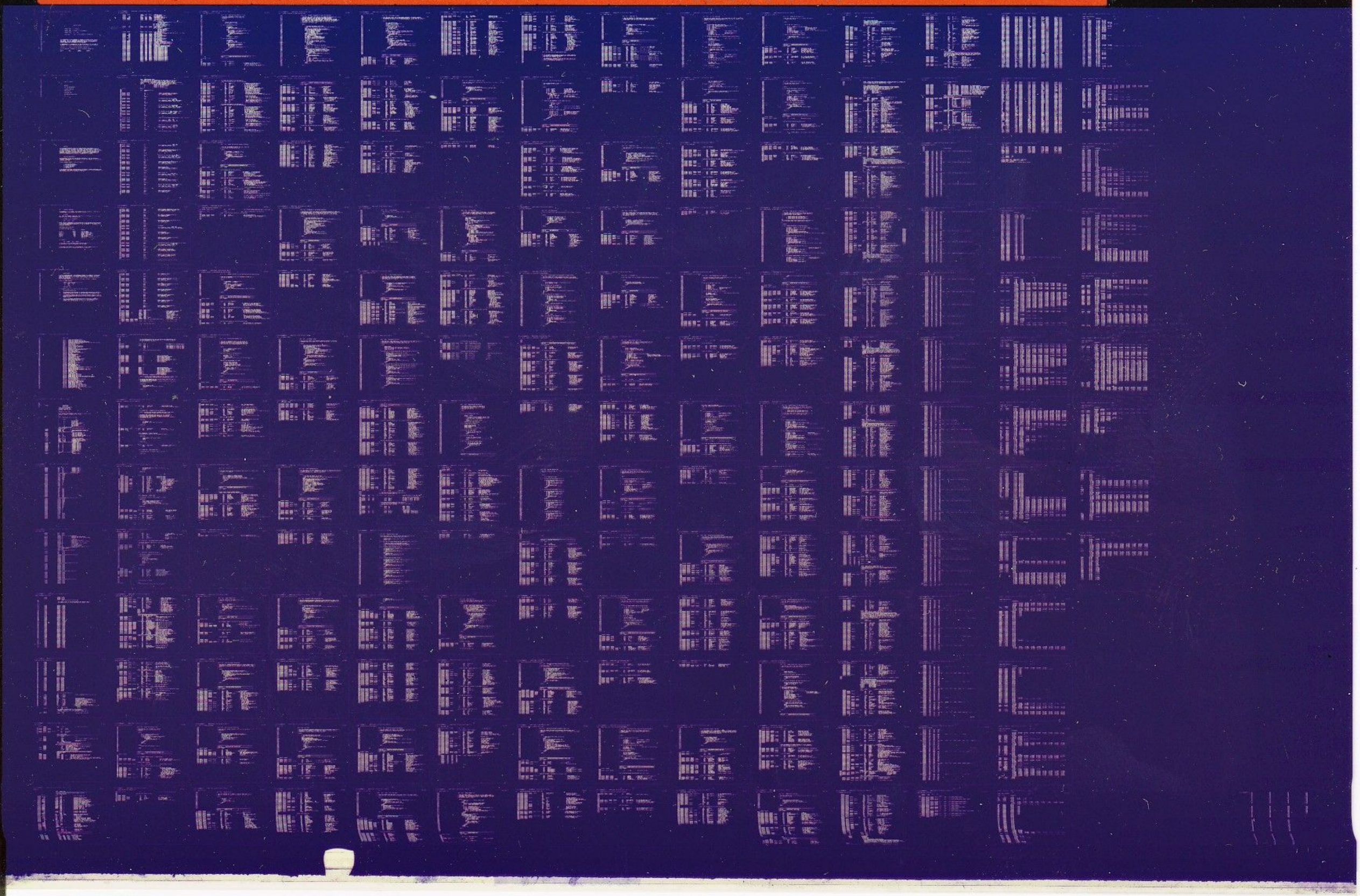
KTJ11-B KTJ11-B DIAGNOSTIC
COKTABO

AH-T876B-MC
FICHE 01 OF 01

FEB 1985
digital
Made In USA



```
                        .REM    &
```

IDENTIFICATION

PRODUCT CODE: AC-T8758-MC

PRODUCT NAME: COKTA80 KTJ11-B DIAGNOSTIC

PRODUCT DATE: JULY, 1984

MAINTAINER: SMALL SYSTEMS DIAGNOSTIC ENGINEERING

```
45
46
47
48                                  TABLE OF CONTENTS
49                                  -----------------
50
51
52                              1.      ABSTRACT
53
54                              2.      RUN-TIME REQUIREMENTS
55
56                              3.      STARTING PROCEDURE
57
58                              4.      ERROR REPORTS
59
60                              5.      EXECUTION TIME
61
62                              6       UBA REGISTER DEFINITIONS
63
64                              7.      TEST LIST
65
```

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 OKTAB?
107
108                     WILL START THE PROGRAM, WHICH WILL THEN PROMT THE OPERATOR FOR THE SOFTWARE
109                     REGISTER SWITCH SETTING DESRIBED BELOW.
110
111                             COKTABO 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
```

```
146
147
148                     4.      ERROR REPORTS
149
150                     IF RAN IN STANDALONE MODE, THE DIAGNOSTIC REPORTS ALL ERRORS TO A FUNCTIONAL
151                     LEVEL AND THEN CONTINUES. FAILING PROGRAM COUNTER, TEST NUMBER, AND ERROR
152                     NUMBER ARE PRINTED OUT FOR ALL ERRORS. WHERE POSSIBLE EXPECTED AND RECEIVED
153                     DATA ARE ALSO PROVIDED.
154                     REFER TO OPERATIONAL SWITCH SETTINS IF ANYTHING DIFFERENT IS REQUIRED.
155
156                     EXAMPLE OF ERROR PRINTOUT:
157
158                             ERROR IN THE M9312 BOOT ROM SECTION
159                              TEST   ERROR   ERROR
160                                #     PC       #
161                              33    12650    31
162
163                     5.      EXECUTION TIME
164
165                     THE DIAGNOSTIC RUNS A FULL PASS IN LESS THAN A MINUTE.
166
167                     6.      UBA REGISTER DEFINITION
168
169                     DDR     THE DIAGNOSTIC DATA REGISTER IS A BUFFER THAT PROVIDES THE ABILITY OF
170                             READING AND WRITING THE DATA TO AND FROM MEMORY IN DIAGNOSTIC MODE.
171
172                     DCSR    THE DIAGNOSTIC CONTROL AND STATUS REGISTER PROVIDES MEANS OF GOING IN
173                             AND OUT OF STANDALONE MODE AND ALSO OF INITIATING DIAGNOSTIC DATI FROM
174                             MEMORY CYCLES.
175
176                     KMCR    THE KTJ11-B MEMORY CONFIGURATION REGISTER IDENTIFIES THE AMOUNT OF
177                             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.
                        ;*
                        $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
```

BASIC DEFINITIONS

```
000100                  PR2=    100             ;;PRIORITY LEVEL 2
000140                  PR3=    140             ;;PRIORITY LEVEL 3
000200                  PR4=    200             ;;PRIORITY LEVEL 4
000240                  PR5=    240             ;;PRIORITY LEVEL 5
000300                  PR6=    300             ;;PRIORITY LEVEL 6
000340                  PR7=    340             ;;PRIORITY LEVEL 7
                        ;*"SWITCH REGISTER" SWITCH DEFINITIONS
100000                  SW15=   100000
040000                  SW14=   40000
020000                  SW13=   20000
010000                  SW12=   10000
004000                  SW11=   4000
002000                  SW10=   2000
001000                  SW09=   1000
000400                  SW08=   400
000200                  SW07=   200
000100                  SW06=   100
000040                  SW05=   40
000020                  SW04=   20
000010                  SW03=   10
000004                  SW02=   4
000002                  SW01=   2
000001                  SW00=   1
001000                  SW9=    SW09
000400                  SW8=    SW08
000200                  SW7=    SW07
000100                  SW6=    SW06
000040                  SW5=    SW05
000020                  SW4=    SW04
000010                  SW3=    SW03
000004                  SW2=    SW02
000002                  SW1=    SW01
000001                  SW0=    SW00
                        ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
100000                  BIT15=  100000
040000                  BIT14=  40000
020000                  BIT13=  20000
010000                  BIT12=  10000
004000                  BIT11=  4000
002000                  BIT10=  2000
001000                  BIT09=  1000
000400                  BIT08=  400
000200                  BIT07=  200
000100                  BIT06=  100
000040                  BIT05=  40
000020                  BIT04=  20
000010                  BIT03=  10
000004                  BIT02=  4
000002                  BIT01=  2
000001                  BIT00=  1
001000                  BIT9=   BIT09
000400                  BIT8=   BIT08
000200                  BIT7=   BIT07
000100                  BIT6=   BIT06
000040                  BIT5=   BIT05
000020                  BIT4=   BIT04
000010                  BIT3=   BIT03
```

BASIC DEFINITIONS

```
000004                  BIT2=   BIT02
000002                  BIT1=   BIT01
000001                  BIT0=   BIT00
                        ;*BASIC "CPU" TRAP VECTOR ADDRESSES
000004                  ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
000010                  RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
000014                  TBITVEC=14              ;;"T" BIT
000014                  TRTVEC= 14              ;;TRACE TRAP
000014                  BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
000020                  IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
000024                  PWRVEC= 24              ;;POWER FAIL
000030                  EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
000034                  TRAPVEC=34              ;;"TRAP" TRAP
000060                  TKVEC=  60              ;;TTY KEYBOARD VECTOR
000064                  TPVEC=  64              ;;TTY PRINTER VECTOR
000240                  PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
  266                   .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
```

K1

MEMORY MANAGEMENT DEFINITIONS

```
                172372                  KDPAR5= 172372
                172374                  KDPAR6= 172374
                172376                  KDPAR7= 172376
    267                                 .SBTTL  UNIBUS MAP REGISTER DEFINITIONS
    268
    269                                 ;*THE LOWER 16 BITS OF THE MAP REGISTERS ARE LABELED 'MAPLXX'
    270                                 ;*THE UPPER 6 BITS OF THE MAP REGISTERS ARE LABELED 'MAPHXX'
    271
    272
    273         170200                  MAPL00 = 170200
    274         170202                  MAPH00 = 170202
    275         170204                  MAPL01 = 170204
    276         170206                  MAPH01 = 170206
    277         170210                  MAPL02 = 170210
    278         170212                  MAPH02 = 170212
    279         170214                  MAPL03 = 170214
    280         170216                  MAPH03 = 170216
    281         170220                  MAPL04 = 170220
    282         170222                  MAPH04 = 170222
    283         170224                  MAPL05 = 170224
    284         170226                  MAPH05 = 170226
    285         170230                  MAPL06 = 170230
    286         170232                  MAPH06 = 170232
    287         170234                  MAPL07 = 170234
    288         170236                  MAPH07 = 170236
    289         170240                  MAPL10 = 170240
    290         170242                  MAPH10 = 170242
    291         170244                  MAPL11 = 170244
    292         170246                  MAPH11 = 170246
    293         170250                  MAPL12 = 170250
    294         170252                  MAPH12 = 170252
    295         170254                  MAPL13 = 170254
    296         170256                  MAPH13 = 170256
    297         170260                  MAPL14 = 170260
    298         170262                  MAPH14 = 170262
    299         170264                  MAPL15 = 170264
    300         170266                  MAPH15 = 170266
    301         170270                  MAPL16 = 170270
    302         170272                  MAPH16 = 170272
    303         170274                  MAPL17 = 170274
    304         170276                  MAPH17 = 170276
    305         170300                  MAPL20 = 170300
    306         170302                  MAPH20 = 170302
    307         170304                  MAPL21 = 170304
    308         170306                  MAPH21 = 170306
    309         170310                  MAPL22 = 170310
    310         170312                  MAPH22 = 170312
    311         170314                  MAPL23 = 170314
    312         170316                  MAPH23 = 170316
    313         170320                  MAPL24 = 170320
    314         170322                  MAPH24 = 170322
    315         170324                  MAPL25 = 170324
    316         170326                  MAPH25 = 170326
    317         170330                  MAPL26 = 170330
    318         170332                  MAPH26 = 170332
    319         170334                  MAPL27 = 170334
    320         170336                  MAPH27 = 170336
```

UNIBUS MAP REGISTER DEFINITIONS

```
                                MAPL30 = 170340
                                MAPH30 = 170342
                                MAPL31 = 170344
                                MAPH31 = 170346
                                MAPL32 = 170350
                                MAPH32 = 170352
                                MAPL33 = 170354
                                MAPH33 = 170356
                                MAPL34 = 170360
                                MAPH34 = 170362
                                MAPL35 = 170364
                                MAPH35 = 170366
                                MAPL36 = 170370
                                MAPH36 = 170372
                                MAPL37 = 170374
                                MAPH37 = 170376

                                MAPL0 = MAPL00
                                MAPH0 = MAPH00
                                MAPL1 = MAPL01
                                MAPH1 = MAPH01
                                MAPL2 = MAPL02
                                MAPH2 = MAPH02
                                MAPL3 = MAPL03
                                MAPH3 = MAPH03
                                MAPL4 = MAPL04
                                MAPH4 = MAPH04
                                MAPL5 = MAPL05
                                MAPH5 = MAPH05
                                MAPL6 = MAPL06
                                MAPH6 = MAPH06
                                MAPL7 = MAPL07
                                MAPH7 = MAPH07

                                .SBTTL UBA SPECIFIC REGISTERS

                                MMR0    = 177572                ; MEMORY MANAGEMENT REGISTER DEFINITIONS
                                MMR1    = 177574                ;
                                MMR2    = 177576                ;
                                MMR3    = 172516                ;
                                UFDSET  = 1                     ; FLAG FOR UFD MODE
                                BCSR    = 177520                ; BOOT/DIAGNOSTIC STATUS REGISTER
                                DCSR    = 177730                ; DIAGNOSTIC CONTROLLER STATUS REGISTER
                                DDR     = 177732                ; DIAGNOSTIC DATA REISTER
                                KMCR    = 177734                ; KTJ11-B MEMORY CONFIGURATION REGISTER
                                CCR     = 177746                ; CACHE CONTOL REGISTER FOR CPU
                                PCR     = 177522                ; PAGE CONTROL REGISTER
                                POLY    = 120001                ; POLYNOMIAL USED FOR CRC ROUTINES
                                SIMLGO  = 170014                ; SIMULTANEOUS GO ADDRESS FOR MULTIPLE UBE'S

                                .SBTTL  TRAP CATCHER
                        000000          .=0
                                ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
                                ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
                                ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
                        000174          .=174
                000174  000000  DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
```

```
000176  000000                  SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER
372     000200                  .=200
373 000200  005037  001160              CLR     $TMP0
374 000204  000137  002500              JMP     @#START
375     000220                  .=220
376 000220  012737  000777  001160      MOV     #777,$TMP0
377 000226  000137  002500              JMP     @#START
378
379                             .SBTTL  ACT11 HOOKS
                                ;;******************************************************************
                                ;HOOKS REQUIRED BY ACT11
        000232                          $SVPC=.                 ;SAVE PC
        000046                          .=46
    000046  023142                      $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
        000052                          .=52
    000052  000000                      .WORD   0               ;;2)SET LOC.52 TO ZERO
        000232                          .=$SVPC                 ;; RESTORE PC
380                             .SBTTL  APT PARAMETER BLOCK
                                ;;******************************************************************
                                ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                ;;******************************************************************
        000232                          .$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)
```

```
    381                                 .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                      $TMP0:  .WORD   0               ;;USER DEFINED
    001162  000000                      $TMP1:  .WORD   0               ;;USER DEFINED
    001164  000000                      $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
    001166  000000                      $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
    001170     207      377     377     $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
    001173     000
    001174     077                      $QUES:  .ASCII  /?/             ;;QUESTION MARK
    001175     015                      $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
    001176     012      000             $LF:    .ASCIZ  <12>            ;;LINE FEED
                                        ;;**********************************************************************
                                        .SBTTL  APT MAILBOX-ETABLE
                                        ;;**********************************************************************
                                        .EVEN
    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                      $ERRTB:
382
383                             .SBTTL  ERROR DEFINITIONS
384                             ;       ERROR 1
385
386 001324  027006                      EM1                     ; TIMEOUT ON ACCESSING A MAP REGISTER
387 001326  031174                      DH1                     ; TEST # ERROR PC  ERROR #  ADDRESS
388 001330  032166                      DT1                     ; TEST,$ERRPC,ERRNUM,$BDADR
389 001332  000000                      0
390                             ;       ERROR 2
391
392 001334  027052                      EM2                     ; MAP REGISTER COULD NOT BE CLEARED
393 001336  031243                      DH2                     ;                               GOOD  BAD
394                                                             ; TEST # ERROR PC  ERROR # DATA  DATA ADDRESS
395 001340  032200                      DT2                     ; TEST,$ERRPC,ERRNUM,$GDDAT,$BDDAT,$BDADR
396 001342  000000                      0
397                             ;       ERROR 3
398
399 001344  027114                      EM3                     ; MAP REGISTER COULD NOT HOLD PATTERN
400 001346  031243                      DH2                     ;                               GOOD  BAD
401                                                             ; TEST # ERROR PC  ERROR # DATA  DATA ADDRESS
402 001350  032200                      DT2                     ; TEST,$ERRPC,ERRNUM,$GDDAT,$BDDAT,$BDADR
403 001352  000000                      0
404                             ;       ERROR 4
405
406 001354  027160                      EM4                     ; MAP REGISTER HAS NOT BEEN ADDRESSED CORRECTLY
407 001356  031335                      DH3                     ;                               GOOD      BAD
408                                                             ; TEST # ERROR PC  ERROR #  ADDRESS  ADDRESS
409 001360  032216                      DT3                     ; TEST,$ERRPC,ERRNUM,$GDADR,$BDADR
410 001362  000000                      0
411                             ;       ERROR 5
412
413 001364  027236                      EM5                     ; THERE WAS NO DIFFERENCE FOUND BETWEEN HI AND LO MAP REGIST
ERS
414 001366  031425                      DH4                     ;                               HI    LOW
415                                                             ; TEST # ERROR PC  ERROR #  MAP   MAP
416 001370  032232                      DT4                     ; TEST,$ERRPC,ERRNUM,$GDDAT,$BDDAT
417 001372  000000                      0
418                             ;       ERROR 6
419
420 001374  027334                      EM6                     ; ERROR IN BITS 3-6,9-14 IN THE DCSR
421 001376  031503                      DH5                     ;                               GOOD  BAD
422                                                             ; TEST # ERROR PC  ERROR #  DATA  DATA
423 001400  032232                      DT4                     ; TEST,$ERRPC,ERRNUM,$GDDAT,$BDDAT
424 001402  000000                      0
425                             ;       ERROR 7
426
427 001404  027377                      EM7                     ; DCSR DID RESPOND PROPERLY ON RESET
```

D2

ERROR DEFINITIONS

```
001406  031503                          DH5                     ;                                       GOOD  BAD
                                                                ; TEST # ERROR PC  ERROR #  DATA  DATA
001410  032232                          DT4                     ; TEST,$ERRPC,ERRNUM,$GDDAT,$BDDAT
001412  000000                          0
                                ;       ERROR 10
                                
001414  027446                          EM10                    ; TIMEOUT HAS OCCURED ON ACCESS TO THE DCSR
001416  031174                          DH1                     ; TEST # ERROR PC  ERROR #  ADDRESS
001420  032166                          DT1                     ; TEST,$ERRPC,ERRNUM,$BDADR
001422  000000                          0
                                ;       ERROR 11
                                
