;*****************************************************************
                                ;*TEST 42       ARBITRATION BETWECN INTERRUPTS FROM 2 UBE
                                ;;*****************************************************************
    015014  000004              TST42:  SCOPE

                                ;
                                ; INITIAL SETUP
                                ;
                                ;
                                ; ARBITRATE BETWEEN 2 INTERRUPTS OF DIFFERENT LEVEL
                                ;
```

```
                                        CMP     #2,UBECT                ; 2 UBE'S?
                                        BNE     TST43           ;;      EXIT, IF NO
                                        JSR     PC,IUBE                 ; INITIALIZE THE UBE

                                        MOV     #240,R3                 ; START AT PRIORITY 6
                                        MOV     #20,R1                  ; UBE AT LEVEL 7
                                        MOV     #BE1SV,@BE1VEC          ; POINT UBE1 VECTOR TO PROGRAM
                                        MOV     #340,@BE1PSW            ; AT PRIORITY 7
                                        MOV     #BE2SV,@BE2VEC          ; POINT UBE2 VECTOR TO PROGRAM
                                        MOV     #340,@BE2PSW            ; AT PRIORITY 7

                                1$:     MOV     #340,PSW                ; . RAISE PRIORITY TO 7
                                        MOV     R1,@BE1CR1              ; . HIGHER PRIORITY TO UBE1
                                        ROR     R1                      ; . GET LOWER PRIORITY
                                        MOV     R1,@BE2CR1              ; . LOWER PRIORITY TO UBE2
                                        BIS     #BIT00,@BE1CR1          ; . SET GO BIT OF UBE #1
```

T42 ARBITRATION BETWEEN INTERRUPTS FROM 2 UBE

```
3737 015120  052777  000001  165050          BIS     #BIT00,@BE2CR1                  ; . SET GO BIT OF UBE #2
3738 015126  010337  177776                  MOV     R3,PSW                          ; . LOWER CPU PRIORITY
3739 015132  000240                          NOP                                     ; . GIVE UBE TIME TO INTERRUPT
3740 015134  000240                          NOP                                     ;
3741 015136  000240                          NOP                                     ;
3742 015140  005737  001724                  TST     BE1INT                          ; . DID UBE1 INTERRUPT
3743 015144  001002                          BNE     5$                              ; . YES,THEN GO MAKE SURE UBE2 DIDN'T
3744 015146  104032                          ERROR   +32                             ; . LOWER ORDER INTERRUPTS HAPPNE BEFORE
3745 015150  000404                          BR      10$                             ; . GO SEE IF WE ARE DONE
3746 015152  005737  001726          5$:     TST     BE2INT                          ; . DID UBE2 INTERRUPT
3747 015156  001401                          BEQ     10$                             ; . NO,THEN GO SEE IF WE ARE DONE
3748 015160  104032                          ERROR   +32                             ; . YES,THEN ERROR IN ARBITRATION
3749 015162  005037  001724          10$:    CLR     BE1INT                          ; . INITIALIZE INTERRUPT FLAGS
3750 015166  005037  001726                  CLR     BE2INT                          ;
3751 015172  162703  000040                  SUB     #40,R3                          ; . DO THE NEXT LEVEL
3752 015176  022703  000100                  CMP     #100,R3                         ; . CPU AT 2 (LAST ONE)?
3753 015202  001333                          BNE     1$                              ; . BRANCH IF NOT YET
3754 015204  005077  164770                  CLR     @BE2CLR                         ; CLEAR ERRORS ON 2ND UBE
3755 015210  000420                          BR      TST43           ;;      GO TO NEXT TEST
3756
3757 015212  012777  000000  164756  BE1SV:  MOV     #0,@BE2CR1                      ; CLEAR PENDING UBE #2 INTERRUPTS
3758 015220  005237  001724                  INC     BE1INT                          ; SET BE1 INTERRUPT FLAG
3759 015224  005037  001726                  CLR     BE2INT                          ; CLEAR BE2 INTERRUPT FLAG
3760 015230  000002                          RTI
3761
3762 015232  012777  000000  164716  BE2SV:  MOV     #0,@BE1CR1                      ; CLEAR PENDING UBE #1 INTERRUPTS
3763 015240  005237  001726                  INC     BE2INT                          ; SET BE2 INTERRUPT FLAG
3764 015244  005037  001724                  CLR     BE1INT                          ; CLEAR BE1 INTERRUPT FLAG
3765 015250  000002                          RTI
3766
```

```
                                .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 1HRU 24/26) IN THE EAROM ON CPU BOARD.
                                ;*
                                ;
                                ;  BGNTST
                                ;
                                ;       LET BCSR<6,5>=<0,1> TO ACCESS EAROM
                                ;       IF @#165002<7,6> NE <1,1> FOR POWER UP CODE 00 THEN
                                ;       .  EXIT TEST
                                ;       ENDIF
                                ;       SAVE PWRVEC
                                ;       POINT PWRVEC TO PROGAM AREA
                                ;       LET @BE1CR2<4> = #1
                                ;       IF NO TRAP TO 24 THEN
                                ;       .  ERROR EXECUTING POWER DOWN THRU UNIBUS
                                ;       ENDIF
                                ;       LET @BE1CR2<4> = #0 FOR POWER UP
                                ;       RESTORE PWRVEC
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------------

                                ;;*************************************************************
                                ;*TEST 43       POWER DOWN TEST
                                ;;*************************************************************
    015252  000004              TST43:  SCOPE
                                ;
                                ; RESTRICTIONS ON RUNNING THIS TEST
                                ;
015254  005737  001206                  TST     $PASS                   ; FIRST PASS
015260  001102                  BNE     TST44           ;;IF NOT 1ST PASS, DON'T DO IT
015262  032737  000040  000052  BIT     #BIT05,@#52                     ; UFD MODE?
015270  001076                  BNE     TST44           ;;EXIT TEST, IF SO
015272  122737  000001  001220  CMPB    @APTENV,$ENV                    ; APT?
015300  001472                  BEQ     TST44           ;;EXIT TEST IF APT
015302  032737  000010  177524  BIT     #BIT03,@#177524                 ; FORCED CONSOLE MODE?
015310  001066                  BNE     TST44           ;;IF YES, EXIT TEST
015312  042737  000100  177520  BIC     #BIT06,BCSR                     ; ENABLE ACCESS THRU 165000
015320  052737  000040  177520  BIS     #BIT05,BCSR                     ; READ EAROM
015326  032737  000100  165000  BIT     #BIT06,@#165000                 ; BATTERY OVERRIDE?
015334  001454                  BEQ     TST44           ;;IF SO, EXIT TEST
015336  032737  000200  165002  BIT     #BIT07,@#165002                 ; POWER UP CODE 00?
015344  001450                  BEQ     TST44           ;;EXIT TEST, IF NOT
015346  032737  000100  165002  BIT     #BIT06,@#165002                 ; POWER UP CODE 00?
015354  001444                  BEQ     TST44           ;;EXIT TEST, IF NOT
015356  004737  002450          JSR     PC,IUBE                         ; INITIALIZE THE UBE
                                ;
                                ; DO POWER DOWN SEQUENCE
                                ;
015362  013737  000024  001160  MOV     PWRVEC,$TMP0                    ; STORE POWER DOWN VECTOR
015370  012737  015422  000024  MOV     #1$,PWRVEC                      ; POINT NEW ONE TO PROGRAM
015376  013737  000026  000026  MOV     PWRVEC+2,@#26                   ; AT PRIORITY 7
015404  052777  000020  164550  BIS     #BIT04,@BE1CR2                  ; START POWER DOWN
```

T43 POWER DOWN TEST

```
 3822 015412  000240                          NOP
 3823 015414  000240                          NOP                     ; WAIT WHAT HAPPENED
 3824 015416  104032                          ERROR   .32             ; NO POWER DOWN FOR THRU UNIBUS
 3825 015420  000417                          BR      10$             ;
 3826 015422  042777  000020  164532  1$:     BIC     @BIT04,@BE1CR2  ; CLEAR POWER DOWN BIT
 3827 015430  062706  000004                  ADD     @4,SP           ; ADJUST STACK POINTER
 3828 015434  012737  015454  000024          MOV     @3$,PWRVEC      ; POINT POWER UP VECTOR
 3829 015442  012701  177777                  MOV     @177777,R1      ; TIMEOUT ROUTINE
 3830 015446  077101                  2$:     SOB     R1,2$           ; WAIT A WHILE
 3831 015450  104032                          ERROR   .32             ; NO POWER UP
 3832 015452  000402                          BR      10$             ;
 3833 015454  012706  001100          3$:     MOV     @1100,SP        ; ADJUST STACK
 3834 015460  013737  001160  000024  10$:    MOV     $TMP0,PWRVEC    ; RESTORE POWER DOWN VECTOR
 3835
```

```
                                .SBTTL TEST - WRONG PARITY TEST

                                ;* THIS TEST CHECKS THAT A WRONG PARITY TRAP CAN BE GENERATED ON
                                ;* DATI CYCLES.
                                ;*
                                ;
                                ;  BGNTST
                                ;
                                ;       SAVE 114 WRONG PARITY VECTOR AND POINT IT TO PROGRAM AREA
                                ;       LET @BE1CC = -1 TO DO 1 CYCLE
                                ;       LET @BE1BA = @$TMP0 FOR THE ADDRESS
                                ;       LET @BE1CR2 = @BE1CR2 SET.BY @BIT12 TO ENABLE WRONG PARITY
                                ;       LET @BE1CR1 = @13041 TO DO 1 DATO FROM BE1CC
                                ;       IF NO TRAP TO 114 THEN
                                ;       .  ERROR GENERATING WRONG PARITY TRAP THRU UNIBUS
                                ;       ENDIF
                                ;       LET @BE1CR2 = @0 TO CLEAR WRONG PARITY BIT
                                ;       RESTORE VECTOR 114
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------------------

                                ;;*****************************************************************
                                ;*TEST 44       WRONG PARITY TEST
                                ;;*****************************************************************
     015466  000004             TST44:  SCOPE
3861
3862 015470  004737  002450             JSR     PC,IUBE         ; INITIALIZE THE UBE
3863 015474  013701  000114             MOV     @@114,R1        ; SAVE PARITY TRAP
3864 015500  012737  015546  000114     MOV     @1$,@@114       ; POINT TO PROGRAM AREA
3865 015506  012737  000340  000116     MOV     @340,@@116      ; AT PRIORITY 7
3866 015514  012777  177777  164430     MOV     @-1,@BE1CC      ; DO 1 CYCLE
3867 015522  012777  001160  164424     MOV     @$TMP0,@BE1BA   ; SETUP ADDRESS REGISTER
3868 015530  052777  010000  164424     BIS     @BIT12,@BE1CR2  ; SET WRONG PARITY BIT
3869 015536  005777  164414             TST     @BE1CR1         ; READ A UBE REGISTER
3870 015542  000240                     NOP                     ; SHOULD CAUSE A PARITY TRAP
3871 015544  104032                     ERROR   .32             ; NO PARITY TRAP
3872 015546  012777  000000  164406  1$:MOV     @0,@BE1CR2      ; CLEAR WRONG PARITY BIT
3873 015554  062706  000004             ADD     @4,SP           ; ADJUST STACK POINTER
3874
```

```
                                .SBTTL TEST - NO SACK TIMEOUT

                                ;* THIS TEST INSURES THAT THE CPU TIMES OUT AND DROPS A GRANT IF NO
                                ;* SACK SIGNAL IS RECEIVED.  IF THE CPU DOES NOT TIMEOUT, THE UBE
                                ;* WILL TIME OUT AND SEND SACK TO PREVENT THE BUS FROM HANGING AND WILL SET
                                ;* THE ERROR BIT IN CR2.
                                ;*
                                ;
                                ;  BGNTST
                                ;
                                ;       LET @BE1CC = -1 TO DO 1 TRANSFER
                                ;       LET @BE1CR2 = @BE1CR2 SET.BY #BIT03 TO INHIBIT SACK
                                ;       LET @BE1CR1 = #6003 TO DO FUN 3
                                ;       WAIT FOR TIMEOUT OR @BE1CC NE -1
                                ;       IF  NO TIMEOUT AND @BE1CR2 SET.BY #BIT07 THEN
                                ;       .  ERROR CPU FAILED TO DO NO SACK TIMEOUT
                                ;       ENDIF
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------------------

                                ;;*****************************************************************
                                ;*TEST 45       NO SACK TIMEOUT
                                ;;*****************************************************************
     015560  000004             TST45:  SCOPE
3899
3900 015562  004737  002450             JSR     PC,IUBE                 ; INITIALIZE THE UBE
3901 015566  012777  000010  164366     MOV     #BIT03,@BE1CR2          ; INHIBIT SACK BIT
3902 015574  005037  177776             CLR     PSW                     ; LOWER PRIORITY
3903 015600  012777  006003  164350     MOV     #6003,@BE1CR1           ; GIVE UP BUS AFTER BECOMING MASTER
3904 015606  012701  007777             MOV     #7777,R1                ; DELAY CONSTANT
3905 015612  077101                 1$: SOB     R1,1$                   ; WAIT IN A LOOP
3906 015614  105777  164342             TSTB    @BE1CR2                 ; NO NO SACK TIMEOUT?
3907 015620  100001                     BPL     2$                      ; IF NO, BRANCH
3908 015622  104032                     ERROR   +32                     ; NO NO SACK TIMEOUT
3909 015624  012777  000000  164330 2$: MOV     #0,@BE1CR2              ; CLEAR INHIBIT SACK BIT
3910 015632  012737  000340  177776     MOV     #340,PSW                ; RESTORE PRIORITY
```

TEST - NO INTERRUPT TEST

```
                                .SBTTL TEST - NO INTERRUPT TEST

                                ;* THIS TEST CHECKS THAT NO INTERRUPT CONDITION DOES NOT HANG THE BUS.
                                ;*
                                ;
                                ;  BGNTST
                                ;
                                ;       PROGRAM UBE TO DO DMA ON BR7
                                ;       IF CYCLE NOT DONE OR INTERRUPT THEN
                                ;       .  ERROR IN NO INTERRUPT LOGIC
                                ;       ENDIF
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------------

                                ;;*****************************************************************
                                ;*TEST 46       PASSIVE RELEASE
                                ;;*****************************************************************
 015640  000004                 TST46:  SCOPE

 015642  004737  002450                 JSR     PC,IUBE                 ; INITIALISE UBE
 015646  012777  177777  164276         MOV     #-1,@BE1CC              ; DO 1 CYCLE
 015654  012777  001160  164272         MOV     #$TMP0,@BE1BA           ; AT ADDRESS $TMP0
 015662  012777  015734  164274         MOV     #10$,@BE1VEC            ; POINT VECTOR
 015670  012737  000003  177776         MOV     #3,PSW                  ; LOWER PRIORITY TO 3
 015676  012777  002021  164252         MOV     #2021,@BE1CR1           ; BR7, DATI
 015704  105777  164246         1$:     TSTB    @BE1CR1                 ; DONE?
 015710  100375                        BPL     1$                      ; IF NOT, WAIT
 015712  012737  000340  177776         MOV     #340,PSW                ; RESTORE PRIORITY
 015720  023777  001160  164222         CMP     $TMP0,@BE1DB            ; DATA OK?
 015726  001404                        BEQ     TST47           ;; IF OK, EXIT TEST
 015730  104032                        ERROR   .32                     ; DATI ON BR7 IS WRONG
 015732  000402                        BR      TST47           ;; GO TO NEXT TEST

 015734  104032                 10$:    ERROR   .32                     ; NO INTERRUPT LOGIC IS WRONG
 015736  000002                        RTI
```

```
                                .SBTTL TEST - UNIBUS DEVICE DATO CYCLE
                                ;
                                ;* THE FOLLOWING TEST WILL SEE IF A UNIBUS DEVICE DATA OUT
                                ;* CAN OCCUR ON THE KTJ11-B MODULE.THE UNIBUS DATA PATH
                                ;* ON THE BOARD IS ALSO TESTED OUT.
                                ;*
                                ;
                                ;  BGNTST
                                ;
                                ;       DISABLE UNIBUS MAPPING
                                ;       LET R0 := POINTER TO WRITE BUFFER
                                ;       LET R1 := POINTER TO PATTERN TABLE
                                ;*
                                ;*  GO DO THE TRANSFERS
                                ;*
                                ;       DO FOR PATTERNS 377,7417,31463,52525,125252
                                ;       .  LET BE1DB := (R1)+                   ;DATA IS CURRENT PATTERN
                                ;       .  LET BE1BA := (R0)+                   ;ADDRESS IS CURRENT ADDRESS OF WRITE BUFFER
                                ;       .  LET BE1CC := -1                      ;SET UBE FOR 1 DATA XFER
                                ;       .  SET UBE FOR NPR-DATO-1 XFER
                                ;       .  SET OFF THE TRANSFER
                                ;       ENDDO
                                ;       LET R0 := POINTER TO WRITE BUFFER
                                ;       LET R1 := POINTER TO PATTERN TABLE
                                ;*
                                ;*  CHECK IF THE TRANSFERS WERE CORRECT
                                ;*
                                ;       DO UNTIL TABLE IS COMPLETE
                                ;       .  IF (R0)+ NEQ (R1)+ THEN
                                ;       .  .  ERROR DATO DID NOT OCCUR CORRECTLY
                                ;       .  ENDIF
                                ;       ENDDO
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------------------------
                                ;;*************************************************************************
                                ;*TEST 47       UNIBUS DEVICE DATO CYCLE TEST
                                ;;*************************************************************************
    015740  000004              TST47:  SCOPE

    015742  004737  002450              JSR     PC,IUBE         ; INITIALIZE THE UBE
                                ;
                                ; INITIALIZE POINTERS FOR THE TRANSFERS
                                ;
    015746  042737  000040  172516      BIC     #BIT5,MMR3      ; MAKE SURE UNIBUS MAPPING IS DISABLED
    015754  012700  001734              MOV     #WRTBUF,R0      ; POINTER TO WRITE BUFFER
    015760  012701  001774              MOV     #PTRN16,R1      ; POINTER TO PATTERN TABLE
                                ;
                                ; EXECUTE THE LOOP TO DO THE TRANSFERS
                                ;
    015764  012737  000002  177730      MOV     #BIT01,DCSR     ; SELECT UNIBUS LINES
```

T47 UNIBUS DEVICE DATO CYCLE TEST

```
4002 015772  012704  000005          MOV   #5,R4           ; SET UP LOOP COUNT
4003 015776  012177  164146    1$:   MOV   (R1)+,@BE1DB    ; . GET CURRENT DATA
4004 016002  010077  164146          MOV   R0,@BE1BA       ; . GET ADDRESS IN THE WRITE BUFFER
4005 016006  005077  164150          CLR   @BE1CR2         ; . CLEAR ADDRESS BITS 16,17
4006 016012  012777  177777  164132 MOV   #-1.,@BE1CC     ; . SET UBE FOR 1 DATA TRANSFER
4007 016020  012777  003041  164130 MOV   #3041,@BE1CR1   ; . SET UBE FOR NPR-DATO-1 XFER
4008
4009                               ;
4010                               ; WAIT FOR THE TRANSFER TO BE COMPLETE
4011                               ;
4012
4013 016026  032777  000200  164122 5$:   BIT   #BIT7,@BE1CR1  ; . . IS THE TRANSFER COMPLETE ?
4014 016034  001774                BEQ   5$              ; . . NO,THEN GO WAIT FOR IT
4015 016036  005737  177732        TST   DDR             ; . . UNIBUS LINES 0?
4016 016042  001415                BEQ   7$              ; . . IF SO, BRANCH
4017 016044  022704  000003        CMP   #3,R4           ; . . SELECTING CONTROL LINES
4018 016050  001004                BNE   6$              ; . . IF NOT, BRANCH
4019 016052  032737  000373  177732 BIT   #373,DDR       ; . . ONLY USED LINES 0'S?
4020 016060  001406                BEQ   7$              ; . . IF SO, BRANCH
4021 016062  013737  177732  001126 6$:   MOV   DDR,$BDDAT     ; . . RECIEVED DATA
4022 016070  005037  001124        CLR   $GDDAT          ; . . EXPECTED DATA
4023 016074  104016                ERROR +16             ; . . ERROR IN UNIBUS LINES
4024 016076  062737  000002  177730 7$:   ADD   #BIT01,DCSR  ; . . THRU ALL COMBINATIONS
4025 016104  032737  000006  177730       BIT   #6,DCSR    ; . . ALL DONE?
4026 016112  001003                BNE   8$              ; . . IF NOT, BRANCH
4027 016114  012737  000002  177730       MOV   #BIT01,DCSR  ; . . OTHERWISE, START OVER
4028 016122  062700  000002        8$:   ADD   #2,R0       ; . . GET THE NEXT ADDRESS TO TRANSFER TO
4029 016126  077455                SOB   R4,1$           ; . HAVE WE GONE THROUGH THE TABLE ?
4030
4031                               ;
4032                               ; CHECK IF THE TRANSFERS WERE COMPLETED CORRECTLY
4033                               ;
4034
4035 016130  012700  001734          MOV   #WRTBUF,R0      ; POINTER TO WRITE BUFFER
4036 016134  012701  001774          MOV   #PTRN16,R1      ; POINTER TO PATTERN TABLE
4037 016140  012704  000005          MOV   #5,R4           ; SET UP LOOP COUNT
4038 016144  022021          10$:    CMP   (R0)+,(R1)+     ; . WAS THE TRANSFER CORRECT ?
4039 016146  001401                  BEQ   15$             ; . YES,THEN SKIP THE ERROR MESSAGE
4040 016150  104032                  ERROR +32             ; . NO, THEN ERROR IN THE TRANSFER
4041 016152  077404          15$:    SOB   R4,10$          ; . HAVE WE CHECKED ALL 5 XFERS ?
4042
4043
4044
```

```
                                .SBTTL TEST - UNIBUS DEVICE DATI CYCLE

                                ;* THIS TEST WILL SEE IF THE KTJ11-B MODULE CAN EXECUTE A
                                ;* UNIBUS DEVICE DATI CYCLE.THE UNIBUS ADDRESS PATH ON THE
                                ;* MODULE IS ALSO TESTED OUT.
                                ;*
                                ;
                                ;  BGNTST
                                ;
                                ;*
                                ;*  INITIALIZE THE POINTERS FOR THE TRANSFERS
                                ;*
                                ;       DISABLE UNIBUS MAPPING
                                ;       LET R0 := POINTER TO 16 BIT PATTERNS
                                ;       LET R1 := (R0)                          ;CURRENT PATTERN
                                ;*
                                ;*  GO DO THE TRANSFER THEN CHECK IF IT OCCURED CORRECTLY
                                ;*
                                ;       LET BE1DB := 0                          ;CLEAR DATA BUFFER
                                ;       LET BE1BA := R0                         ;READ FROM DATA PATTERN
                                ;       LET BE1CC := -1                         ;1 DATA XFER
                                ;       SETUP UBE TO DO NPR-DATI-1 DATA XFER
                                ;       SET OFF THE TRANSFER
                                ;       IF BE1DB NEQ #377  THEN
                                ;       .  ERROR DATA READ IN WAS INCORRECT
                                ;       ENDIF
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------------------

                                ;;*****************************************************************
                                ;*TEST 50       UNIBUS DEVICE DATI CYCLE TEST
                                ;;*****************************************************************
    016154  000004              TST50:  SCOPE

                                ;
                                ; INITIALIZE POINTERS FOR THE TRANSFERS
                                ;