001424  027520                          EM11                    ; KMCR BITS 0-5,8 DID NOT GET SET CORRECTLY
001426  031243                          DH2                     ;                                       GOOD  BAD
                                                                ; TEST # ERROR PC  ERROR # DATA  DATA ADDRESS
001430  032200                          DT2                     ; TEST,$ERRPC,ERRNUM,$GDDAT,$BDDAT,$BDADR
001432  000000                          0
                                ;       ERROR 12
                                
001434  027572                          EM12                    ; TIMEOUT HAS OCCURED ON ACCESS TO THE KMCR
001436  031174                          DH1                     ; TEST # ERROR PC  ERROR #  ADDRESS
001440  032166                          DT1                     ; TEST,$ERRPC,ERRNUM,$BDADR
001442  000000                          0
                                ;       ERROR 13
                                
001444  027644                          EM13                    ; ERROR IN DATA PATH
001446  031565                          DH13                    ; TEST # ERROR PC ERROR # PATTERN DDR
001450  032232                          DT4                     ; TEST,$ERRPC,ERRNUM,$GDDAT,$BDDAT
001452  000000                          0
                                ;       ERROR 14
                                
001454  027713                          EM14                    ; ERROR IN DATA OUT
001456  031503                          DH5                     ;                                       GOOD  BAD
                                                                ; TEST # ERROR PC  ERROR #  DATA  DATA
001460  032232                          DT4                     ; TEST,$ERRPC,ERRNUM,$GDDAT,$BDDAT
001462  000000                          0
                                ;       ERROR 15
                                
001464  027743                          EM15                    ; ERROR IN DATA IN
001466  031565                          DH13                    ; TEST # ERROR PC  ERROR # PATTERN DDR
001470  032232                          DT4                     ; TEST,$ERRPC,ERRNUM,$GDDAT,$BDDAT
001472  000000                          0
                                ;       ERROR 16
                                
001474  027772                          EM16                    ; DDR NOT ZERO WHEN DCSR SELECTS UNIBUS LINES
001476  031503                          DH5                     ;                                       GOOD  BAD
                                                                ; TEST # ERROR PC  ERROR #  DATA  DATA
001500  032232                          DT4                     ; TEST,$ERRPC,ERRNUM,$GDDAT,$BDDAT
001502  000000                          0
                                ;       ERROR 17
                                
001504  030054                          EM17                    ; ERROR IN SETTING DCSR<7>
001506  031640                          DH16                    ; TEST # ERROR PC ERROR #
001510  032246                          DT16                    ; TEST,$ERRPC,ERRNUM
001512  000000                          0
                                ;       ERROR 20
```

ERROR DEFINITIONS

```
001514  030107                          EM20            ; ERROR IN UNIQUE ADDRESSING OF REGISTERS
001516  031677                          DH20            ; TEST # ERROR PC ERROR # KMCR PAIR FAILED
001520  032256                          DT20            ; TEST,$ERRPC,ERRNUM,KMCR, $BDADR
001522  000000                          0
                                ;       ERROR 21

001524  030165                          EM21            ; REG. PAIR 31. PERFORMS RELOCATION
001526  031640                          DH16            ; TEST # ERROR PC ERROR #
001530  032246                          DT16            ; TEST,$ERRPC,ERRNUM
001532  000000                          0
                                ;       ERROR 22

001534  030232                          EM22            ; NXM CONDITION COULDN'T BE CREATED THRU ALU
001536  031640                          DH16            ; TEST # ERROR PC ERROR #
001540  032246                          DT16            ; TEST,$ERRPC,ERRNUM
001542  000000                          0
                                ;       ERROR 23

001544  030306                          EM23            ; ALU ERROR
001546  031757                          DH23            ;                                       GOOD  BAD

001550  032302                          DT23            ; TEST,$ERRPC,ERRNUM,$GDDAT,$BDDAT,KIPAR6,$BDADR
001552  000000                          0
                                ;       ERROR 24

001554  030320                          EM24            ; CPU CACHE ERROR
001556  031640                          DH16            ; TEST # ERROR PC ERROR #
001560  032246                          DT16            ; TEST,$ERRPC,ERRNUM
001562  000000                          0
                                ;       ERROR 25

001564  030340                          EM25            ; KMCR<4-0> DOESN'T DISABLE MAIN MEMORY
001566  031640                          DH16            ; TEST # ERROR PC ERROR #
001570  032246                          DT16            ; TEST,$ERRPC,ERRNUM
001572  000000                          0
                                ;       ERROR 26

001574  030410                          EM26            ; KMCR DOES NOT REFLECT EXPECTED STATUS OF THE CACHE/
001576  031503                          DH5             ;                                       GOOD  BAD

001600  032232                          DT4             ; TEST,$ERRPC,ERRNUM,$GDDAT,$BDDAT
001602  000000                          0
                                ;       ERROR 27

001604  030473                          EM27            ; ERROR IN CACHE TAG REGISTERS
001606  032070                          DH27            ; TEST # ERROR PC  ERROR # ADDRESS
001610  032322                          DT27            ; TEST,$ERRPC,ERRNUM,MAPH01,MAPL01
001612  000000                          0
                                ;       ERROR 30

001614  030530                          EM30            ; ERROR IN THE DMA CACHE DATA RAMS
001616  031174                          DH1             ; TEST # ERROR PC  ERROR # ADDRESS
001620  032336                          DT30            ; TEST,$ERRPC,ERRNUM,MAPL00
001622  000000                          0
                                ;       ERROR 31

001624  030571                          EM31            ; ERROR IN THE M9312 BOOT ROM SECTION
```

F2

ERROR DEFINITIONS

```
                                        DH16                    ; TEST # ERROR PC  ERROR #
                                        DT16                    ; TEST,$ERRPC,ERRNUM
                                        0
                                ;       ERROR 32

                                        EM32                    ; ERROR IN ARBITRATION LOGIC THRU UBE
                                        DH16                    ; TEST # ERROR PC  ERROR #
                                        DT16                    ; TEST,$ERRPC,ERRNUM
                                        0
                                ;       ERROR 33

                                        EM33                    ; ERROR TRYING TO DO DMA CYCLES THRU UBE
                                        DH16                    ; TEST # ERROR PC  ERROR #
                                        DT16                    ; TEST,$ERRPC,ERRNUM
                                        0
                                ;       ERROR 34

                                        EM34                    ; ERROR IN THE UNIBUS MEMORY TEST
                                        DH16                    ; TEST # ERROR PC  ERROR #
                                        DT16                    ; TEST,$ERRPC,ERRNUM
                                        0
                                ;       ERROR 35

                                        EM35                    ; UNEXPECTED TIMEOUT HAS OCCURED
                                        DH16                    ; TEST # ERROR PC  ERROR #
                                        DT21                    ; TEST,$BDADR,ERRNUM
                                        0
                                ;       ERROR 36

                                        EM36                    ; UNIBUS TIMEOUT DID NOT OCCUR
                                        DH16                    ; TEST # ERROR PC  ERROR #
                                        DT16                    ; TEST,$ERRPC,ERRNUM
                                        0
                                ;       ERROR 37

                                        EM37                    ; DCSR<3> DIDN'T DISABLE UBA ROM'S
                                        DH16                    ; TEST # ERROR PC  ERROR #
                                        DT16                    ; TEST,$ERRPC,ERRNUM
                                        0

                                .SBTTL  GLOBAL VARIABLES
                                ERRNUM: .WORD   0                       ; ERROR NUMBER FOR REPORT
                                TEST:   .WORD   0                       ; TEST NUMBER FOR REPORT
                                PMIS:   .WORD   0                       ; PMI MEMORY SIZE
                                UBECT:  .WORD   0                       ; NUMBER OF UBE'S
                                BE1INT: .WORD   0                       ; UBE #1 INTERRUPT FLAG
                                BE2INT: .WORD   0                       ; UBE #2 INTERRUPT FLAG
                                TOUT:   .WORD   0                       ; TIMEOUT FLAG
                                MCSR:   .WORD   172100                  ; POINTER TO MEMORY CSR
                                WRTBUF: .BLKW   20                      ; WRITE BUFFER
000377  007417  031463  PTRN16: .WORD   377,7417,31463,52525,125252,0    ; 16-BIT BINARY DIVIDE
052525  125252  000000
000017  000014  000025  PTRN6:  .WORD   17,14,25,52,0                    ; 6-BIT BINARY DIVIDE
000052  000000

                                .SBTTL  CACHE RAM IMAGE TABLE
```

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
```

SUBROUTINE - DIAGNOSTIC_DATA_OUT SUBROUTINE

```
                                        ;
                                        ;-------------------------------------------------------------------------
                                        
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
                                ;
                                ;-------------------------------------------------------------

                                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?
```

SUBROUTINE - MEMORY SIZE

```
    763 002402  001367                     BNE     1$                      ; . IF NOT, BRANCH
    764 002404  012701  000040              MOV     #40,R1                  ; IF 2M PRESENT, ONLY 21ST MASKED
    765 002410  000402                      BR      4$                      ; BRANCH
    766 002412  005726              3$:     TST     (SP)+                   ; RESTORE STACK
    767 002414  005726                      TST     (SP)+                   ;
    768 002416  010237  000004      4$:     MOV     R2,ERRVEC               ; RESTORE TIMEOUT VECTOR
    769 002422  000207                      RTS     PC                      ; RETURN
    770
    771                             .SBTTL SUBROUTINE - SIZE UNIBUS MEMORY FROM KMCR
    772
    773                             ;* ON RETURN FROM THIS SUBROUTINE R2 WOULD HAVE PAR VALUE
    774                             ;* OF UNIBUS MEMORY
    775                             ;
    776 002424  013701  177734      UMSIZ:  MOV     KMCR,R1                 ; SAVE KMCR
    777 002430  042701  000040      1$:     BIC     #BIT05,R1               ; LEAVE ONLY # OF PAGES
    778 002434  012702  007600              MOV     #7600,R2                ; START W/O UNIBUS MEMORY
    779 002440  162702  000200      100$:   SUB     #200,R2                 ; . SUBTRACT 4K
    780 002444  077103                      SOB     R1,100$                 ; . FOR ALL PAGES PRESENT
    781 002446  000207                      RTS     PC
    782
    783
    784                             .SBTTL SUBROUTINE - INITIALIZE THE UBE'S
    785
    786                             ;* THIS ROUTINE IS USED TO INITIALIZE THE UBE'S
    787                             ;
    788                             ;  BGNROUTINE
    789                             ;
    790                             ;       SAVE R0,R1
    791                             ;       LET R0 := FIRST UBE ADDRESS
    792                             ;       DO FOR (R0) := BE1DB TO BE1CR1
    793                             ;       .  LET (R0) := 0
    794                             ;       ENDDO
    795                             ;       RESTORE R1,R0
    796                             ;       RETURN
    797                             ;
    798                             ;  ENDROUTINE
    799                             ;
    800                             ;---------------------------------------------------------------
    801
    802 002450                      IUBE:
        002450  010046                      MOV     R0,-(SP)        ;;PUSH R0 ON STACK
        002452  010146                      MOV     R1,-(SP)        ;;PUSH R1 ON STACK
    803 002454  013700  002150              MOV     BE1DB,R0        ; POINT R0 TO UBE REGISTERS
    804 002460  012701  000010              MOV     #10,R1          ; SET UP A LOOP COUNTER
    805 002464  012720  000000      5$:     MOV     #0,(R0)+        ; CLEAR OUT A REGISTER
    806 002470  077103                      SOB     R1,5$           ; HAVE WE INITIALIZED THE UBE ?
    807 002472  012601                      MOV     (SP)+,R1        ;;POP STACK INTO R1
        002474  012600                      MOV     (SP)+,R0        ;;POP STACK INTO R0
    808 002476  000207                      RTS     PC
    809
    810
    811                             ;
    812                             ; STARTING POINT OF PROGRAM
    813                             ;
    814
    815 002500                      START:
                                    ;; LCP/ORION ROUTINE TO SAVE EMTULATOR AND PRIORITY
```

K2

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$:     EMT     42                      ;; GET ADDRESS OF XXDP DCA TABLE
002544  005060  000042                  CLR     42(R0)                  ;; CLR XXDP+ "DRSERR"
002550  013737  000030  002566          MOV     30,SAV30                ;; SAVE EMULATOR ADDRESS
002556  013737  000032  002570          MOV     32,SAV32                ;; SAVE EMULATOR PRIORITY LEVEL
002564  000404                          BR      VMKOR                   ;; GET AROUND TAG AREA
002566  000000          SAV30:  .WORD   0                               ;; PUT EMULATOR INFO HERE
002570  000000          SAV32:  .WORD   0                               ;; PUT PRIORITY LOCATION        HERE
002572  000000          UFDFLG: .WORD   0                               ;; USER FRIENDLY MODE FLAG
002574  000000          UQUIET: .WORD   0                               ;; UFD QUIET MODE FLAG
002576                  VMKOR:
                        ;*******************************************************************************
  816                   .SBTTL  INITIALIZE THE COMMON TAGS
                        ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
002576  012706  001100          MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
002602  005026                  CLR     (R6)+           ;;CLEAR MEMORY LOCATION
002604  022706  001140          CMP     #SWR,R6 ;;DONE?
002610  001374                  BNE     .-6             ;;LOOP BACK IF NO
002612  012706  001100          MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                        ;;INITIALIZE A FEW VECTORS
002616  012737  023162  000020  MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
002624  012737  000340  000022  MOV     #340,@#IOTVEC+2 ;;LEVEL 7
002632  012737  025536  000030  MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
002640  012737  000340  000032  MOV     #340,@#EMTVEC+2 ;;LEVEL 7
002646  012737  026544  000034  MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
002654  012737  000340  000036  MOV     #340,@#TRAPVEC+2;LEVEL 7
002662  012737  026630  000024  MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
002670  012737  000340  000026  MOV     #340,@#PWRVEC+2 ;;LEVEL 7
002676  013737  023022  023014  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
```

L2

INITIALIZE THE COMMON TAGS

```
        003030  132737  000200  001221          BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
        003036  001403                          BEQ     67$             ;;YES,USE NON-APT SWITCH
        003040  012737  001222  001140          MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
        003046                          67$:
    817 003046  005737  002574                  TST     UQUIET          ; ARE WE UNDER UFD QUIET MODE ?
    818 003052  001056                          BNE     1$              ; YES,THEN SKIP THE DIAGNOSTIC TITLE PRINTOUT
    819 003054  122737  000001  001220          CMPB    #APTENV,$ENV    ; APT?
    820 003062  001452                          BEQ     1$              ; IF YES, SKIP PRINTOUT
    821                             .SBTTL  TYPE PROGRAM NAME
                                    ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
        003064  005227  177777                  INC     #-1             ;;FIRST TIME?
        003070  001047                          BNE     68$             ;;BRANCH IF NO
        003072  022737  023142  000042          CMP     #$ENDAD,@#42    ;;ACT-11?
        003100  001443                          BEQ     68$             ;;BRANCH IF YES
        003102  104401  003150                  TYPE    ,69$            ;;TYPE ASCIZ STRING
                                .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
        003106  005737  000042                  TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
        003112  001012                          BNE     70$             ;;BRANCH IF YES
        003114  123727  001220  000001          CMPB    $ENV,#1         ;;ARE WE RUNNING UNDER APT?
        003122  001406                          BEQ     70$             ;;BRANCH IF YES
        003124  023727  001140  000176          CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
        003132  001005                          BNE     71$             ;;BRANCH IF NO
        003134  104406                          GTSWR                   ;;GET SOFT-SWR SETTINGS
        003136  000403                          BR      71$
        003140  112737  000001  001134  70$:    MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
        003146                          71$:
        003146  000420                          BR      68$             ;;GET OVER THE ASCIZ
                                        ;;69$:  .ASCIZ  <CRLF>* COKTABO KTJ11-B DIAGNOSTIC *<CRLF>
        003210                          68$:
    822 003210  012737  002234  000004  1$:     MOV     #TIMOUT,@#4     ; SETUP TIMEOUT VECTOR FOR UNEXPECTED TIMEOUT
    823 003216  012737  000340  000006          MOV     #340,@#6        ;
    824 003224  005037  001716                  CLR     TEST            ; TEST # FOR ERROR REPORTING
    825                             ; SIZE MEMORY IN 4K PAGES
    826 003230  004737  002244                  JSR     PC,MAPPR        ; REMAP PROGRAM AREA
    827 003234  052737  000060  172516          BIS     #BIT05!BIT04,MMR3       ; ENABLE RELOCATION AND 22BITS
    828 003242  005237  177572                  INC     MMR0            ; ENABLE MMU
    829 003246  004737  002332                  JSR     PC,MEMSIZ       ; GET HIGHEST 128K OF MEMORY
    830 003252  005037  177572                  CLR     MMR0            ; DISABLE MMU
    831 003256  005037  172516                  CLR     MMR3            ;
    832 003262  013737  172354  001720          MOV     KIPAR6,PMIS     ; STORE HIGHEST PAGE
    833 003270                          RESTART:
    834 003270  012706  001100                  MOV     #1100,SP        ; SET UP THE STACK POINTER
    835
    836
    837
```

TEST - UNIBUS MAP REGISTER TESTS

```
                                .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
                                ;;*************************************************************************
    003274  000004              TST1:   SCOPE
                                
                                ;
                                ; SET UP MEMORY TIMEOUT VECTOR TO ERROR ROUTINE
                                ;
    003276  013705  000004              MOV     @#4,R5                  ; SAVE THE TIMEOUT VECTOR
    003302  012737  003374  000004      MOV     #20$,@#4                ; SET UP TIMEOUT VECTOR TO ERROR ROUTINE
    003310  012737  000340  000006      MOV     #340,@#6                ; SET UP TIMEOUT PRIORITY
                                ;
                                ; DISABLE UNIBUS MAPPING
                                ;
    003316  042737  000040  172516      BIC     #BIT5,@#SR3             ; DISABLE UNIBUS MAPPING
                                ;
                                ; TRY TO ACCESS ALL THE UNIBUS MAP REGISTERS
                                ;
    003324  005001                      CLR     R1                      ; INITIALIZE LOOP COUNT
    003326  012700  170200      5$:     MOV     #MAPL00,R0              ; . R0 POINTS TO MAP REGISTER #0
    003332  005720              10$:    TST     (R0)+                   ; . . TRY TO READ MAP REGISTER
    003334  020027  170400      12$:    CMP     R0,#170400              ; . . HAVE WE READ ALL THE MAP REGISTERS
    003340  001374                      BNE     10$                     ; . . NO THEN GO READ THE NEXT ONE
    003342  005701                      TST     R1                      ; . HAVE WE DONE THIS TWICE ?
    003344  001406                      BEQ     15$                     ; . GO TURN ON UNIBUS MAP AND TRY AGAIN
    003346  042737  000040  172516      BIC     #BIT5,SR3               ; . TURN OFF UNIBUS MAPPING
    003354  010537  000004              MOV     R5,@#4                  ; RESTORE THE TIMEOUT VECTOR
    003360  000417                      BR      TST2            ;; . GO TO NEXT TEST
    003362  005201              15$:    INC     R1                      ; . INCREMENT LOOP COUNT
    003364  052737  000040  172516      BIS     #BIT5,SR3               ; . TURN ON UNIBUS MAP
```

N2

T1 UNIBUS MAP REGISTER RESPONSE TEST

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

```
                                .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
                                ;;*************************************************************
003420  000004                  TST2:   SCOPE

                                ;
                                ; CLEAR ALL MAP REGISTERS
                                ;
003422  012700  170200                  MOV     #MAPL00,R0              ; GET ADDRESS TO 1ST MAP REGISTER
003426  005020                  5$:     CLR     (R0)+                   ; . CLEAR THE MAP REGISTER
003430  020027  170400                  CMP     R0,#170400              ; . HAVE WE CLEARED ALL THE MAP REGISTERS ?
003434  001374                          BNE     5$                      ; . NO THEN GO CLEAR THE NEXT ONE
```