016156  004737  002450                  JSR     PC,IUBE                         ; INITIALIZE THE UBE
016162  042737  000040  172516          BIC     #BIT5,MMR3      ; DISABLE UNIBUS MAPPING

                                ;
                                ; GO DO THE TRANSFER
                                ;

016176  005077  163746          5$:     CLR     @BE1DB          ; . INITIALIZE THE DATA BUFFER.
016202  010077  163746                  MOV     R0,@BE1BA       ; . GET ADDRESS WE WANT TO READ
016206  005077  163750                  CLR     @BE1CR2         ; . CLEAR ADDRESS BITS 16,17
016212  012777  177777  163732          MOV     #-1,@BE1CC      ; . SET UBE FOR 1 TRANSFER
016220  012777  002041  163730          MOV     #2041,@BE1CR1   ; . SET UBE FOR NPR-DATI-1 XFER

                                ;
```

J8

T50 UNIBUS DEVICE DATI CYCLE TEST

```
                                        ; WAIT FOR THE TRANSFER TO COMPLETE
                                        ;
                                        ;
016226  032777  000200  163722  10$:    BIT     #BIT7,@BE1CR1   ; . . IS THE TRANSFER COMPLETE ?
016234  001774                          BEQ     10$             ; . . WAIT FOR IT TO BE COMPLETE
                                        ;
                                        ; CHECK THE TRANSFER
                                        ;
016236  027720  163706                  CMP     @BE1DB,(R0)+    ; . WAS THE TRANSFER CORRECT ?
016242  001401                          BEQ     TST51           ;;      YES,THEN GO TO THE NEXT TEST
016244  104032                          ERROR   +32             ; . NO,THEN ERROR IN THE TRANSFER
```

TEST - UNIBUS DEVICE DATO CYCLE WITH RELOCATION ENABLED

```
                                .SBTTL TEST - UNIBUS DEVICE DATO CYCLE WITH RELOCATION ENABLED
                                ;
                                ;* THIS TEST WILL SEE IF THE KTJ11-B MODULE CAN EXECUTE
                                ;* A UNIBUS DEVICE DATO CYCLE THROUGH THE UNIBUS MAP.
                                ;*
                                ;
                                ;  BGNTST
                                ;       IF ALL UNIBUS MEMORY THEN GO TO END OF PASS
                                ;*
                                ;*  SET UP UBE AND UNIBUS MAP REGISTERS FOR TRANSFER
                                ;*
                                ;       LET BE1BA := 0                  ;POINT TO UMR0
                                ;       LET BE1DB := 125252             ;DATA PATTERN IS 125252
                                ;       LET BE1CC := -8.                ;DO 8 XFERS
                                ;       SET UBE TO DO NPR-DATO-2 DATA XFERS
                                ;       LET MAPL00 := ADDRESS OF WRITE BUFFER
                                ;       LET MAPH00 := 0
                                ;*
                                ;*      GO DO THE TRANSFER
                                ;*
                                ;       SET OFF TRANSFER
                                ;       WAIT TILL XFER IS DONE
                                ;*
                                ;*      CHECK IF THE TRANSFER OCCURED CORRECTLY
                                ;*
                                ;       LET R0 := POINTER TO WRITE BUFFER
                                ;       DO FOR R1 := 1 TO 8
                                ;       .  IF (R0)+ NEQ #125252 THEN
                                ;       .  .  ERROR DATO DID NOT EXECUTE CORRECTLY
                                ;       .  ENDIF
                                ;       ENDDO
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------------
                                ;;*************************************************************
                                ;*TEST 51       UNIBUS DEVICE DATO CYCLE WITH RELOCATION ENABLED
                                ;;*************************************************************
    016246  000004              TST51:  SCOPE

                                        MOV     KMCR,R1         ; STORE KMCR
                                        BIC     #177700,R1      ; LEAVE ONLY BITS <5-0>
                                        CMP     #77,R1          ; ALL UNIBUS MEMORY?
                                        BNE     1$              ; IF NOT, BRANCH
                                        JMP     $EOP            ; IF ALL UNIBUS, SKIP TILL END OF PASS
                                1$:     JSR     PC,IUBE         ; INITIALIZE THE UBE

                                ;
                                ; SET UP UBE AND UNIBUS MAP REGISTERS FOR TRANSFER
                                ;

                                        CLR     @BE1BA          ; POINT UBE ADDRESS TO UNIBUS MAP
                                        CLR     @BE1CR2         ; REGISTER #0
                                        MOV     #125252,@BE1DB  ; DATA PATTERN IS 125252
                                        MOV     #-8.,@BE1CC     ; DO 8. TRANSFERS
                                        MOV     #WRTBUF,MAPL00  ; SET UP LOW MAP REGISTER
```

T51 UNIBUS DEVICE DATO CYCLE WITH RELOCATION ENABLED

```
4172 016330  005037  170202                    CLR     MAPH00          ; SET UP HIGH MAP REGISTER
4173 016334  052737  000040  172516            BIS     #BIT5,MMR3      ; ENABLE UNIBUS MAPPING
4174
4175                                 ;
4176                                 ; GO DO THE TRANSFER
4177                                 ;
4178
4179 016342  012777  003041  163606            MOV     #3041,@8E1CR1   ; DO A NPR-DATO-1 XFER
4180 016350  032777  000200  163600  10$:       BIT     #BIT7,@8E1CR1   ; IS THE TRANSFER COMPLETE ?
4181 016356  001774                            BEQ     10$             ; WAIT FOR IT TO COMPLETE
4182
4183                                 ;
4184                                 ; CHECK IF THE TRANSFER OCCURED CORRECTLY
4185                                 ;
4186
4187 016360  042737  000040  172516            BIC     #BIT5,MMR3      ; DISABLE UNIBUS MAPPING
4188 016366  012700  001734                    MOV     #WRTBUF,R0      ; GET POINTER TO WRITE BUFFER
4189 016372  012701  000010                    MOV     #8.,R1          ; SET UP LOOP COUNTER
4190 016376  022027  125252          15$:      CMP     (R0)+,#125252   ; . SEE IF TRANSFER OCCURED CORRECTLY ?
4191 016402  001401                            BEQ     20$             ; . YES,THEN SKIP ERROR MESSAGE
4192 016404  104032                            ERROR   +32             ; . NO,THEN ERROR IN TRANSFER
4193 016406  077105                  20$:      SOB     R1,15$          ; . HAVE WE CHECKED ALL THE TRANSFERS
4194
4195
```

TEST - UNIBUS DEVICE DATI CYCLE WITH RELOCATION ENABLED

```
                                .SBTTL TEST - UNIBUS DEVICE DATI CYCLE WITH RELOCATION ENABLED
                                
                                ;* THIS TEST WILL SEE IF THE KTJ11-B MODULE CAN EXECUTE
                                ;* A UNIBUS DEVICE DATI CYCLE THROUGH THE UNIBUS MAP WITH
                                ;* CACHE DISABLED.
                                ;*
                                ;
                                ;  BGNTST
                                ;
                                ;*
                                ;*  SET UP THE UBE FOR A DATI CYCLE
                                ;*
                                ;       LET BE1BA := 0          ;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 MAPL00 := R0                ;POINT UMR #0 TO DATA PATTERN
                                ;       LET MAPH00 := 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 (R0)+ THEN
                                ;       .  ERROR DATI THROUGH UNIBUS MAP DID NOT EXECUTE CORRECTLY
                                ;       ENDIF
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------------------
                                
                                
                                ;;*****************************************************************
                                ;*TEST 52       UNIBUS DEVICE DATI CYCLE WITH RELOCATION ENABLED
                                ;;*****************************************************************
    016410  000004              TST52:  SCOPE
                                
 016412  004737  002450                 JSR     PC,IUBE         ; INITIALIZE THE UBE
                                
                                ;
                                ; SET UP UBE AND UNIBUS MAP REGISTERS FOR A DATI CYCLE
                                ;
                                
 016416  005077  163532                 CLR     @BE1BA          ; POINT UBE TO THE FIRST UNIBUS MAP
 016422  005077  163534                 CLR     @BE1CR2         ; REGISTER
 016426  005077  163516                 CLR     @BE1DB          ; INITIALIZE THE DATA BUFFER
 016432  012777  177777  163512         MOV     #-1,@BE1CC      ; SET FOR 1 TRANSFER
 016440  012737  001774  170200         MOV     #PTRN16,MAPL00  ; POINT MAP REGISTERS TO PATTERN TABLE
```

T52 UNIBUS DEVICE DATI CYCLE WITH RELOCATION ENABLED

```
4251 016446  005037  170202                    CLR     MAPH00          ;
4252 016452  052737  000040  172516            BIS     #BIT5,MMR3      ; ENABLE UNIBUS MAPPING
4253
4254                                ;
4255                                ; GO DO THE TRANSFER
4256                                ;
4257
4258 016460  012777  002041  163470            MOV     #2041,@BE1CR1   ; SET UBE FOR NPR-DATI-1 XFER
4259 016466  032777  000200  163462  5$:       BIT     #BIT7,@BE1CR1   ; IS THE TRANSFER COMPLETE
4260 016474  001774                            BEQ     5$              ; NO,THEN WAIT FOR IT TO BE COMPLETE
4261 016476  042737  000040  172516            BIC     #BIT5,@#MMR3    ; YES,THEN DISABLE UNIBUS MAPPING
4262
4263                                ;
4264                                ; CHECK THE TRANSFER
4265                                ;
4266
4267 016504  027727  163440  000377            CMP     @BE1DB,#377     ; DID THE TRANSFER HAPPEN CORRECTLY ?
4268 016512  001401                            BEQ     TST53           ;; GO TO THE NEXT TEST
4269 016514  104032                            ERROR   +32             ; TRANSFER DID NOT OCCUR CORRECTLY
4270
4271
4272
```

```
4274                            .SBTTL TEST - ALU TEST USING UBE
4275
4276                            ;* THIS TEST WILL CHECK THE FIRST 3 ALU'S EACH ONE AT A TIME. AT FIRST THE
4277                            ;* PATTERN THAT DOES NOT RESULT IN OVERFLOW WILL BE USED. THE SECOND PATTERN
4278                            ;* GENERATES OVERFLOW INTO THE NEXT ALU.
4279                            ;*
4280                            ;
4281                            ;  BGNTST
4282                            ;
4283                            ;       IN 18-BIT MODE DON'T RUN THIS TEST
4284                            ;       LET R1 := 1001 (PATTERN WITHOUT OVERFLOW FROM ALU)
4285                            ;       LET R2 := 0101 (MAP REGISTER PATTERN)
4286                            ;       LET R3 := @MAPL00
4287                            ;       LET R4 := 0110 (OVERFLOW)
4288                            ;       LET R5 := 1011
4289                            ;       DO FOR 3 PATTERNS
4290                            ;       . LET BE1BA := R1               ;POINT ADDRESS TO MAPX00
4291                            ;       . LET BE1CR2 := 0               ;
4292                            ;       . LET BE1DB := 0                ;INITIALIZE DATA BUFFER
4293                            ;       . LET BE1CC := -1               ;SET FOR 1 XFER
4294                            ;       . SET UBE FOR A NPR-DATI-1 XFER
4295                            ;*
4296                            ;*  SET UP THE MAP REGISTER FOR THE TRANSFER WITHOUT OVERFLOW FROM ALU
4297                            ;*
4298                            ;       . LET (R3) := R2                ;POINT UMR @0 TO DATA PATTERN
4299                            ;       . LET 2(R3) := 0
4300                            ;       . ENABLE UNIBUS MAPPING
4301                            ;*
4302                            ;*  GO DO THE TRANSFER
4303                            ;*
4304                            ;       . SET OFF THE TRANSFER
4305                            ;       . WAIT FOR TRANSFER TO COMPLETE
4306                            ;       . DISABLE UNIBUS MAPPING
4307                            ;*
4308                            ;*  CHECK IF THE TRANSFER OCCURED CORRECTLY
4309                            ;*
4310                            ;       . IF BE1DB NEQ (R1+R2) THEN
4311                            ;       . . ERROR DATI THROUGH UNIBUS MAP DID NOT EXECUTE CORRECTLY
4312                            ;       . ENDIF
4313                            ;*
4314                            ;* NOW CHECK CARRY OUT FROM ALU
4315                            ;*
4316                            ;       . LET BE1BA := R4               ;POINT ADDRESS TO MAPX00
4317                            ;       . LET BE1CR2 := 0               ;
4318                            ;       . LET BE1DB := 0                ;INITIALIZE DATA BUFFER
4319                            ;       . LET BE1CC := -1               ;SET FOR 1 XFER
4320                            ;       . SET UBE FOR A NPR-DATI-1 XFER
4321                            ;*
4322                            ;*  SET UP THE MAP REGISTER FOR THE TRANSFER
4323                            ;*
4324                            ;       . LET (R3) := R5                ;POINT UMR @0 TO DATA PATTERN
4325                            ;       . LET 2(R3) := 0
4326                            ;       . ENABLE UNIBUS MAPPING
4327                            ;*
4328                            ;*  GO DO THE TRANSFER
4329                            ;*
4330                            ;       . SET OFF THE TRANSFER
```

TEST - ALU TEST USING UBE

```
4331                                    ;       . WAIT FOR TRANSFER TO COMPLETE
4332                                    ;       . DISABLE UNIBUS MAPPING
4333                                    ;*
4334                                    ;*  CHECK IF THE TRANSFER OCCURED CORRECTLY
4335                                    ;*
4336                                    ;       . IF BE1DB NEQ (R4+R5) THEN
4337                                    ;       . . ERROR DATI THROUGH UNIBUS MAP DID NOT EXECUTE CORRECTLY
4338                                    ;       . ENDIF
4339                                    ;*
4340                                    ;* CHANGE THE PATTERN
4341                                    ;*
4342                                    ;       . SHIFT R1,R2,R4,R5 4 TIMES AND POINT TO MAPLO3 AND MAPLO6
4343                                    ;       . POINT R3 TO MAPLO3 AND THEN TO MAPLO6
4344                                    ;      ENDDO
4345                                    ;
4346                                    ;  ENDTST
4347                                    ;
4348                                    ;-------------------------------------------------------------------
4349
4350
4351                                    ;;*****************************************************************
                                        ;*TEST 53       ALU TEST USING UBE
                                        ;;*****************************************************************
     016516  000004                     TST53:  SCOPE
4352
4353 016520  032737  000040  177734             BIT     #BIT05,KMCR     ; 18 BIT MODE?
4354 016526  001402                             BEQ     10$             ; IF NOT, CONTINUE
4355 016530  000137  017146                     JMP     20$             ; OTHERWISE, EXIT
4356 016534  005037  172354             10$:    CLR     KIPAR6          ; FOR ERROR REPORTING
4357 016540  004737  002450                     JSR     PC,IUBE         ; INITIALIZE THE UBE
4358 016544  012700  000022                     MOV     #22,R0          ; INITIAL PATTERN FOR ADDRESS LINES
4359 016550  012701  000012                     MOV     #12,R1          ; INITIAL PATTERN FOR MAP REGISTER
4360 016554  012703  170200                     MOV     #MAPL00,R3      ; FIRST REGISTER PAIR TO BE USED
4361 016560  012704  000014                     MOV     #14,R4          ; PATTERN FOR THE OVERFLOW FOR A. LINES
4362 016564  012705  000026                     MOV     #26,R5          ; PATTERN FOR MAP PAIR
4363 016570  005037  001162                     CLR     $TMP1           ; LOCATION TO SELECT REGISTER PAIR
4364                                    ;
4365                                    ; SET UP UBE AND UNIBUS MAP REGISTERS FOR A DATI CYCLE
4366                                    ;
4367
4368 016574  010077  163354             1$:     MOV     R0,@#BE1BA      ; POINT UBE TO THE PATTERN
4369 016600  005077  163356                     CLR     @#BE1CR2        ; REGISTER
4370 016604  005077  163340                     CLR     @#BE1DB         ; INITIALIZE THE DATA BUFFER
4371 016610  012777  177777  163334             MOV     #-1,@#BE1CC     ; SET FOR 1 TRANSFER
4372 016616  010113                             MOV     R1,(R3)         ; POINT MAP REGISTERS TO PATTERN
4373 016620  005063  000002                     CLR     2(R3)           ; CLEAR HIGH MAP REGISTER
4374 016624  052737  000040  172516             BIS     #BIT5,MMR3      ; ENABLE UNIBUS MAPPING
4375
4376                                    ;
4377                                    ; GO DO THE TRANSFER
4378                                    ;
4379
4380 016632  012777  002041  163316             MOV     #2041,@#BE1CR1  ; SET UBE FOR NPR-DATI-1 XFER
4381 016640  032777  000200  163310  2$:     BIT     #BIT7,@#BE1CR1  ; IS THE TRANSFER COMPLETE
4382 016646  001774                             BEQ     2$              ; NO,THEN WAIT FOR IT TO BE COMPLETE
4383 016650  042737  000040  172516             BIC     #BIT5,@#MMR3    ; YES,THEN DISABLE UNIBUS MAPPING
4384
```

```
                                        ;
                                        ; CHECK THE TRANSFER
                                        ;
                                            MOV     R0,$TMP0        ; FIND ADDRESS ACCESSED
                                            BIC     #160000,$TMP0   ; MAKE SURE THAT USING ALL 16 BITS FOR ADDRESS
                                            ADD     R1,$TMP0        ;
                                            CMP     @#BE1DB,@$TMP0  ; DID THE TRANSFER HAPPEN CORRECTLY ?
                                            BEQ     3$              ; IF SO, BRANCH
                                            MOV     @#BE1DB,$BDDAT  ; WRONG DATA
                                            MOV     @$TMP0,$GDDAT   ; EXPECTED DATA
                                            MOV     $TMP0,$BDADR    ; ADDRESS USED
                                            ERROR   .23             ; TRANSFER DID NOT OCCUR CORRECTLY
                                        ;
                                        ; NOW CHECK THE SAME ALU FOR OVERFLOW
                                        ;
                                        3$: MOV     R4,@#BE1BA      ; POINT UBE TO THE PATTERN
                                            CLR     @#BE1CR2        ; REGISTER
                                            CLR     @#BE1DB         ; INITIALIZE THE DATA BUFFER
                                            MOV     #-1,@#BE1CC     ; SET FOR 1 TRANSFER
                                            MOV     R5,(R3)         ; POINT MAP REGISTERS TO PATTERN
                                            CLR     2(R3)           ; CLEAR HIGH MAP REGISTER
                                            BIS     #BIT5,MMR3      ; ENABLE UNIBUS MAPPING

                                        ;
                                        ; GO DO THE TRANSFER
                                        ;

                                            MOV     #2041,@#BE1CR1  ; SET UBE FOR NPR-DATI-1 XFER
                                        4$: BIT     #BIT7,@#BE1CR1  ; IS THE TRANSFER COMPLETE
                                            BEQ     4$              ; NO,THEN WAIT FOR IT TO BE COMPLETE
                                            BIC     #BIT5,@#MMR3    ; YES,THEN DISABLE UNIBUS MAPPING

                                        ;
                                        ; CHECK THE TRANSFER
                                        ;

                                            MOV     R4,$TMP0        ; FIND ADDRESS ACCESSED
                                            BIC     #160000,$TMP0   ; MAKE SURE THAT USING ALL 16 BITS FOR ADDRESS
                                            ADD     R5,$TMP0        ;
                                            CMP     @#BE1DB,@$TMP0  ; DID THE TRANSFER HAPPEN CORRECTLY ?
                                            BEQ     5$              ; IF SO, BRANCH
                                            MOV     @#BE1DB,$BDDAT  ; WRONG DATA
                                            MOV     @$TMP0,$GDDAT   ; EXPECTED DATA
                                            MOV     $TMP0,$BDADR    ; ADDRESS USED
                                            ERROR   .23             ; TRANSFER DID NOT OCCUR CORRECTLY
                                        ;
                                        ; CHANGE THE PATTERN
                                        ;
                                        5$: CMP     #2000,PMIS      ; MORE THAN 32K OF PMI MEMORY?
                                            BHIS    TST54           ;; IF NOT, EXIT TEST
                                            ASH     #4,R0           ; TO GET TO NEXT ALU
                                            ASH     #4,R1           ; BY SHIFYING 4 TIMES
                                            ASH     #4,R4           ;
                                            ASH     #4,R5           ;
                                            ADD     #12.,R3         ; GET NEXT REGISTER PAIR
                                            ADD     #60000,$TMP1    ;
```

T53 ALU TEST USING UBE

```
4442 017126  053700  001162              BIS     $TMP1,R0        ; SET TO SELECT NEXT REGISTER PAIR
4443 017132  053704  001162              BIS     $TMP1,R4        ;
4444 017136  022737  020000  001162      CMP     #20000,$TMP1    ; ALL 3 ALU'S DONE?
4445 017144  001213                      BNE     1$              ; IF NOT, BRANCH
4446 017146                      20$:
```

```
                                .SBTTL TEST - CARRY PROPOGATION TEST USING UBE
                                ;* THIS TEST VALIDATES THAT CARRY CAN BE PROPOGATED THRU ALL ALU'S CORRECTLY.
                                ;* THE RESULT IS OBTAINED FROM LOCATION 0.
                                ;
                                ;  BGNTST
                                ;
                                ;*
                                ;*  SET UP THE UBE FOR A DATI CYCLE
                                ;*
                                ;       LET @#0 := #52525
                                ;       LET BE1BA := #2
                                ;       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 MAPLO0 := #177777           ;POINT UMR #0 TO DATA PATTERN
                                ;       LET MAPHO0 := #77
                                ;       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 @#0 (52525) THEN
                                ;       .  ERROR DATI THROUGH UNIBUS MAP DID NOT EXECUTE CORRECTLY
                                ;       ENDIF
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------------------------
                                ;;*********************************************************************
                                ;*TEST 54        CARRY PROPOGATION TEST USING UBE
                                ;;*********************************************************************
    017146  000004              TST54:  SCOPE

                                        CLR     KIPAR6          ; CLEAR FOR ERROR REPORTS
    017150  005037  172354              JSR     PC,IUBE         ; INITIALIZE THE UBE
    017154  004737  002450

                                ;
                                ; SET UP UBE AND UNIBUS MAP REGISTERS FOR A DATI CYCLE
                                ;

    017160  012737  052525  000000      MOV     #52525,@#0      ; TEST LOCATION
    017166  012777  000002  162760      MOV     #2,@BE1BA       ; POINT UBE TO THE FIRST UNIBUS MAP
    017174  005077  162762              CLR     @BE1CR2         ; REGISTER
    017200  005077  162744              CLR     @BE1DB          ; INITIALIZE THE DATA BUFFER
    017204  012777  177777  162740      MOV     #-1,@BE1CC      ; SET FOR 1 TRANSFER
    017212  012737  177777  170200      MOV     #177777,MAPLO0  ; POINT MAP REGISTERS TO OVERFLOW
    017220  012737  000077  170202      MOV     #77,MAPHO0      ;
```

T54 CARRY PROPOGATION TEST USING UBE