```
                                        ;
                                        ; DO A READ,WRITE,READ ON THE REGISTERS TO
                                        ; VERIFY ALL THE BITS OF THE  LOW MAP REGISTERS
                                        ;
003436  012700  170200          MOV     #MAPLOO,R0          ; GET ADDRESS TO 1ST MAP REGISTER
003442  005710          10$:    TST     (R0)                ; . IS THE MAP REGISTER CLEAR ?
003444  001410                  BEQ     15$                 ; . YES,THEN GO CHECK THE BITS
003446  011037  001126          MOV     (R0),$BDDAT         ; . NO,THEN GET THE BAD DATA
003452  012737  000000  001124  MOV     #0,$GDDAT           ; .  GET THE GOOD DATA
003460  010037  001122          MOV     R0,$BDADR           ; .  GET THE ADDRESS THAT DID NOT CLEAR
003464  104002                  ERROR   ;2                  ; .  ERROR - REGISTER DID NOT CLEAR
003466  012701  001774  15$:    MOV     #PTRN16,R1          ; . GET POINTER TO PATTERN TABLE
003472  011110          20$:    MOV     (R1),(R0)           ; . . MOVE PATTERN TO MAP REGISTER
003474  012102                  MOV     (R1)+,R2            ; . . MOVE COPY OF PATTERN TO R2
003476  042702  000001          BIC     #BIT0,R2            ; . . CLEAR BIT0 OF THE PATTERN
003502  020210                  CMP     R2,(R0)             ; . . DID PATTERN GET WRITTEN CORRECTLY ?
003504  001410                  BEQ     25$                 ; . . YES,THEN GO SEE IF ALL PATTERNS HAVE BEEN WRIT
TEN
003506  011037  001126          MOV     (R0),$BDDAT         ; . . NO,GET THE BAD DATA
003512  005741                  TST     -(R1)               ; . .  POINT TO THE GOOD DATA
003514  012137  001124          MOV     (R1)+,$GDDAT        ; . .  GET THE GOOD DATA,AND POINT R1 TO THE NEXT DA
TA PATTERN
003520  010037  001122          MOV     R0,$BDADR           ; . .  GET THE ADDRESS OF DATA FAULT
003524  104003                  ERROR   ;3                  ; . .  ERROR - REGISTER COULD NOT HOLD PATTERN
003526  005711          25$:    TST     (R1)                ; . . ARE WE THROUGH THE TABLE ?
003530  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
                                        ;
003532  062700  000002          ADD     #2,R0               ; . POINT TO THE HIGH MAPPING REGISTER
003536  005710          30$:    TST     (R0)                ; . IS THE MAP REGISTER CLEAR ?
003540  001410                  BEQ     35$                 ; . YES,THEN GO CHECK THE BITS
003542  011037  001126          MOV     (R0),$BDDAT         ; . NO,THEN GET THE BAD DATA
003546  012737  000000  001124  MOV     #0,$GDDAT           ; .  GET THE GOOD DATA
003554  010037  001122          MOV     R0,$BDADR           ; .  GET THE ADDRESS THAT DID NOT CLEAR
003560  104002                  ERROR   ;2                  ; .  ERROR - REGISTER DID NOT CLEAR
003562  012701  002010  35$:    MOV     #PTRN6,R1           ; . GET POINTER TO PATTERN TABLE
003566  011110          40$:    MOV     (R1),(R0)           ; . . MOVE PATTERN TO MAP REGISTER
003570  012102                  MOV     (R1)+,R2            ; . . MOVE COPY OF THE PATTERN TO R2
003572  042702  177700          BIC     #177700,R2          ; . . MASK OUT THE UPPER 10 BITS
003576  020210                  CMP     R2,(R0)             ; . . DID PATTERN GET WRITTEN CORRECTLY ?
003600  001410                  BEQ     45$                 ; . . YES,THEN GO SEE IF ALL PATTERNS HAVE BEEN WRIT
TEN
003602  011037  001126          MOV     (R0),$BDDAT         ; . . NO,GET THE BAD DATA
003606  005741                  TST     -(R1)               ; . .  POINT TO THE GOOD DATA
003610  012137  001124          MOV     (R1)+,$GDDAT        ; . .  GET THE GOOD DATA,AND POINT R1 BACK TO THE NE
XT PATTERN
003614  010037  001122          MOV     R0,$BDADR           ; . .  GET THE ADDRESS OF DATA FAULT
003620  104003                  ERROR   ;3                  ; . .  ERROR - REGISTER COULD NOT HOLD PATTERN
003622  005711          45$:    TST     (R1)                ; . . ARE WE THROUGH THE TABLE ?
003624  001360                  BNE     40$                 ; . . NO THEN GO WRITE NEXT PATTERN
                                        ;
                                        ; SEE IF WE HAVE CHECKED ALL THE MAP REGISTERS
                                        ;
003626  062700  000002          ADD     #2,R0               ; . POINT TO NEXT REGISTER
003632  020027  170400          CMP     R0,#170400          ; . ARE WE THROUGH ALL THE MAP REGISTERS ?
003636  001301                  BNE     10$                 ; . GO CHECK THE NEXT REGISTER
```

TEST - UNIBUS MAP REGISTER ADDRESS UNIQUENESS

```
                                .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
                                ;;*****************************************************************
003640  000004                  TST3:   SCOPE
                                ;
                                ; WRITE ADDRESS INTO ALL MAP REGISTERS
                                ;
003642  012700  170200                  MOV     @#MAPLOO,R0             ; GET ADDRESS OF 1ST MAP REGISTER
003646  012701  170200                  MOV     @#MAPLOO,R1             ; GET ADDRESS OF 1ST MAP REGISTER
003652  010021                  5$:     MOV     R0,(R1)+                ; . WRITE ADDRESS TO ITSELF
003654  062700  000002                  ADD     #2,R0                   ; . GET THE NEXT ADDRESS
003660  022700  170400                  CMP     #170400,R0              ; . HAVE WE WRITTEN TO ALL MAP REGISTERS ?
003664  001372                          BNE     5$                      ; . NO,THEN GO WRITE TO NEXT ADDRESS
                                ;
                                ; MAKE SURE EACH LOW MAP REGISTER CONTAINS IT'S ADDRESS
                                ;
003666  012700  170200                  MOV     @#MAPLOO,R0             ; GET ADDRESS TO 1ST MAP REGISTER
003672  012701  170200                  MOV     @#MAPLOO,R1             ; GET ADDRESS TO 1ST MAP REGISTER
003676  020021                  10$:    CMP     R0,(R1)+                ; . WAS THE ADDRESS WRITTEN TO THE MAP REGISTER ?
003700  001406                          BEQ     15$                     ; . YES,THEN GO CHECK THE HIGH MAP REGISTER
003702  005740                          TST     -(R0)                   ; . NO GET THE ADDRESS THAT WAS WRITTEN
003704  010037  001120                  MOV     R0,$GDADR               ; .  GET THE GOOD ADDRESS
003710  012037  001122                  MOV     (R0)+,$BDADR            ; .  GET THE BAD ADDRESS,AND POINT BACK TO NEXT ADDR
ESS
003714  104004                          ERROR   +4                      ; .  ERROR IN ADDRESSING OF MAP REGISTERS
                                ;
                                ; MAKE SURE EACH HIGH MAP REGISTER CONTAINS IT'S ADDRESS
                                ;
003716  062700  000002          15$:    ADD     #2,R0                   ; . POINT R0 TO HIGH MAP REGISTER
003722  010002                          MOV     R0,R2                   ; . PUT A COPY OF IT INTO R2
003724  042702  177700                  BIC     #177700,R2              ; . MASK OUT THE UPPER 10 BITS
003730  020221                          CMP     R2,(R1)+                ; . WAS THE ADDRESS WRITTEN TO THE HIGH MAP REGISTER
 ?
003732  001406                          BEQ     20$                     ; . YES,THEN GO CHECK IF WE ARE THROUGH ALL THE MAP
REGISTERS
003734  005740                          TST     -(R0)                   ; . NO GET THE ADDRESS THAT WAS WRITTEN
003736  010037  001120                  MOV     R0,$GDADR               ; .  GET THE GOOD ADDRESS
```

E3

T3 UNIBUS MAP REGISTER UNIQUENESS TEST

```
   1071 003742  012037  001122                MOV     (R0)+,$BDADR            ; .  GET THE BAD ADDRESS,AND POINT BACK TO NEXT ADDR
ESS
   1072 003746  104004                         ERROR   +4                      ; .  ERROR IN ADDRESSING OF MAP REGISTERS
   1073                                 ;
   1074                                 ; CHECK TO SEE IF WE HAVE CHECKED ALL MAP REGISTERS
   1075                                 ;
   1076 003750  062700  000002         20$:    ADD     #2,R0                   ; . GET THE NEXT REGISTER
   1077 003754  020027  170400                 CMP     R0,#170400              ; . HAVE WE CHECKED ALL THE MAPPING REGISTERS
   1078 003760  001346                         BNE     10$                     ; . NO,THEN GO CHECK THE NEXT REGISTER
   1079
   1080
   1081
```

F3

TEST - UNIQENESS BETWEEN LOW MAP REGISTERS AND HIGH MAP REGIST

```
                                .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
                                ;;*****************************************************************
003762  000004                  TST4:   SCOPE
                                ;
                                ; WRITE #77777 TO ALL ADDRESSES
                                ;
003764  012700  170200                  MOV     #MAPL00,R0              ; GET ADDRESS OF FIRST MAP REGISTER
003770  012720  077777          5$:     MOV     #77777,(R0)+            ; . WRITE 77777 TO MAP REGISTER
003774  020027  170400                  CMP     R0,#170400              ; . HAVE WE WRITTEN TO ALL REGISTERS ?
004000  001373                          BNE     5$                      ; . NO,THEN GO WRITE TO THE NEXT REGISTER ?
                                ;
                                ; MAKE SURE THAT HIGH MAP REGISTER BITS 6-15 ARE CLEARED
                                ;
004002  012700  170200                  MOV     #MAPL00,R0              ; GET ADDRESS OF FIRST MAP REGISTER
004006  012002                  10$:    MOV     (R0)+,R2                ; . GET CONTENTS OF LOW MAP REGISTER
004010  012003                          MOV     (R0)+,R3                ; . GET CONTENTS OF HIGH MAP REGISTER
004012  160302                          SUB     R3,R2                   ; . DID THE REGISTERS GET ADDRESSED CORRECTLY ?
004014  003007                          BGT     15$                     ; . YES,THEN GO SEE IF WE HAVE CHECKED ALL THE MAP R
EGISTERS
004016  005740                          TST     -(R0)                   ; . NO,
004020  005740                          TST     -(R0)                   ; .  POINT TO THE LOW MAP REGISTER
004022  012037  001124                  MOV     (R0)+,#GDDAT            ; .  GET CONTENTS OF LOW MAP REGISTER
004026  012037  001126                  MOV     (R0)+,#BDDAT            ; .  GET CONTENTS OF HIGH MAP REGISTER
004032  104005                          ERROR   +5                      ; .  ERROR - MAP REGISTERS WERE NOT ADDRESSED CORREC
TLY
                                ;
                                ; SEE IF WE HAVE CHECKED ALL THE MAP REGISTER PAIRS
                                ;
004034  020027  170400          15$:    CMP     R0,#170400              ; . HAVE WE CHECKED ALL THE REGISTERS ?
004040  001362                          BNE     10$                     ; . 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
                                ;;*****************************************************************
004042  000004                  TST5:   SCOPE
                                ;
                                ; SET UP MEMORY TIMEOUT VECTOR
                                ;
004044  012737  004216  000004          MOV     #20$,@#4                ; SET UP VECTOR PC TO ERROR ROUTINE
004052  012737  000340  000006          MOV     #340,@#6                ; SET UP VECTOR PSW TO PRIORITY 7
                                ;
                                ; READ ADDRESS OF DCSR TO SEE IF IT TIMES OUT
```

T5 DCSR REGISTER TEST

```
1190                                    ;
1191 004060  005737  177730             TST     @#DCSR                  ; WILL THE DCSR TIMEOUT ?
1192 004064  012737  002234  000004     MOV     #TIMOUT,@#4             ; RESTORE TIMEOUT VECTOR
1193                                    ;
1194                                    ; TRY TO WRITE ONES TO BITS 3-6 AND 9-14 ( BIT 03 SHOULD STAY WRITTEN)
1195                                    ;
1196 004072  052737  077170  177730     BIS     #77170,@#DCSR           ; TRY TO WRITE TO READ ONLY BITS
1197 004100  032737  077160  177730     BIT     #77160,@#DCSR           ; DID THEY GET WRITTEN TO ?
1198 004106  001412                     BEQ     5$                      ; NO,THEN GO CHECK THE EFFECT OF A RESET ON THE DCSR
1199 004110  013701  177730             MOV     @#DCSR,R1               ; GET COPY OF DCSR INTO R1
1200 004114  042701  100607             BIC     #100607,R1              ; MASK OUT UNTESTED BITS
1201 004120  010137  001126             MOV     R1,$BDDAT               ; PUT THEM INTO BAD DATA
1202 004124  012737  000010  001124     MOV     #10,$GDDAT              ; GOOD DATA
1203 004132  104006                     ERROR   .+6                     ; ERROR - IN BITS 3-6,9-14 OF DCSR
1204                                    ;
1205                                    ; CACHE THE RESET INSTRUCTION
1206                                    ;
1207 004134  012737  000200  001124  5$: MOV     #BIT7,$GDDAT            ; ONLY 7 SHOULD BE SET AFTER RESET
1208 004142  053737  177730  001124     BIS     DCSR,$GDDAT             ; SET BOOT BITS
1209 004150  005737  004162             TST     10$                     ; CACHE THE RESET
1210                                    ;
1211                                    ; SET UP DCSR,DO A RESET,MAKE SURE PROPER RESPONSE OCCURS
1212                                    ;
1213 004154  052737  000006  177730     BIS     #6,@#DCSR               ; SET BIT 7 AND CLEAR BIT 1,2
1214 004162  000005                 10$: RESET                           ; DO A BUS INIT
1215 004164  032737  000200  177730     BIT     #BIT7,@#DCSR            ; DID BIT 7 GET SET ?
1216 004172  001001                     BNE     15$                     ; YES,THEN GO CHECK IF BITS 1,2 GOT CLEAR
1217 004174  104007                     ERROR   .+7                     ; NO,THEN ERROR - BIT 7 DIDN'T GET SET
1218 004176  032737  000006  177730 15$: BIT     #6,@#DCSR               ; DID BIT 1,2 GET CLEARED ?
1219 004204  001410                     BEQ     TST6            ;; YES,THEN GO TO NEXT TEST
1220 004206  013737  177730  001126     MOV     @#DCSR,$BDDAT           ; NO,THEN GET CONTENTS OF THE DCSR
1221 004214  104007                     ERROR   .+7                     ;  ERROR - BIT 1,2 DID NOT GET CLEAR
1222                                    ;
1223                                    ; ERROR ROUTINE
1224                                    ;
1225 004216  012737  177730  001122 20$: MOV     #DCSR,$BDADR            ; GET ADDRESS OF THE DCSR
1226 004224  104010                     ERROR   .+10                    ; ERROR - DCSR REGISTER ACCESS HAS TIMED OUT
1227
1228
```

TEST - KMCR BITS TEST

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

                                ;;*****************************************************************
                                ;*TEST 6        KMCR MEMORY CONFIGURATION BITS TEST
                                ;;*****************************************************************
004226  000004                  TST6:   SCOPE
                                ;
                                ; SETUP TIMEOUT VECTOR,TO ERROR ROUTINE
                                ; VERIFY THAT AT LEAST SOME PMI MEMORY PRESENT
                                ;
004230  012737  004374  000004          MOV     #10$,@#4                ; SET VECTOR PC TO ERROR ROUTINE
004236  012737  000340  000006          MOV     #340,@#6                ; SET VECTOR PSW TO ERROR ROUTINE
004244  013737  177734  001160          MOV     KMCR,$TMP0              ; STORE KMCR
004252  012737  002234  000004          MOV     #TIMOUT,@#4             ; RESTORE TIMEOUT VECTOR
004260  042737  177700  001160          BIC     #177700,$TMP0           ; CLEAR ALL BUT U.MEMORY
004266  022737  000077  001160          CMP     #77,$TMP0               ; NO PMI MEMORY?
004274  001003                          BNE     2$                      ; IF SOME, GO TO THE NEXT TEST
004276  005037  001720                  CLR     PMIS                    ; OTHERWISE FLAG NO PMI MEMORY
004302  000440                          BR      TST7            ;; AND EXIT TEST
                                ;
                                ; CHECK THAT NOT IN DIAGNOSTIC MODE WRITE ACCESS TO KMCR<5-0> IS DISABLED
                                ;
004304  013701  177734          2$:     MOV     KMCR,R1                 ; STORE PATTERN FROM KMCR TO R1
004310  005101                          COM     R1                      ; COMPLEMENT THE PATTERN
004312  042701  177700                  BIC     #177700,R1              ; CLEAR BITS <15-6>
004316  042737  000400  177730          BIC     #BIT08,DCSR             ; MAKE SURE THAT NOT IN DIAGN. MODE
004324  042701  000010                  BIC     #BIT03,R1               ; ENABLE 32K, IN CASE OF ERROR
004330  050137  177734                  BIS     R1,KMCR                 ; TRY TO WRITE TO KMCR<5-0>
004334  020137  177734                  CMP     R1,KMCR                 ; WRITTEN OK?
004340  001014                          BNE     1$                      ; IF DIDN'T WRITE, BRANCH
004342  013737  177734  001124          MOV     KMCR,$GDDAT             ; NO,THEN GET THE GOOD DATA
004350  010137  001126                  MOV     R1,$BDDAT               ;  GET THE BAD DATA
004354  012737  177734  001122          MOV     #KMCR,$BDADR            ;  GET THE ADDRESS OF THE KMCR
004362  013737  001160  177734          MOV     $TMP0,KMCR              ; RESTORE KMCR
004370  104011                          ERROR   +11                     ; NO,THEN ERROR - BITS DID NOT GET SET CORRECTLY IN
THE KMCR
004372                          1$:
004372  000404                          BR      TST7            ;;  GO TO THE NEXT TEST
                                ;
```

J3


```
                                        ; TIMEOUT ROUTINE
                                        ;
004374  012737  177734  001122  10$:    MOV     #KMCR,$BDADR        ; GET THE ADDRESS OF THE KMCR
004402  104012                          ERROR   +12                 ; ERROR - MEMORY TIMEOUT HAS OCCURED
```

K3

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
                                ;;*************************************************************************
004404  000004                  TST7:   SCOPE
                                
                                ;
                                ; SET UP MEMORY TIMEOUT VECTOR AND PUT UBA INTO DIAGNOSTIC MODE
                                ;
004406  005737  001720                  TST     PMIS                    ; ANY PMI MEMORY?
004412  001425                  BEQ     TST10           ;; IF NONE, EXIT TEST
004414  012737  004446  000004  MOV     #10$,@#4                ; SET UP TIMEOUT VECTOR TO END OF TEST
004422  012737  000340  000006  MOV     #340,@#6                ;
004430  052737  000400  177730  BIS     #BIT8,DCSR              ; PUT UBA INTO DIAGNOSTIC MODE
                                
                                ;
                                ; READ A UNIBUS LOCATION
                                ;
                                