```
4502 017226  052737  000040  172516          BIS     #BIT5,MMR3      ; ENABLE UNIBUS MAPPING
4503
4504                                 ;
4505                                 ; GO DO THE TRANSFER
4506                                 ;
4507
4508 017234  012777  002041  162714          MOV     #2041,@BE1CR1   ; SET UBE FOR NPR-DATI-1 XFER
4509 017242  032777  000200  162706  5$:     BIT     #BIT7,@BE1CR1   ; IS THE TRANSFER COMPLETE
4510 017250  001774                          BEQ     5$              ; NO,THEN WAIT FOR IT TO BE COMPLETE
4511 017252  042737  000040  172516          BIC     #BIT5,@#MMR3    ; YES,THEN DISABLE UNIBUS MAPPING
4512
4513                                 ;
4514                                 ; CHECK THE TRANSFER
4515                                 ;
4516
4517 017260  027727  162664  052525          CMP     @BE1DB,#052525  ; DID THE TRANSFER HAPPEN CORRECTLY ?
4518 017266  001411                          BEQ     TST55           ;; GO TO THE NEXT TEST
4519 017270  017737  162654  001126          MOV     @BE1DB,$BDDAT   ; WRONG DATA
4520 017276  012737  052525  001124          MOV     #52525,$GDDAT   ; EXPECTED DATA
4521 017304  005037  001122                  CLR     $BDADR          ; ADDRESS USED
4522 017310  104023                          ERROR   .23             ; TRANSFER DID NOT OCCUR CORRECTLY
4523
```

```
                                .SBTTL TEST - NXM TEST USING UBE
                                ;* THIS TEST CHECKS THAT ACCESSING NXM MEMORY CAN BE DONE CORRECTLY THRU
                                ;* UBE. TEST LOCATION USED IS 17760000.
                                ;
                                ;  BGNTST
                                ;
                                ;*
                                ;*  SET UP THE UBE FOR A DATI CYCLE
                                ;*
                                ;       LET BE1BA := 0          ;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 MAPL00 := #160000           ;POINT UMR #0 TO DATA PATTERN
                                ;       LET MAPH00 := #77
                                ;       ENABLE UNIBUS MAPPING
                                ;*
                                ;*  GO DO THE TRANSFER
                                ;*
                                ;       SET OFF THE TRANSFER
                                ;*
                                ;*  CHECK IF THE TRANSFER OCCURED CORRECTLY
                                ;*
                                ;       IF #BIT08 NOTSET IN @BE1CR2 THEN
                                ;       .  ERROR DATI TO NXM IS WRONG
                                ;       ENDIF
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------------------

                                ;;*****************************************************************
                                ;*TEST 55       NXM TEST USING UBE WITH RELOCATION ENABLED
                                ;;*****************************************************************
    017312  000004              TST55:  SCOPE
4561 017314  004737  002450            JSR     PC,IUBE         ; INITIALIZE THE UBE
4562
4563                            ;
4564                            ; SET UP UBE AND UNIBUS MAP REGISTERS FOR A DATI CYCLE
4565                            ;
4566 017320  012777  017450  162636    MOV     #10$,@BE1VEC    ; POINT INTERUPT VECTOR
4567 017326  012777  000340  162632    MOV     #340,@BE1PSW    ;
4568 017334  005077  162614            CLR     @BE1BA          ; POINT UBE TO THE FIRST UNIBUS MAP
4569 017340  005077  162616            CLR     @BE1CR2         ; REGISTER
4570 017344  005077  162600            CLR     @BE1DB          ; INITIALIZE THE DATA BUFFER
4571 017350  012777  177777  162574    MOV     #-1,@BE1CC      ; SET FOR 1 TRANSFER
4572 017356  012737  160000  170200    MOV     #160000,MAPL00  ; POINT MAP REGISTERS TO NXM 17760000
4573 017364  012737  000077  170202    MOV     #77,MAPH00      ;
4574 017372  052737  000040  172516    BIS     #BIT5,MMR3      ; ENABLE UNIBUS MAPPING
4575                            ;
4576                            ; GO DO THE TRANSFER
4577                            ;
4578 017400  012777  002041  162550    MOV     #2041,@BE1CR1   ; SET UBE FOR NPR-DATI-1 XFER
```

```
KTJ11-B DIAGNOSTIC      MACRO M1200  25-JUL-84 17:10  PAGE 55-1                    SEQ 0112

T55     NXM TEST USING UBE WITH RELOCATION ENABLED

  4579 017406  032777  000200  162542  5$:     BIT     @#BIT7,@#BE1CR1   ; IS THE TRANSFER COMPLETE
  4580 017414  001774                         BEQ     5$                ; NO,THEN WAIT FOR IT TO BE COMPLETE
  4581 017416  032777  000400  162536         BIT     @#BIT08,@#BE1CR2  ; NXM ?
  4582 017424  001001                         BNE     6$                ; IF SO, BRANCH
  4583 017426  104022                         ERROR   +22               ; NXM NOT SET
  4584 017430  106427  000140          6$:    MTPS    #140              ; ALLOW INTERRUPTS
  4585 017434  000240                         NOP
  4586 017436  000240                         NOP
  4587 017440  104022                         ERROR   +22               ; NXM WITHOUT INTERRUPT
  4588 017442  000404                         BR      12$               ;
  4589 017444  106427  000340                 MTPS    #340              ; RESTORE PRIORITY
  4590                                 ;
  4591                                 ; CHECK THE TRANSFER
  4592                                 ;
  4593 017450  062706  000004          10$:    ADD     #4,SP             ; ADJUST STACK
  4594 017454  042737  000040  172516  12$:    BIC     #BIT5,@#MMR3      ; YES,THEN DISABLE UNIBUS MAPPING
  4595
```

```
                                .SBTTL TEST - RELOCATION WITH UNIBUS MEMORY

                                ;* THIS TEST WILL CHECK THAT MAPPING REGISTER PAIRS THAT ARE
                                ;* ASSOCIATED WITH UNIBUS MEMORY DO NOT PERFORM RELOCATION.
                                ;* IF ANY UNIBUS MEMORY PRESENT, DATI CYCLES TO THAT MEMORY
                                ;* WILL BE CHECKED.
                                ;*
                                ;
                                ;  BGNTST
                                ;
                                ;       DO FOR KMCR = #67,27 (18-BIT AND 22-BIT MODES, 32K OF PMI MEMORY)
                                ;       DO WITH MAPPING PAIRS 10 TO 36
                                ;       .  DO DATI TO 0 USING SELCTED MAPPING PAIR
                                ;       .  IF NO TIMEOUT THEN
                                ;       .  .  ERROR DISABLED REGISTER PAIRS DO RELOCATION
                                ;       .  ENDIF
                                ;       ENDDO
                                ;       IF ANY UNIBUS MEMORY PRESENT THEN
                                ;       .  DO DATI WITH RELOCATION ENABLED
                                ;       .  IF RELOCATED THEN
                                ;       .  .  ERROR
                                ;       .  ENDIF
                                ;       ENDIF
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------

                                ;;*********************************************************************
                                ;*TEST 56       RELOCATION WITH UNIBUS MEMORY
                                ;;*********************************************************************
    017462  000004              TST56:  SCOPE
                                
                                        MOV     KMCR,$TMP1      ; STORE KMCR
                                        MOV     KMCR,R1         ; STORE KMCR
                                        BIC     #177740,R1      ; LEAVE ONLY <4-0>
                                        MOV     #MAPL37,$TMP0   ; START WITH HIGHEST ADDRESS
                                        BIT     #37,R1          ; ANY UNIBUS MEMRORY?
                                        BEQ     101$            ; IF NOT, BRANCH
                                100$:   SUB     #4,$TMP0        ; GET LOWER REGISTER PAIR
                                        SOB     R1,100$         ; LEAVE IN $TMP0 HIGHEST AVAILABLE PAIR
                                101$:   BIS     #BIT08,DCSR     ; ENABLE DIAGN. MODE
                                        MOV     #67,KMCR        ; 18-BIT, 32K PMI
                                        BIC     #BIT08,DCSR     ; DISABLE DIAGN. MODE
                                ;
                                ; SETUP MAPPING REGISTERS FOR INITIAL CONDITIONS
                                ;
                                        BIS     #BIT5,MMR3      ; ENABLE UNIBUS MAPPING
                                1$:     MOV     #MAPL10,R1      ; . POINT TO MAPL10
                                        CLR     R2              ; . ADDRESS BITS <15-0> REG. 10
                                        MOV     #1,R3           ; . ADDRESS FOR CR2<17-16>
                                2$:     JSR     PC,IUBE         ; . . INITIALIZE THE UBE
                                
                                ;
                                ; SET UP UBE AND UNIBUS MAP REGISTERS FOR A DATI CYCLE
                                ;
```

T56 RELOCATION WITH UNIBUS MEMORY

```
4651 017574  010277  162354                MOV     R2,@BE1BA       ; . . POINT UBE TO FIRST MAP REGISTER
4652 017600  005077  162356                CLR     @BE1CR2         ; . . REGISTER
4653 017604  010377  162352                MOV     R3,@BE1CR2      ; . . LOAD ADDRESS <17-16>
4654 017610  005077  162334                CLR     @BE1DB          ; . . INITIALIZE THE DATA BUFFER
4655 017614  012777  177777  162330        MOV     #-1,@BE1CC      ; . . SET FOR 1 TRANSFER
4656 017622  005021                        CLR     (R1)+           ; . . CLEAR LOW MAP REGISTER
4657 017624  005021                        CLR     (R1)+           ; . . AND HIGH MAP TOO
4658 017626  012777  017714  162330        MOV     #10$,@BE1VEC    ; . . LOAD INTERRUPT VECTOR
4659 017634  012777  000340  162324        MOV     #340,@BE1PSW    ;
4660                                ;
4661                                ; GO DO THE TRANSFER
4662                                ;
4663
4664 017642  012777  002041  162306        MOV     #2041,@BE1CR1   ; . . SET UBE FOR NPR-DATI-1 XFER
4665 017650  032777  000200  162300  5$:   BIT     #BIT7,@BE1CR1   ; . . IS THE TRANSFER COMPLETE
4666 017656  001774                        BEQ     5$              ; . . NO,THEN WAIT FOR IT TO BE COMPLETE
4667 017660  032777  000400  162274        BIT     #BIT08,@BE1CR2  ; . . NXM SET?
4668 017666  001002                        BNE     6$              ; . . IF SO, BRANCH
4669 017670  104022                        ERROR   +22             ; . . NO NXM
4670 017672  000412                        BR      11$             ;
4671 017674  106427  000140          6$:   MTPS    #140            ; . . LOWER PRIORITY
4672 017700  000240                        NOP                     ; . . WAIT FOR INTERRUPT
4673 017702  000240                        NOP                     ;
4674 017704  104022                        ERROR   +22             ; . . NO INTERRUPT ON NXM
4675 017706  106427  000340                MTPS    #340            ; . . RAISE PRIORITY
4676 017712  000402                        BR      11$             ;
4677 017714  062706  000004          10$:  ADD     #4,SP           ; . . ADJUST STACK
4678 017720  062702  020000          11$:  ADD     #20000,R2       ; . . SELECT NEXT MAP PAIR REGISTER
4679 017724  103001                        BCC     12$             ; . . IF NO CARRY, BRANCH
4680 017726  005203                        INC     R3              ; . . OTHERWISE INCREMENT ADDRESS <17-16>
4681 017730  023701  001160          12$:  CMP     $TMP0,R1        ; . . ALL DONE?
4682 017734  101315                        BHI     2$              ; . . IF LESS THAN, BRANCH
4683                                ;
4684                                ; CHECK IF DONE WITH 22-BIT MODE AND 18-BIT
4685                                ;
4686 017736  032737  000040  177734        BIT     #BIT05,KMCR     ; . 22-BIT MODE DONE?
4687 017744  001412                        BEQ     13$             ; . IF YES, BRANCH TO EXIT
4688 017746  052737  000400  177730        BIS     #BIT08,DCSR     ; . ENABLE DIAGN. MODE
4689 017754  012737  000027  177734        MOV     #27,KMCR        ; . 22-BIT, 32K PMI
4690 017762  042737  000400  177730        BIC     #BIT08,DCSR     ; . DISABLE DIAGN. MODE
4691 017770  000672                        BR      1$              ; . DO AGAIN IN 22-BIT MODE
4692                                ;
4693                                ; IF ANY UNIBUS MEMORY PRESENT, DO RELOCATION THRU IT
4694                                ;
4695 017772  022737  170374  001160  13$:   CMP     #MAPL37,$TMP0   ; ANY UNIBUS MEMORY?
4696 020000  001432                        BEQ     200$            ; IF NO, EXIT
4697 020002  012777  140000  162144        MOV     #140000,@BE1BA  ; TRY TO DO DATI THRU
4698 020010  012777  000003  162144        MOV     #3,@BE1CR2      ;  THRU MAP 36 REGISTER
4699 020016  005037  170370                CLR     MAPL36          ; CLEAR MAPPING REGISTER
4700 020022  005037  170372                CLR     MAPH36          ;  PAIR
4701 020026  012737  123456  000000        MOV     #123456,@#0     ; SET UP PATTERN AT 0
4702 020034  005077  162110                CLR     @BE1DB          ; CLEAR DATA REGISTER
4703 020040  012777  002071  162110        MOV     #2071,@BE1CR1   ; START DATI
4704 020046  105777  162104          20$:  TSTB    @BE1CR1         ; WAIT TILL
4705 020052  100375                        BPL     20$             ;  DONE
4706 020054  022777  123456  162066        CMP     #123456,@BE1DB  ; DATA CAME FROM @#0?
4707 020062  001001                        BNE     200$            ; IF NOT, BRANCH
```

L9

T56 RELOCATION WITH UNIBUS MEMORY

```
 4708 020064  104033                          ERROR   +33             ; RELOCATED UNIBUS MEMORY
 4709 020066  052737  000400  177730  200$:   BIS     @BIT08,DCSR     ; ENABLE DIAGN. MODE
 4710 020074  013737  001162  177734          MOV     $TMP1,KMCR      ; RESTORE KMCR
 4711 020102  042737  000400  177730          BIC     @BIT08,DCSR     ; DISABLE DIAGN. MODE
```

```
                                .SBTTL MAIN MEMORY DISABLE THRU UBE

                                ;* THIS TEST CHECKS THAT A MAIN MEMORY RESPONSE IS DISABLED WHENEVER
                                ;* THE APPROPRIATE BITS IN KMCR ARE SET. THE FIRST 32K ARE NOT GOING
                                ;* TO BE CHECKED. THE DMA DATI CYCLES WILL BE DONE THAT SHOULD CAUSE
                                ;* NXM BIT TO BE SET IN THE UBE REGISTER.
                                ;
                                ;  BGNTST
                                ;
                                ;       SIZE THE MEMORY AVAILABLE
                                ;       DO FOR ALL MEMORY PAGES
                                ;       .  LET KMCR DISABLE THAT PAGE IN MEMORY
                                ;       .  DO DATI TO THAT LOCATION
                                ;       .  IF NO TIMEOUT THEN
                                ;       .  . ERROR DISABLED MEMORY DOES NOT TIMEOUT
                                ;       .  ENDIF
                                ;       ENDDO
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------------

                                ;;*****************************************************************
                                ;*TEST 57       MAIN MEMORY DISABLE THRU UBE
                                ;;*****************************************************************
020110  000004                  TST57:  SCOPE

020112  005737  001720                  TST     PMIS                    ; ANY PMI MEMORY?
020116  001561                          BEQ     TST60           ;; IF NONE, EXIT TEST
                                ;
                                ; ENABLE MMU AND SIZE MEMORY
                                ;
020120  004737  002244                  JSR     PC,MAPPR        ; REMAP PROGRAM AREA
020124  052737  000001  177572          BIS     #BIT00,MMR0     ; ENABLE MMU
020132  052737  000020  172516          BIS     #BIT04,MMR3     ; ENABLE 22-BIT MODE
020140  013702  001720                  MOV     PMIS,R2         ; STORE HIGHEST PAGE
020144  042737  000001  177572          BIC     #BIT00,MMR0     ; DISABLE MMU
                                ;
                                ; SETUP MAPPING REGISTERS AND KMCR
                                ;
020152  052737  000040  172516          BIS     #BIT5,@#MMR3    ; ENABLE UNIBUS MAPPING
020160  013737  177734  001160          MOV     KMCR,$TMP0      ; SAVE KMCR
020166  052737  000400  177730          BIS     #BIT08,DCSR     ; ENABLE DIAGNOSTIC MODE
020174  012737  000067  177734          MOV     #67,KMCR        ; 18-BIT, >32K DISABLED
020202  042737  000400  177730          BIC     #BIT08,DCSR     ; DISABLE DIAGNOSTIC MODE
020210  005037  170200                  CLR     MAPL00          ; CLEAR LOW MAP REGISTER
020214  012737  000001  170202          MOV     #1,MAPH00       ; POINT TO 32K
020222  012777  020334  161734          MOV     #20$,@BE1VEC    ; . LOAD INTERRUPT VECTOR
020230  012777  000340  161730          MOV     #340,@BE1PSW    ;
020236  012737  002000  172354          MOV     #2000,KIPAR6    ; INITIAL POINTER TO ABOVE 32K
                                ;
                                ; DO DATI TO DISABLED MEMORY
                                ;
020244  004737  002450          10$:    JSR     PC,IUBE         ; INTIALISE UBE
020250  005077  161700                  CLR     @BE1BA          ; . ADDRESS TO ACCESS THRU MAP 0
020254  012777  177777  161670          MOV     #-1,@BE1CC      ; . ONE CYCLE
020262  012777  002041  161666          MOV     #2041,@BE1CR1   ; . SET UBE FOR NPR-DATI-1 XFER
```

T57 MAIN MEMORY DISABLE THRU UBE

```
 4767 020270  032777  000200  161660  14$:    BIT     #BIT7,@BE1CR1   ; . IS THE TRANSFER COMPLETE
 4768 020276  001774                          BEQ     14$             ; . NO,THEN WAIT FOR IT TO BE COMPLETE
 4769 020300  032777  000400  161654          BIT     #BIT08,@BE1CR2  ; . NXM SET?
 4770 020306  001002                          BNE     15$             ; . IF SO, BRANCH
 4771 020310  104022                          ERROR   +22             ; . NO NXM
 4772 020312  000412                          BR      25$             ;
 4773 020314  106427  000140          15$:    MTPS    #140            ; . LOWER PRIORITY
 4774 020320  000240                          NOP                     ; . WAIT FOR INTERRUPT
 4775 020322  000240                          NOP                     ;
 4776 020324  104022                          ERROR   +22             ; . NO INTERRUPT ON NXM
 4777 020326  106427  000340                  MTPS    #340            ; . RAISE PRIORITY
 4778 020332  000402                          BR      25$             ; . BRANCH
 4779 020334  062706  000004          20$:    ADD     #4,SP           ; . ADJUST STACK
 4780                                 ;
 4781                                 ; CHANGE PAGE ACCESSED AND CHECK END CONDITIONS
 4782                                 ;
 4783 020340  023702  172354          25$:    CMP     KIPAR6,R2       ; . ALL PAGES DONE?
 4784 020344  103032                          BHIS    30$             ; . IF YES, EXIT
 4785 020346  062737  004000  170200          ADD     #4000,MAPL00    ; . INCREMENT IN 1K
 4786 020354  103002                          BCC     26$             ; . IF NO CARRY, BRANCH
 4787 020356  005237  170202                  INC     MAPH00          ; . OTHERWISE INCR. HIGH MAP
 4788 020362  032737  017777  170200  26$:    BIT     #17777,MAPL00   ; . IS IT ON 4K BOUNDARY?
 4789 020370  001017                          BNE     27$             ; . IF NOT, BRANCH
 4790 020372  062737  000200  172354          ADD     #200,KIPAR6     ; . INCREMENT COUNTER
 4791 020400  052737  000400  177730          BIS     #BIT08,DCSR     ; . ENABLE DIAG. MODE
 4792 020406  005337  177734                  DEC     KMCR            ; . ACCESS NEXT HIGHER LOCATION
 4793 020412  032737  000040  177734          BIT     #BIT05,KMCR     ; . ALL 128K DONE?
 4794 020420  001404                          BEQ     30$             ; . IF SO, BRANCH
 4795 020422  042737  000400  177730          BIC     #BIT08,DCSR     ; . DISABLE DIAG. MODE
 4796 020430  000705                  27$:    BR      10$             ; . DO FOR THE NEXT PAGE
 4797 020432  042737  000040  172516  30$:    BIC     #BIT05,MMR3     ; DISABLE MAPPING
 4798 020440  052737  000400  177730          BIS     #BIT08,DCSR     ; ENABLE DIAG. MODE
 4799 020446  013737  001160  177734          MOV     $TMP0,KMCR      ; RESTORE KMCR
 4800 020454  042737  000400  177730          BIC     #BIT08,DCSR     ; DISABLE DIAG. MODE
 4801
```

```
                                .SBTTL TEST - UNIBUS DEVICE DATOB CYCLE
                                ;
                                ;* THIS TEST WILL SEE IF THE KTJ11-B MODULE CAN EXECUTE
                                ;* A UNIBUS DEVICE DATOB .
                                ;*
                                ;
                                ;  BGNTST
                                ;
                                ;*
                                ;*  SET UP UBE FOR A DATOB CYCLE
                                ;*
                                ;       LET R1 := ADDRESS OF THE WRITE BUFFER
                                ;       LET (R1) := 0                   ;CLEAR OUT THE LOCATION
                                ;       LET BE1DB := 77777              ;DATA PATTERN IS 77777
                                ;       LET BE1CC := -1                 ;SET FOR 1 DATA XFER
                                ;       LET BE1BA := R1                 ;ADDRESS IS FIRST LOCATION IN WRITE BUFFER
                                ;       SET UBE FOR NPR-DATOB-1 XFER
                                ;       SET OFF THE TRANSFER
                                ;       WAIT FOR THE TRANSFER TO COMPLETE
                                ;*
                                ;*  CHECK IF THE TRANSFER OCCURED CORRECTLY
                                ;*
                                ;       IF (R1) NEQ @377 THEN
                                ;       .  ERROR TRANSFER DID NOT EXECUTE CORRECTLY
                                ;       ENDIF
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------------------------

                                ;;*************************************************************************
                                ;*TEST 60        UNIBUS DEVICE DATOB CYCLE TEST
                                ;;*************************************************************************
    020462  000004              TST60:  SCOPE

                                        JSR     PC,IUBE         ; INITIALIZE THE UBE
                                        MOV     @WRTBUF,R1      ; GET POINTER TO WRITE BUFFER
                                        CLR     (R1)            ; CLEAR OUT THE LOCATION
    020476  012777  077777  161444      MOV     @77777,@BE1DB   ; SET DATA BUFFER UP
    020504  012777  177777  161440      MOV     @-1,@BE1CC      ; SET FOR 1 DATA TRANSFER
    020512  010177  161436              MOV     R1,@BE1BA       ; POINT UBE XFER TO WRITE BUFFER
                                ;
                                ; DO THE TRANSFER
                                ;
    020516  012777  003441  161432      MOV     @3441,@BE1CR1   ; SET UP UBE FOR NPR-DATOB-1 XFER
    020524  032777  000200  161424  5$: BIT     @BIT7,@BE1CR1   ; IS THE TRANSFER COMPLETE ?
    020532  001774                     BEQ     5$              ; NO,THEN WAIT FOR IT TO COMPLETE
                                ;
                                ; CHECK THE TRANSFER
                                ;
    020534  021127  000377              CMP     (R1),@377       ; WAS THE TRANSFER CORRECT ?
    020540  001401                      BEQ     TST61           ;; YES GO TO THE NEXT TEST
    020542  104033                      ERROR   .33             ; ERROR TRANSFER WAS INCORRECT
```

```
                                .SBTTL TEST - UNIBUS DEVICE DATIP CYCLE

                                ;* THIS TEST WILL SEE IF THE KTJ11-B MODULE CAN EXECUTE
                                ;* A UNIBUS DEVICE DATIP CYCLE.
                                ;*
                                ;
                                ;  BGNTST
                                ;*
                                ;*  INITIALIZE WRITE BUFFER FOR DATIP CYCLE
                                ;*
                                ;       LET R1 := POINTER TO WRITE BUFFER
                                ;       DO FOR R2 := 1 TO 8
                                ;       . LET (R1)+ := #52525
                                ;       ENDDO
                                ;*
                                ;*  SET UP UBE TO DO 8 DATIP'S TO WRITE BUFFER
                                ;*
                                ;       LET BE1BA := ADDRESS OF WRITE BUFFER
                                ;       LET BE1CC := -8.                ;SET FOR 8 TRANSFERS
                                ;       SET UBE FOR NPR-DATIP-1 XFER
                                ;       SET OFF THE TRANSFER
                                ;       WAIT FOR TRANSFER TO BE COMPLETE
                                ;*
                                ;*  CHECK IF DATIP WAS EXECUTED CORRECTLY
                                ;*
                                ;       LET R1 := ADDRESS OF WRITE BUFFER
                                ;       DO FOR R2 := 1 TO 8
                                ;       . IF (R1)+ NEQ 125252 THEN
                                ;       . . ERROR TRANSFER DID NOT EXECUTE CORRECTLY
                                ;       . ENDIF
                                ;       ENDDO
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------------------

                                ;;*****************************************************************
                                ;*TEST 61       UNIBUS DEVICE DATIP CYCLE TEST
                                ;;*****************************************************************
    020544  000004              TST61:  SCOPE

                                        JSR     PC,IUBE         ; INITIALIZE THE UBE

                                ;
                                ; INITIALIZE WRITE BUFFER FOR DATIP CYCLE
                                ;

    020552  012701  001734              MOV     #WRTBUF,R1      ; GET POINTER TO WRITE BUFFER
    020556  012702  000010              MOV     #8.,R2          ; SET UP LOOP COUNTER
    020562  012721  052525      5$:     MOV     #52525,(R1)+    ; . INITIALIZE WRITE BUFFER LOCATION
    020566  077203                      SOB     R2,5$           ; . HAVE WE DONE IT 8 TIMES ?

                                ;
                                ; SET UP UBE TO 8 DATIP'S TO WRITE BUFFER
                                ;

    020570  012777  001734  161356      MOV     #WRTBUF,@BE1BA  ; SET UP UBE TO WRITE BUFFER
```

T61 UNIBUS DEVICE DATIP CYCLE TEST

```
4910 020576  005077  161360                    CLR     @#BE1CR2          ;
4911 020602  012777  177770  161342            MOV     #-8.,@#BE1CC      ; SET UP FOR 8 TRANSFERS
4912 020610  012777  002441  161340            MOV     #2441,@#BE1CR1    ; SET UP UBE FOR NPR-DATIP-1 XFER
4913 020616  032777  000200  161332  10$:      BIT     #BIT7,@#BE1CR1    ; IS THE TRANSFER COMPLETE ?
4914 020624  001774                            BEQ     10$               ; NO,THEN WAIT FOR IT TO BE COMPLETE
4915
4916                                 ;
4917                                 ; CHECK IF DATIP WAS EXECUTED CORRECTLY
4918                                 ;
4919
4920 020626  012701  001734                    MOV     #WRTBUF,R1        ; GET POINTER TO WRITE BUFFER
4921 020632  012702  000010                    MOV     #8.,R2            ; SET UP LOOP COUNTER
4922 020636  022127  125252  15$:      CMP     (R1)+,#125252     ; . WAS THE TRANSFER CORRECT ?
4923 020642  001401                            BEQ     20$               ; . YES,THEN GO SEE IF WE ARE DONE CHECKING
4924 020644  104033                            ERROR   +33               ; . NO,THEN ERROR IN THE TRANSFER
4925 020646  077205  20$:      SOB     R2,15$            ; . HAVE WE CHECKED ALL THE XFERS
4926
4927
```

```
                        .SBTTL TEST - UNIBUS DEVICE I/O PAGE READ CYCLE

                        ;* THIS TEST  WILL SEE IF THE KTJ11-B MODULE CAN EXECUTE
                        ;* A UNIBUS DEVICE I/O PAGE READ CYCLE. THIS CONSISTS OF
                        ;* THREE DIFFERENT TYPES OF READS :
                        ;*
                        ;*      1. UNIBUS DEVICE PMI I/O PAGE READ CYCLE
                        ;*      2. UNIBUS DEVICE BOOT ROM READ CYCLE
                        ;*      3. UNIBUS DEVICE MAP REGISTER READ CYCLE.
                        ;*
                        ;* IT ALSO WILL CHECK THAT A UNIBUS DEVICE READ CYCLE NOT
                        ;* OF THE ABOVE THREE TYPES SHOULD CAUSE A TIMEOUT.
                        ;*
                        ;
                        ;  BGNTST
                        ;*
                        ;*  DO A UNIBUS DEVICE PMI I/O PAGE READ CYCLE
                        ;*
                        ;       LET BE1BA := ADDRESS OF CSR OF MEMORY BOARD
                        ;       LET BE1CC := -1                         ;SET FOR 1 TRANSFER
                        ;       SET UBE FOR A NPR-DATI-1 DATA XFER
                        ;       SET OFF THE TRANSFER
                        ;       WAIT FOR THE TRANSFER TO BE COMPLETE
                        ;       IF BE1DB NEQ CONTENTS OF MEMORY CSR THEN
                        ;       .  ERROR UNIBUS DEVICE PMI I/O PAGE READ CYCLE DID NOT EXECUTE CORRECTLY
                        ;       ENDIF
                        ;*
                        ;*  DO A UNIBUS DEVICE BOOT ROM READ CYCLE
                        ;*
                        ;       LET BE1BA := 773000
                        ;       LET BE1CC := -1                         ;SET FOR 1 TRANSFER
                        ;       SET UBE FOR A NPR-DATI-1 DATA XFER
                        ;       SET OFF THE TRANSFER
                        ;       WAIT FOR THE TRANSFER TO BE COMPLETE
                        ;       IF BE1DB NEQ @#173000 THEN
                        ;       .  ERROR UNIBUS DEVICE BOOT ROM READ CYCLE DID NOT EXECUTE CORRECTLY
                        ;       ENDIF
                        ;*
                        ;*  DO A UNIBUS DEVICE MAP REGISTER READ CYCLE
                        ;*
                        ;       LET BE1BA := 770200                     ;UNIBUS MAP REGISTER #0
                        ;       LET BE1CC := -1                         ;SET FOR 1 TRANSFER
                        ;       SET UBE FOR A NPR-DATI-1 DATA XFER
                        ;       SET OFF THE TRANSFER
                        ;       WAIT FOR THE TRANSFER TO BE COMPLETE
                        ;       IF BE1DB NEQ @#770200 THEN
                        ;       .  ERROR UNIBUS DEVICE MAP REGISTER READ CYCLE DID NOT EXECUTE CORRECTLY
                        ;       ENDIF
                        ;*
                        ;*    DO AN ILLEGAL UNIBUS DEVICE I/O PAGE READ CYCLE
                        ;*
                        ;       LET BE1BA := 777730,777732,777734       ;DCSR,DDR,KMCR ADDRESS
                        ;       LET BE1CC := -1                         ;1 DATA TRANSFER
                        ;       SET UBE FOR NPR-DATI-1 DATA XFER
                        ;       SET OFF THE TRANSFER
                        ;       WAIT FOR THE TRANSFER TO BE COMPLETE
                        ;       IF TRANSFER DID NOT TIMEOUT THEN
```

```
                                        ;       . ERROR AN ILLEGAL UNIBUS DEVICE CYCLE HAS OCCURED
                                        ;       ENDIF
                                        ;
                                        ;  ENDTST
                                        ;
                                        ;-------------------------------------------------------------------

                                        ;;*****************************************************************
                                        ;*TEST 62       UNIBUS DEVICE I/O PAGE READ CYCLE
                                        ;;*****************************************************************
     020650  000004                     TST62:  SCOPE

                                                JSR     PC,IUBE         ; INITIALIZE THE UBE

                                        ;
                                        ; DO A UNIBUS DEVICE PMI I/O PAGE CYCLE
                                        ;
     020656  013777  001732  161270             MOV     MCSR,@BE1BA     ; GET ADDRESS OF MEMORY CSR ADDRESS
     020664  012777  000003  161270             MOV     #3,@BE1CR2      ; SET THE UPPER TWO ADDRESS BITS
     020672  012777  177777  161252             MOV     #-1,@BE1CC      ; SET FOR 1 TRANSFER
     020700  012777  002041  161250             MOV     #2041,@BE1CR1   ; SET UBE FOR NPR-DATI-1 XFER
     020706  032777  000200  161242  5$:        BIT     #BIT7,@BE1CR1   ; ARE WE DONE THE TRANSFER ?
     020714  001774                             BEQ     5$              ; NO,THEN FOR WAIT FOR IT TO
                                        ;
                                        ; CHECK OUT THE TRANSFER
                                        ;
     020716  027777  161010  161224             CMP     @MCSR,@BE1DB    ; WAS THE TRANSFER CORRECT
     020724  001401                             BEQ     10$             ; YES,THEN GO DO BOOT ROM READ
     020726  104033                             ERROR   +33             ; NO,THEN ERROR IN READ CYCLE
                                        ;
                                        ; DO A UNIBUS DEVICE BOOT ROM READ CYCLE
                                        ;
     020730  012777  173000  161216  10$:        MOV     #173000,@BE1BA  ; GET ADDRESS OF OF THE BOOT ROM
     020736  012777  000003  161216             MOV     #3,@BE1CR2      ; SET THE UPPER TWO ADDRESS BITS
     020744  012777  177777  161200             MOV     #-1,@BE1CC      ; SET FOR 1 TRANSFER
     020752  012777  002041  161176             MOV     #2041,@BE1CR1   ; SET UBE FOR NPR-DATI-1 XFER
     020760  032777  000200  161170  15$:        BIT     #BIT7,@BE1CR1   ; ARE WE DONE THE TRANSFER ?
     020766  001774                             BEQ     15$             ; NO,THEN FOR WAIT FOR IT TO
                                        ;
                                        ; CHECK OUT THE TRANSFER
                                        ;
     020770  052737  000200  177520             BIS     #BIT07,BCSR     ; SELECT RIGHT ROM
     020776  023777  173000  161144             CMP     @#173000,@BE1DB ; WAS THE TRANSFER CORRECT
     021004  001401                             BEQ     20$             ; YES,THEN GO DO UNIBUS DEVICE MAP REGISTER READ
     021006  104033                             ERROR   +33             ; NO,THEN ERROR IN READ CYCLE
                                        ;
                                        ; DO A UNIBUS DEVICE MAP REGISTER READ
                                        ;
     021010  012777  170200  161136  20$:        MOV     #170200,@BE1BA  ; GET ADDRESS OF THE MAP REGISTER
     021016  012777  000003  161136             MOV     #3,@BE1CR2      ; SET THE UPPER TWO ADDRESS BITS
     021024  012777  177777  161120             MOV     #-1,@BE1CC      ; SET FOR 1 TRANSFER
     021032  012777  002041  161116             MOV     #2041,@BE1CR1   ; SET UBE FOR NPR-DATI-1 XFER
     021040  032777  000200  161110  25$:        BIT     #BIT7,@BE1CR1   ; ARE WE DONE THE TRANSFER ?
     021046  001774                             BEQ     25$             ; NO,THEN FOR WAIT FOR IT TO
                                        ;
                                        ; CHECK OUT THE TRANSFER
                                        ;
```

T62 UNIBUS DEVICE I/O PAGE READ CYCLE

```
5040 021050  023777  170200  161072          CMP     @#170200,@BE1DB ; WAS THE TRANSFER CORRECT
5041 021056  001401                          BEQ     30$             ; YES,THEN GO DO ILLEGAL UNIBUS DEVICE I/O READ
5042 021060  104033                          ERROR   .33             ; NO,THEN ERROR IN READ CYCLE
5043                                 ;
5044                                 ; DO AN ILLEGAL UNIBUS DEVICE I/O PAGE READ
5045                                 ;
5046 021062  012777  021164  161074  30$:    MOV     #40$,@BE1VEC    ; SET UP INTERRUPT VECTOR
5047 021070  012777  000340  161070          MOV     #340,@BE1PSW    ;
5048 021076  012701  000003                  MOV     #3,R1           ; DO FOR 3 REGISTERS
5049 021102  012702  177730                  MOV     #177730,R2      ; STARTING WITH DCSR (THEN DDR, KMCR)
5050 021106  010277  161042          31$:    MOV     R2,@BE1BA       ; GET ADDRESS OF THE DIAGNOSTIC REGISTER
5051 021112  012777  000003  161042          MOV     #3,@BE1CR2      ; SET THE UPPER TWO ADDRESS BITS
5052 021120  012777  177777  161024          MOV     #-1,@BE1CC      ; SET FOR 1 TRANSFER
5053 021126  012777  002041  161022          MOV     #2041,@BE1CR1   ; SET UBE FOR NPR-DATI-1 XFER
5054 021134  106427  000140                  MTPS    #140            ; LOWER PRIORITY
5055 021140  032777  000200  161010  35$:    BIT     #BIT7,@BE1CR1   ; ARE WE DONE THE TRANSFER ?
5056 021146  001774                          BEQ     35$             ; NO,THEN FOR WAIT FOR IT TO
5057 021150  000240                          NOP                     ; WAIT FOR INTERRUPT
5058 021152  000240                          NOP                     ;
5059 021154  104033                          ERROR   .33             ; ERROR KTJ11 RESPONDED TO AN ILLEGAL ADDRESS
5060 021156  106427  000340                  MTPS    #340            ; RAISE PRIORITY BACK
5061 021162  000407                          BR      45$             ;
5062 021164  062706  000004          40$:    ADD     #4,SP           ; CLEAN UP THE STACK AND THEN GO DO NEXT TEST
5063 021170  032777  000400  160764          BIT     #BIT08,@BE1CR2  ; NXM SET?
5064 021176  001001                          BNE     45$             ; IF SET, BRANCH
5065 021200  104033                          ERROR   .33             ; NXM NOT SET ON ILLEGAL REFERENCE
5066 021202  005722                  45$:    TST     (R2)+           ; GET ADDRESS OF NEXT REGISTER
5067 021204  004737  002450                  JSR     PC,IUBE         ; CLEAN UP UBE
5068 021210  077142                          SOB     R1,31$          ; DO FOR ALL 3 REGISTERS
5069
```

```
5071                                    .SBTTL TEST - UNIBUS DEVICE I/O PAGE WRITE CYCLE
5072
5073                                    ;* THIS TEST  WILL SEE IF THE KTJ11-B MODULE CAN EXECUTE
5074                                    ;* A UNIBUS DEVICE I/O PAGE WRITE CYCLE. THIS CONSISTS OF
5075                                    ;* TWO DIFFERENT TYPES OF WRITES :
5076                                    ;*
5077                                    ;*      1. UNIBUS DEVICE PMI I/O PAGE WRITE CYCLE
5078                                    ;*      2. UNIBUS DEVICE MAP REGISTER WRITE CYCLE.
5079                                    ;*
5080                                    ;* IT ALSO WILL CHECK THAT A UNIBUS DEVICE WRITE CYCLE NOT
5081                                    ;* OF THE ABOVE TWO TYPES SHOULD CAUSE A TIMEOUT.
5082                                    ;*
5083                                    ;
5084                                    ;  BGNTST
5085                                    ;
5086                                    ;*
5087                                    ;*  DO A UNIBUS DEVICE PMI I/O PAGE WRITE CYCLE
5088                                    ;*
5089                                    ;       LET BE1DB := 52525
5090                                    ;       LET BE1BA := ADDRESS OF CSR OF MEMORY BOARD
5091                                    ;       LET BE1CC := -1                         ;SET FOR 1 TRANSFER
5092                                    ;       SET UBE FOR A NPR-DATO-1 DATA XFER
5093                                    ;       SET OFF THE TRANSFER
5094                                    ;       WAIT FOR THE TRANSFER TO BE COMPLETE
5095                                    ;       IF CONTENTS OF MEMORY CSR NEQ 52525 THEN
5096                                    ;       .  ERROR UNIBUS DEVICE PMI I/O PAGE WRITE CYCLE DID NOT EXECUTE CORRECTLY
5097                                    ;       ENDIF
5098                                    ;*
5099                                    ;*  DO A UNIBUS DEVICE MAP REGISTER WRITE CYCLE
5100                                    ;*
5101                                    ;       LET BE1DB := 52525                      ;DATA PATTERN
5102                                    ;       LET BE1BA := 770200                     ;UNIBUS MAP REGISTER #0
5103                                    ;       LET BE1CC := -1                         ;SET FOR 1 TRANSFER
5104                                    ;       SET UBE FOR A NPR-DATO-1 DATA XFER
5105                                    ;       SET OFF THE TRANSFER
5106                                    ;       WAIT FOR THE TRANSFER TO BE COMPLETE
5107                                    ;       IF MAP REGISTER 0 NEQ #52525 THEN
5108                                    ;       .  ERROR UNIBUS DEVICE MAP REGISTER READ CYCLE DID NOT EXECUTE CORRECTLY
5109                                    ;       ENDIF
5110                                    ;*
5111                                    ;*  DO A UNIBUS DEVICE TO BOOT ROM WRITE CYCLE
5112                                    ;*
5113                                    ;       LET BE1BA := 773000
5114                                    ;       LET BE1CC := -1                         ;SET FOR 1 TRANSFER
5115                                    ;       SET UBE FOR A NPR-DATO-1 DATA XFER
5116                                    ;       SET OFF THE TRANSFER
5117                                    ;       WAIT FOR THE TRANSFER TO BE COMPLETE
5118                                    ;       IF NO TIMEOUT THEN
5119                                    ;       .  ERROR UNIBUS DEVICE BOOT ROM WRITE CYCLE DID NOT EXECUTE CORRECTLY
5120                                    ;       ENDIF
5121                                    ;*
5122                                    ;*  DO AN ILLEGAL UNIBUS DEVICE I/O PAGE WRITE CYCLE
5123                                    ;*
5124                                    ;       LET BE1DB := 52525                      ;DATA PATTERN
5125                                    ;       LET BE1BA := 777730,777732,777734       ;DCSR, DDR, KMCR ADDRESSES
5126                                    ;       LET BE1CC := -1                         ;1 DATA TRANSFER
5127                                    ;       SET UBE FOR NPR-DATO-1 DATA XFER
```

```
                                ;       SET OFF THE TRANSFER
                                ;       WAIT FOR THE TRANSFER TO BE COMPLETE
                                ;       IF TRANSFER DID NOT TIMEOUT THEN
                                ;       . ERROR AN ILLEGAL UNIBUS DEVICE CYCLE HAS OCCURED
                                ;       ENDIF
                                ;       DISABLE UNIBUS MAPPING
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------------------

                                ;;*****************************************************************
                                ;*TEST 63       UNIBUS DEVICE I/O PAGE WRITE CYCLE
                                ;;*****************************************************************
021212  000004                  TST63:  SCOPE

021214  004737  002450                  JSR     PC,IUBE         ; INITIALIZE THE UBE
                                ;
                                ; DO A UNIBUS DEVICE PMI I/O PAGE CYCLE
                                ;
021220  017737  160506  001160          MOV     @MCSR,$TMP0     ; STORE MEMORY CSR
021226  013777  001732  160720          MOV     MCSR,@BE1BA     ; GET ADDRESS OF MEMORY CSR
021234  012777  000003  160720          MOV     #3,@BE1CR2      ; SET UPPER TWO ADDRESS BITS
021242  012777  040000  160700          MOV     #BIT14,@BE1DB   ; GET WRITE PATTERN
021250  012777  177777  160674          MOV     #-1,@BE1CC      ; SET FOR 1 XFER
021256  012777  003041  160672          MOV     #3041,@BE1CR1   ; SET UBE FOR NPR-DATO-1 XFER
021264  032777  000200  160664  5$:     BIT     #BIT7,@BE1CR1   ; IS THE TRANSFER COMPLETE ?
021272  001774                          BEQ     5$              ; NO,THEN WAIT FOR IT TO COMPLETE
                                ;
                                ; CHECK THE TRANSFER
                                ;
021274  032777  040000  160430          BIT     #BIT14,@MCSR    ; WAS THE TRANSFER CORRECT ?
021302  001001                          BNE     10$             ; YES,THEN GO DO A MAP REGISTER WRITE
021304  104033                          ERROR   +33             ; NO,THEN ERROR IN THE TRANSFER
                                ;
                                ; DO A UNIBUS DEVICE MAP REGISTER WRITE CYCLE
                                ;
021306  013777  001160  160416  10$:    MOV     $TMP0,@MCSR     ; RESTORE MEMORY CSR
021314  005037  170200                  CLR     MAPL00          ; CLEAR MAP REGISTER PAIR
021320  005037  170202                  CLR     MAPH00          ;
021324  012777  170200  160622          MOV     #MAPL00,@BE1BA  ; GET ADDRESS OF MAP REGISTER
021332  012777  000003  160622          MOV     #3,@BE1CR2      ; SET UPPER TWO ADDRESS BITS
021340  012777  052525  160602          MOV     #52525,@BE1DB   ; GET WRITE PATTERN
021346  012777  177776  160576          MOV     #-2,@BE1CC      ; SET FOR 2 XFER
021354  012777  003041  160574          MOV     #3041,@BE1CR1   ; SET UBE FOR NPR-DATO-1 XFER
021362  032777  000200  160566  15$:    BIT     #BIT7,@BE1CR1   ; IS THE TRANSFER COMPLETE ?
021370  001774                          BEQ     15$             ; NO,THEN WAIT FOR IT TO COMPLETE
                                ;
                                ; CHECK THE TRANSFER
                                ;
021372  023727  170200  052524          CMP     MAPL00,#52524   ; WAS THE TRANSFER CORRECT ?
021400  001401                          BEQ     16$             ; YES,THEN GO DO A MAP REGISTER WRITE
021402  104033                          ERROR   +33             ; NO,THEN ERROR IN THE TRANSFER
021404  022737  000025  170202  16$:    CMP     #25,MAPH00      ; SECOND WORD OK TOO?
021412  001401                          BEQ     17$             ; IF SO, BRANCH
021414  104033                          ERROR   +33             ; NO,THEN ERROR IN THE TRANSFER
                                ;
```

T63 UNIBUS DEVICE I/O PAGE WRITE CYCLE