004436  005737  170002          TST     @#170002                ; TRY TO READ UBE REGISTER ?
004442  104036                  ERROR   +36                     ; ERROR - SHOULD TIMEOUT HERE
004444  000402                  BR      11$                     ; GO TO THE NEXT TEST
                                
                                ;
                                ; TIMEOUT ROUTINE
                                ;
                                
004446  005726          10$:    TST     (SP)+                   ; CLEAN UP THE STACK
004450  005726                  TST     (SP)+                   ;
004452  012737  002234  000004  11$:    MOV     #TIMOUT,@#4             ; RESTORE TIMEOUT VECTOR
004460  042737  000400  177730          BIC     #BIT8,DCSR              ;
                                
                                
```

TEST - DATA OUT WITHOUT RELOCATION

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

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
                                ;;*********************************************************************
004624  000004                  TST11:  SCOPE

004626  005737  001720                  TST     PMIS                    ; ANY PMI MEMORY?
004632  001433                          BEQ     TST12           ;; IF NONE, EXIT TEST
004634  052737  000400  177730          BIS     #BIT08,DCSR             ; MAKE SURE DIAGNOSTIC MODE IS ON
004642  042737  000006  177730          BIC     #BIT02!BIT01,DCSR       ; MAKE SURE NPR REGISTER IS SELECTED
004650  012737  052525  001124          MOV     #52525,$GDDAT           ; USE ALTERNATING 1'S AND 0'S
004656  052737  000001  177730          BIS     #BIT00,DCSR             ; SET GO BIT
004664  013737  001124  001160          MOV     $GDDAT,$TMP0            ; STORE PATTERN IN TEST LOCATON
                                ;
                                ; NOW COMPARE THE RESULTS
                                ;
004672  023737  177732  001124          CMP     DDR,$GDDAT              ; DDR GOT DATA OK?
004700  001004                          BNE     1$                      ; IF NO, ERROR
004702  042737  000400  177730          BIC     #BIT08,DCSR
004710  000404                          BR      TST12           ;;EXIT
004712  013737  177732  001126  1$:     MOV     DDR,$BDDAT              ; STORE DDR FOR ERROR REPORTS
004720  104015                          ERROR   +15                     ; ERROR IN DATA IN
```

TEST - CONTENT OF DDR

```
                                .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
                                ;;*************************************************************
004722  000004                  TST12:  SCOPE
1448 004724  005737  001720             TST     PMIS                            ; ANY PMI MEMORY?
1449 004730  001445             BEQ     TST13           ;; IF NONE, EXIT TEST
1450 004732  052737  000400  177730     BIS     #BIT08,DCSR                     ; MAKE SURE IN DIAG. MODE
1451 004740  052737  000002  177730     BIS     #BIT01,DCSR                     ; START WITH 1 IN DCSR
1452 004746  012703  000003             MOV     #3,R3                           ; DO 3 TIMES
1453 004752  005037  001124             CLR     $GDDAT                          ; RECIEVED DATA 0
1454 004756  005001                 1$: CLR     R1                              ; . ADDRESS 0
1455 004760  004737  002222             JSR     PC,DDIN                         ; . DO DIAGNOSTIC DATI
1456 004764  022703  000001             CMP     #1,R3                           ; . CONTROL LINES SELECTED?
1457 004770  001403             BEQ     2$                              ; . IF YES, CHECK IT
1458 004772  005737  177732             TST     DDR                             ; . ALL ZEROES?
1459 004776  001413             BEQ     3$                              ; . IF YES, BRANCH
1460 005000  012737  177400  001124 2$: MOV     #177400,$GDDAT                  ; . EXPECTED PATTERN
1461 005006  032737  000373  177732     BIT     #373,DDR                        ; . SELECT ONLY USED
1462 005014  001404             BEQ     3$                              ; . IF ALL ZEROES, BRANCH
1463 005016  013737  177732  001126     MOV     DDR,$BDDAT                      ; . STORE RECIEVED DATA
1464 005024  104016             ERROR   +16                             ; . ERROR IN UNIBUS LINES
1465 005026  062737  000002  177730 3$: ADD     #BIT01,DCSR                     ; . GET NEXT SET OF UNIBUS LINES
1466 005034  077330             SOB     R3,1$                           ; . DO FOR ALL COMBINATIONS
1467 005036  042737  000400  177730     BIC     #BIT8,DCSR                      ; .
1468
```

TEST - INDIRECT ACCESSING OF UNIBUS MAP REGISTERS

```
                        .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 #00 (THIS LOCATION IS ACTUALLY USED)
                        ;       .  LET PATTERN = R0 (ADDRESS OF LOW REGISTER)
                        ;       .  CALL DIAGNOSTIC_DATA_OUT <(R1),PATTERN>
                        ;       .  IF TIMEOUT THEN
                        ;       .  .  ERROR IN UNIQUE ADDRESSING OF REGISTERS
                        ;       .  ENDIF
                        ;       .  IF #00 NE PATTERN THEN
                        ;       .  .  ERROR PERFORMING DATA OUT
                        ;       .  ENDIF
                        ;       .  LET (R0)+ = #160000
                        ;       .  LET (R0)+ = #77 TO MAP JUST USED PAIR TO 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

```
1527                                    ;------------------------------------------------------------------
1528
1529                                    ;;*****************************************************************
                                        ;*TEST 13       INDIRECT ACCESSING OF UNIBUS MAP REGISTERS
                                        ;;*****************************************************************
     005044  000004                     TST13:  SCOPE
1530
1531 005046  005737  001720                     TST     PMIS                    ; ANY PMI MEMORY?
1532 005052  001576                             BEQ     TST14           ;; IF NONE, EXIT TEST
1533 005054  032737  000040  000052             BIT     #BIT05,@#52             ; UFD MODE ?
1534 005062  001406                             BEQ     1$                      ; BRANCH, IF NOT
1535 005064  032737  000040  177734             BIT     #BIT05,KMCR             ; 18 BIT MODE ?
1536 005072  001402                             BEQ     1$                      ; BRANCH, IF NOT
1537 005074  000137  005430                     JMP     10$                     ; EXIT TEST
1538 005100  052737  000400  177730  1$:        BIS     #BIT08,DCSR             ; SET DIAGNOSTIC MODE
1539 005106  042737  000006  177730             BIC     #BIT02!BIT01,DCSR       ; SELECT NPR THRU DDR
1540                                    ;
1541                                    ; POINT EACH OF MAPPING REGISTER PAIRS TO 17760000
1542                                    ;
1543 005114  012701  000040                     MOV     #32.,R1                 ; DO FOR 32. REGISTER PAIRS
1544 005120  012702  170200                     MOV     #MAPL00,R2              ; POINT TO ADDRESS OF THE FIRST REGISTER
1545 005124  012722  160000             2$:     MOV     #160000,(R2)+           ; . 160000 -> LO REGISTER
1546 005130  012722  000077                     MOV     #77,(R2)+               ; . 77 -> HI REGISTER
1547 005134  077105                             SOB     R1,2$                   ; . CONTINUE UNTILL ALL DONE
1548                                    ;
1549                                    ; INITIALISE TO DO RELOCATION IN 22 BIT MODE
1550                                    ;
1551 005136  012737  002234  000004             MOV     #TIMOUT,@#4             ; POINT TIMEOUT ROUTINE
1552 005144  004737  002244                     JSR     PC,MAPPR                ; REMAP PROGRAM AREA
1553 005150  005237  177572                     INC     MMR0                    ; TURN ON MMU
1554 005154  052737  000040  172516             BIS     #BIT05,MMR3             ; ENABLE RELOCATION
1555 005162  012700  170200                     MOV     #MAPL00,R0              ; POINT R0 TO FIRST REGISTER
1556 005166  012737  000000  172354             MOV     #0, KIPAR6              ; LET KIPAR6 SELECT MAP PAIR 0
1557 005174  012710  000000                     MOV     #0,(R0)                 ; MOVE 0 TO LOW ADDRESS
1558 005200  012760  000000  000002             MOV     #0,2(R0)                ; MOVE 0 TO HI ADDRESS
1559 005206  012701  140000                     MOV     #140000,R1              ; TO ACCESS THRU KIPAR6
1560                                    ;
1561                                    ; STORE # OF REGISTERS TO TEST IN R4
1562                                    ;
1563 005212  032737  000040  000052             BIT     #BIT05,@#52             ; UFD MODE ?
1564 005220  001010                             BNE     3$                      ; IF YES, BRANCH
1565 005222  013737  177734  001162             MOV     KMCR,$TMP1              ; SAVE KMCR
1566 005230  005037  177734                     CLR     KMCR                    ; DO NOT DISABLE ANY REGISTERS
1567 005234  012704  000036                     MOV     #30.,R4                 ; DO FOR ALL REGISTERS
1568 005240  000407                             BR      4$                      ; BRANCH AROUND
1569 005242  113703  177734             3$:     MOVB    KMCR,R3                 ; STORE # OF DISABLED REGISTERS
1570 005246  042703  177740                     BIC     #177740,R3              ; LEAVE JUST BITS <4-0>
1571 005252  012704  000036                     MOV     #30.,R4                 ; STORE MAX. # OF REGISTERS
1572 005256  160304                             SUB     R3,R4                   ; GET # OF ACCESSABLE REGISTERS
1573                                    ;
1574                                    ; DO DATA OUT FOR EACH REGISTER PAIR UNDER TEST
1575                                    ;
1576 005260  005003                     4$:     CLR     R3                      ; START WITH REGISTER PAIR 0
1577 005262  005037  000000             5$:     CLR     @#0                     ; . CLEAR LOCATION UNDER TEST
1578 005266  005037  001730                     CLR     TOUT                    ; . CLEAR TIMEOUT FLAG
1579 005272  010037  001124                     MOV     R0,$GDDAT               ; . PATTERN IS ADDRESS OF LO REGISTER
1580 005276  004737  002210                     JSR     PC,DDOUT                ; . DO DIAGN. DATA OUT
```

T13 INDIRECT ACCESSING OF UNIBUS MAP REGISTERS

```
1581 005302  005737  001730                    TST     TOUT            ; . TIMEOUT FLAG SET?
1582 005306  001411                            BEQ     6$              ; . IF NOT BRANCH
1583 005310  013737  172354  001122            MOV     KIPAR6,$BDADR   ; . STORE # OF PAIR FOR ERRORS
1584 005316  012705  000007                    MOV     #7,R5           ; . PREPARE TO SHIFT TO GET TO LOWER BITS
1585 005322  006237  001122          11$:      ASR     $BDADR          ; . . SHIFT ONCE
1586 005326  077503                            SOB     R5,11$          ; . . SHIFT 7 TIMES
1587 005330  104020                            ERROR   ^20             ; . IN UNIQUE ADDRESSING OF REGISTERS
1588 005332  023737  000000  001124  6$:       CMP     @#0,$GDDAT      ; . DATA OUT OK?
1589 005340  001404                            BEQ     7$              ; . IF YES, BRANCH
1590 005342  013737  000000  001126            MOV     @#0,$BDDAT      ; . STORE FOR ERROR REPORTS
1591 005350  104013                            ERROR   ^13             ; . PERFORMING DATA OUT
1592 005352  012720  160000          7$:       MOV     #160000,(R0)+   ; . POINT JUST USED REGISTER
1593 005356  012720  000077                    MOV     #77,(R0)+       ; . TO NXM
1594 005362  012710  000000                    MOV     #0,(R0)         ; . POINT NEXT REGISTER PAIR
1595 005366  012760  000000  000002            MOV     #0,2(R0)        ; . TO LOCATION 0
1596 005374  062737  000200  172354            ADD     #200,KIPAR6     ; . ACCESS NEXT REGISTER PAIR
1597 005402  062703  000001                    ADD     #1,R3           ; . INCREMENT COUNT
1598 005406  020304                            CMP     R3,R4           ; . ALL AVAILABLE PAIRS DONE?
1599 005410  003724                            BLE     5$              ; . BRANCH IF NOT
1600 005412  032737  000040  000052            BIT     #BIT05,@#52     ; CHECK TO SEE IF KMCR WAS SAVED
1601 005420  001003                            BNE     10$             ; NO, DON'T RESTORE KMCR
1602 005422  013737  001162  177734            MOV     $TMP1,KMCR      ; RESTORE KMCR
1603 005430  005037  177572          10$:      CLR     MMR0            ; DISABLE MMU
1604 005434  042737  000040  172516            BIC     #BIT05,MMR3     ; DISABLE MAPPING
1605 005442  042737  000400  177730            BIC     #BIT8,DCSR      ; EXIT DIAGNOSTIC MODE
```

TEST - DISABLING OF THE MAPPING REGISTERS

```
                                .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
                                ;;*****************************************************************
     005450  000004             TST14:  SCOPE

                                        TST     PMIS                    ; ANY PMI MEMORY?
     005452  005737  001720             BEQ     TST15           ;; IF NONE, EXIT TEST
     005456  001473                     BIT     #BIT05,@#52             ; UFD MODE ?
     005460  032737  00004C  000052     BEQ     1$                      ; BRANCH, IF NOT
     005466  001404                     BIT     #BIT05,KMCR             ; 18 BIT MODE ?
     005470  032737  000040  177734     BNE     TST15           ;;EXIT TEST IF 18 BIT MODE
     005476  001063
                                ;
                                ; INITIALISE MMU AND DIAGNOSTIC REGISTERS
                                ;
     005500  052737  000400  177730 1$:     BIS     #BIT08,DCSR             ; SET DIAGNOSTIC MODE
     005506  042737  000006  177730         BIC     #BIT02!BIT01,DCSR       ; SELECT NPR THRU DDR
     005514  004737  002244             JSR     PC,MAPPR                ; REMAP PROGRAM AREA
     005520  005237  177572             INC     MMR0                    ; TURN ON MMU
     005524  052737  000040  172516     BIS     #BIT05,MMR3             ; ENABLE RELOCATION
                                ;
                                ; TRY TO DO RELOCATION THRU REGISTER PAIR 31.
                                ;
     005532  012737  005614  000004     MOV     #2$,@#4                 ; INITIALISE TIMEOUT ROUTINE
     005540  012701  140000             MOV     #140000,R1              ; ACCESS THRU KIPAR6
```

T14 DISABLING REGISTERS

```
 1661 005544  012737  007600  172354          MOV     #7600,KIPAR6            ; ACCESS REG. PAIR 31.
 1662 005552  012737  125252  001124          MOV     #125252,$GDDAT          ; STORE PATTERN
 1663 005560  005037  000000                  CLR     @#0                     ; CLEAR LOCATION UNDER TEST
 1664 005564  012737  000000  170374          MOV     #0,MAPL37               ; LET MAP PAIR 31.
 1665 005572  012737  000000  170376          MOV     #0,MAPH37               ; POINT TO 0
 1666 005600  013737  001124  177732          MOV     $GDDAT,DDR              ; STORE PATTERN IN DDR
 1667 005606  005011                          CLR     (R1)                    ; EXTENAL READ TO PROVIDE ADDRESS
 1668 005610  104021                          ERROR   +21                     ; RELOCATION PERFORMED THRU REG. 31.
 1669 005612  000402                          BR      3$                      ; EXIT TEST
 1670 005614  005726                  2$:     TST     (SP)+                   ; RESTORE STACK
 1671 005616  005726                          TST     (SP)+
 1672 005620  005037  177572          3$:     CLR     MMR0                    ; DISABLE MMU
 1673 005624  042737  000040  172516          BIC     #BIT05,MMR3             ; DISABLE MAPPING
 1674 005632  012737  002234  000004          MOV     #TIMOUT,@#4             ; RESTORE TIMEOUT
 1675 005640  042737  000400  177730          BIC     #BIT8,DCSR              ; EXIT DIAGNOSTIC MODE
```

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 = #077
                                ;       .  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
                                ;;*************************************************************************
     005646   000004           TST15:  SCOPE

005650   005737   001720               TST     PMIS                    ; ANY PMI MEMORY?
005654   001476                       BEQ     TST16           ;; IF NONE, EXIT TEST
005656   052737   000400   177730      BIS     #BIT8,DCSR              ; SELECT DIAGNOSTIC MODE
005664   004737   002244               JSR     PC,MAPPR                ; REMAP PROGRAM AREA
005670   005237   177572               INC     MMR0                    ; ENABLE MMU
                                ;
                                ; DECIDE WHETHER TO  ACCESS WITH RELOCATION OR NOT
                                ;
005674   012701   140000               MOV     #140000,R1              ; ACCESS THRU KIPAR6
005700   032737   000040   000052      BIT     #BIT05,@#52             ; UFD MODE?
005706   001410                       BEQ     1$                      ; IF NOT, BRANCH
005710   032737   000040   177734      BIT     #BIT05,KMCR             ;  3 BIT MODE?
005716   001412                       BEQ     2$                      ; IF NOT BRANCH
005720   012737   177600   172354      MOV     #177600,KIPAR6          ; KIPAR6 HAS NXM
005726   000422                       BR      3$                      ; GO DO IT
005730   013737   177734   001162  1$: MOV     KMCR,$TMP1              ; SAVE KMCR
```

T15 NXM MEMORY TIMEOUT

```
 1731 005736  042737  000040  177734          BIC     #BIT05,KMCR                     ; IN NOT UFD, DO IN 22 BITS
 1732 005744  012737  000000  172354  2$:     MOV     #0,KIPAR6                       ; KIPAR6 ACCESS MAP REG. 0
 1733 005752  012737  160000  170200          MOV     #160000,MAPL00                  ; MAP PAIR HAS NXM
 1734 005760  012737  000077  170202          MOV     #77,MAPH00
 1735 005766  052737  000040  172516          BIS     #BIT05,MMR3                     ; ENABLE RELOCATION
 1736                                  ;
 1737                                  ; DO ACTUAL DATA OUT CYCLE
 1738                                  ;
 1739 005774  005037  177732          3$:     CLR     DDR                             ; PROMT DIAG. DATO
 1740 006000  005011                          CLR     (R1)                            ; WRITE TO NXM
 1741 006002  032737  100000  177730          BIT     #BIT15,DCSR                     ; NXM SET?
 1742 006010  001001                          BNE     5$                              ; IF YES, BRANCH
 1743 006012  104022                          ERROR   +22                             ; NXM ACCESS DOES NOT SET DCSR<15>
 1744 006014  005037  177572          5$:     CLR     MMR0                            ; DISABLE MMU
 1745 006020  042737  000040  172516          BIC     #BIT05,MMR3                     ; AND MAPPING TOO
 1746 006026  032737  000040  000052          BIT     #BIT05,@#52                     ; UFD MODE
 1747 006034  001006                          BNE     TST16           ;;IF NOT UFD, GO TO NEXT TEST
 1748 006036  013737  001162  177734          MOV     $TMP1,KMCR                      ; OTHERWISE, RESTORE KMCR
 1749 006044  042737  000400  177730          BIC     #BIT8,DCSR                      ; EXIT DIAGNOSTIC MODE
 1750
```

TEST - CARRY PROPOGATION TEST

```
                                .SBTTL TEST - CARRY PROPOGATION TEST
                                ;
                                ;* THIS TEST CHECKS THE CARRY PROPOGATION THRU THE ALU. MAPPING REGISTER PAIR 0
                                ;* IS LOADED WITH ALL 1'S.  LOCATION 1 IS ACCESSED. THE RESULT OF THE DATO
                                ;* CYCLE IS SUPPOSED TO BE LOADED IN LOCATION 0.
                                ;
                                ;  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>
                                ;       POINT TIMEOUT_ROUTINE TO VECTOR 4
                                ;       CALL MAP_PROGRAM_AREA
                                ;       LET MMR0<0> = 1 TO ENABLE MMU
                                ;       LET MMR3<5> = 1 TO ENABLE RELOCATION
                                ;       LET KIPAR6 = #0 TO ACCESS THRU MAP REG.0
                                ;       LET R1 = #140001 TO ACCESS THRU KIPA6
                                ;       LET LO MAP REG. = #177777
                                ;       LET HI MAP REG. = #77
                                ;       LET PATTERN = #125252
                                ;       CALL DIAGNOSTIC_DATA_OUT<(R1),PATTERN>
                                ;       LET MMR0<0> = #0 TO DISABLE MMU
                                ;       IF @0 NE #125252 THEN
                                ;       .  ERROR IN CARRY PROPOGATION
                                ;       ENDIF
                                ;
                                ;  ENDTST
                                ;
                                ;-------------------------------------------------------------------------
                                ;;*************************************************************************
                                ;*TEST 16        CARRY PROPOGATION TEST
                                ;;*************************************************************************
    006052  000004              TST16:  SCOPE

                                        TST     PMIS                    ; ANY PMI MEMORY?
    006054  005737  001720              BEQ     TST17           ;; IF NONE, EXIT TEST
    006060  001470                      BIT     #BIT05,@#52             ; UFD MODE ?
    006062  032737  000040  000052      BEQ     1$                      ; BRANCH, IF NOT
    006070  001404                      BIT     #BIT05,KMCR             ; 18 BIT MODE ?
    006072  032737  000040  177734      BNE     TST17           ;; GO TO NEXT TEST, IF YES
    006100  001060
                                ;
                                ; INITIALISE MMU AND DIAGNOSTIC REGISTERS
                                ;
    006102  052737  000400  177730  1$:     BIS     #BIT08,DCSR             ; SET DIAGNOSTIC MODE
    006110  042737  000006  177730          BIC     #BIT02!BIT01,DCSR       ; SELECT NPR THRU DDR
    006116  004737  002244              JSR     PC,MAPPR                ; REMAP PROGRAM AREA
    006122  005237  177572              INC     MMR0                    ; TURN ON MMU
    006126  052737  000040  172516      BIS     #BIT05,MMR3             ; ENABLE RELOCATION
                                ;
                                ; ACCESS LOCATION 0 THRU RELOCATION REGISTERS
                                ;
    006134  005037  172354              CLR     KIPAR6                  ; CLEAR KIPAR6
    006140  012701  140002              MOV     #140002,R1              ; ACCESS THRU KIPAR6
    006144  012737  177777  170200      MOV     #177777,MAPLOO          ; ALL ONE'S TO LO REG
```

T16 CARRY PROPOGATION TEST

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

```
                                .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
                                ;;*************************************************************************
    006242  000004              TST17:  SCOPE

006244  005737  001720                  TST     PMIS                            ; ANY PMI MEMORY?
006250  001521                          BEQ     TST20           ;; IF NONE, EXIT TEST
006252  032737  000040  000052          BIT     #BIT05,@#52                     ; UFD MODE
006260  001404                          BEQ     1$                              ; IF NOT, BRANCH
006262  032737  000040  177734          BIT     #BIT05,KMCR                     ; 18 BIT MODE
006270  001111                          BNE     TST20           ;; IF 18 BIT MODE, DON'T DO THIS TEST
                                ;
                                ; INITIALISE MMU AND RELOCATION
                                ;
006272  052737  000400  177730  1$:     BIS     #BIT08,DCSR                     ; SET DIAGNOSTIC MODE
006300  042737  000006  177730          BIC     #BIT02!BIT01,DCSR               ; SELECT NPR THRU DDR
006306  004737  002244                  JSR     PC,MAPPR                        ; REMAP PROGRAM AREA
006312  052737  000060  172516          BIS     #BIT05!BIT04,MMR3               ; ENABLE RELOCATION AND 22BITS
006320  005237  177572                  INC     MMR0                            ; ENABLE MMU
                                ;
                                ; POINT ALL MAPPING REGISTERS TO ALL 1'S
                                ;
006324  012701  170200                  MOV     #MAPL00,R1                      ; START WITH 0
006330  012703  000036                  MOV     #36,R3                          ; DO FOR ALL REGISTERS
006334  012721  177776          2$:     MOV     #177776,(R1)+                   ; LOW MAP REGISTER
006340  012721  000077                  MOV     #77,(R1)+                       ; HIGH MAP REGISTER
006344  077305                          SOB     R3,2$                           ; DO FOR ALL
```

```
                                        ;
                                        ; START WITH DATI ON A WORD BOUNDARY
                                        ;
006346  005037  172354                  CLR     KIPAR6                  ; ACCESS REGISTER PAIR 0
006352  012701  140002                  MOV     #140002,R1              ; START WITH 0 THRU KIPAR6
006356  000401                          BR      4$                      ;
006360  005721                  3$:     TST     (R1)+                   ; . DO IN WORD INCREMENTS
006362  004737  002222          4$:     JSR     PC,DDIN                 ; . DO DATI
006366  010137  001122                  MOV     R1,$BDADR               ; . STORE JUST ACCESSED ADDRESS
006372  162737  000002  001122          SUB     #2,$BDADR               ; . GET RID OF OVERFLOW
006400  027737  172516  177732          CMP     @$BDADR,DDR             ; . PROPER DATA?
006406  001412                          BEQ     7$                      ; . IF EQUAL, BRANCH
006410  017737  172506  001124          MOV     @$BDADR,$GDDAT          ; . STORE RECEIVED DATA
006416  013737  177732  001126          MOV     DDR,$BDDAT              ; . STORE EXPECTED DATA
006424  042737  160000  001122          BIC     #BIT15!BIT14!BIT13,$BDADR ; . STRIP OF PAR BITS
006432  104023                          ERROR   +23                     ; . ERROR IN RELOCATION
                                        ;
                                        ; DECIDE WHICH CYCLE SHOULD OCCUR NEXT
                                        ;
006434  005737  172354          7$:     TST     KIPAR6                  ; . FIRST TIME 1K BOUNDARY?
006440  001011                          BNE     10$                     ; . IF NOT, BRANCH
006442  022701  144000                  CMP     #144000,R1              ; . LESS THAN 1K DONE?
006446  103744                          BLO     3$                      ; . DO IN WORD INCREMENTS
006450  012701  140002                  MOV     #140002,R1              ; . NOW DO INCREMENTING THRU PAR
006454  012737  000040  172354          MOV     #40,KIPAR6              ; . FIRST 1K
006462  000737                          BR      4$                      ; . GO DO COMPARE
006464  062737  000040  172354  10$:     ADD     #40,KIPAR6              ; . ACCESS NEXT 1K
006472  022737  000200  172354          CMP     #200,KIPAR6             ; . OVER 4K BOUNDARY?
006500  003330                          BGT     4$                      ; . IF NOT, GO DO DATI
006502  005037  177572                  CLR     MMR0                    ; DISABLE MMU
006506  042737  000400  177730          BIC     #BIT8,DCSR              ; EXIT DIAGNOSTIC MODE
```

```
                                .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
                                ;;*************************************************************
     006514  000004             TST20:  SCOPE
                                ;
                                ; INITIALISE MMU AND RELOCATION
                                ;
 006534  052737  000400  177730 1$:     BIS     #BIT08,DCSR                     ; SET DIAGNOSTIC MODE
 006542  042737  000006  177730         BIC     #BIT02!BIT01,DCSR               ; SELECT NPR THRU DDR
 006550  004737  002244             JSR     PC,MAPPR                        ; REMAP PROGRAM AREA
 006554  052737  000060  172516         BIS     #BIT05!BIT04,MMR3               ; ENABLE RELOCATION AND 22BITS
 006562  005237  177572             INC     MMR0                            ; ENABLE MMU
 006566  013702  001720             MOV     PMIS,R2                         ; STORE SIZE OF PMI MEMORY
 006572  010237  001160             MOV     R2,$TMP0                        ; CREATE A PATTERN
 006576  012703  004200             MOV     #4200,R3                        ; CONSTANT FOR PAR
 006602  040203                     BIC     R2,R3                           ; MASK OUT NXM
 006604  072227  177766             ASH     #-10.,R2                        ; FOR MAPH00
                                ;
                                ; DO FOR ALL COMBINATIONS POSSIBLE, 4 BITS AT A TIME
                                ;
 006610  012701  140000             MOV     #140000,R1                      ; ACCESS THRU KIPAR6
 006614  005037  172354             CLR     KIPAR6                          ; ACCESS REGISTER PAIR 0
```

```
 1961 006620  005037  170200                CLR     MAPLOO                          ; CLEAR MAP REGISTERS
 1962 006624  005037  170202                CLR     MAPHOO                          ;
 1963 006630  005037  001162                CLR     $TMP1                           ; CLEAR STORAGE FOR PAR
 1964 006634  005037  001122                CLR     $BDADR                          ; FIRST ADDRESS USED
 1965 006640  005037  001124                CLR     $GDDAT                          ; CLEAR PATTERN
 1966 006644  004737  002222          2$:   JSR     PC,DDIN                         ; . DO DATI
 1967 006650  013737  001162  172354        MOV     $TMP1,KIPAR6                    ; . GET ADDED PAR
 1968 006656  023777  177732  172236        CMP     DDR,@$BDADR                     ; . DATI OK?
 1969 006664  001435                        BEQ     10$                             ; . IF SO, BRANCH
 1970 006666  013737  177732  001126        MOV     DDR,$BDDAT                      ; . STORE RECIEVED DATA
 1971 006674  017737  172222  001124        MOV     @$BDADR,$GDDAT                  ; . STORE EXPECTED DATA
 1972 006702  042737  160000  001122        BIC     #BIT15!BIT14!BIT13,$BDADR       ; . STRIP OF PAR BITS
 1973 006710  104023                        ERROR   +23                             ;
 1974 006712  005037  001124                CLR     $GDDAT                          ;
 1975 006716  000420                        BR      10$                             ;
 1976 006720  004737  002210          5$:   JSR     PC,DDOUT                        ; . DO DATO
 1977 006724  013737  001162  172354        MOV     $TMP1,KIPAR6                    ; . GET ADDED PAR
 1978 006732  023777  001124  172162        CMP     $GDDAT,@$BDADR                  ; . DATI OK?
 1979 006740  001407                        BEQ     10$                             ; . IF SO, BRANCH
 1980 006742  017737  172154  001126        MOV     @$BDADR,$BDDAT                  ; . STORE EXPECTED DATA
 1981 006750  042737  160000  001122        BIC     #BIT15!BIT14!BIT13,$BDADR       ; . STRIP OF PAR BITS
 1982 006756  104023                        ERROR   +23                             ;
 1983                                  ;
 1984                                  ; DECIDE WHAT ACCESS NEXT
 1985                                  ;
 1986 006760  005037  172354          10$:  CLR     KIPAR6                          ; . CLEAR PAR
 1987 006764  062701  001042                ADD     #1042,R1                        ; . START INCREMENTING
 1988 006770  062737  021042  170200        ADD     #21042,MAPLOO                   ; . EVERY 4TH BIT
 1989 006776  005537  170202                ADC     MAPHOO                          ; . DON'T FORGET CARRY
 1990 007002  062737  000002  170202        ADD     #2,MAPHOO                       ; . FOR ALL REGISTERS
 1991 007010  010137  001122                MOV     R1,$BDADR                       ; . SAVE ADDRESS USED
 1992 007014  063737  170200  001122        ADD     MAPLOO,$BDADR                   ; . ADD UNIBUS MAP REG.
 1993 007022  052737  140000  001122        BIS     #140000,$BDADR                  ; . MAKE SURE KIPAR6 SELECTED
 1994 007030  042737  020000  001122        BIC     #20000,$BDADR                   ;
 1995 007036  022701  150420                CMP     #150420,R1                      ; . OVERFLOW FROM ADDITION?
 1996 007042  001003                        BNE     15$                             ; . IF NO, BRANCH
 1997 007044  062737  000200  001162        ADD     #200,$TMP1                      ; . ADD OVERFLOW FOR PAR
 1998 007052  060337  001162          15$:  ADD     R3,$TMP1                        ; . ADD INCR. TO PAR
 1999 007056  022737  002000  001162        CMP     #2000,$TMP1                     ; . LESS THAN 32K?
 2000 007064  003267                        BGT     2$                              ; . IF YES, DO DATI
 2001 007066  005237  001124                INC     $GDDAT                          ; . INCREMENT PATTERN
 2002 007072  023737  001160  001162        CMP     $TMP0,$TMP1                     ; . OVER AVAILABLE MEMORY?
 2003 007100  003307                        BGT     5$                              ; . IF NOT, DO ANOTHER DATO
 2004 007102  005037  177572          25$:  CLR     MMR0                            ; DISABLE MMU
 2005 007106  042737  000400  177730        BIC     #BIT8,DCSR              ; EXIT DIAGNOSTIC MODE
```

```
                                .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
                                ;;*****************************************************************
    007114  000004              TST21:  SCOPE
2051 007116  005737  001720             TST     PMIS                    ; ANY PMI MEMORY?
2052 007122  001002             BNE     20$                             ; IF YES, DO THE TEST
2053 007124  000137  007574     JMP     100$                            ; OTHERWISE EXIT TEST
2054 007130  052737  000400  177730  20$:    BIS     #BIT08,DCSR         ; STANDALONE MODE
2055 007136  004737  002244             JSR     PC,MAPPR                ; REMAP PROGRAM AREA
2056 007142  005237  177572             INC     MMR0                    ; ENABLE MMU
2057 007146  113701  177734             MOVB    KMCR,R1                 ; SAVE KMCR
2058 007152  042701  177740             BIC     #177740,R1              ; R1 HAS # OF PAGES DISABLED
2059                                    ;
2060                                    ; GET THE HIGHEST NXM LOCATION
```

T21 MAIN MEMORY DISABLE

```
                                        ;
007156  032737  000040  177734          BIT     #BIT05,KMCR             ; 18 BIT MODE?
007164  001404                          BEQ     1$                      ; IF NOT, BRANCH
007166  012737  007400  172354          MOV     #7400,KIPAR6            ; NXM FOR 18 BITS
007174  000403                          BR      2$                      ; GO DO IT
007176  012737  177400  172354  1$:     MOV     #177400,KIPAR6          ; NXM FOR 22 BITS
                                        ;
                                        ; GO THRU ALL PAGES OF MEMORY DISABLED
                                        ;
007204  105701                  2$:     TSTB    R1                      ; ANY UNIBUS MEMORY?
007206  001420                          BEQ     6$                      ; IF NO, BRANCH
007210  012737  007226  000004          MOV     #4$,#4                  ; POINT TIMEOUT VECTOR TO PROGRAM
007216  005737  140000          3$:     TST     #140000                 ; . ACCESS THRU KIPAR6
007222  104025                          ERROR   .25                     ; . KMCR<4-0> DOES NOT DISABLE MAIN MEMORY
007224  000402                          BR      5$                      ; .
007226  005726                  4$:     TST     (SP)+                   ; . ADJUST STACK
007230  005726                          TST     (SP)+                   ; .
007232  162737  000200  172354  5$:     SUB     #200,KIPAR6             ; . ACCESS NEXT LOWER PAGE
007240  077112                          SOB     R1,3$                   ; . DO FOR ALL PAGES IN KMCR
007242  012737  002234  000004          MOV     #TIMOUT,#4              ; RESTORE TIMEOUT VECTOR
                                        ;
                                        ; IN NON-UFD MODE, CHECK KMCR<5-0>
                                        ;
007250  032737  000040  000052  6$:     BIT     #BIT05,#52              ; UFD MODE?
007256  001146                          BNE     TST22           ;; IF YES, GO TO NEXT TEST
007260  013737  177734  001162          MOV     KMCR,$TMP1              ; SAVE KMCR
007266  012701  000067                  MOV     #67,R1                  ; HIGHEST VALUE (32K OF MEMORY)
007272  110137  177734          7$:     MOVB    R1,KMCR                 ; . WRITE TO KMCR
007276  120137  177734                  CMPB    R1,KMCR                 ; . WRITTEN OK?
007302  001426                          BEQ     8$                      ; . IF SO, BRANCH
007304  010137  001124                  MOV     R1,$GDDAT               ; . NO,THEN GET THE GOOD DATA
007310  052737  000200  001124          BIS     #BIT07,$GDDAT           ; . IN CASE REBOOT BIT SET
007316  123737  001124  177734          CMPB    $GDDAT,KMCR             ; . WAS THAT THE ONLY WRONG?
007324  001415                          BEQ     8$                      ; . IF YES, BRANCH
007326  010137  001124                  MOV     R1,$GDDAT               ; . IF NOT, RESTORE PATTERN WRITTEN
007332  013737  177734  001126          MOV     KMCR,$BDDAT             ; . GET THE BAD DATA
007340  012737  177734  001122          MOV     #KMCR,$BDADR            ; . GET THE ADDRESS OF THE KMCR
007346  013737  001160  177734          MOV     $TMP0,KMCR              ; . RESTORE REGISTER
007354  104011                          ERROR   .11                     ; . ERROR IN KMCR<5-0>
007356  000446                          BR      12$                     ; . EXIT TEST ON ERROR
007360  005301                  8$:     DEC     R1                      ; . LAST LOCATION (0)?
007362  002343                          BGE     7$                      ; . IF NOT, BRANCH
                                        ;
                                        ; IN NON-UFD MODE, GO THRU THE PAGE JUST ABOVE 32K
                                        ;
007364  052737  000020  172516          BIS     #BIT04,MMR3             ; ENABLE 22-BIT
007372  013702  001720                  MOV     PMIS,R2                 ; STORE HIGHEST PAGE
007376  005037  170200                  CLR     MAPL00                  ; CLEAR MAP REG. 0
007402  012737  000001  170202          MOV     #1,MAPH00               ; POINTS TO 32K
007410  052737  000040  172516          BIS     #BIT05,MMR3             ; ENABLE MAPPING
007416  005001                          CLR     R1                      ; ADDRESS FOR DMA CYCLES
007420  012737  007452  000004          MOV     #10$,#4                 ; TIMEOUT TO PROGRAM
007426  012737  002000  172354          MOV     #2000,KIPAR6            ; PAGE JUST ABOVE 32K
007434  012737  000067  177734          MOV     #67,KMCR                ; DISABLE HIGHER THEN 32K
                                        ;
                                        ; DO A CPU CYCLE TO DISABLED MEMORY
```

T21 MAIN MEMORY DISABLE

```
2118
2119 007442  005737  140000          9$:   TST     @#140000            ; . ACCESS DISABLED MEMORY
2120 007446  104025                        ERROR   .25                 ; . IN DISABLING MEMORY THRU KMCR<4-0>
2121 007450  000411                        BR      12$                 ;
2122 007452  005726                  10$:  TST     (SP)+               ; . RESTORE STACK
2123 007454  005726                        TST     (SP)+
2124                                  ;
2125                                  ; DO A DMA DATI CYCLE TO DISABLED MEMORY
2126                                  ;
2127 007456  004737  002222          11$:  JSR     PC,@DDIN            ; . DIAGNOSTIC DATI
2128 007462  032737  100000  177730        BIT     @#BIT15,@DCSR       ; . NXM SET?
2129 007470  001001                        BNE     12$                 ; . IF SET, BRANCH
2130 007472  104025                        ERROR   .25                 ; . IN DMA MEMORY NOT DISABLED
2131                                  ;
2132                                  ; CHECK WHETHER ALL PAGES IN MEMORY VERIFIED
2133                                  ;
2134 007474  023702  172354          12$:  CMP     @KIPAR6,R2          ; . IS IT IN MEMORY?
2135 007500  103020                        BHIS    14$                 ; . IF NOT, BRANCH
2136 007502  062737  020000  170200        ADD     @#20000,@MAPLOO     ; . ACCESS NEXT 4K
2137 007510  103002                        BCC     13$                 ; . IF DIDN'T EXCEED 32K, BRANCH
2138 007512  005237  170202                INC     @MAPHOO             ; . OTHERWISE, INCREMENT MAP REGISTER
2139 007516  005337  177734          13$:  DEC     @KMCR               ; . ENABLE NEXT 4K OF MEMORY
2140 007522  032737  000040  177734        BIT     @#BIT05,@KMCR       ; . ALL 128K DONE?
2141 007530  001404                        BEQ     14$                 ; . IF SO, BRANCH
2142 007532  062737  000200  172354        ADD     @#200,@KIPAR6       ; . ACCESS NEXT 4K
2143 007540  000740                        BR      9$                  ; . TRY TO DO IT
2144 007542  012737  002234  000004  14$:  MOV     @#TIMOUT,@#4        ; RESTORE TIMEOUT VECTOR
2145 007550  013737  001162  177734        MOV     $TMP1,@KMCR         ; RESTORE KMCR
2146 007556  005037  172516                CLR     @MMR3               ; DISABLE 22-BIT MODE
2147 007562  005037  177572                CLR     @MMR0               ; DISABLE MMU
2148 007566  042737  000400  177730        BIC     @#BIT8,@DCSR        ; EXIT DIAGNOSTIC MODE
2149 007574                          100$:
```