```
                                        ; DO A WRITE CYCLE TO BOOT ROM THAT SHOULD CAUSE A TIMEOUT
                                        ;
021416  012777  173000  160530  17$:    MOV     #173000,@BE1BA  ; ADDRESS TO WRITE TO
021424  012777  177777  160520          MOV     #-1,@BE1CC      ; 1 CYCLE
021432  012777  021500  160524          MOV     #19$,@BE1VEC    ; POINT INTERRUPT VECTOR TO PROGRAM
021440  012777  000340  160520          MOV     #340,@BE1PSW    ; AT PRIORITY 7
021446  012777  003041  160502          MOV     #3041,@BE1CR1   ; 1 DATO-NPR
021454  106427  000140                  MTPS    #140            ; LOWER PRIRITY
021460  105777  160472          18$:    TSTB    @BE1CR1         ; DONE?
021464  100375                          BPL     18$             ; WAIT TILL DONE
021466  000240                          NOP
021470  104033                          ERROR   +33             ; DIDN'T TIMEOUT ON BOOT ROM WRITE
021472  106427  000340                  MTPS    #340            ; RAISE PRIORITY
021476  000407                          BR      20$             ;
021500  062706  000004          19$:    ADD     #4,SP           ; ADJUST STACK
021504  032777  000400  160450          BIT     #BIT08,@BE1CR2  ; NXM SET?
021512  001001                          BNE     20$             ; IF SET, BRANCH
021514  104033                          ERROR   +33             ; NXM NOT SET ON BOOT ROM WRITE
                                        ;
                                        ; DO AN ILLEGAL UNIBUS DEVICE I/O PAGE WRITE CYCLE
                                        ;
021516  004737  002450          20$:    JSR     PC,IUBE         ; CLEAN UP UBE
021522  012701  000003                  MOV     #3,R1           ; DO FOR DCSR,DDR,KMCR
021526  012702  177730                  MOV     #177730,R2      ; START WITH DCSR
021532  012777  021640  160424          MOV     #30$,@BE1VEC    ; SET UP TIMEOUT VECTOR
021540  012777  000340  160420          MOV     #340,@BE1PSW    ;
021546  010277  160402          21$:    MOV     R2,@BE1BA       ; GET ADDRESS OF THE DIAGN. REGISTER
021552  012777  000003  160402          MOV     #3,@BE1CR2      ; SET UPPER TWO ADDRESS BITS
021560  012777  052525  160362          MOV     #52525,@BE1DB   ; GET WRITE PATTERN
021566  012777  177777  160356          MOV     #-1,@BE1CC      ; SET FOR 1 XFER
021574  052737  000040  172516          BIS     #BIT5,MMR3      ; ENABLE UNIBUS MAPPING
021602  012777  003041  160346          MOV     #3041,@BE1CR1   ; SET UBE FOR NPR-DATO-1 XFER
021610  005037  177776                  CLR     PSW             ; LOWER PRIORITY
021614  032777  000200  160334  25$:    BIT     #BIT7,@BE1CR1   ; IS THE TRANSFER COMPLETE ?
021622  001774                          BEQ     25$             ; NO,THEN WAIT FOR IT TO COMPLETE
021624  000240                          NOP                     ; WAIT FOR INTERRUPT
021626  000240                          NOP                     ;
021630  104033                          ERROR   +33             ; ERROR - AN ILLEGAL CYCLE DID NOT TIMEOUT
021632  106427  000340                  MTPS    #340            ; RAISE PRIORITY BACK
021636  000407                          BR      35$             ;
021640  062706  000004          30$:    ADD     #4,SP           ; ADJUST THE STACK AND GO TO NEXT TEST
021644  032777  000400  160310          BIT     #BIT08,@BE1CR2  ; NXM SET?
021652  001001                          BNE     35$             ; IF SET, BRANCH
021654  104033                          ERROR   +33             ; NXM NOT SET ON ILLEGAL REFERENCE
021656  005722                  35$:    TST     (R2)+           ; GET NEXT DIAGNOSTIC REGISTER
021660  004737  002450                  JSR     PC,IUBE         ; INITIALISE UBE
021664  077150                          SOB     R1,21$          ; DO FOR ALL OF THEM
```

```
                                .SBTTL TEST - MAPPING REGISTERS TEST USING UBE

                                ;* THIS TEST WRITES DIFFERENT PATTERNS TO MAPPING REGISTERS USING
                                ;* DATI AND DATO CYCLES THRU UBE.
                                ;
                                ;  BGNTST
                                ;
                                ;       ENABLE UNIBUS MAPPING
                                ;       WRITE ALL MAPPING REGISTERS WITH "10" PATTERN
                                ;       READ ALL REGISTERS BACK CLEARING THE ONE JUST READ
                                ;       IF ANY REGISTER DOES NOT HAVE THE PATTERN
                                ;       .  ERROR IN WRITING TO MAP REGISTERS
                                ;       ENDIF
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------

                                ;;*********************************************************************
                                ;*TEST 64       MAPPING REGISTERS TEST USING UBE
                                ;;*********************************************************************
 021666  000004                 TST64:  SCOPE
5249 021670  004737  002450            JSR     PC,IUBE         ; CLEAN UP UBE
                                ;
                                ; DO DATO TO ALL 100 REGISTERS WITH 125252 AS PATTERN
                                ;
5253 021674  012777  170200  160252    MOV     #MAPL00,@8E18A  ; THE STARTING ADDRESS
5254 021702  012777  177700  160242    MOV     #-100,@8E1CC    ; 80 WORDS TO WRITE TO
5255 021710  012777  125252  160232    MOV     #125252,@8E1DB  ; THE PATTERN TO BE WRITTEN
5256 021716  012777  000003  160236    MOV     #3,@8E1CR2      ; ADDRESS <17-16>
5257 021724  012777  003041  160224    MOV     #3041,@8E1CR1   ; GO DO DATO'S
5258 021732  105777  160220        1$: TSTB    @8E1CR1         ; DONE BIT SET?
5259 021736  100375                    BPL     1$              ; WAIT
                                ;
                                ; CHECK THAT ALL REGISTERS WRITTEN OK
                                ;
5263 021740  012701  170200            MOV     #MAPL00,R1      ; START WITH MAP REGISTER 1
5264 021744  010177  160204            MOV     R1,@8E18A       ; STARTING ADDRESS
5265 021750  012777  177777  160174 2$: MOV     #-1,@8E1CC      ; . DO JUST 1 CYCLE
5266 021756  012777  002041  160172    MOV     #2041,@8E1CR1   ; . DATI FROM LOW MAP REGISTER
5267 021764  105777  160166        3$: TSTB    @8E1CR1         ; . DONE BIT SET?
5268 021770  100375                    BPL     3$              ; . IF NOT, WAIT
5269 021772  022777  125252  160150    CMP     #125252,@8E1DB  ; . READ OK?
5270 022000  001401                    BEQ     4$              ; . IF SO, BRANCH
5271 022002  104033                    ERROR   +33             ; . ERROR IN WRITING AND READING MAP REGISTERS
5272 022004  012777  177777  160140 4$: MOV     #-1,@8E1CC      ; . NOW DO HIGH MAP REGISTER
5273 022012  012777  002041  160136    MOV     #2041,@8E1CR1   ; . DATI FROM HIGH REGISTER
5274 022020  105777  160132        5$: TSTB    @8E1CR1         ; . DONE BIT SET?
5275 022024  100375                    BPL     5$              ; . IF NOT, WAIT
5276 022026  022777  000052  160114    CMP     #52,@8E1DB      ; . READ OK?
5277 022034  001401                    BEQ     6$              ; . IF SO, BRANCH
5278 022036  104033                    ERROR   +33             ; . ERROR IN WRITING AND READING MAP REGISTERS
5279 022040  005021                6$: CLR     (R1)+           ; . CLEAR REGISTER JUST WRITTEN
5280 022042  005021                    CLR     (R1)+           ;
5281 022044  022701  170376            CMP     #MAPH37,R1      ; . ALL DONE?
5282 022050  101337                    BHI     2$              ; . IF NOT, BRANCH
```

```
5284                            .SBTTL TEST - UNIBUS DEVICE DATI CYCLE WITH CACHE ENABLED
5285
5286                            ;* THIS TEST WILL SEE IF THE CACHE OPERATES CORRECTLY
5287                            ;* WHEN A UNIBUS DEVICE READ CYCLE IS DONE
5288                            ;*
5289                            ;
5290                            ;  BGNTST
5291                            ;
5292                            ;       ENABLE THE CACHE
5293                            ;       ENABLE UNIBUS MAPPING
5294                            ;*
5295                            ;*  CHECK INITIAL SETTING OF THE CACHE VALID BITS
5296                            ;*
5297                            ;       SELECT THE CACHE VALID BITS
5298                            ;       IF KMCR <15-9> NEQ ^B0000000 THEN
5299                            ;       .  ERROR IN THE CACHE
5300                            ;       ENDIF
5301                            ;*
5302                            ;*  CHECK INITIAL SETTING OF THE CACHE AVAILABILITY BITS
5303                            ;*
5304                            ;       SELECT THE CACHE AVAILABLE SET BITS
5305                            ;       IF KMCR <14-9> NEQ ^B111111 THEN
5306                            ;       .  ERROR IN THE CACHE
5307                            ;       ENDIF
5308                            ;*
5309                            ;*  GO DO TWO CONSECUTIVE READS
5310                            ;*  THE FIRST WILL ALLOCATE SET A
5311                            ;*  THE SECOND WILL CAUSE A CACHE HIT IN SET A
5312                            ;*
5313                            ;       LET MAP REGISTER 1 POINT TO LOCATION 2000
5314                            ;       LET BE1BA := 20000                      ;OCTAL BOUNDARY READ
5315                            ;       LET BE1CR2 := 0                         ;CLEAR UPPER 2 ADDRESS BITS
5316                            ;       LET BE1CC := -2                         ;SET FOR TWO TRANSFERS
5317                            ;       SET UBE FOR NPR-DATI-1 XFER
5318                            ;       SET OFF THE TRANSFER
5319                            ;       WAIT FOR THE TRANSFER TO COMPLETE
5320                            ;       DISABLE UNIBUS MAPPING
5321                            ;       TURN OFF THE CACHE
5322                            ;       SELECT THE CACHE VALID BITS
5323                            ;*
5324                            ;*  CHECK IF THE CACHE OPERATED CORRECTLY
5325                            ;*
5326                            ;       IF KMCR <15-9> NEQ ^B1001000 THEN
5327                            ;       .  ERROR CACHE DID NOT GET UPDATED CORRECTLY
5328                            ;       ENDIF
5329                            ;       SELECT THE CACHE AVAILBILITY BITS
5330                            ;       IF KMCR <15-9> NEQ ^B1000111 THEN
5331                            ;       .  ERROR CACHE DID NOT OPERATE CORRECTLY
5332                            ;       ENDIF
5333                            ;
5334                            ;  ENDTST
5335                            ;
5336                            ;-----------------------------------------------
5337
5338                            ;;*****************************************************************
                                ;*TEST 65       UNIBUS DEVICE CYCLE WITH CACHE ENABLED
                                ;;*****************************************************************
```

T65 UNIBUS DEVICE CYCLE WITH CACHE ENABLED

```
        022052  000004                  TST65:  SCOPE
   5339
   5340 022054  004737  002450                  JSR     PC,IUBE         ; INITIALIZE THE UBE
   5341 022060  052737  000100  177734          BIS     #BIT6,KMCR      ; ENABLE THE DMA CACHE
   5342 022066  032737  000100  177734          BIT     #BIT6,KMCR      ; IS THE CACHE PRESENT ?
   5343 022074  001002                          BNE     1$              ; IF YES, CONTINUE WITH TEST
   5344 022076  000137  022502                  JMP     $EOP            ; OTHERWISE, DO END OF PASS
   5345 022102  012737  160000  170374  1$:     MOV     #160000,MAPL37  ; INITIALIZE MAP REGISTERS
   5346 022110  012737  000077  170376          MOV     #77,MAPH37      ;
   5347 022116  005037  170200                  CLR     MAPL00          ;
   5348 022122  005037  170202                  CLR     MAPH00          ;
   5349 022126  052737  000040  172516          BIS     #BIT5,MMR3      ; ENABLE UNIBUS MAPPING
   5350 022134  052737  000400  177730          BIS     #BIT08,DCSR     ; ENABLE DIAGNOSTIC MODE
   5351 022142  042737  000400  177734          BIC     #BIT08,KMCR     ; MAKE SURE THAT VALID BITS SET
   5352 022150  042737  000400  177730          BIC     #BIT08,DCSR     ; DISABLE DIAGNOSTIC MODE
   5353 022156                          5$:
   5354                                 ;
   5355                                 ; GO DO TWO CONSECUTIVE READS STARTING AT A OCTAL BOUNDARY.
   5356                                 ; THE FIRST READ WILL ALLOCATE CACHE SET A. THE SECOND READ
   5357                                 ; WILL CAUSE A CACHE HIT IN SET A
   5358                                 ;
   5359 022156  012737  000200  170204  10$:     MOV     #200,MAPL01     ; POINT MAP REGISTER TO LOCATION 200
   5360 022164  005037  170206                  CLR     MAPH01
   5361 022170  012777  020000  157756          MOV     #20000,@BE1BA   ; SET TRANSFER ADDRESS TO OCTAL BOUNDARY
   5362 022176  012777  000000  157756          MOV     #0,@BE1CR2      ; CLEAR UPPER 2 ADDRESS BITS
   5363 022204  012777  177776  157740          MOV     #-2,@BE1CC      ; SET UBE FOR TWO TRANSFERS
   5364                                 ;
   5365                                 ; GO DO THE TRANSFER
   5366                                 ;
   5367 022212  012777  002041  157736          MOV     #2041,@BE1CR1   ; SET UBE FOR NPR-DATI-1 XFER
   5368 022220  032777  000200  157730  15$:     BIT     #BIT7,@BE1CR1   ; IS THE TRANSFER COMPLETE ?
   5369 022226  001774                          BEQ     15$             ; NO,THEN WAIT FOR IT TO BE COMPLETE
   5370                                 ;
   5371                                 ; CHECK THE VALID BITS IN THE KMCR
   5372                                 ;
   5373 022230  013703  177734                  MOV     KMCR,R3         ; PUT KMCR INTO R3 FOR MASKING
   5374 022234  042703  000777                  BIC     #777,R3         ; MASK OUT BITS 0-8
   5375 022240  022703  110000                  CMP     #110000,R3      ; ARE THE BITS SET CORRECTLY ?
   5376 022244  001401                          BEQ     20$             ; YES,THEN GO CHECK THE AVAILIBILITY BITS
   5377 022246  104033                          ERROR   +33             ; NO,THEN ERROR IN DMA CACHE
   5378 022250  005037  172516          20$:    CLR     MMR3            ; DISABLE MAPPING
   5379 022254  042737  000100  177734          BIC     #BIT06,KMCR     ; AND CACHE
   5380 022262  023777  000202  157660          CMP     @#202,@BE1DB    ; DATA READ OK?
   5381 022270  001401                          BEQ     TST66           ;;IF OK, EXIT TEST
   5382 022272  104033                          ERROR   +33             ; OTHERWISE ERROR IN CACHE
   5383
```

```
                                .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
                                ;;*****************************************************************
    022274  000004              TST66:  SCOPE

 022276  004737  002450                 JSR     PC,IUBE                 ; CLEAR UBE
 022302  052737  000005  172100         BIS     #BIT02!BIT00,@#172100   ; SET WRONG PARITY IN MEMORY CSR
 022310  042737  003740  172100         BIC     #3740,@#172100          ; CLEAR CHECK BITS
 022316  005037  000000                 CLR     @#0                     ; WRITE LOCATION 0 WITH WRONG PARITY
 022322  042737  000004  172100         BIC     #BIT02,@#172100         ; CLEAR WRONG PARITY
 022330  052737  000040  172516         BIS     #BIT05,MMR3             ; ENABLE MEMORY MAPPING
 022336  042737  000100  177734         BIC     #BIT06,KMCR             ; DISABLE JUST TO MAKE SURE
 022344  052737  000100  177734         BIS     #BIT06,KMCR             ; ENABLE CACHE
 022352  012737  000000  170204         MOV     #0,MAPL01               ; POINT MAP REGISTER TO LOCATION 200
 022360  005037  170206                 CLR     MAPH01
 022364  012777  020000  157562         MOV     #20000,@BE1BA           ; SET TRANSFER ADDRESS TO OCTAL BOUNDARY
 022372  012777  000000  157562         MOV     #0,@BE1CR2              ; CLEAR UPPER 2 ADDRESS BITS
 022400  012777  177777  157544         MOV     #-1,@BE1CC              ; SET UBE FOR TWO TRANSFERS
                                ;
                                ; GO DO THE TRANSFER
                                ;
 022406  012777  002041  157542         MOV     #2041,@BE1CR1           ; SET UBE FOR NPR-DATI-1 XFER
 022414  032777  000200  157534 15$:    BIT     #BIT7,@BE1CR1           ; IS THE TRANSFER COMPLETE ?
 022422  001774                         BEQ     15$                     ; NO,THEN WAIT FOR IT TO BE COMPLETE
 022424  032737  177000  177734         BIT     #177000,KMCR            ; WAS ALLOCATED?
 022432  001401                         BEQ     16$                     ; IF NOT, BRANCH
 022434  104032                         ERROR   +32                     ; IF WAS, REPORT ERROR
 022436  005037  000000         16$:    CLR     @#0                     ; REWRITE 0 WITH RIGHT PARITY
 022442  005037  172516                 CLR     MMR3                    ; CLEAR MAPPING ENABLED
 022446  042737  000001  172100         BIC     #BIT00,@#172100

 022454                         UBEM:
                                ;;*****************************************************************
                                ;*TEST 67       DUMMY TEST
                                ;;*****************************************************************
    022454  000004              TST67:  SCOPE
 022456  022737  000001  001206         CMP     #1,$PASS                        ; SECOND PASS?
 022464  001006                         BNE     $EOP                            ; IF NOT, GOTO EOP
 022466  012737  000012  022530         MOV     #12,$EOPCT                      ; AFTER 1ST PASS
 022474  012737  000012  022536         MOV     #12,$ENDCT                      ; PRINT EVERY 10TH
```

```
                                        ;
                                        ; END OF PASS ROUTINE
                                        ;
                                        .SBTTL  END OF PASS ROUTINE
                                        ;;**********************************************************************
                                        ;*INCREMENT THE PASS NUMBER ($PASS)
                                        ;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
                                        ;*TYPE "END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYY"
                                        ;*WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS
                                        ;*IF THERES A MONITOR GO TO IT
                                        ;*IF THERE ISN'T JUMP TO RESTART
022502                                  $EOP:
022502  000004                                  SCOPE
022504  005037  001102                         CLR     $TSTNM          ;;ZERO THE TEST NUMBER
022510  005037  001164                         CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
022514  005237  001206                         INC     $PASS           ;;INCREMENT THE PASS NUMBER
022520  042737  100000  001206                BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
022526  005327                                  DEC     (PC)+           ;;LOOP?
022530  000001                          $EOPCT: .WORD   1
022532  003063                                  BGT     $DOAGN          ;;YES
022534  012737                                  MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
022536  000001                          $ENDCT: .WORD   1
022540  022530                                  $EOPCT
022542  104401  022550                         TYPE    ,65$            ;;TYPE ASCIZ STRING
022546  000407                                  BR      64$             ;;GET OVER THE ASCIZ
                                        ;;65$:  .ASCIZ  <12><15>/END PASS #/
022566                                  64$:
022566  013746  001206                         MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
                                                                        ;;TYPE PASS NUMBER
022572  104405                                  TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
022574  104401  022602                         TYPE    ,67$            ;;TYPE ASCIZ STRING
022600  000421                                  BR      66$             ;;GET OVER THE ASCIZ
                                        ;;67$:  .ASCIZ  /  TOTAL ERRORS SINCE LAST REPORT /
022644                                  66$:
022644  013746  001112                         MOV     $ERTTL,-(SP)    ;;SAVE $ERTTL FOR TYPEOUT
                                                                        ;;TOTAL NUMBER OF ERRORS
022650  104405                                  TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
022652  104401  001175                         TYPE    ,$CRLF          ;;TYPE CARRIAGE RETURN, LINE FEED
022656  005037  001112                         CLR     $ERTTL          ;;CLEAR ERROR TOTAL
022662  013700  000042                  $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
022666  001405                                  BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
022670  000005                                  RESET                   ;;CLEAR THE WORLD
022672  004710                          $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
022674  000240                                  NOP                     ;;SAVE ROOM
022676  000240                                  NOP                     ;;FOR
022700  000240                                  NOP                     ;;ACT11
022702                                  $DOAGN:
022702  000137                                  JMP     @(PC)+          ;;RETURN
022704  003272                          $RTNAD: .WORD   RESTART
022706     377     377     000          $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
                                                .EVEN

                                        ;
                                        ; SYSMAC ROUTINES
                                        ;
```

```
 5445                                   .SBTTL  SCOPE HANDLER ROUTINE
                                        ;;**********************************************************************
                                        ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
                                        ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
                                        ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
                                        ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                        ;*SW14=1         LOOP ON TEST
                                        ;*SW11=1         INHIBIT ITERATIONS
                                        ;*SW09=1         LOOP ON ERROR
                                        ;*SW08=1         LOOP ON TEST IN SWR<5:0>
                                        ;*CALL
                                        ;*      SCOPE            ;;SCOPE=IOT
 022712                                 $SCOPE:
 022712  104407                                 CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
 022714  032777  040000  156216         1$:     BIT     @#BIT14,@SWR    ;;LOOP ON PRESENT TEST?
 022722  001117                                 BNE     $OVER           ;;YES IF SW14=1
                                        ;@@@@@START OF CODE FOR THE XOR TESTER@@@@@
 022724  000416                         $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                                                        ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
 022726  013746  000004                         MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
 022732  012737  022752  00000(                  MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
 022740  005737  177060                         TST     @#177060        ;;TIME OUT ON XOR?
 022744  012637  000004                         MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
 022750  000466                                 BR      $SVLAD          ;;GO TO THE NEXT TEST
 022752  022626                         5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
 022754  012637  000004                         MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
 022760  000426                                 BR      7$              ;;LOOP ON THE PRESENT TEST
 022762                                 6$:;@@@@@END OF CODE FOR THE XOR TESTER@@@@@
 022762  032777  000400  156150                 BIT     @#BIT08,@SWR    ;;LOOP ON SPEC. TEST?
 022770  001407                                 BEQ     2$              ;;BR IF NO
 022772  017746  156142                         MOV     @SWR,-(SP)      ;;SET DESIRED TEST NUM. FROM SWR
 022776  042716  000300                         BIC     #$SWRMK,(SP)    ;;STRIP AWAY UNDESIRED BITS
 023002  122637  001102                         CMPB    (SP)+,$TSTNM    ;;ON THE RIGHT TEST?
 023006  001465                                 BEQ     $OVER           ;;BR IF YES
 023010  105737  001103                 2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
 023014  001421                                 BEQ     3$              ;;BR IF NO
 023016  123737  001115  001103                 CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
 023024  101015                                 BHI     3$              ;;BR IF NO
 023026  032777  001000  156104                 BIT     @#BIT09,@SWR    ;;LOOP ON ERROR?
 023034  001404                                 BEQ     4$              ;;BR IF NO
 023036  013737  001110  001106         7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
 023044  000446                                 BR      $OVER
 023046  105037  001103                 4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
 023052  005037  001164                         CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
 023056  000415                                 BR      1$              ;;ESCAPE TO THE NEXT TEST
 023060  032777  004000  156052         3$:     BIT     @#BIT11,@SWR    ;;INHIBIT ITERATIONS?
 023066  001011                                 BNE     1$              ;;BR IF YES
 023070  005737  001206                         TST     $PASS           ;;IF FIRST PASS OF PROGRAM
 023074  001406                                 BEQ     1$              ;;      INHIBIT ITERATIONS
 023076  005237  001104                         INC     $ICNT           ;;INCREMENT ITERATION COUNT
 023102  023737  001164  001104                 CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
 023110  002024                                 BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
 023112  012737  000001  001104         1$:     MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
 023120  013737  023176  001164                 MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
 023126  105237  001102                 $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
 023132  113737  001102  001204                 MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
 023140  011637  001106                         MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
```

SCOPE HANDLER ROUTINE