TEST - CACHE PRESENCE

```
                                .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
                                ;;*****************************************************************
007574  000004                  TST22:  SCOPE
                                ;
007576  005737  001720                  TST     PMIS                            ; ANY PMI MEMORY?
007602  001440                  BEQ     TST23           ;; IF NONE, EXIT TEST
007604  052737  000100  177734          BIS     #BIT06,KMCR                     ; SET CACHE ENABLE BIT
007612  032737  000100  177734          BIT     #BIT06,KMCR                     ; DID IT GET SET?
007620  001031                  BNE     TST23           ;;      IF CACHE PRESENT, EXIT TEST
007622  005737  001206                  TST     $PASS                           ; FIRST PASS?
007626  001014                  BNE     1$                                      ; IF YES, SKIP PRINTOUT
007630  122737  000001  001220          CMPB    @APTENV,$ENV                    ; IN APT MODE?
007636  001410                  BEQ     1$                                      ; IF SO, SKIP PRINTOUT
007640  032737  000040  000052          BIT     #BIT05,@052                     ; UFD MODE ?
007646  001004                  BNE     1$                                      ; IF YES, BRANCH
007650  104401  007664                  TYPE    .NOCAH                          ; TYPE NO CACHE MESSAGE
007654  104401  001175                  TYPE    .$CRLF                          ;
007660  000137  013210          1$:     JMP     @BETST                          ; IF NO CACHE, SKIP ALL CACHE TSTS
                                ;
007664      040     116     117 NOCAH:  .ASCIZ  / NO CACHE SEEN/
007667      040     103     101
007672      103     110     105
007675      040     123     105
007700      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
                                ;;*****************************************************************
    007704  000004              TST23:  SCOPE
2212
2213 007706  005737  001720             TST     PMIS                            ; ANY PMI MEMORY?
2214 007712  001471             BEQ     TST24           ;; IF NONE, EXIT TEST
2215 007714  052737  000400  177730     BIS     #BIT8,DCSR                      ; SELECT DIAGNOSTIC MODE
2216 007722  042737  000040  172516     BIC     #BIT05,MMR3                     ; FIRST DO WITH RELOCATION DISABLED
2217 007730  000403             BR      2$                              ; GO DO IT
2218 007732  052737  000040  172516  1$:    BIS     #BIT05,MMR3                     ; NOW DO WITH RELOCATION ENABLED
2219 007740  042737  000100  177734  2$:    BIC     #BIT06,KMCR                     ; MAKE SURE THAT CACHE IS DISABLED
2220 007746  042737  000400  177734     BIC     #BIT08,KMCR                     ; READ VALID BITS FIRST
2221 007754  032737  017000  177734     BIT     #BIT12!BIT11!BIT10!BIT9,KMCR    ; ALL VALID BITS CLEAR?
2222 007762  001412             BEQ     3$                              ; IF YES, BRANCH
2223 007764  013737  177734  001126     MOV     KMCR,$BDDAT                     ; STORE KMCR
2224 007772  013737  177734  001124     MOV     KMCR,$GDDAT                     ; STORE TO PRESERVE LOW BYTE
2225 010000  042737  017000  001124     BIC     #BIT12!BIT11!BIT10!BIT9,$GDDAT  ; HIGH BYTE CLEAR
2226 010006  104026             ERROR   .26                             ; VALID BITS NOT CLEAR WITH KMCR<6>=0
2227 010010  052737  000400  177734  3$:    BIS     #BIT08,KMCR                     ; NOW READ AVAILABILITY BITS
2228 010016  013737  177734  001160     MOV     KMCR,$TMP0                      ; STORE KMCR
2229 010024  122737  000177  001161     CMPB    #177,$TMP0+1                    ; ALL SET?
2230 010032  001412             BEQ     4$                              ; IF YES, BRANCH
2231 010034  013737  177734  001126     MOV     KMCR,$BDDAT                     ; STORE KMCR
2232 010042  013737  177734  001124     MOV     KMCR,$GDDAT                     ; STORE TO PRESERVE LOW BYTE
2233 010050  152737  000177  001124     BISB    #177,$GDDAT                     ; HIGH BYTE SET
2234 010056  104026             ERROR   .26                             ; AVAIL. BITS NOT SET WITH KMCR<6>=0
2235 010060  032737  000040  172516  4$:    BIT     #BIT05,MMR3                     ; DONE WITH RELOCATION ENABLED?
2236 010066  001721             BEQ     1$                              ; IF NOT, BRANCH
2237 010070  042737  000400  177730     BIC     #BIT8,DCSR                      ; EXIT DIAGNOSTIC MODE
2238
2239
```

```
                        .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
                        ;
                        ;-------------------------------------------------------------
```

```
2298                                    ;;*************************************************************************
                                        ;*TEST 24       AVAILABILITY OF SETS
                                        ;;*************************************************************************
     010076  000004                     TST24:  SCOPE
2299
2300 010100  005737  001720                     TST     PMIS                    ; ANY PMI MEMORY?
2301 010104  001002                             BNE     20$                     ; IF SOME, DO THE TEST
2302 010106  000137  010524                     JMP     100$                    ; OTHERWISE EXIT
2303 010112  052737  000400  177730  20$:      BIS     #BIT8,DCSR              ; SELECT DIAGNOSTIC MODE
2304 010120  052737  000100  177734             BIS     #BIT06,KMCR             ; ENABLE CACHE
2305 010126  012702  010526                     MOV     #VALTBL,R2              ; POINT TO VALID BITS
2306 010132  012703  010542                     MOV     #TOPTBL,R3              ; POINT TO AVAILABILITY BITS
2307 010136  052737  000040  172516             BIS     #BIT05,MMR3             ; ENABLE RELOCATION
2308 010144  012737  000200  170200             MOV     #200,MAPL00             ; POINT MAP00 TO FIRST 8KB
2309 010152  012737  000000  170202             MOV     #0,MAPH00               ; IN HIGH AND LOW MAP REGISTERS
2310 010160  013737  177734  001124             MOV     KMCR,$GDDAT             ; PRESERVE LOW BYTE
2311 010166  105037  001125                     CLRB    $GDDAT+1                ; CLEAR TO REPORT ERRORS, IF ANY
2312                                    ;
2313                                    ; ALLOCATE 4 SETS BY READING THEM THRU DIAGNOSTIC_DATA_IN
2314                                    ;
2315 010172  012701  000000                     MOV     #0,R1                   ; START READING MEMORY FROM 0
2316 010176  004737  002222             1$:     JSR     PC,DDIN                 ; . ALLOCATE IN CACHE NEXT 8 WORDS
2317 010202  042737  000400  177734             BIC     #BIT08,KMCR             ; . READ VALID BITS
2318 010210  013737  177734  001126             MOV     KMCR,$BDDAT             ; . STORE KMCR
2319 010216  122237  001127                     CMPB    (R2)+,$BDDAT+1          ; . VALID BITS OK?
2320 010222  001404                             BEQ     2$                      ; . IF YES, BRANCH
2321 010224  105742                             TSTB    -(R2)                   ; . GET THE PATTERN JUST WRITTEN
2322 010226  112237  001125                     MOVB    (R2)+,$GDDAT+1          ; . STORE HIGH BYTE
2323 010232  104026                             ERROR   +26                     ; . ERROR IN VALID BITS
2324 010234  052737  000400  177734  2$:       BIS     #BIT08,KMCR             ; . READ AVAILABILITY BIT
2325 010242  013737  177734  001126             MOV     KMCR,$BDDAT             ; . STORE KMCR
2326 010250  122337  001127                     CMPB    (R3)+,$BDDAT+1          ; . AVAILABILITY BITS OK?
2327 010254  001404                             BEQ     3$                      ; . IF YES, BRANCH
2328 010256  105743                             TSTB    -(R3)                   ; . GET THE PATTERN JUST WRITTEN
2329 010260  112337  001125                     MOVB    (R3)+,$GDDAT+1          ; . STORE HIGH BYTE
2330 010264  104026                             ERROR   +26                     ; . ERROR IN AVAILABILITY BITS
2331 010266  062701  000020             3$:     ADD     #20,R1                  ; . DO FOR NEXT OCTAL BOUNDARY
2332 010272  022701  000100                     CMP     #100,R1                 ; . ALL 4 DONE?
2333 010276  003337                             BGT     1$                      ; . IF NOT, BRANCH
2334                                    ;
2335                                    ; NOW READ 3RD AND THEN 8TH LOCATION FROM CACHE
2336                                    ;
2337 010300  012701  000004                     MOV     #4,R1                   ; START WITH 3RD LOCATION A SET
2338 010304  042737  000400  177734  4$:       BIC     #BIT08,KMCR             ; . READ VALID BITS
2339 010312  004737  002222                     JSR     PC,DDIN                 ; . READ FROM CACHE
2340 010316  013737  177734  001126             MOV     KMCR,$BDDAT             ; . STORE KMCR
2341 010324  122237  001127                     CMPB    (R2)+,$BDDAT+1          ; . VALID BITS OK?
2342 010330  001404                             BEQ     5$                      ; . IF YES, BRANCH
2343 010332  105742                             TSTB    -(R2)                   ; . GET THE PATTERN JUST WRITTEN
2344 010334  112237  001125                     MOVB    (R2)+,$GDDAT+1          ; . STORE HIGH BYTE
2345 010340  104026                             ERROR   +26                     ; . ERROR IN VALID BITS
2346 010342  052737  000400  177734  5$:       BIS     #BIT08,KMCR             ; . READ AVAILABILITY BIT
2347 010350  013737  177734  001126             MOV     KMCR,$BDDAT             ; . STORE KMCR
2348 010356  122337  001127                     CMPB    (R3)+,$BDDAT+1          ; . AVAILABILITY BITS OK?
2349 010362  001404                             BEQ     6$                      ; . IF YES, BRANCH
2350 010364  105743                             TSTB    -(R3)                   ; . GET THE PATTERN JUST WRITTEN
2351 010366  112337  001125                     MOVB    (R3)+,$GDDAT+1          ; . STORE HIGH BYTE
```

T24 AVAILABILITY OF SETS

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

```
                        .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
                                ;;*************************************************************
     010556  000004             TST25:  SCOPE

 010560  005737  001720                 TST     PMIS                    ; ANY PMI MEMORY?
 010564  001002                         BNE     20$                     ; IF SOME, DO THE TEST
 010566  000137  011266                 JMP     100$                    ; OTHERWISE, EXIT
 010572  052737  000400  177730 20$:    BIS     #BIT8,DCSR              ; SELECT DIAGNOSTIC MODE
 010600  052737  000040  172516         BIS     #BIT05,MMR3             ; ENABLE RELOCATION
 010606  052737  000100  177734         BIS     #BIT06,KMCR             ; ENABLE CACHE
 010614  012737  000200  170200         MOV     #200,MAPL00             ; POINT MAP00 TO FIRST 8KB
 010622  012737  000000  170202         MOV     #0,MAPH00               ; IN HIGH AND LOW MAP REGISTERS
 010630  013737  177734  001124         MOV     KMCR,$GDDAT             ; STORE LOW BYTE OF KMCR
                                ;
                                ; ALLOCATE ALL SETS MAKING A MOST AVAILABLE
                                ;
 010636  012701  000000                 MOV     #0,R1                   ; START WITH 0 IN SET A
 010642  004737  002222         1$:     JSR     PC,DDIN                 ; . ALLOCATE A SET
 010646  062701  000020                 ADD     #20,R1                  ; . GET READY FOR NEXT SET
 010652  022701  000100                 CMP     #100,R1                 ; . ALL 4 DONE?
 010656  003371                         BGT     1$                      ; . IF NOT, BRANCH
                                ;
                                ; TRY TO BRING A NEW SET TO A
                                ;
 010660  012701  000100                 MOV     #100,R1                 ; TRY TO DO DATI
 010664  042737  000400  177734         BIC     #BIT08,KMCR             ; READ VALID BITS
 010672  004737  002222                 JSR     PC,DDIN                 ; ALLOCATE IN CACHE
 010676  013737  177734  001126         MOV     KMCR,$BDDAT             ; STORE KMCR
 010704  122737  000036  001127         CMPB    #36,$BDDAT+1            ; ALL SETS VALID?
 010712  001404                         BEQ     2$                      ; IF OK, BRANCH
 010714  112737  000036  001125         MOVB    #36,$GDDAT+1            ; STORE EXPECTED PATTERN
 010722  104026                         ERROR   +26                     ; DEALLOCATING A
 010724  052737  000400  177734 2$:     BIS     #BIT08,KMCR             ; READ AVAILABILITY BITS
 010732  013737  177734  001126         MOV     KMCR,$BDDAT             ; STORE KMCR
 010740  122737  000017  001127         CMPB    #17,$BDDAT+1            ; MRU:A,D,C,B
 010746  001404                         BEQ     3$                      ; IF OK, BRANCH
 010750  112737  000017  001125         MOVB    #17,$GDDAT+1            ; STORE EXPECTED PATTERN
 010756  104026                         ERROR   +26                     ; DEALLOCATING A
                                ;
                                ; TRY TO BRING A NEW SET TO B
                                ;
 010760  012701  000120         3$:     MOV     #120,R1                 ; PREPARE FOR NEXT READ
 010764  042737  000400  177734         BIC     #BIT08,KMCR             ; READ VALID BITS
 010772  004737  002222                 JSR     PC,DDIN                 ; READ INTO CACHE
 010776  013737  177734  001126         MOV     KMCR,$BDDAT             ; STORE KMCR
 011004  122737  000036  001127         CMPB    #36,$BDDAT+1            ; ALL VALID?
 011012  001404                         BEQ     4$                      ; IF YES, BRANCH
```

```
2514 011014  112737  000036  001125          MOVB    #36,$GDDAT+1            ; STORE EXPECTED PATTERN
2515 011022  104026                          ERROR   +26                     ; DEALLOCATING B
2516 011024  052737  000400  177734  4$:     BIS     #BIT08,KMCR             ; READ AVAILABILITY BITS
2517 011032  013737  177734  001126          MOV     KMCR,$BDDAT             ; STORE KMCR
2518 011040  122737  000103  001127          CMPB    #103,$BDDAT+1           ; MRU:B,A,D,C
2519 011046  001404                          BEQ     5$                      ; IF OK, BRANCH
2520 011050  112737  000103  001125          MOVB    #103,$GDDAT+1           ; STORE EXPECTED PATTERN
2521 011056  104026                          ERROR   +26                     ; DEALLOCATING B
2522                                 ;
2523                                 ; TRY TO BRING A NEW SET TO C
2524                                 ;
2525 011060  012701  000140          5$:     MOV     #140,R1                 ; PREPARE FOR NEXT READ
2526 011064  042737  000400  177734          BIC     #BIT08,KMCR             ; READ VALID BITS
2527 011072  004737  002222                  JSR     PC,DDIN                 ; READ INTO CACHE
2528 011076  013737  177734  001126          MOV     KMCR,$BDDAT             ; STORE KMCR
2529 011104  122737  000036  001127          CMPB    #36,$BDDAT+1            ; ALL VALID?
2530 011112  001404                          BEQ     6$                      ; IF YES, BRANCH
2531 011114  112737  000036  001125          MOVB    #36,$GDDAT+1            ; STORE EXPECTED PATTERN
2532 011122  104026                          ERROR   +26                     ; DEALLOCATING C
2533 011124  052737  000400  177734  6$:     BIS     #BIT08,KMCR             ; READ AVAILABILITY BITS
2534 011132  013737  177734  001126          MOV     KMCR,$BDDAT             ; STORE KMCR
2535 011140  122737  000151  001127          CMPB    #151,$BDDAT+1           ; MRU:C,B,A,D
2536 011146  001404                          BEQ     7$                      ; IF OK, BRANCH
2537 011150  112737  000151  001125          MOVB    #151,$GDDAT+1           ; STORE EXPECTED PATTERN
2538 011156  104026                          ERROR   +26                     ; DEALLOCATING C
2539                                 ;
2540                                 ; TRY TO BRING A NEW SET TO D
2541                                 ;
2542 011160  012701  000160          7$:     MOV     #160,R1                 ; PREPARE FOR NEXT READ
2543 011164  042737  000400  177734          BIC     #BIT08,KMCR             ; READ VALID BITS
2544 011172  004737  002222                  JSR     PC,DDIN                 ; READ INTO CACHE
2545 011176  013737  177734  001126          MOV     KMCR,$BDDAT             ; STORE KMCR
2546 011204  122737  000036  001127          CMPB    #36,$BDDAT+1            ; ALL VALID?
2547 011212  001404                          BEQ     8$                      ; IF YES, BRANCH
2548 011214  112737  000036  001125          MOVB    #36,$GDDAT+1            ; STORE EXPECTED PATTERN
2549 011222  104026                          ERROR   +26                     ; DEALLOCATING D
2550 011224  052737  000400  177734  8$:     BIS     #BIT08,KMCR             ; READ AVAILABILITY BITS
2551 011232  013737  177734  001126          MOV     KMCR,$BDDAT             ; STORE KMCR
2552 011240  042737  000400  177730          BIC     #BIT08,DCSR             ; EXIT DIAGNOSTIC MODE
2553 011246  122737  000177  001127          CMPB    #177,$BDDAT+1           ; MRU:D,C,B,A
2554 011254  001404                          BEQ     TST26           ;;      IF OK, EXIT TEST
2555 011256  112737  000177  001125          MOVB    #177,$GDDAT+1           ; STORE EXPECTED PATTERN
2556 011264  104026                          ERROR   +26                     ; DEALLOCATING D
2557 011266                          100$:
2558
```

```
                                .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
                                ;;**************************************************************************
011266  000004                  TST26:  SCOPE

011270  005737  001720                  TST     PMIS                            ; ANY PMI MEMORY?
011274  001460                  BEQ     TST27                   ;; IF NONE, EXIT TEST
011276  052737  000100  177734  BIS     #BIT06,KMCR             ; MAKE SURE THAT CACHE IS ENABLED
011304  052737  000400  177730  BIS     #BIT08,DCSR             ; SELECT DIAGNOSTIC MODE
011312  042737  000040  172516  BIC     #BIT05,MMR3             ; DISABLE RELOCATION
011320  042737  000400  177734  BIC     #BIT08,KMCR             ; SELECT VALID BITS
011326  042737  000400  177734  BIC     #BIT08,KMCR             ; READ VALID BITS FIRST
011334  013737  177734  001124  MOV     KMCR,$GDDAT             ; STORE LOW BYTE
011342  105037  001125          CLRB    $GDDAT+1                ; CLEAR HIGH BYTE
011346  032737  017000  177734  BIT     #BIT12!BIT11!BIT10!BIT9,KMCR    ; ALL VALID BITS CLEAR?
011354  001404                  BEQ     3$                      ; IF YES, BRANCH
011356  013737  177734  001126  MOV     KMCR,$BDDAT             ; STORE KMCR
011364  104026                  ERROR   +26                     ; VALID BITS NOT CLEAR WITH KMCR<6>=0
011366  052737  000400  177734  3$:     BIS     #BIT08,KMCR             ; NOW READ AVAILABILITY BITS
011374  013737  177734  001126  MOV     KMCR,$BDDAT             ; STORE KMCR
011402  122737  000177  001127  CMPB    #177,$BDDAT+1           ; ALL SET?
011410  001404                  BEQ     4$                      ; IF YES, EXIT TEST
011412  112737  000177  001125  MOVB    #177,$GDDAT+1           ; EXPECTED PATTERN
011420  104026                  ERROR   +26                     ; AVAIL. BITS NOT SET WITH KMCR<6>=0
011422  042737  000100  177734  4$:     BIC     #BIT06,KMCR             ; DISABLE CACHE
011430  042737  000400  177730  BIC     #BIT08,DCSR             ; EXIT DIAGNOSTIC MODE
```

```
                                .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
                                ;;*****************************************************************
    011436  000004              TST27:  SCOPE
                                
011440  005737  001720                  TST     PMIS                    ; ANY PMI MEMORY?
011444  001002                          BNE     20$                     ; IF SOME, DO THE TEST
011446  000137  012120                  JMP     100$                    ; OTHERWISE EXIT
011452  052737  000400  177730  20$:    BIS     #BIT08,DCSR             ; SELECT DIAGNOSTIC MODE
011460  052737  000100  177734          BIS     #BIT06,KMCR             ; CACHE STILL ENABLED
011466  052737  000040  172516          BIS     #BIT05,MMR3             ; ENABLE RELOCATION
011474  012737  000200  170200          MOV     #200,MAPL00             ; POINT MAP00 TO FIRST 8KB
011502  012737  000000  170202          MOV     #0,MAPH00               ; IN HIGH AND LOW MAP REGISTERS
011510  042737  000400  177734          BIC     #BIT08,KMCR             ; SELECT VALID BITS
011516  013702  177734                  MOV     KMCR,R2                 ; SAVE VALID BITS OF KMCR
011522  052737  000400  177734          BIS     #BIT08,KMCR             ; SELECT AVAILABILITY BITS
011530  013703  177734                  MOV     KMCR,R3                 ; SAVE AVAILABILITY BITS
                                ;
                                ; ACCESS LOCATION NOT IN CACHE
                                ;
011534  012701  000200                  MOV     #200,R1                 ; ACCESS ADDRESS
011540  013737  000400  001124          MOV     #0400,$GDDAT            ; THE PATTERN TO BE WRITTEN
011546  004737  002210                  JSR     PC,DDOUT                ; DO DIAGNOSTIC DATA OUT
011552  042737  000400  177734          BIC     #BIT08,KMCR             ; SELECT VALID BITS
011560  020237  177734                  CMP     R2,KMCR                 ; KMCR CHANGED?
011564  001406                          BEQ     1$                      ; IF NOT, BRANCH
011566  010237  001124                  MOV     R2,$GDDAT               ; EXPECTED PATTERN
011572  013737  177734  001126          MOV     KMCR,$BDDAT             ; RECIEVED PATTERN
011600  104026                          ERROR   +26                     ; WRITE MISSES AFFETS CACHE
011602  052737  000400  177734  1$:     BIS     #BIT08,KMCR             ; SELECT AVAILABILITY BITS
011610  020337  177734                  CMP     R3,KMCR                 ; KMCR CHANGED?
011614  001406                          BEQ     2$                      ; IF OK, BRANCH
```

T27 WRITE CYCLES AND CACHE

```
2662 011616 010337 001124                MOV     R3,$GDDAT                ; EXPECTED PATTERN
2663 011622 013737 177734 001126         MOV     KMCR,$BDDAT              ; RECIEVED PATTERN
2664 011630 104026                       ERROR   .26                      ; WRITE MISSES AFFETS CACHE
2665                              ;
2666                              ; CHECK THAT WRITE HIT INVALIDATES CACHE
2667                              ;
2668 011632 042737 000100 177734 2$:     BIC     @BIT06,KMCR              ; DISABLE CACHE TO GET TO KNOW STATE
2669 011640 052737 000100 177734         BIS     @BIT06,KMCR              ; ENABLE CACHE
2670 011646 012701 000000                MOV     @0,R1                    ; ADDRESS 0
2671 011652 004737 002222                JSR     PC,DDIN                  ; ALLOCATE CACHE SET A
2672 011656 012701 000020                MOV     @20,R1                   ; ADDRESS 20
2673 011662 004737 002222                JSR     PC,DDIN                  ; AND ALLOCATE SET B
2674 011666 005001                       CLR     R1                       ; ACCESS SET A
2675 011670 013737 000200 001124         MOV     @@200,$GDDAT             ; THE PATTERN TO BE WRITTEN
2676 011676 004737 002210                JSR     PC,DDOUT                 ; INVALIDATE CACHE
2677 011702 032737 100000 177734         BIT     @BIT15,KMCR              ; HIT?
2678 011710 001012                       BNE     3$                       ; IF YES, BRANCH
2679 011712 013737 177734 001124         MOV     KMCR,$GDDAT              ; PRESERVE LOW BYTE IF ERROR
2680 011720 112737 000363 001124         MOVB    @363,$GDDAT              ; HIT, B LEAST AVAILABLE
2681 011726 013737 177734 001126         MOV     KMCR,$BDDAT              ; RECIEVED PATTERN
2682 011734 104026                       ERROR   .26                      ; WRITE HIT WAS NOT RECORDED
2683 011736 042737 000400 177734 3$:     BIC     @BIT08,KMCR              ; READ VALID BITS
2684 011744 032737 010000 177734         BIT     @BIT12,KMCR              ; A VALID?
2685 011752 001412                       BEQ     4$                       ; IF INVALIDATED, BRANCH
2686 011754 013737 177734 001124         MOV     KMCR,$GDDAT              ; PRESERVE LOW BYTE IF ERROR
2687 011762 112737 000010 001125         MOVB    @BIT03,$GDDAT+1          ; MISS, B VALID
2688 011770 013737 177734 001126         MOV     KMCR,$BDDAT              ; RECIEVED PATTERN
2689 011776 104026                       ERROR   .26                      ; WRITE HIT DOES NOT INVALIDATE
2690 012000 013737 177734 001124 4$:     MOV     KMCR,$GDDAT              ; PRESERVE LOW BYTE IF ERROR
2691 012006 112737 000200 001125         MOVB    @200,$GDDAT+1            ; HIT RECIEVED
2692 012014 013737 000220 000220         MOV     @@220,@@220              ; CPU WRITE TO SET B
2693 012022 032737 100000 177734         BIT     @BIT15,KMCR              ; HIT?
2694 012030 001004                       BNE     5$                       ; IF YES, BRANCH
2695 012032 013737 177734 001126         MOV     KMCR,$BDDAT              ; STORE KMCR
2696 012040 104026                       ERROR   .26                      ; HIT NOT RECORDED
2697 012042 032737 004000 177734 5$:     BIT     @BIT11,KMCR              ; B STILL VALID?
2698 012050 001412                       BEQ     6$                       ; IF NOT, EXIT TEST
2699 012052 013737 177734 001124         MOV     KMCR,$GDDAT              ; IN CASE HIT WENT AWAY
2700 012060 142737 000030 001125         BICB    @BIT04!BIT03,$GDDAT+1    ; CLEAR ALL EXTRA
2701 012066 013737 177734 001126         MOV     KMCR,$BDDAT              ; RECIEVED PATTERN
2702 012074 104026                       ERROR   .26                      ; CPU WRITE DOES NOT INVALIDATE
2703 012076 042737 000100 177734 6$:     BIC     @BIT06,KMCR              ; DISABLE CACHE
2704 012104 042737 000040 172516         BIC     @BIT05,MMR3              ; AND MAPPING
2705 012112 042737 000400 177730         BIC     @BIT08,DCSR              ; EXIT DIAGNOSTIC MODE
2706 012120                      100$:
```

```
                                .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
                                ;;*****************************************************************
012120  000004                  TST30:  SCOPE

012122  005737  001720                  TST     @PMIS                   ; ANY PMI MEMORY?
012126  001502                          BEQ     TST31           ;; IF NONE, EXIT TEST

                                ;
                                ; ENABLE CACHE AND ALLOCATE SET A
                                ;
012130  052737  000400  177730          BIS     #BIT08,@DCSR            ; SELECT DIAGNOSTIC MODE
012136  052737  000040  172516          BIS     #BIT05,@MMR3            ; ENABLE RELOCATION
012144  052737  000100  177734          BIS     #BIT06,@KMCR            ; ENABLE CACHE
012152  012737  000200  170200          MOV     #200,@MAPL00            ; POINT MAP00 TO FIRST 8KB
012160  012737  000000  170202          MOV     #0,@MAPH00              ; IN HIGH AND LOW MAP REGISTERS
012166  012701  000000                  MOV     #0,R1                   ; ALLOCATE 200-216
012172  004737  002222                  JSR     PC,@DDIN                ; IN CACHE

                                ;
                                ; TRY TO CHECK WHETHER NON-ZERO INDEX EFFECT CACHE
                                ;
012176  042737  000400  177734          BIC     #BIT08,@KMCR            ; SELECT VALID BITS
012204  013737  177734  001124          MOV     @KMCR,$GDDAT            ; SAVE VALID BITS OF KMCR
012212  052737  000400  177734          BIS     #BIT08,@KMCR            ; SELECT AVAILABILITY BITS
012220  013703  177734                  MOV     @KMCR,R3                ; SAVE AVAILABILITY BITS
012224  012701  000102                  MOV     #102,R1                 ; TRY CACHING 102
012230  004737  002222                  JSR     PC,@DDIN                ; DO DIAGNOSTIC DATA IN
012234  042737  000400  177734          BIC     #BIT08,@KMCR            ; SELECT VALID BITS
012242  023737  001124  177734          CMP     $GDDAT,@KMCR            ; KMCR CHANGED?
012250  001404                          BEQ     1$                      ; IF NOT, BRANCH
012252  013737  177734  001126          MOV     @KMCR,$BDDAT            ; RECIEVED PATTERN
012260  104026                          ERROR   .26                     ; NON-ZERO INDEX AFFECTS CACHE
012262  052737  000400  177734  1$:     BIS     #BIT08,@KMCR            ; SELECT AVAILABILITY BITS
012270  020337  177734                  CMP     R3,@KMCR                ; KMCR CHANGED?
012274  001406                          BEQ     2$                      ; IF NOT, EXIT TEST
012276  010337  001124                  MOV     R3,$GDDAT               ; EXPECTED DATA
012302  013737  177734  001126          MOV     @KMCR,$BDDAT            ; RECIEVEED DATE
012310  104026                          ERROR   .26                     ; NON-ZERO INDEX AFFECTS CACHE
```

T30 DMA READ WITH INDEX NOT ZERO

```
2762 012312  042737  000100  177734  2$:     BIC     #BIT06,KMCR             ; DISABLE CACH
2763 012320  042737  000040  172516          BIC     #BIT05,MMR3             ; AND MAPPING
2764 012326  042737  000400  177730          BIC     #BIT08,DCSR             ; EXIT DIAGNOSTIC MODE
2765
```

```
                                .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
                                ;;*************************************************************************
012334  000004                  TST31:  SCOPE
2812
2813 012336  005737  001720            TST     PMIS                            ; ANY PMI MEMORY?
2814 012342  001002                    BNE     1$                              ; YES
2815 012344  000137  012654            JMP     100$                            ; IF NONE, EXIT TEST
2816 012350  032737  000040  177734  1$:       BIT     #BIT05,KMCR             ; 18-BIT MODE?
2817 012356  001177                    BNE     TST32           ;; IF YES, EXIT TEST
2818 012360  052737  000400  177730            BIS     #BIT08,DCSR             ; SELECT DIAGNOSTIC MODE
2819 012366  004737  002244                    JSR     PC,MAPPR                ; REMAP PROGRAM AREA
2820 012372  005237  177572                    INC     MMR0                    ; ENABLE MMU
```

T31 TAG REGISTERS

```
2821 012376 052737 000020 172516          BIS     #BIT04,MMR3                     ; ENABLE 22-BIT
2822 012404 022737 000040 001720          CMP     #40,PMIS                        ; IF 2M PRESENT, ONLY 21ST MASKED
2823 012412 001412                        BEQ     5$                              ; DON'T CONSTRUCT MASK
2824                                      ;
2825                                      ; IF LESS THEN 2M OF MEMORY, MAKE UP A MASK FOR MAP HIGH REGISTERS
2826                                      ;
2827 012414 000241                        CLC                                     ; CLEAR CARRY
2828 012416 013701 001720                 MOV     PMIS,R1                         ; SAVE KIPAR6
2829 012422 006101                 4$:    ROL     R1                              ; . ROTATE LEFT R1
2830 012424 053701 001720                 BIS     PMIS,R1                         ; . AND ADD BIT AT FIRST POSITION
2831 012430 103374                        BHIS    4$                              ; . IF CARRY NOT SET, BRANCH
2832 012432 000301                        SWAB    R1                              ; <15-8><---><7-0>
2833 012434 006001                        ROR     R1                              ; NOW GET ADDRESS BITS <21-16>
2834 012436 006001                        ROR     R1                              ; FROM <21-14>
2835                                      ;
2836                                      ; ALLOCATE DIFFERENT PATTERNS IN TAG REGISTERS
2837                                      ;
2838 012440 042737 000100 177734   5$:    BIC     #BIT06,KMCR                     ; DISABLE CACHE
2839 012446 052737 000100 177734          BIS     #BIT06,KMCR                     ; ENABLE CACHE
2840 012454 042737 000400 177734          BIC     #BIT08,KMCR                     ; MAKE SURE THAT VALID READ
2841 012462 052737 000040 172516          BIS     #BIT05,MMR3                     ; ENABLE MAPPING
2842 012470 012702 012656                 MOV     #TBLML,R2                       ; POINTER FOR LOW MAP PATTERNS
2843 012474 012703 012716                 MOV     #TBLMH,R3                       ; POINTER FOR HIGH MAP PATTERNS
2844 012500 010137 001160                 MOV     R1,$TMP0                        ; SAVE THE MASK
2845 012504 012704 000004                 MOV     #4,R4                           ; COUNTER FOR 16 PATTERNS
2846 012510 012701 020000          6$:    MOV     #20000,R1                       ; . TO ACCESS THRU MAP REG. 1
2847 012514 012705 000004                 MOV     #4,R5                           ; . DO EACH FOR AT A TIME
2848 012520 012237 170204          7$:    MOV     (R2)+,MAPL01                    ; . . LOAD LOW MAP
2849 012524 012337 170206                 MOV     (R3)+,MAPH01                    ; . . LOAD HIGH MAP
2850 012530 043737 001160 170206          BIC     $TMP0,MAPH01                    ; . . MASK OUT NXM
2851 012536 004737 002222                 JSR     PC,DDIN                         ; . . DO DIAGNOSTIC DATI
2852 012542 077512                        SOB     R5,7$                           ; . . DO FOR ALL 4 PATTERNS
2853 012544 013701 177734                 MOV     KMCR,R1                         ; . STORE SETS VALID?
2854 012550 005101                        COM     R1                              ; . COMPLEMENT VALID BITS
2855 012552 032701 017000                 BIT     #17000,R1                       ; . THAT'S ALL VALID BITS
2856 012556 001401                        BEQ     8$                              ; . IF ALL SET BEFORE COM, BRANCH
2857 012560 104027                        ERROR   +27                             ; . NOT ALL VALID BITS SET
2858                                      ;
2859                                      ; NOW INVALIDATE BY WRITING TO THE SAME LOCATIONS
2860                                      ;
2861 012562 162702 000010          8$:    SUB     #10,R2                          ; . MOVE POINTER TO -4
2862 012566 162703 000010                 SUB     #10,R3                          ; . MOVE HIGH MAP POINTER TOO
2863 012572 012705 000004                 MOV     #4,R5                           ; . VALIDATE ALLOCATED PATTERNS
2864 012576 012701 020000                 MOV     #20000,R1                       ; . POINTER TO MAP 1
2865 012602 012237 170204          9$:    MOV     (R2)+,MAPL01                    ; . . LOAD LOW MAP
2866 012606 012337 170206                 MOV     (R3)+,MAPH01                    ; . . LOAD HIGH MAP
2867 012612 043737 001160 170206          BIC     $TMP0,MAPH01                    ; . . MASK OUT NXM
2868 012620 004737 002222                 JSR     PC,DDIN                         ; . . DO DAIGN. DATA IN
2869 012624 032737 100000 177734          BIT     #BIT15,KMCR                     ; . . HIT?
2870 012632 001001                        BNE     10$                             ; . . IF HIT, BRANCH
2871 012634 104027                        ERROR   +27                             ; . . IN TAG REGISTERS
2872 012636 077517                 10$:   SOB     R5,9$                           ; . . DO FOR ALL 4 PATTERNS
2873 012640 077455                        SOB     R4,6$                           ; . REPEAT FOR ALL 16 PATTERNS
2874 012642 005037 177572                 CLR     MMR0                            ; DISABLE MMU
2875 012646 042737 000400 177730          BIC     #BIT08,DCSR                     ; EXIT DIAGNOSTIC MODE
2876 012654                        100$:
     012654 000440                        BR      TST32           ;;      EXIT TEST
```

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

```
2893                            .SBTTL  TEST - CACHE RAM BIT PATTERN TEST
2894
2895                            ;* THIS TEST IS DONE TO CHECK THE OPERATION OF
2896                            ;* THE CACHE RAM . BIT PATTERNS 0,177777,52525
2897                            ;* AND 125252 ARE WRITTEN TO ALL OF THE MEMORY
2898                            ;* USED AND THEN VERIFIED
2899                            ;
2900                            ;  BGNTST
2901                            ;
2902                            ;*
2903                            ;* GO SET UP THE CACHE IMAGE TABLE
2904                            ;*
2905                            ;       LET R0 := #CTBLE
2906                            ;       REPEAT
2907                            ;       .  IF R0 POINTS TO OCTAL BOUNDARY ADDRESS THEN
2908                            ;       .  .  WE HAVE FOUND THE BEGINNING OF THE TABLE
2909                            ;       .  ELSE
2910                            ;       .  .  POINT R0 TO THE NEXT ADDRESS
2911                            ;       .  ENDIF
2912                            ;       UNTIL WE HAVE FOUND BEGINNING OF THE TABLE
2913                            ;*
2914                            ;* SET UP BOARD FOR THE TRANSFERS
2915                            ;*
2916                            ;       ENABLE UNIBUS MAPPING
2917                            ;       ENABLE DIAGNOSTIC MODE
2918                            ;       ENABLE CACHE
2919                            ;       DO FOR PATTERN := 0,177777,52525,125252
2920                            ;*
2921                            ;* INITIALIZE THE CACHE TABLE
2922                            ;*
2923                            ;       .  DO FOR ALL OF THE CACHE TABLE
2924                            ;       .  .  WRITE PATTERN TO TABLE
2925                            ;       .  ENDDO
2926                            ;*
2927                            ;* ALLOCATE ALL THE CACHE
2928                            ;*
2929                            ;       .  DO UNTIL ALL CACHE ALLOCATED
2930                            ;       .  .  DO A DIAGNOSTIC DATI NPR CYCLE (ON OCTAL BOUNDARY)
2931                            ;       .  ENDDO
2932                            ;*
2933                            ;* CHECK THAT THE DATA GOT CACHED CORRECTLY
2934                            ;*
2935                            ;       .  DO FOR ALL OF THE CACHE TABLE
2936                            ;       .  .  DO A DIAGNOSTIC DATI NPR CYCLE
2937                            ;       .  .  IF DDR NEQ PATTERN THEN
2938                            ;       .  .  .  ERROR IN CACHE RAM
2939                            ;       .  .  ENDIF
2940                            ;       .  ENDDO
2941                            ;       ENDDO
2942                            ;
2943                            ;  ENDTST
2944                            ;
2945                            ;---------------------------------------------------------------
2946
2947                            ;;*************************************************************
                                ;*TEST 32        CACHE RAM BIT TEST
                                ;;*************************************************************
```