```
023144  011637  001110                  $OVER:  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
023150  005037  001166                          CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
023154  112737  000001  001115                  MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
023162  013777  001102  155752                  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
023170  013716  001106                          MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
023174  000002                                  RTI                     ;;FIXES PS
023176  000001                          $MXCNT: 1                       ;;MAX. NUMBER OF ITERATIONS
   5446                                 .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
                                        ;*
023200  105737  001157          $TYPE:  TSTB    $TPFLG                  ;;IS THERE A TERMINAL?
023204  100002                          BPL     1$                      ;;BR IF YES
023206  000000                          HALT                            ;;HALT HERE IF NO TERMINAL
023210  000430                          BR      3$                      ;;LEAVE
023212  010046                  1$:     MOV     R0,-(SP)                ;;SAVE R0
023214  017600  000002                  MOV     @2(SP),R0               ;;GET ADDRESS OF ASCIZ STRING
023220  122737  000001  001220          CMPB    #APTENV,$ENV            ;;RUNNING IN APT MODE
023226  001011                          BNE     62$                     ;;NO,GO CHECK FOR APT CONSOLE
023230  132737  000100  001221          BITB    #APTSPOOL,$ENVM         ;;SPOOL MESSAGE TO APT
023236  001405                          BEQ     62$                     ;;NO,GO CHECK FOR CONSOLE
023240  010037  023250                  MOV     R0,61$                  ;;SETUP MESSAGE ADDRESS FOR APT
023244  004737  025656                  JSR     PC,$ATY3                ;;SPOOL MESSAGE TO APT
023250  000000                  61$:    .WORD   0                       ;;MESSAGE ADDRESS
023252  132737  000040  001221  62$:    BITB    #APTCSUP,$ENVM          ;;APT CONSOLE SUPPRESSED
023260  001003                          BNE     60$                     ;;YES,SKIP TYPE OUT
023262  112046                  2$:     MOVB    (R0)+,-(SP)             ;;PUSH CHARACTER TO BE TYPED ONTO STACK
023264  001005                          BNE     4$                      ;;BR IF IT ISN'T THE TERMINATOR
023266  005726                          TST     (SP)+                   ;;IF TERMINATOR POP IT OFF THE STACK
023270  012600                  60$:    MOV     (SP)+,R0                ;;RESTORE R0
023272  062716  000002          3$:     ADD     #2,(SP)                 ;;ADJUST RETURN PC
023276  000002                          RTI                             ;;RETURN
023300  122716  000011          4$:     CMPB    #HT,(SP)                ;;BRANCH IF <HT>
023304  001430                          BEQ     8$
023306  122716  000200                  CMPB    #CRLF,(SP)              ;;BRANCH IF NOT <CRLF>
023312  001006                          BNE     5$
023314  005726                          TST     (SP)+                   ;;POP  <CR><LF> EQUIV
023316  104401                          TYPE                            ;;TYPE A CR AND LF
023320  001175                          $CRLF
023322  105037  023530                  CLRB    $CHARCNT                ;;CLEAR CHARACTER COUNT
023326  000755                          BR      2$                      ;;GET NEXT CHARACTER
023330  004737  023412          5$:     JSR     PC,$TYPEC               ;;GO TYPE THIS CHARACTER
023334  123726  001156          6$:     CMPB    $FILLC,(SP)+            ;;IS IT TIME FOR FILLER CHARS.?
023340  001350                          BNE     2$                      ;;IF NO GO GET NEXT CHAR.
023342  013746  001154                  MOV     $NULL,-(SP)             ;;GET # OF FILLER CHARS. NEEDED
                                                                        ;;AND THE NULL CHAR.
```

TYPE ROUTINE

```
023346  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
023352  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
023354  004737  023412                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
023360  105337  023530                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
023364  000770                          BR      7$              ;;LOOP
                                ;HORIZONTAL TAB PROCESSOR
023366  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
023372  004737  023412          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
023376  132737  000007  023530          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
023404  001372                          BNE     9$              ;;TAB STOP
023406  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
023410  000724                          BR      2$              ;;GET NEXT CHARACTER
023412                          $TYPEC:
023412  105777  155526                  TSTB    @$TKS           ;;CHAR IN KYBD BUFFER?          ;MJD001
023416  100022                          BPL     10$             ;;BR IF NOT                     ;MJD001
023420  017746  155522                  MOV     @$TKB,-(SP)     ;;GET CHAR                      ;MJD001
023424  042716  177600                  BIC     #177600,(SP)    ;;STRIP EXTRANEOUS BITS         ;MJD001
023430  122716  000023                  CMPB    #$XOFF,(SP)     ;;WAS CHAR XOFF                 ;MJD001
023434  001012                          BNE     102$            ;;BR IF NOT                     ;MJD001
023436                          101$:                                                           ;MJD001
023436  105777  155502                  TSTB    @$TKS           ;;WAIT FOR CHAR                 ;MJD001
023442  100375                          BPL     101$                                            ;MJD001
023444  117716  155476                  MOVB    @$TKB,(SP)      ;;GET CHAR                      ;MJD001
023450  042716  177600                  BIC     #177600,(SP)    ;;STRIP IT                      ;MJD001
023454  122716  000021                  CMPB    #$XON,(SP)      ;;WAS IT XON?                   ;MJD001
023460  001366                          BNE     101$            ;;BR IF NOT                     ;MJD001
023462                          102$:                                                           ;MJD001
023462  005726                          TST     (SP)+           ;;FIX STACK                     ;MJD001
023464                          10$:                                                            ;MJD001
023464  105777  155460                  TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
023470  100375                          BPL     10$                                             ;MJD001
023472  116677  000002  155452          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
023500  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
023506  001003                          BNE     1$              ;;BRANCH IF NO
023510  105037  023530                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
023514  000406                          BR      $TYPEX          ;;EXIT
023516  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
023524  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
023526  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
023530  000000                  $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
023532  000207                  $TYPEX: RTS     PC
5447                            .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
                                ;;******************************************************************
                                ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
                                ;*OCTAL (ASCII) NUMBER AND TYPE IT.
                                ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
                                ;*CALL:
                                ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                ;*      TYPOS                   ;;CALL FOR TYPEOUT
                                ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
                                ;*      .BYTE   M               ;;M=1 OR 0
                                ;*                                      ;;1=TYPE LEADING ZEROS
                                ;*                                      ;;0=SUPPRESS LEADING ZEROS
                                ;*
                                ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
                                ;*$TYPOS OR $TYPOC
                                ;*CALL:
```

```
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPON                   ;;CALL FOR TYPEOUT
                                        ;*
                                        ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPOC                   ;;CALL FOR TYPEOUT
023534  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
023540  116637  000001  023757          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
023546  112637  023761                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
023552  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
023556  000406                          BR      $TYPON
023560  112737  000001  023757  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
023566  112737  000006  023761          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
023574  112737  000005  023756  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
023602  010346                          MOV     R3,-(SP)        ;;SAVE R3
023604  010446                          MOV     R4,-(SP)        ;;SAVE R4
023606  010546                          MOV     R5,-(SP)        ;;SAVE R5
023610  113704  023761                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
023614  005404                          NEG     R4
023616  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
023622  110437  023760                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
023626  113704  023757                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
023632  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
023636  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
023640  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
023642  000404                          BR      3$              ;;GO DO MSB
023644  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
023646  006105                          ROL     R5
023650  006105                          ROL     R5
023652  010503                          MOV     R5,R3
023654  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
023656  105337  023760                  DECB    $OMODE          ;;TYPE THIS DIGIT?
023662  100016                          BPL     7$              ;;BR IF NO
023664  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
023670  001002                          BNE     4$              ;;TEST FOR 0
023672  005704                          TST     R4              ;;SUPPRESS THIS 0?
023674  001403                          BEQ     5$              ;;BR IF YES
023676  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
023700  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
023704  052703  000040          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
023710  110337  023754                  MOVB    R3,8$           ;;SAVE FOR TYPING
023714  104401  023754                  TYPE    ,8$             ;;GO TYPE THIS DIGIT
023720  105337  023756          7$:     DECB    $OCNT           ;;COUNT BY 1
023724  003347                          BGT     2$              ;;BR IF MORE TO DO
023726  002402                          BLT     6$              ;;BR IF DONE
023730  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
023732  000744                          BR      2$              ;;GO DO THE LAST DIGIT
023734  012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
023736  012604                          MOV     (SP)+,R4        ;;RESTORE R4
023740  012603                          MOV     (SP)+,R3        ;;RESTORE R3
023742  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
023750  012616                          MOV     (SP)+,(SP)
023752  000002                          RTI                     ;;RETURN
023754     000                  8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
023755     000                          .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
023756     000          $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
```

BINARY TO OCTAL (ASCII) AND TYPE

```
                                $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
                                $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
  5448                          .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
023762                          $TYPDS:
023762  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
023764  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
023766  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
023770  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
023772  010546                        MOV     R5,-(SP)        ;;PUSH R5 ON STACK
023774  012746  020200                MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
024000  016605  000020                MOV     20(SP),R5       ;;GET THE INPUT NUMBER
024004  100004                        BPL     1$              ;;BR IF INPUT IS POS.
024006  005405                        NEG     R5              ;;MAKE THE BINARY NUMBER POS.
024010  112766  000055  000001        MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
024016  005000                1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
024020  012703  024176                MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
024024  112723  000040                MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
024030  005002                2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
024032  016001  024166                MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
024036  160105                3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
024040  002402                        BLT     4$              ;;BR IF DONE
024042  005202                        INC     R2              ;;INCREASE THE BCD DIGIT BY 1
024044  000774                        BR      3$
024046  060105                4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
024050  005702                        TST     R2              ;;CHECK IF BCD DIGIT=0
024052  001002                        BNE     5$              ;;FALL THROUGH IF 0
024054  105716                        TSTB    (SP)            ;;STILL DOING LEADING 0'S?
024056  100407                        BMI     7$              ;;BR IF YES
024060  106316                5$:     ASLB    (SP)            ;;MSD?
024062  103003                        BCC     6$              ;;BR IF NO
024064  116663  000001  177777        MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
024072  052702  000060          6$:   BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
024076  052702  000040          7$:   BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
024102  110223                        MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
024104  005720                        TST     (R0)+           ;;JUST INCREMENTING
024106  020027  000010                CMP     R0,#10          ;;CHECK THE TABLE INDEX
024112  002746                        BLT     2$              ;;GO DO THE NEXT DIGIT
024114  003002                        BGT     8$              ;;GO TO EXIT
024116  010502                        MOV     R5,R2           ;;GET THE LSD
024120  000764                        BR      6$              ;;GO CHANGE TO ASCII
024122  105726                8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
024124  100003                        BPL     9$              ;;BR IF NO
024126  116663  177777  177776        MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
024134  105013                9$:     CLRB    (R3)            ;;SET THE TERMINATOR
024136  012605                        MOV     (SP)+,R5        ;;POP STACK INTO R5
024140  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
024142  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
024144  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
```

CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```
024146  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
024150  104401  024176                  TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
024154  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
024162  012616                          MOV     (SP)+,(SP)
024164  000002                          RTI                     ;;RETURN TO USER
024166  023420                  $DTBL:  10000.
024170  001750                          1000.
024172  000144                          100.
024174  000012                          10.
024176                          $DBLK:  .BLKW   4
 5449                           .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.
024206  022737  000176  001140  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
024214  001074                          BNE     15$             ;;BRANCH IF NO
024216  105777  154722                  TSTB    @$TKS           ;;CHAR THERE?
024222  100071                          BPL     15$             ;;IF NO, DON'T WAIT AROUND
024224  117746  154716                  MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
024230  042716  177600                  BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
024234  022726  000007                  CMP     #7,(SP)+        ;;IS IT A CONTROL G?
024240  001062                          BNE     15$             ;;NO, RETURN TO USER
024242  123727  001134  000001          CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
024250  001456                          BEQ     15$             ;;BRANCH IF YES
024252  104401  024743                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
024256  104401  024750          $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
024262  013746  000176                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
024266  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
024270  104401  024761                  TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
024274  005046                  19$:    CLR     -(SP)           ;;CLEAR COUNTER
024276  005046                          CLR     -(SP)           ;;THE NEW SWR
024300  105777  154640          7$:     TSTB    @$TKS           ;;CHAR THERE?
024304  100375                          BPL     7$              ;;IF NOT TRY AGAIN
024306  117746  154634                  MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
024312  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
024316  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
024322  001005                          BNE     10$             ;;BRANCH IF NOT
024324  104401  024736                  TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
024330  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
024334  000757                          BR      19$             ;;LET'S TRY IT AGAIN
024336  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
024342  001022                          BNE     16$             ;;BRANCH IF NO
024344  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
024350  001403                          BEQ     11$             ;;BRANCH IF YES
024352  016677  000002  154560          MOV     2(SP),@SWR      ;;SAVE NEW SWR
024360  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
024364  104401  001175          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
024370  123727  001135  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
024376  001003                          BNE     15$             ;;BRANCH IF NOT
024400  012777  000100  154536          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
024406  000002                  15$:    RTI                     ;;RETURN
024410  004737  023412          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
024414  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
```

TTY INPUT ROUTINE

```
024420  002420                          BLT     18$             ;;BRANCH IF YES
024422  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
024426  003015                          BGT     18$             ;;BRANCH IF YES
024430  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
024434  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
024440  001403                          BEQ     17$             ;;BRANCH IF YES
024442  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
024444  006316                          ASL     (SP)            ;;   CHAR OVER TO MAKE
024446  006316                          ASL     (SP)            ;;   ROOM FOR NEW ONE.
024450  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
024454  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
024460  000707                          BR      7$              ;;GET THE NEXT ONE
024462  104401  001174          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
024466  000720                          BR      20$             ;;SIMULATE CONTROL-U
                                .DSABL  LSB
                                ;;**************************************************************************
                                ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
                                ;*CALL:
                                ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
                                ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
                                ;*                              ;;WITH PARITY BIT STRIPPED OFF
                                ;
024470  011646                  $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
024472  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
024500  105777  154440          1$:     TSTB    @$TKS           ;;WAIT FOR
024504  100375                          BPL     1$              ;;A CHARACTER
024506  117766  154434  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
024514  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
024522  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
024530  001013                          BNE     3$              ;;BRANCH IF NO
024532  105777  154406          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
024536  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
024540  117746  154402                  MOVB    @#177562,-(SP)  ;;GET CHARACTER
024544  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
024550  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
024554  001366                          BNE     2$              ;;IF NOT DISCARD IT
024556  000750                          BR      1$              ;;YES, RESUME
024560  026627  000004  000021  3$:     CMP     4(SP),#$XON     ;;IS IT A RANDOM XON?                   ;RAN001
024566  001744                          BEQ     1$              ;;BRANCH IF YES                         ;RAN001
024570  026627  000004  000140          CMP     4(SP),#140      ;;IS IT UPPER CASE?
024576  002407                          BLT     4$              ;;BRANCH IF YES
024600  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
024606  003003                          BGT     4$              ;;BRANCH IF YES
024610  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
024616  000002                  4$:     RTI                     ;;GO BACK TO USER
                                ;;**************************************************************************
                                ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
                                ;*CALL:
                                ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
                                ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
                                ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
024620  010346                  $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
024622  012703  024726          1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
024626  022703  024736          2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
024632  101405                          BLOS    4$              ;;BR IF YES
024634  104410                          RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
024636  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
```

TTY INPUT ROUTINE

```
024640  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
024644  001003                  BNE     3$              ;;SKIP IF NOT
024646  104401  001174          4$:     TYPE    ,$QUES          ;;TYPE A '?'
024652  000763                  BR      1$              ;;CLEAR THE BUFFER AND LOOP
024654  111337  024724          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
024660  104401  024724                  TYPE    ,9$
024664  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
024670  001356                  BNE     2$              ;;LOOP IF NOT RETURN
024672  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
024676  104401  001176                  TYPE    ,$LF            ;;TYPE A LINE FEED
024702  012603                  MOV     (SP)+,R3        ;;RESTORE R3
024704  011646                  MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
024706  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
024714  012766  024726  000004          MOV     #$TTYIN,4(SP)
024722  000002                  RTI                     ;;RETURN
024724     000          9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
024725     000                  .BYTE   0               ;;TERMINATOR
024726                  $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
024736     136     125     015  $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL "U"
024741     012     000
024743     136     107     015  $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL "G"
024746     012     000
024750     015     012     123  $MSWR:  .ASCIZ  <15><12>/SWR = /
024753     127     122     040
024756     075     040     000
024761     040     040     116  $MNEW:  .ASCIZ  /  NEW = /
024764     105     127     040
024767     075     040     000
5450                            .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
                                ;;**************************************************************
                                ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
                                ;*CHANGE IT TO BINARY.
                                ;*CALL:
                                ;*      RDOCT                   ;;READ AN OCTAL NUMBER
                                ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
                                ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT
024772  011646                  $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
024774  016666  000004  000002          MOV     4(SP),2(SP)     ;;INPUT NUMBER
025002  010046                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
025004  010146                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
025006  010246                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
025010  104411                  1$:     RDLIN                   ;;READ AN ASCIZ LINE
025012  012600                  MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
025014  005001                  CLR     R1              ;;CLEAR DATA WORD
025016  005002                  CLR     R2
025020  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
025022  001412                  BEQ     3$              ;;IF ZERO GET OUT
025024  006301                  ASL     R1              ;;*2
025026  006102                  ROL     R2
025030  006301                  ASL     R1              ;;*4
025032  006102                  ROL     R2
025034  006301                  ASL     R1              ;;*8
025036  006102                  ROL     R2
025040  042716  177770          BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
025044  062601                  ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
025046  000764                  BR      2$              ;;LOOP
025050  005726                  3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
```

READ AN OCTAL NUMBER FROM THE TTY

```
025052  010166  000012                  MOV     R1,12(SP)       ;;SAVE THE RESULT
025056  010237  025072                  MOV     R2,$HIOCT
025062  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
025064  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
025066  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
025070  000002                          RTI                     ;;RETURN
025072  000000                  $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
 5451                                   .SBTTL  READ A DECIMAL NUMBER FROM THE TTY
                                        ;;**************************************************************
                                        ;*THIS ROUTINE WILL READ A DECIMAL (ASCII) NUMBER FROM THE TTY AND
                                        ;*CHANGE IT TO BINARY. IF TOO MANY CHARACTERS OR ANY ILLEGAL CHARACTERS
                                        ;*ARE READ A "?" FOLLOWED BY A CARRIAGE RETURN-LINE FEED WILL BE TYPED.
                                        ;*THE COMPLETE NUMBER MUST BE RETYPED. THE INPUT IS TERMINATED BY THE
                                        ;*USER TYPING A CARRIAGE RETURN. THE RANGE OF THE INPUT NUMBER IS
                                        ;*POSITIVE 32767 TO NEGATIVE 32768.
                                        ;*CALL:
                                        ;*      RDDEC                   ;;READ A DECIMAL NUMBER
                                        ;*      RETURN HERE             ;;NUMBER IS ON TOP OF THE STACK
                                        ;
025074  011646                  $RDDEC: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR
025076  016666  000004  000002          MOV     4(SP),2(SP)     ;;THE INPUT NUMBER
025104  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
025106  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
025110  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
025112  104411                  1$:     RDLIN                   ;;READ AN ASCIZ LINE
025114  012600                          MOV     (SP)+,R0        ;;ADDRESS OF 1ST CHAR.
025116  010037  025242                  MOV     R0,6$           ;;SAVE INCASE OF BAD INPUT
025122  005046                          CLR     -(SP)           ;;CLEAR DATA WORD
025124  005002                          CLR     R2              ;;SIGN SET POSITIVE
025126  122710  000055                  CMPB    #'-,(R0)        ;;SEE IF A MINUS SIGN WAS TYPED
025132  001001                          BNE     2$              ;;BR IF NO MINUS SIGN
025134  112002                          MOVB    (R0)+,R2        ;;SAVE FOR LATER USE
025136  112001                  2$:     MOVB    (R0)+,R1        ;;PICKUP THIS CHARACTER
025140  001424                          BEQ     3$              ;;GET OUT IF ZERO
025142  122701  000060                  CMPB    #'0,R1          ;;MAKE SURE THIS CHARACTER
025146  003032                          BGT     5$              ;;IS A DIGIT BETWEEN 0 & 9
025150  122701  000071                  CMPB    #'9,R1
025154  002427                          BLT     5$
025156  032716  170000                  BIT     #^C7777,(SP)    ;;DON'T LET NUMBER GET TO BIG
025162  001024                          BNE     5$              ;;BR IF NUMBER WOULD OVERFLOW
025164  006316                          ASL     (SP)            ;;*2
025166  011646                          MOV     (SP),-(SP)      ;;SAVE FOR LATER
025170  006316                          ASL     (SP)            ;;*4
025172  006316                          ASL     (SP)            ;;*8.
025174  062616                          ADD     (SP)+,(SP)      ;;*10.
025176  102416                          BVS     5$              ;;OVERFLOW ISN'T ALLOWED
025200  162701  000060                  SUB     #'0,R1          ;;STRIP AWAY THE ASCII JUNK
025204  060116                          ADD     R1,(SP)         ;;ADD IN THIS DIGIT
025206  102412                          BVS     5$              ;;OVERFLOW ISN'T ALLOWED
025210  000752                          BR      2$              ;;LOOP
025212  005702                  3$:     TST     R2              ;;CHECK IF NUMBER IS NEG
025214  001401                          BEQ     4$              ;;BR IF NO
025216  005416                          NEG     (SP)            ;;YES--NEGATE THE NUMBER
025220  012666  000012          4$:     MOV     (SP)+,12(SP)    ;;SAVE THE RESULT
025224  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
025226  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
025230  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
```

READ A DECIMAL NUMBER FROM THE TTY

```
025232  000002                          RTI                     ;;RETURN
025234  005726                  5$:     TST     (SP)+           ;;CLEAN PARTIAL NUMBER FROM STACK
025236  105010                          CLRB    (R0)            ;;SET A TERMINATOR
025240  104401                          TYPE                    ;;TYPE THE INPUT UP TO BAD CHAR.
025242  000000                  6$:     .WORD   0               ;;POINTER GOES HERE
025244  104401  001174                  TYPE    ,$QUES          ;;"?" "CR" &"LF"
025250  000720                          BR      1$              ;;TRY AGAIN
   5452                         .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
025252                          $ERROR:
025252  005737  002574                  TST     UQUIET          ;;TEST FOR USER-QUIET MODE
025256  001403                          BEQ     9$              ;;BRANCH IF FIELD-SERVICE MODE
025260  005000                          CLR     R0              ;;IN CASE R0 HAS A #3 IN IT (^C)
025262  004737  025466                  JSR     PC,ABORT        ;;TEST FOR ABORT CONDITION
025266                          9$:
025266  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
025270  105237  001103          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
025274  001775                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
025276  013777  001102  153636          MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
025304  032777  002000  153626          BIT     #BIT10,@SWR     ;;BELL ON ERROR?
025312  001402                          BEQ     1$              ;;NO - SKIP
025314  104401  001170                  TYPE    ,$BELL          ;;RING BELL
025320  005237  001112          1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
025324  011637  001116                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
025330  162737  000002  001116          SUB     #2,$ERRPC
025336  117737  153554  001114          MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
025344  032777  020000  153566          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
025352  001004                          BNE     20$             ;;SKIP TYPEOUTS
025354  004737  025626                  JSR     PC,ERTYPE       ;;GO TO USER ERROR ROUTINE
025360  104401  001175                  TYPE    ,$CRLF
025364                          20$:
025364  122737  000001  001220          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
025372  001007                          BNE     2$              ;;NO,SKIP APT ERROR REPORT
025374  113737  001114  025406          MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
025402  004737  025666                  JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
025406     000                  21$:    .BYTE   0
025407     000                          .BYTE   0
025410  000777                  22$:    BR      22$             ;;APT ERROR LOOP
025412  005777  153522          2$:     TST     @SWR            ;;HALT ON ERROR
025416  100002                          BPL     3$              ;;SKIP IF CONTINUE
025420  000000                          HALT                    ;;HALT ON ERROR!
025422  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
025424  032777  001000  153506  3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
025432  001402                          BEQ     4$              ;;BR IF NO
025434  013716  001110                  MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
025440  005737  001166          4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
025444  001402                          BEQ     5$              ;;BR IF NONE
```

```
025446  013716  001166                  MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
025452                          5$:
025452  022737  022672  000042          CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
025460  001001                          BNE     6$              ;;BRANCH IF NO
025462  000000                          HALT                    ;;YES
025464                          6$:
025464  000002                          RTI                     ;;RETURN
                                .SBTTL  ABORT ROUTINE FOR LCP/ORION UFD MODE
025466  005737  002572          ABORT:  TST     UFDFLG          ;TEST FOR USER FRIENDLY MODE
025472  001454                          BEQ     NOABRT          ;IF NOT UFD THEN CONTINUE NORMAL OPERATION
025474  020027  000032                  CMP     R0,#32          ;IS IT A ↑Z ?
025500  001443                          BEQ     ABORTZ          ;JUST GO BACK TO CHAIN IF IT IS (NO ERROR)
025502  020027  000003                  CMP     R0,#3           ;IS IS A ↑C ?
025506  001404                          BEQ     ABORTC          ;BR TO LOAD ↑C ON XXDP+ STACK (NO ERROR)
025510  005737  002574                  TST     UQUIET          ;TEST FOR USER-QUIET MODE
025514  001443                          BEQ     NOABRT          ;IF FIELD-SERVICE MODE, CONTINUE NORMAL OPERATION
                                                                ; BECAUSE FIELD-SERVICE MODE DOES NOT QUIT ON ERROR
025516  000422                          BR      ABORTE          ;SET DRSERR THEN LEAVE
025520  013737  002566  000030  ABORTC: MOV     SAV30,30        ;RESTORE EMT LOCATION (30)
025526  013737  002570  000032          MOV     SAV32,32        ;RESTORE EMT PRIORITY LOCATION (32)
025534  104043                          EMT     +43             ;GET XXDP STACK LOC. INTO R0 FROM MONITOR
025536  005720                  1$:     TST     (R0)+           ;FIND END OF STACK
025540  001376                          BNE     1$
025542  112760  000057  177777          MOVB    #'/,-1(R0)      ;LOAD SLASH OVER ZERO
025550  112720  000136                  MOVB    #'↑,(R0)+       ;LOAD UPARROW
025554  112720  000103                  MOVB    #'C,(R0)+       ;LOAD C
025560  105010                          CLRB    (R0)            ;MAKE NEW END TO STACK
025562  000412                          BR      ABORTZ          ;NOW LEAVE
025564  013737  002566  000030  ABORTE: MOV     SAV30,30        ;RESTORE EMT LOCATION (30)
025572  013737  002570  000032          MOV     SAV32,32        ;RESTORE EMT PRIORITY LOCATION (32)
025600  104042                          EMT     +42             ;GET DCA LOCATION INTO R0 FROM MONITOR
025602  012760  177777  000042          MOV     #-1,42(R0)      ;SET A -1 INTO LOCATION DRSERR IN MONITOR
025610  013700  000042  ABORTZ: MOV     @#42,R0         ;AND PUT THE MONITOR RETURN ADDRESS IN R0
025614  005037  000042                  CLR     @#42            ;CLEAR MONITOR RETURN FLAG
025620  000137  022672                  JMP     $ENDAD          ;RETURN TO MONITOR-DO NOT PUSH STACK HERE
025624  000207                  NOABRT: RTS     PC              ;IF NOTUFD RETURN TO MAINLINE
5453                            ;* THIS ROUTINE LOADS TEST NUMBER AND ERROR NUMBER AND THEN CALLS $ERRTYP
5454 025626  113737  001102  001716  ERTYPE: MOVB    $TSTNM,TEST                     ;;LOAD TEST NUMBER
5455 025634  113737  001114  001714          MOVB    $ITEMB,ERRNUM                   ;;AND ERROR NUMBER
5456 025642  004737  026116                  JSR     PC,$ERRTYP                      ;;GO REPORT ERRORS
5457 025646  000207                          RTS     PC                              ;;RETURN
5458                            .SBTTL  APT COMMUNICATIONS ROUTINE
                                ;;**********************************************************************
025650  112737  000001  026114  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
025656  112737  000001  026112  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
025664  000403                          BR      $ATYC
025666  112737  000001  026114  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
025674                          $ATYC:
025674  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
025676  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
025700  105737  026112                  TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
025704  001450                          BEQ     5$              ;;IF NOT:  BR
025706  122737  000001  001220          CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
025714  001031                          BNE     3$              ;;IF NOT:  BR
025716  132737  000100  001221          BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
025724  001425                          BEQ     3$              ;;IF NOT:  BR
025726  017600  000004                  MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
```

```
025732  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
025740  005737  001200          1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
025744  001375                  BNE     1$              ;;IF NOT:  WAIT
025746  010037  001214          MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
025752  105720          2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
025754  001376                  BNE     2$
025756  163700  001214          SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
025762  006200                  ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
025764  010037  001216          MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
025770  012737  000004  001200  MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
025776  000413                  BR      5$
026000  017637  000004  026024  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
026006  062766  000002  000004  ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
026014  013746  177776          MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
026020  004737  023200          JSR     PC,$TYPE        ;;CALL TYPE MACRO
026024  000000          4$:     .WORD   0
026026                  5$:
026026  105737  026114  10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
026032  001416                  BEQ     12$             ;;IF NOT:  BR
026034  005737  001220          TST     $ENV            ;;RUNNING UNDER APT?
026040  001413                  BEQ     12$             ;;IF NOT:  BR
026042  005737  001200  11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
026046  001375                  BNE     11$             ;;IF NOT:  WAIT
026050  017637  000004  001202  MOV     @4(SP),$FATAL   ;;GET ERROR #
026056  062766  000002  000004  ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
026064  005237  001200          INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
026070  105037  026114  12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
026074  105037  026113          CLRB    $LFLG           ;;CLEAR LOG FLAG
026100  105037  026112          CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
026104  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
026106  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
026110  000207                  RTS     PC              ;;RETURN
026112     000  $MFLG:  .BYTE   0               ;;MESSG. FLAG
026113     000  $LFLG:  .BYTE   0               ;;LOG FLAG
026114     000  $FFLG:  .BYTE   0               ;;FATAL FLAG
                        .EVEN
        000200          APTSIZE=200
        000001          APTENV=001
        000100          APTSPOOL=100
        000040          APTCSUP=040
5459                    .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
                        ;;**************************************************************
                        ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($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.
026116                  $ERRTYP:
026116  104401  001175          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
026122  010046                  MOV     R0,-(SP)        ;;SAVE R0
026124  005000                  CLR     R0              ;;PICKUP THE ITEM INDEX
026126  153700  001114          BISB    @#$ITEMB,R0
026132  001004                  BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
                                                        ;;TYPE THE PC OF THE ERROR
026134  013746  001116          MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
                                                        ;;ERROR ADDRESS
026140  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
026142  000426                  BR      6$              ;;GET OUT
026144  005300          1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
```

ERROR MESSAGE TYPEOUT ROUTINE

```
026146  006300                          ASL     R0              ;;      WORK FOR THE ERROR TABLE
026150  006300                          ASL     R0
026152  006300                          ASL     R0
026154  062700  001324                  ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
026160  012037  026170                  MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
026164  001404                          BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
026166  104401                          TYPE                    ;;TYPE THE "ERROR MESSAGE"
026170  000000                  2$:     .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
026172  104401  001175                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
026176  012037  026206          3$:     MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
026202  001404                          BEQ     5$              ;;SKIP TYPEOUT IF 0
026204  104401                          TYPE                    ;;TYPE THE "DATA HEADER"
026206  000000                  4$:     .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
026210  104401  001175                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
026214  011000                  5$:     MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
026216  001004                          BNE     7$              ;;GO TYPE THE DATA
026220  012600                  6$:     MOV     (SP)+,R0        ;;RESTORE R0
026222  104401  001175                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
026226  000207                          RTS     PC              ;;RETURN
026230                          7$:
026230  013046                          MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
026232  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
026234  005710                          TST     (R0)            ;;IS THERE ANOTHER NUMBER?
026236  001770                          BEQ     6$              ;;BR IF NO
026240  104401  026246                  TYPE    ,8$             ;;TYPE TWO(2) SPACES
026244  000771                          BR      7$              ;;LOOP
026246    040     040     000   8$:     .ASCIZ  /  /            ;;TWO(2) SPACES
                                        .EVEN
5460                                .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.
026252  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
026254  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
026260  005740                          TST     -(R0)           ;;BACKUP BY 2
026262  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
026264  006300                          ASL     R0              ;;POSITION FOR INDEXING
026266  016000  026306                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
026272  000200                          RTS     R0              ;;GO TO ROUTINE
                                    ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
026274  011646                  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
026276  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
026304  000002                          RTI                     ;;RESTORE THE PSW
                                .SBTTL  TRAP TABLE
                                ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
                                ;*BY THE "TRAP" INSTRUCTION.
                                ;       ROUTINE
                                ;       -------
026306  026274                  $TRPAD: .WORD   $TRAP2
026310  023200                          $TYPE   ;;CALL-TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
026312  023560                          $TYPOC  ;;CALL-TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
026314  023534                          $TYPOS  ;;CALL-TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
026316  023574                          $TYPON  ;;CALL-TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
026320  023762                          $TYPDS  ;;CALL-TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
026322  024256                          $GTSWR  ;;CALL-GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING
```

TRAP TABLE

```
026324  024206                          $CKSWR  ;;CALL-CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
026326  024470                          $RDCHR  ;;CALL-RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
026330  024620                          $RDLIN  ;;CALL-RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
026332  024772                          $RDOCT  ;;CALL-RDOCT    TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
026334  025074                          $RDDEC  ;;CALL-RDDEC    TRAP+13(104413) READ A DECIMAL NUMBER FROM TTY
 5461                           .SBTTL  POWER DOWN AND UP ROUTINES
                                ;;**********************************************************************
                                ;POWER DOWN ROUTINE
026336  012737  026476  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
026344  012737  000340  000026         MOV     #340,@#PWRVEC+2 ;;PRIO:7
026352  010046                       MOV     R0,-(SP)        ;;PUSH R0 ON STACK
026354  010146                       MOV     R1,-(SP)        ;;PUSH R1 ON STACK
026356  010246                       MOV     R2,-(SP)        ;;PUSH R2 ON STACK
026360  010346                       MOV     R3,-(SP)        ;;PUSH R3 ON STACK
026362  010446                       MOV     R4,-(SP)        ;;PUSH R4 ON STACK
026364  010546                       MOV     R5,-(SP)        ;;PUSH R5 ON STACK
026366  017746  152546               MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
026372  010637  026502               MOV     SP,$SAVR6       ;;SAVE SP
026376  012737  026410  000024       MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
026404  000000                       HALT
026406  000776                       BR      .-2             ;;HANG UP
                                ;;**********************************************************************
                                ;POWER UP ROUTINE
026410  012737  026476  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
026416  013706  026502               MOV     $SAVR6,SP       ;;GET SP
026422  005037  026502               CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
026426  005237  026502         1$:   INC     $SAVR6          ;;WAIT FOR THE INC
026432  001375                       BNE     1$              ;;OF  WORD
026434  012677  152500               MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
026440  012605                       MOV     (SP)+,R5        ;;POP STACK INTO R5
026442  012604                       MOV     (SP)+,R4        ;;POP STACK INTO R4
026444  012603                       MOV     (SP)+,R3        ;;POP STACK INTO R3
026446  012602                       MOV     (SP)+,R2        ;;POP STACK INTO R2
026450  012601                       MOV     (SP)+,R1        ;;POP STACK INTO R1
026452  012600                       MOV     (SP)+,R0        ;;POP STACK INTO R0
026454  012737  026336  000024       MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
026462  012737  000340  000026       MOV     #340,@#PWRVEC+2 ;;PRIO:7
026470  104401                       TYPE                    ;REPORT THE POWER FAILURE
026472  026504               $PWRMG: .WORD   $POWER          ;;POWER FAIL MESSAGE POINTER
026474  000002                       RTI
026476  000000               $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
026500  000776                       BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
026502  000000               $SAVR6: 0                       ;;PUT THE SP HERE
026504     015     012     120  $POWER: .ASCIZ  <15><12>"POWER"
026507     117     127     105
026512     122     000
                                .EVEN
 5462
 5463                           ;
 5464                           ; ERROR MESSAGES
 5465                           ;
 5466
 5467 026514     124     111     115  EM1:    .ASCIZ  /TIMEOUT ON ACCESSING A MAP REGISTER/
      026517     105     117     125
      026522     124     040     117
      026525     116     040     101
      026530     103     103     105
```

POWER DOWN AND UP ROUTINES

```
                026533  123  123  111
                026536  116  107  040
                026541  101  040  115
                026544  101  120  040
                026547  122  105  107
                026552  111  123  124
                026555  105  122  000
   5468 026560  115  101  120  EM2:    .ASCIZ  /MAP REGISTER COULD NOT BE CLEARED/
                026563  040  122  105
                026566  107  111  123
                026571  124  105  122
                026574  040  103  117
                026577  125  114  104
                026602  040  116  117
                026605  124  040  102
                026610  105  040  103
                026613  114  105  101
                026616  122  105  104
                026621  000
   5469 026622  115  101  120  EM3:    .ASCIZ  /MAP REGISTER COULD NOT HOLD PATTERN/
                026625  040  122  105
                026630  107  111  123
                026633  124  105  122
                026636  040  103  117
                026641  125  114  104
                026644  040  116  117
                026647  124  040  110
                026652  117  114  104
                026655  040  120  101
                026660  124  124  105
                026663  122  116  000
   5470 026666  115  101  120  EM4:    .ASCIZ  /MAP REGISTER HAS NOT BEEN ADDRESSED CORRECTLY/
                026671  040  122  105
                026674  107  111  123
                026677  124  105  122
                026702  040  110  101
                026705  123  040  116
                026710  117  124  040
                026713  102  105  105
                026716  116  040  101
                026721  104  104  122
                026724  105  123  123
                026727  105  104  040
                026732  103  117  122
                026735  122  105  103
                026740  124  114  131
                026743  000
   5471 026744  124  110  105  EM5:    .ASCIZ  /THERE WAS NO DIFFERENCE FOUND BETWEEN HI AND LO MAP REGISTERS/
                026747  122  105  040
                026752  127  101  123
                026755  040  116  117
                026760  040  104  111
                026763  106  106  105
                026766  122  105  116
                026771  103  105  040
                026774  106  117  125
                026777  116  104  040
```

```
        027002  102  105  124
        027005  127  105  105
        027010  116  040  110
        027013  111  040  101
        027016  116  104  040
        027021  114  117  040
        027024  115  101  120
        027027  040  122  105
        027032  107  111  123
        027035  124  105  122
        027040  123  000
  5472  027042  105  122  122  EM6:    .ASCIZ  /ERROR IN BITS 3-6,9-14 IN THE DCSR/
        027045  117  122  040
        027050  111  116  040
        027053  102  111  124
        027056  123  040  063
        027061  055  066  054
        027064  071  055  061
        027067  064  040  111
        027072  116  040  124
        027075  110  105  040
        027100  104  103  123
        027103  122  000
  5473  027105  104  103  123  EM7:    .ASCIZ  /DCSR DID NOT RESPOND PROPORLY ON RESET/
        027110  122  040  104
        027113  111  104  040
        027116  116  117  124
        027121  040  122  105
        027124  123  120  117
        027127  116  104  040
        027132  120  122  117
        027135  120  117  122
        027140  114  131  040
        027143  117  116  040
        027146  122  105  123
        027151  105  124  000
  5474  027154  124  111  115  EM10:   .ASCIZ  /TIMEOUT HAS OCCURED ON ACCESS TO THE DCSR/
        027157  105  117  125
        027162  124  040  110
        027165  101  123  040
        027170  117  103  103
        027173  125  122  105
        027176  104  040  117
        027201  116  040  101
        027204  103  103  105
        027207  123  123  040
        027212  124  117  040
        027215  124  110  105
        027220  040  104  103
        027223  123  122  000
  5475  027226  113  115  103  EM11:   .ASCIZ  /KMCR BITS 0-5,8 DID NOT GET SET CORRECTLY/
        027231  122  040  102
        027234  111  124  123
        027237  040  060  055
        027242  065  054  070
        027245  040  104  111
        027250  104  040  116
```

```
        027253  117  124  040
        027256  107  105  124
        027261  040  123  105
        027264  124  040  103
        027267  117  122  122
        027272  105  103  124
        027275  114  131  000
5476    027300  124  111  115  EM12:   .ASCIZ  /TIMEOUT HAS OCCURED ON ACCESS TO THE KMCR/
        027303  105  117  125
        027306  124  040  110
        027311  101  123  040
        027314  117  103  103
        027317  125  122  105
        027322  104  040  117
        027325  116  040  101
        027330  103  103  105
        027333  123  123  040
        027336  124  117  040
        027341  124  110  105
        027344  040  113  115
        027347  103  122  000
5477    027352  105  122  122  EM13:   .ASCIZ  /ERROR IN DATA PATH PERFORMING DATA OUT/
        027355  117  122  040
        027360  111  116  040
        027363  104  101  124
        027366  101  040  120
        027371  101  124  110
        027374  040  120  105
        027377  122  106  117
        027402  122  115  111
        027405  116  107  040
        027410  104  101  124
        027413  101  040  117
        027416  125  124  000
5478    027421  105  122  122  EM14:   .ASCIZ  /ERROR IN DATA OUT CYCLE/
        027424  117  122  040
        027427  111  116  040
        027432  104  101  124
        027435  101  040  117
        027440  125  124  040
        027443  103  131  103
        027446  114  105  000
5479    027451  105  122  122  EM15:   .ASCIZ  /ERROR IN DATA IN CYCLE/
        027454  117  122  040
        027457  111  116  040
        027462  104  101  124
        027465  101  040  111
        027470  116  040  103
        027473  131  103  114
        027476  105  000
5480    027500  104  104  122  EM16:   .ASCIZ  /DDR IS NOT 0, WHEN DCRS<2-1> SELECTS UNIBUS LINES/
        027503  040  111  123
        027506  040  116  117
        027511  124  040  060
        027514  054  040  127
        027517  110  105  116
        027522  040  104  103
```

```
        027525  122  123  074
        027530  062  055  061
        027533  076  040  123
        027536  105  114  105
        027541  103  124  123
        027544  040  125  116
        027547  111  102  125
        027552  123  040  114
        027555  111  116  105
        027560  123  000
  5481  027562  104  103  123  EM17:   .ASCIZ  /DCSR<07> DOES NOT BECOME 1/
        027565  122  074  060
        027570  067  076  040
        027573  104  117  105
        027576  123  040  116
        027601  117  124  040
        027604  102  105  103
        027607  117  115  105
        027612  040  061  000
  5482  027615  111  116  104  EM20:   .ASCIZ  /INDIRECT ADDRESSING OF MAPPING REGISTER PAIRS/
        027620  111  122  105
        027623  103  124  040
        027626  101  104  104
        027631  122  105  123
        027634  123  111  116
        027637  107  040  117
        027642  106  040  115
        027645  101  120  120
        027650  111  116  107
        027653  040  122  105
        027656  107  111  123
        027661  124  105  122
        027664  040  120  101
        027667  111  122  123
        027672  000
  5483  027673  122  105  107  EM21:   .ASCIZ  /REGISTER PAIR 40 PERFORMS RELOCATION/
        027676  111  123  124
        027701  105  122  040
        027704  120  101  111
        027707  122  040  064
        027712  060  040  120
        027715  105  122  106
        027720  117  122  115
        027723  123  040  122
        027726  105  114  117
        027731  103  101  124
        027734  111  117  116
        027737  000
  5484  027740  116  130  115  EM22:   .ASCIZ  /NXM CONDITION COULD NOT BE CREATED THRU ALU/
        027743  040  103  117
        027746  116  104  111
        027751  124  111  117
        027754  116  040  103
        027757  117  125  114
        027762  104  040  116
        027765  117  124  040
        027770  102  105  040
```

POWER DOWN AND UP ROUTINES

```
                027773  103     122     105
                027776  101     124     105
                030001  104     040     124
                030004  110     122     125
                030007  040     101     114
                030012  125     000
   5485         030014  101     114     125     EM23:   .ASCIZ  /ALU ERROR/
                030017  040     105     122
                030022  122     117     122
                030025  000
   5486         030026  103     120     125     EM24:   .ASCIZ  /CPU CACHE ERROR/
                030031  040     103     101
                030034  103     110     105
                030037  040     105     122
                030042  122     117     122
                030045  000
   5487         030046  113     115     103     EM25:   .ASCIZ  /KMCR<4-0> DON'T DISABLE MEMORY RESPONSE/
                030051  122     074     064
                030054  055     060     076
                030057  040     104     117
                030062  116     047     124
                030065  040     104     111
                030070  123     101     102
                030073  114     105     040
                030076  115     105     115
                030101  117     122     131
                030104  040     122     105
                030107  123     120     117
                030112  116     123     105
                030115  000
   5488         030116  113     115     103     EM26:   .ASCIZ  /KMCR DOES NOT REFLECT EXPECTED STATUS OF THE CACHE/
                030121  122     040     104
                030124  117     105     123
                030127  040     116     117
                030132  124     040     122
                C30135  105     106     114
                030140  105     103     124
                030143  040     105     130
                030146  120     105     103
                030151  124     105     104
                030154  040     123     124
                030157  101     124     125
                030162  123     040     117
                030165  106     040     124
                030170  110     105     040
                030173  103     101     103
                030176  110     105     000
   5489         030201  105     122     122     EM27:   .ASCIZ  /ERROR IN CACHE TAG REGISTERS/
                030204  117     122     040
                030207  111     116     040
                030212  103     101     103
                030215  110     105     040
                030220  124     101     107
                030223  040     122     105
                030226  107     111     123
                030231  124     105     122
                030234  123     000
```

```
 5490 030236  105  122  122  EM30:   .ASCIZ  /ERROR IN THE DMA CACHE DATA RAMS/
      030241  117  122  040
      030244  111  116  040
      030247  124  110  105
      030252  040  104  115
      030255  101  040  103
      030260  101  103  110
      030263  105  040  104
      030266  101  124  101
      030271  040  122  101
      030274  115  123  000
 5491 030277  105  122  122  EM31:   .ASCIZ  /ERROR IN THE M9312 BOOT ROM SECTION/
      030302  117  122  040
      030305  111  116  040
      030310  124  110  105
      030313  040  115  071
      030316  063  061  062
      030321  040  102  117
      030324  117  124  040
      030327  122  117  115
      030332  040  123  105
      030335  103  124  111
      030340  117  116  000
 5492 030343  105  122  122  EM32:   .ASCIZ  /ERROR IN ARBITRATION LOGIC USING UBE/
      030346  117  122  040
      030351  111  116  040
      030354  101  122  102
      030357  111  124  122
      030362  101  124  111
      030365  117  116  040
      030370  114  117  107
      030373  111  103  040
      030376  125  123  111
      030401  116  107  040
      030404  125  102  105
      030407  000
 5493 030410  105  122  122  EM33:   .ASCIZ  /ERROR TRYING TO EXECUTE DMA CYCLES THRU UBE/
      030413  117  122  040
      030416  124  122  131
      030421  111  116  107
      030424  040  124  117
      030427  040  105  130
      030432  105  103  125
      030435  124  105  040
      030440  104  115  101
      030443  040  103  131
      030446  103  114  105
      030451  123  040  124
      030454  110  122  125
      030457  040  125  102
      030462  105  000
 5494 030464  105  122  122  EM34:   .ASCIZ  /ERROR IN THE UNIBUS MEMORY TEST/
      030467  117  122  040
      030472  111  116  040
      030475  124  110  105
      030500  040  125  116
      030503  111  102  125
```

```
        030506  123  040  115
        030511  105  115  117
        030514  122  131  040
        030517  124  105  123
        030522  124  000
  5495  030524  125  116  105  EM35:   .ASCIZ  /UNEXPECTED TIMEOUT HAS OCCURED/
        030527  130  120  105
        030532  103  124  105
        030535  104  040  124
        030540  111  115  105
        030543  117  125  124
        030546  040  110  101
        030551  123  040  117
        030554  103  103  125
        030557  122  105  104
        030562  000
  5496  030563  125  116  111  EM36:   .ASCIZ  /UNIBUS TIMEOUT DID NOT WORK CORRECTLY/
        030566  102  125  123
        030571  040  124  111
        030574  115  105  117
        030577  125  124  040
        030602  104  111  104
        030605  040  116  117
        030610  124  040  127
        030613  117  122  113
        030616  040  103  117
        030621  122  122  105
        030624  103  124  114
        030627  131  000
  5497  030631  104  103  123  EM37:   .ASCIZ  /DCSR<3> DID NOT DISABLE UBA ROM RESPONSE/
        030634  122  074  063
        030637  076  040  104
        030642  111  104  040
        030645  116  117  124
        030650  040  104  111
        030653  123  101  102
        030656  114  105  040
        030661  125  102  101
        030664  040  122  117
        030667  115  040  122
        030672  105  123  120
        030675  117  116  123
        030700  105  000
  5498
  5499  030702  124  105  123  DH1:    .ASCII  /TEST   ERROR   ERROR   ADDRESS/<12><15>
        030705  124  011  105
        030710  122  122  117
        030713  122  011  105
        030716  122  122  117
        030721  122  011  101
        030724  104  104  122
        030727  105  123  123
        030732  012  015
  5500  030734  040  040  043          .ASCIZ  /  #      PC      #/
        030737  011  040  040
        030742  120  103  011
        030745  040  040  043
```

POWER DOWN AND UP ROUTINES

```
                030750  000
          5501  030751  124  105  123  DH2:   .ASCII  /TEST   ERROR   ERROR   GOOD    BAD/<12><15>
                030754  124  011  105
                030757  122  122  117
                030762  122  011  105
                030765  122  122  117
                030770  122  011  107
                030773  117  117  104
                030776  011  102  101
                031001  104  012  015
          5502  031004  040  040  043         .ASCIZ  / #     PC      #     DATA    DATA    ADDRESS/
                031007  011  040  040
                031012  120  103  011
                031015  040  040  043
                031020  011  104  101
                031023  124  101  011
                031026  104  101  124
                031031  101  011  101
                031034  104  104  122
                031037  105  123  123
                031042  000
          5503  031043  124  105  123  DH3:   .ASCII  /TEST   ERROR   ERROR   GOOD    BAD/<12><15>
                031046  124  011  105
                031051  122  122  117
                031054  122  011  105
                031057  122  122  117
                031062  122  011  107
                031065  117  117  104
                031070  011  102  101
                031073  104  012  015
          5504  031076  040  040  043         .ASCIZ  / #     PC      #     ADDRESS ADDRESS/
                031101  011  040  040
                031104  120  103  011
                031107  040  040  043
                031112  011  101  104
                031115  104  122  105
                031120  123  123  011
                031123  101  104  104
                031126  122  105  123
                031131  123  000
          5505  031133  124  105  123  DH4:   .ASCII  /TEST   ERROR   ERROR   HI      LOW/<12><15>
                031136  124  011  105
                031141  122  122  117
                031144  122  011  105
                031147  122  122  117
                031152  122  011  110
                031155  111  011  114
                031160  117  127  012
                031163  015
          5506  031164  040  040  043         .ASCIZ  / #     PC      #     MAP     MAP/
                031167  011  040  040
                031172  120  103  011
                031175  040  040  043
                031200  011  115  101
                031203  120  011  115
                031206  101  120  000
          5507  031211  124  105  123  DH5:   .ASCII  /TEST   ERROR   ERROR   GOOD    BAD/<12><15>
```

```
        031214  124  011  105
        031217  122  122  117
        031222  122  011  105
        031225  122  122  117
        031230  122  011  107
        031233  117  117  104
        031236  011  102  101
        031241  104  012  015
  5508  031244  040  040  043          .ASCIZ  /  #     PC      #     DATA    DATA/
        031247  011  040  040
        031252  120  103  011
        031255  040  040  043
        031260  011  104  101
        031263  124  101  011
        031266  104  101  124
        031271  101  000
  5509  031273  124  105  123  DH13:   .ASCII  /TEST   ERROR   ERROR   DDR     PATTERN/<12><15>
        031276  124  011  105
        031301  122  122  117
        031304  122  011  105
        031307  122  122  117
        031312  122  011  104
        031315  104  122  011
        031320  120  101  124
        031323  124  105  122
        031326  116  012  015
  5510  031331  040  040  043          .ASCIZ  /  #     PC      #/
        031334  011  040  040
        031337  120  103  011
        031342  040  040  043
        031345  000
  5511  031346  124  105  123  DH16:   .ASCII  /TEST   ERROR   ERROR/<12><15>
        031351  124  011  105
        031354  122  122  117
        031357  122  011  105
        031362  122  122  117
        031365  122  012  015
  5512  031370  040  040  043          .ASCIZ  /  #     PC      #/
        031373  011  040  040
        031376  120  103  011
        031401  040  040  043
        031404  000
  5513  031405  124  105  123  DH20:   .ASCII  /TEST   ERROR   ERROR   KMCR PAIR/<12><15>
        031410  124  011  105
        031413  122  122  117
        031416  122  011  105
        031421  122  122  117
        031424  122  011  113
        031427  115  103  122
        031432  040  120  101
        031435  111  122  012
        031440  015
  5514  031441  040  040  043          .ASCIZ  /  #     PC      #     FAILED/
        031444  011  040  040
        031447  120  103  011
        031452  040  040  043
        031455  011  106  101
```

POWER DOWN AND UP ROUTINES

```
                031460  111     114     105
                031463  104     000
5515            031465  124     105     123     DH23:   .ASCII  /TEST   ERROR   ERROR   GOOD    BAD     ADDRESS/<12><15>
                031470  124     011     105
                031473  122     122     117
                031476  122     011     105
                031501  122     122     117
                031504  122     011     107
                031507  117     117     104
                031512  011     102     101
                031515  104     011     101
                031520  104     104     122
                031523  105     123     123
                031526  012     015
5516            031530  040     040     043             .ASCIZ  /  #     PC      #     DATA    DATA    <22-16> <13-0>/
                031533  011     040     040
                031536  120     103     011
                031541  040     040     043
                031544  011     104     101
                031547  124     101     011
                031552  104     101     124
                031555  101     011     074
                031560  062     062     055
                031563  061     066     076
                031566  011     074     061
                031571  063     055     060
                031574  076     000
5517            031576  124     105     123     DH27:   .ASCII  /TEST   ERROR   ERROR   ADDRESS ADDRESS/<12><15>
                031601  124     011     105
                031604  122     122     117
                031607  122     011     105
                031612  122     122     117
                031615  122     011     101
                031620  104     104     122
                031623  105     123     123
                031626  011     101     104
                031631  104     122     105
                031634  123     123     012
                031637  015
5518            031640  040     040     043             .ASCIZ  /  #     PC      #     <22-16> <15-0>/
                031643  011     040     040
                031646  120     103     011
                031651  040     040     043
                031654  011     074     062
                031657  062     055     061
                031662  066     076     011
                031665  074     061     065
                031670  055     060     076
                031673  000
5519                                            .EVEN
5520            031674  001716  001116  001714  DT1:    .WORD   TEST,$ERRPC,ERRNUM,$BDADR,0
                031702  001122  000000
5521            031706  001716  001116  001714  DT2:    .WORD   TEST,$ERRPC,ERRNUM,$GDDAT,$BDDAT,$BDADR,0
                031714  001124  001126  001122
                031722  000000
5522            031724  001716  001116  001714  DT3:    .WORD   TEST,$ERRPC,ERRNUM,$GDADR,$BDADR,0
                031732  001120  001122  000000
```

POWER DOWN AND UP ROUTINES

```
5523 031740  001716  001116  001714  DT4:    .WORD   TEST,$ERRPC,ERRNUM,$GDDAT,$BDDAT,0
     031746  001124  001126  000000
5524 031754  001716  001116  001714  DT16:   .WORD   TEST,$ERRPC,ERRNUM,0
     031762  000000
5525 031764  001716  001116  001714  DT20:   .WORD   TEST,$ERRPC,ERRNUM,KMCR,$BDADR,0
     031772  177734  001122  000000
5526 032000  001716  001122  001714  DT21:   .WORD   TEST,$BDADR,ERRNUM,0
     032006  000000
5527 032010  001716  001116  001714  DT23:   .WORD   TEST,$ERRPC,ERRNUM,$GDDAT,$BDDAT,KIPAR6,$BDADR,0
     032016  001124  001126  172354
     032024  001122  000000
5528 032030  001716  001116  001714  DT27:   .WORD   TEST,$ERRPC,ERRNUM,MAPH01,MAPL01,0
     032036  170206  170204  000000
5529 032044  001716  001116  001714  DT30:   .WORD   TEST,$ERRPC,ERRNUM,MAPL00,0
     032052  170200  000000
5530         000001                  .END
```

SYMBOL TABLE

```
ABASE = 000000
ABORT   025466
ABORTC  025520
ABORTE  025564
ABORTZ  025610
ACDW1 = 000000
ACDW2 = 000000
ACPUOP= 000000
ADDW0 = 000000
ADDW1 = 000000
ADDW10= 000000
ADDW11= 000000
ADDW12= 000000
ADDW13= 000000
ADDW14= 000000
ADDW15= 000000
ADDW2 = 000000
ADDW3 = 000000
ADDW4 = 000000
ADDW5 = 000000
ADDW6 = 000000
ADDW7 = 000000
ADDW8 = 000000
ADDW9 = 000000
ADEVCT= 000000
ADEVM = 000000
AENV  = 000000
AENVM = 000000
AFATAL= 000000
AMADR1= 000000
AMADR2= 000000
AMADR3= 000000
AMADR4= 000000
AMAMS1= 000000
AMAMS2= 000000
AMAMS3= 000000
AMAMS4= 000000
AMSGAD= 000000
AMSGLG= 000000
AMSGTY= 000000
AMTYP1= 000000
AMTYP2= 000000
AMTYP3= 000000
AMTYP4= 000000
APASS = 000000
APRIOR= 000000
APTCSU= 000040
APTENV= 000001
APTSIZ= 000200
APTSPO= 000100
ASWREG= 000000
ATESTN= 000000
AUNIT = 000000
AUSWR = 000000
AVECT1= 000000
AVECT2= 000000
BCSR  = 177520
BE1BA   002154
BE1CC   002152
BE1CLR  002160
BE1CR1  002156
BE1CR2  002162
BE1DB   002150
BE1INT  001724
BE1PSW  002166
BE1SV   015212
BE1VEC  002164
BE2BA   002174
BE2CC   002172
BE2CLR  002200
BE2CR1  002176
BE2CR2  002202
BE2DB   002170
BE2INT  001726
BE2PSW  002206
BE2SV   015232
BE2VEC  002204
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BPTVEC= 000014
CCR   = 177746
CKSWR = 104407
CR    = 000015
CRLF  = 000200
CTBLE   002022
DCSR  = 177730
DDIN    002222
DDISP = 177570
DDOUT   002210
DDR   = 177732
DH1     030702
DH13    031273
DH16    031346
DH2     030751
DH20    031405
DH23    031465
DH27    031576
DH3     031043
DH4     031133
DH5     031211
DISPLA  001142
DISPRE  000174
DSWR  = 177570
DT1     031674
DT16    031754
DT2     031706
DT20    031764
DT21    032000
DT23    032010
DT27    032030
DT3     031724
DT30    032044
DT4     031740
EMTSAV  002500
EMTVEC= 000030
EM1     026514
EM10    027154
EM11    027226
EM12    027300
EM13    027352
EM14    027421
EM15    027451
EM16    027500
EM17    027562
EM2     026560
EM20    027615
EM21    027673
EM22    027740
EM23    030014
EM24    030026
EM25    030046
EM26    030116
EM27    030201
EM3     026622
EM30    030236
EM31    030277
EM32    030343
EM33    030410
EM34    030464
EM35    030524
EM36    030563
EM37    030631
EM4     026666
EM5     026744
EM6     027042
EM7     027105
ERRNUM  001714
ERROR = 104000
ERRVEC= 000004
ERTYPE  025626
GTSWR = 104406
HT    = 000011
IOTVEC= 000020
IUBE    002450
KDPAR0= 172360
KDPAR1= 172362
KDPAR2= 172364
KDPAR3= 172366
KDPAR4= 172370
KDPAR5= 172372
KDPAR6= 172374
KDPAR7= 172376
KDPDR0= 172320
KDPDR1= 172322
KDPDR2= 172324
KDPDR3= 172326
KDPDR4= 172330
KDPDR5= 172332
KDPDR6= 172334
KDPDR7= 172336
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KMCR  = 177734
LF    = 000012
MAPH0 = 170202
MAPH00= 170202
MAPH01= 170206
MAPH02= 170212
MAPH03= 170216
MAPH04= 170222
MAPH05= 170226
MAPH06= 170232
MAPH07= 170236
MAPH1 = 170206
MAPH10= 170242
MAPH11= 170246
MAPH12= 170252
MAPH13= 170256
MAPH14= 170262
MAPH15= 170266
MAPH16= 170272
MAPH17= 170276
MAPH2 = 170212
MAPH20= 170302
MAPH21= 170306
MAPH22= 170312
MAPH23= 170316
MAPH24= 170322
MAPH25= 170326
MAPH26= 170332
MAPH27= 170336
MAPH3 = 170216
MAPH30= 170342
MAPH31= 170346
MAPH32= 170352
MAPH33= 170356
MAPH34= 170362
MAPH35= 170366
MAPH36= 170372
MAPH37= 170376
MAPH4 = 170222
MAPH5 = 170226
MAPH6 = 170232
MAPH7 = 170236
MAPL0 = 170200
MAPL00= 170200
MAPL01= 170204
MAPL02= 170210
MAPL03= 170214
MAPL04= 170220
MAPL05= 170224
MAPL06= 170230
MAPL07= 170234
MAPL1 = 170204
MAPL10= 170240
MAPL11= 170244
MAPL12= 170250
MAPL13= 170254
MAPL14= 170260
MAPL15= 170264
MAPL16= 170270
MAPL17= 170274
MAPL2 = 170210
MAPL20= 170300
MAPL21= 170304
MAPL22= 170310
MAPL23= 170314
MAPL24= 170320
MAPL25= 170324
MAPL26= 170330
MAPL27= 170334
MAPL3 = 170214
MAPL30= 170340
MAPL31= 170344
MAPL32= 170350
MAPL33= 170354
MAPL34= 170360
```

SYMBOL TABLE

```
MAPL35= 170364   SWREG   000176   TST30  011704   $BASE  001254   $GDADR 001120
MAPL36= 170370   SW0   = 000001   TST31  012104   $BDADR 001122   $GDDAT 001124
MAPL37= 170374   SW00  = 000001   TST32  012506   $BDDAT 001126   $GET42 022662
MAPL4 = 170220   SW01  = 000002   TST33  012724   $BELL  001170   $GTSWR 024256
MAPL5 = 170224   SW02  = 000004   TST34  013350   $CDW1  001260   $HD    = 000001
MAPL6 = 170230   SW03  = 000010   TST35  013550   $CDW2  001262   $HIBTS 000232
MAPL7 = 170234   SW04  = 000020   TST36  013736   $CHARC 023530   $HIOCT 025072
MAPPR   002244   SW05  = 000040   TST37  014040   $CKSWR 024206   $ICNT  001104
MCSR    001732   SW06  = 000100   TST4   003764   $CMTAG 001100   $ILLUP 026476
MEMSIZ  002332   SW07  = 000200   TST40  014320   $CM3   = 000000   $INTAG 001135
MMR0  = 177572   SW08  = 000400   TST41  014660   $CM4   = 000002   $ITEMB 001114
MMR1  = 177574   SW09  = 001000   TST42  015014   $CNTLG 024743   $LF    001176
MMR2  = 177576   SW1   = 000002   TST43  015252   $CNTLU 024736   $LFLG  026113
MMR3  = 172516   SW10  = 002000   TST44  015466   $CPUOP 001226   $LPADR 001106
MMVEC = 000250   SW11  = 004000   TST45  015560   $CRLF  001175   $LPERR 001110
NOABRT  025624   SW12  = 010000   TST46  015640   $DBLK  024176   $MADR1 001232
NOCAH   007544   SW13  = 020000   TST47  015740   $DDW0  001264   $MADR2 001236
NROM    013312   SW14  = 040000   TST5   004044   $DDW1  001266   $MADR3 001242
OBADR   002140   SW15  = 100000   TST50  016154   $DDW10 001310   $MADR4 001246
PCR   = 177522   SW2   = 000004   TST51  016246   $DDW11 001312   $MAIL  001200
PIRQ  = 177772   SW3   = 000010   TST52  016410   $DDW12 001314   $MAMS1 001230
PIRQVE= 000240   SW4   = 000020   TST53  016516   $DDW13 001316   $MAMS2 001234
PMIS    001720   SW5   = 000040   TST54  017146   $DDW14 001320   $MAMS3 001240
POLY  = 120001   SW6   = 000100   TST55  017312   $DDW15 001322   $MAMS4 001244
PR0   = 000000   SW7   = 000200   TST56  017462   $DDW2  001270   $MBADR 000234
PR1   = 000040   SW8   = 000400   TST57  020110   $DDW3  001272   $MFLG  026112
PR2   = 000100   SW9   = 001000   TST6   004230   $DDW4  001274   $MNEW  024761
PR3   = 000140   TBITVE= 000014   TST60  020462   $DDW5  001276   $MSGAD 001214
PR4   = 000200   TBLMH   012446   TST61  020544   $DDW6  001300   $MSGLG 001216
PR5   = 000240   TBLML   012406   TST62  020650   $DDW7  001302   $MSGTY 001200
PR6   = 000300   TEST    001716   TST63  021212   $DDW8  001304   $MSWR  024750
PR7   = 000340   TIMOUT  002234   TST64  021666   $DDW9  001306   $MTYP1 001231
PS    = 177776   TKVEC = 000060   TST65  022052   $DEVCT 001210   $MTYP2 001235
PSW   = 177776   TOPTBL  010372   TST66  022274   $DEVM  001256   $MTYP3 001241
PTRN16  001774   TOUT    001730   TST67  022454   $DOAGN 022702   $MTYP4 001245
PTRN6   002010   TPVEC = 000064   TST7   004406   $DTBL  024166   $MXCNT 023176
PWRVEC= 000024   TRAPVE= 000034   TYPDS = 104405   $ENDAD 022672   $NULL  001154
RDCHR = 104410   TRTVEC= 000014   TYPE  = 104401   $ENDCT 022536   $NWTST= 000001
RDDEC = 104413   TST1    003276   TYPOC = 104402   $ENULL 022706   $OCNT  023756
RDLIN = 104411   TST10   004462   TYPON = 104404   $ENV   001220   $OMODE 023760
RDOCT = 104412   TST11   004612   TYPOS = 104403   $ENVM  001221   $OVER  023162
RESTAR  003272   TST12   004700   UBECT   001722   $EOP   022502   $PASS  001206
RESVEC= 000010   TST13   005014   UBEM    022454   $EOPCT 022530   $PASTM 000240
R6    =%000006   TST14   005402   UBETST  012724   $ERFLG 001103   $POWER 026504
R7    =%000007   TST15   005572   UFDFLG  002572   $ERMAX 001115   $PWRDN 026336
SAV30   002566   TST16   005762   UFDSET= 000001   $ERROR 025252   $PWRMG 026472
SAV32   002570   TST17   006144   UMSIZ   002424   $ERRPC 001116   $PWRUP 026410
SCOPE = 000004   TST2    003422   UQUIET  002574   $ERRTB 001324   $QUES  001174
SIMLGO= 170014   TST20   006410   VALTBL  010356   $ERRTY 026116   $RDCHR 024470
SR0   = 177572   TST21   007002   VMKOR   002576   $ERTTL 001112   $RDDEC 025074
SR1   = 177574   TST22   007454   WRTBUF  001734   $ESCAP 001166   $RDLIN 024620
SR2   = 177576   TST23   007564   $APTHD  000232   $ETABL 001220   $RDOCT 024772
SR3   = 172516   TST24   007742   $ATYC   025674   $ETEND 001324   $RDSZ  = 000010
STACK = 001100   TST25   010406   $ATY1   025650   $FATAL 001202   $RTNAD 022704
START   002500   TST26   011102   $ATY3   025656   $FFLG  026114   $SAVR6 026502
STKLMT= 177774   TST27   011236   $ATY4   025666   $FILLC 001156   $SCOPE  022712
SWR     001140   TST3    003642   $AUTOB  001134   $FILLS 001155   $SETUP= 000137
```

```
KTJ11-B DIAGNOSTIC      MACRO M1200  25-JUL-84 17:10  PAGE 64-29                    SEQ 0159

SYMBOL TABLE

$STUP = 177777          $TKS    001144          $TRAP2  026274          $TYPEC  023412          $VECT1  001250
$SVLAD  023126          $TMPO   001160          $TRP  = 000014          $TYPEX  023532          $VECT2  001252
$SVPC = 000232          $TMP1   001162          $TRPAD  026306          $TYPOC  023560          $XOFF = 000023
$SWR  = 167400          $TN   = 000070          $TSTM   000236          $TYPON  023574          $XON  = 000021
$SWREG  001222          $TPB    001152          $TSTNM  001102          $TYPOS  023534          $XTSTR  022724
$SWRMK= 000300          $TPFLG  001157          $TTYIN  024726          $UNIT   001212          $$GET4= 000000
$TESTN  001204          $TPS    001150          $TYPDS  023762          $UNITM  000242          $OFILL  023757
$TIMES  001164          $TRAP   026252          $TYPE   023200          $USWR   001224          .$X   = 000232
$TKB    001146

. ABS.  032056      000
        000000      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  56448 WORDS  ( 221 PAGES)
DYNAMIC MEMORY:  20060 WORDS  ( 77 PAGES)
ELAPSED TIME:  00:03:35
COKTAA,COKTAA/NL:TOC/-SP=ORION.MLB/ML,COKTAA.P11/DS:GBL
.
```