T32 CACHE RAM BIT TEST

```
                012756  000004                  TST32:  SCOPE
   2948 012760  005737  001720                          TST     PMIS            ; ANY PMI MEMORY?
   2949 012764  001511                          BEQ     TST33           ;; IF NONE, EXIT TEST
   2950                                         ;
   2951                                         ; FIND THE FIRST OCTAL BOUNDARY ADDRESS IN THE TABLE
   2952                                         ;
   2953 012766  052737  000400  177730          BIS     #BIT08,DCSR     ; SELECT DIAGNOSTIC MODE
   2954 012774  012700  002022                  MOV     #CTBLE,R0       ; GET POINTER TO THE START OF THE SPACE ALLOCATED FOR THE CA
CHE TABLE
   2955 013000  010001                  5$:     MOV     R0,R1           ; SAVE IT IN R1 FOR WORKING
   2956 013002  042701  177760                  BIC     #177760,R1      ; MASK IN THE LEAST SIGNIFICANT DIGIT
   2957 013006  005701                          TST     R1              ; IS THE ADDRESS ON AN OCTAL BOUNDARY ?
   2958 013010  001402                          BEQ     10$             ; YES,THEN GO SET UP THE TABLE ADDRESSES
   2959 013012  005720                          TST     (R0)+           ; NO,THEN GET THE NEXT ADDRESS
   2960 013014  000771                          BR      5$              ; . GO CHECK IF IT FALLS ON AN OCTAL BOUNDARY
   2961 013016  012702  002140          10$:    MOV     #OBADR,R2       ; GET POINTER TO THE OCTAL BOUNDARY TABLE
   2962 013022  012703  000004                  MOV     #4,R3           ; SET UP THE LOOP COUNTER
   2963 013026  010022                  12$:    MOV     R0,(R2)+        ; . GO PUT ADDRESS INTO THE TABLE
   2964 013030  062700  000020                  ADD     #20,R0          ; . GET NEXT OCTAL BOUNDARY
   2965 013034  077304                          SOB     R3,12$          ; . FILL UP THE TABLE ?
   2966                                         ;
   2967                                         ; INITIALIZE THE BOARD FOR THE TRANSFERS
   2968                                         ;
   2969 013036  052737  000040  172516          BIS     #BIT5,@#MMR3    ; ENABLE UNIBUS MAPPING
   2970 013044  005037  170202                  CLR     MAPH00          ; CLEAR HIGH MAP REGISTER
   2971 013050  052737  000400  177730          BIS     #BIT8,DCSR      ; ENABLE DIAGNOSTIC MODE
   2972 013056  052737  000100  177734          BIS     #BIT6,KMCR      ; ENABLE THE CACHE
   2973 013064  012700  001774                  MOV     #PTRN16,R0      ; GET POINTER TO THE CACHE PATTERN TABLE
   2974 013070  013701  002140          15$:    MOV     OBADR,R1        ; GET POINTER TO THE CACHE IMAGE TABLE
   2975                                         ;
   2976                                         ; INITIALIZE THE CACHE IMAGE TABLE TO CURRENT PATTERN
   2977                                         ;
   2978 013074  012702  000040                  MOV     #40,R2          ; . SET UP LOOP COUNTER
   2979 013100  011021                  20$:    MOV     (R0),(R1)+      ; . . MOVE PATTERN TO TABLE
   2980 013102  077202                          SOB     R2,20$          ; . . HAVE WE DONE IT 40 TIMES ?
   2981                                         ;
   2982                                         ; ALLOCATE ALL THE CACHE MEMORY AVAILABLE
   2983                                         ;
   2984 013104  012702  002140                  MOV     #OBADR,R2       ; . GET ADDRESS OF OCTAL BOUNDARIRES WITHIN THE TAB;E
   2985 013110  012703  000004                  MOV     #4,R3           ; . SET UP LOOP COUNTER
   2986 013114  005001                          CLR     R1              ; . NON-MAPPED ADDRRESS=0
   2987 013116  012237  170200          25$:    MOV     (R2)+,MAPL00    ; . . SET UP MAP FOR DIAG NPR DATI CALL
   2988 013122  004737  002222                  JSR     PC,DDIN         ; . . GO DO A DIAGNOSTIC DATA IN CYCLE
   2989 013126  077305                          SOB     R3,25$          ; . . HAVE WE DONE IT 4 TIMES ?
   2990                                         ;
   2991                                         ; CHECK THE CACHE RAM IMAGE PATTERN
   2992                                         ;
   2993 013130  012703  000040                  MOV     #40,R3          ; . SET UP LOOP COUNTER
   2994 013134  013702  002140                  MOV     OBADR,R2        ; . GET ADDRESS OF CACHE IMAGE TABLE
   2995                                                                 ;   (THIS WHOLE TABLE SHOULD BE CACHED)
   2996 013140  005001                          CLR     R1              ; . USE ADDRESS 0
   2997 013142  010237  170200          30$:    MOV     R2,MAPL00       ; . . GET ADDRESS FPR DIAG. DATA IN CYCLE
   2998 013146  004737  002222                  JSR     PC,DDIN         ; . . GO DO A DIAGNOSTIC DATA IN
   2999 013152  032737  100000  177734          BIT     #BIT15,KMCR     ; . . HIT?
   3000 013160  001001                          BNE     35$             ; . . IF YES, BRANCH
   3001 013162  104030                          ERROR   +30             ; . . OTHERWISE, ERROR
   3002 013164  023722  177732          35$:    CMP     DDR,(R2)+       ; . . DID IT GET STORED CORRECTLY ?
   3003 013170  001401                          BEQ     40$             ; . . YES,THEN GO CHECK THE NEXT LOCATION
```

J6

T32 CACHE RAM BIT TEST

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

```
                        .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
                        ;
                        ;-------------------------------------------------------------------------

013210                  UBETST:
                        ;;*************************************************************************
                        ;*TEST 33       BOOT ROMS TEST
                        ;;*************************************************************************
013210  000004          TST33:  SCOPE
                        ;
                        ; PREPARE TO DO CRC
                        ;
013212  042737  000400  177730          BIC     #BIT08,DCSR             ; LEAVE STANDALONE MODE
013220  052737  000200  177520          BIS     #BIT07,BCSR             ; TO ENABLE BOOT ROMS
013226  005037  177522                  CLR     PCR                     ; TO READ PAGE 0
013232  012704  173000                  MOV     #173000,R4              ; R4 PONTS TO FIRST ADDRESS
013236  005037  001160                  CLR     $TMP0                   ; CLEAR EMPTY SOCKET COUNT
                        ;
                        ; DO FOR EACH POSSIBLE ROM
                        ;
```

```
 3064 013242  022704  173776          1$:     CMP     #173776,R4              ; . ALL ROM'S DONE?
 3065 013246  103475                          BLO     11$                     ; . IF SO, EXIT
 3066 013250  005001                          CLR     R1                      ; . CLEAR COUNTER THRU A ROM
 3067 013252  005005                          CLR     R5                      ; . INITIALIZE PARTIAL CRC
 3068                                         ;
 3069                                         ; CHECK FOR EMPTY SOCKETS AND EXIT IF SO
 3070                                         ;
 3071 013254  022714  161777                  CMP     #161777,(R4)            ; . EMPTY SOCKET?
 3072 013260  001027                          BNE     3$                      ; . IF NOT, BRANCH TO DO CRC
 3073 013262  005737  001206                  TST     $PASS                   ; . FIRST PASS?
 3074 013266  001017                          BNE     2$                      ; . IF NOT, BRANCH
 3075 013270  122737  000001  001220          CMPB    #APTENV,$ENV            ; IN APT MODE?
 3076 013276  001413                          BEQ     2$                      ; IF SO, SKIP PRINTOUT
 3077 013300  032737  000040  000052          BIT     #BIT05,@#52             ; . IN UFD MODE?
 3078 013306  001007                          BNE     2$                      ; . IF IN UFD, BRANCH
 3079 013310  010446                          MOV     R4,-(SP)                ; . PUT NUMBER TO TYPE OUT
 3080 013312  104401  013576                  TYPE    ,NROM                   ; . TYPE ASCII MESSAGE
 3081 013316  104403                          TYPOS                           ; . CALL TYPE OUT ROUTINE
 3082 013320     006                          .BYTE   6                       ; . TYPE 6 DIGITS
 3083 013321     000                          .BYTE   0                       ; . SUPRESS LEADING 0'S
 3084 013322  104401  001175                  TYPE    ,$CRLF                  ; . CARRIAGE RETURN
 3085 013326  005237  001160          2$:     INC     $TMP0                   ; . INCREMENT EMPTY SOCKET COUNT
 3086 013332  062704  000200                  ADD     #200,R4                 ; . PREPARE TO DO NEXT ROM
 3087 013336  000741                          BR      1$                      ; . BRANCH TO DO NEXT ROM
 3088                                         ;
 3089                                         ; IN EACH ROM CHECK LOCATION 24 AND DON'T DO CRC ON IT
 3090                                         ;
 3091 013340  022701  000024          3$:     CMP     #24,R1                  ; . LOCATION 24 ACCESSED?
 3092 013344  001007                          BNE     5$                      ; . IF NO, GO DO CRC
 3093 013346  022724  173000                  CMP     #173000,(R4)+           ; . PROPER DATA AT 24?
 3094 013352  001401                          BEQ     4$                      ; . IF YES, BRANCH
 3095 013354  104031                          ERROR   +31                     ; . WRONG DATA AT 24
 3096 013356  005201                  4$:     INC     R1                      ; . INCREMENT FOR AN EXTRA BYTE
 3097 013360  000137  013420                  JMP     8$                      ; . DO NOT DO CRC FOR 24
 3098                                         ;
 3099                                         ; DO CRC FOR EACH BYTE
 3100                                         ;
 3101 013364  112403                  5$:     MOVB    (R4)+,R3                ; . STORE CORRECT DATA IN R3
 3102 013366  012702  000010                  MOV     #8.,R2                  ; . NUMBER OF BITS PER BYTE
 3103 013372  000241                  6$:     CLC                             ; . . CLEAR CARRY
 3104 013374  006005                          ROR     R5                      ; . . LOW BIT PARTIAL TO CARRY
 3105 013376  006003                          ROR     R3                      ; . . CARRY TO BYTE AND BYTE TO CARRY
 3106 013400  102006                          BVC     7$                      ; . . XOR OF PARIAL AND BYTE LOW BITS
 3107 013402  012746  120001                  MOV     #POLY,-(SP)             ; . . XOR POLY TO PARTIAL (4 INSTRUCTIONS)
 3108 013406  040516                          BIC     R5,(SP)                 ; . . NOT PARTIAL AND POLY
 3109 013410  042705  120001                  BIC     #POLY,R5                ; . . NOT POLY AND PARTIAL
 3110 013414  052605                          BIS     (SP)+,R5                ; . . POLY XOR PARTIAL
 3111 013416  077213                  7$:     SOB     R2,6$                   ; . . DECREMENT BIT COUNT AND CONTINUE
 3112 013420  005201                  8$:     INC     R1                      ; . COUNT BYTES
 3113                                         ;
 3114                                         ; IF A ROM DONE, CHECK FOR 0 IN R5
 3115                                         ;
 3116 013422  022701  000200                  CMP     #128.,R1                ; . ALL 64 WORDS DONE?
 3117 013426  003344                          BGT     3$                      ; . IF NOT, BRANCH
 3118 013430  005705                          TST     R5                      ; . IF YES, CRC 0?
 3119 013432  001401                          BEQ     10$                     ; . IF CRC = 0, BRANCH
 3120 013434  104031                          ERROR   +31                     ; . CRC FOR A ROM NOT EQUAL TO 0
```

M6


```
3121 013436  000137  013242          10$:   JMP     1$                      ; . DO FOR NEXT ROM
3122                                 ;
3123                                 ; TRY TO WRITE TO GET A TIMEOUT
3124                                 ;
3125 013442  013737  000004  001160  11$:   MOV     ERRVEC,$TMP0            ; SAVE TIMEOUT VECTOR
3126 013450  012737  013466  000004        MOV     #15$,ERRVEC             ; POINT NEW TO PROGRAM
3127 013456  005037  173000          14$:   CLR     @#173000                ; TRY TO WRITE TO 1ST PAGE OF ROM
3128 013462  104031                        ERROR   +31                     ; WRITE ACCESS TO ROM'S DIDN'T TIMEOUT
3129 013464  000402                        BR      16$                     ;
3130 013466  005726          15$:   TST     (SP)+                   ; RESTORE STACK
3131 013470  005726                 TST     (SP)+                   ;
3132 013472  032737  000400  177730  16$:   BIT     #BIT08,DCSR             ; OUT OF STANDALONE MODE?
3133 013500  001013                        BNE     20$                     ; IF NOT, EXIT TEST
3134 013502  013702  177734                MOV     KMCR,R2                 ; MAKE SURE THAT
3135 013506  042702  177700                BIC     #177700,R2              ; AT LEAST SOME OF
3136 013512  022702  000077                CMP     #77,R2                  ; MEMORY PMI
3137 013516  001404                        BEQ     20$                     ; IF NOT, BRANCH
3138 013520  052737  000400  177730        BIS     #BIT08,DCSR             ; DO 1 MORE TIME IN STANDALONE MODE
3139 013526  000753                        BR      14$                     ;
3140 013530  052737  000010  177730  20$:   BIS     #BIT03,DCSR             ; DISABLE UBA ROM RESPONSE
3141 013536  012737  013554  000004        MOV     #25$,ERRVEC             ; POINT NEW TIMEOUT VECTOR
3142 013544  005737  173000                TST     @#173000                ; READ BOOT ROM
3143 013550  104037                        ERROR   +37                     ; DSCR<3> DIDN'T DISABLE UBA ROM
3144 013552  000402                        BR      26$                     ;
3145 013554  005726          25$:   TST     (SP)+                   ; RESTORE STACK
3146 013556  005726                 TST     (SP)+
3147 013560  042737  000410  177730  26$:   BIC     #BIT08!BIT03,DCSR       ; LEAVE STANDALONE MODE
3148 013566  013737  001160  000004        MOV     $TMP0,ERRVEC            ; RESTORE STACK
3149 013574  000417                        BR      TST34                   ;; EXIT TEST
3150
3151 013576      101      104      104  NROM:   .ASCIZ  /ADDRESS OF EMPTY UBA SOCKET /
     013601      122      105      123
     013604      123      040      117
     013607      106      040      105
     013612      115      120      124
     013615      131      040      125
     013620      102      101      040
     013623      123      117      103
     013626      113      105      124
     013631      040      000
3152                                 .EVEN
```

```
                                .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
                                ;;*************************************************************************
013634  000004                  TST34:  SCOPE
                                ;
                                ; CHECK IF ALL NON-UNIBUS MEMORY
                                ;
013636  C32737  G00037  177734          BIT     #37,KMCR                ; BITS <5-0> CLEAR?
013644  001473                        BEQ     TST35           ;;      IF YES, SKIP TEST
                                ;
                                ; CHECK IF ALL UNIBUS MEMORY
                                ;
013646  013701  177734                  MOV     KMCR,R1                 ; SAVE KMCR
```

```
3208 013652  042701  177700                BIC     #177700,R1              ; LEAVE ONLY <5-0>
3209 013656  022701  000077                CMP     #77,R1                  ; BITS <5-0> ALL SET?
3210 013662  001005                        BNE     1$                      ; IF NOT, BRANCH
3211 013664  012701  001600                MOV     #1600,R1                ; DON'T TEST FIRST 32K
3212 013670  012702  007600                MOV     #7600,R2                ; 248KB MAXIMUM CONFIGURATION
3213 013674  000420                        BR      2$                      ; GO DO TEST
3214 013676  004737  002424          1$:   JSR     PC,UMSIZ                ; SIZE UNIBUS MEMORY
3215                                       ;
3216                                       ; ON RETURN R2 HAS PAR VALUE FOR UNIBUS MEMORY
3217                                       ;
3218 013702  010201                        MOV     R2,R1                   ; STORE LOWER BOUNDARY
3219 013704  012702  007600                MOV     #7600,R2                ; HIGHER BOUNDARY
3220 013710  032737  000040  177734        BIT     #BIT05,KMCR             ; 18 BIT MODE?
3221 013716  001007                        BNE     2$                      ; IF YES, GO DO TEST
3222 013720  052701  170000                BIS     #170000,R1              ; EXTEND TO 22 BITS
3223 013724  052702  170000                BIS     #170000,R2              ; FOR HIGHER BOUNDARY TOO
3224 013730  052737  000020  172516        BIS     #BIT04,MMR3             ; ENABLE 22 BIT MAPPING
3225                                       ;
3226                                       ; NOW WRITE MEMORY WITH 01 PATTERN, VERIFY IT AND DO THE SAME FOR 10 PATTERN
3227                                       ;
3228 013736  004737  002244          2$:   JSR     PC,MAPPR                ; REMAP PROGRAM TO FIRST 32K
3229 013742  005237  177572                INC     MMR0                    ; ENABLE MMU
3230 013746  010137  172354                MOV     R1,KIPAR6               ; START AT LOWER BOUNDARY
3231 013752  012704  140000          3$:   MOV     #140000,R4              ; . START 4K PAGE
3232 013756  012714  125252          4$:   MOV     #125252,(R4)            ; . . WRITE FIRST PATTERN
3233 013762  022714  125252                CMP     #125252,(R4)            ; . . WRITEN OK?
3234 013766  001401                        BEQ     5$                      ; . . IF OK, BRANCH
3235 013770  104034                        ERROR   .34                     ; . . IN UNIBUS MEMORY
3236 013772  005114                5$:     COM     (R4)                    ; . . COMPLEMENT INITIAL PATTERN
3237 013774  022714  052525                CMP     #52525,(R4)             ; . . NEW PATTERN OK?
3238 014000  001401                        BEQ     6$                      ; . . IF OK, BRANCH
3239 014002  104034                        ERROR   .34                     ; . . IN UNIBUS MEMORY
3240 014004  005724                6$:     TST     (R4)+                   ; . . GET NEXT MEMORY LOCATION
3241 014006  022704  160000                CMP     #160000,R4              ; . . LAST ONE IN 4K PAGE?
3242 014012  101361                        BHI     4$                      ; . . IF NOT, BRANCH
3243 014014  062737  000200  172354        ADD     #200,KIPAR6             ; . GET NEXT PAGE
3244 014022  020237  172354                CMP     R2,KIPAR6               ; . LAST PAGE DONE?
3245 014026  101351                        BHI     3$                      ; . IF NOT, BRANCH
3246 014030  005037  177572                CLR     MMR0                    ; DISABLE MMU
3247
3248                                       .SBTTL UBE TESTS FOR THE UNIBUS ADAPTER BOARD
3249
3250                                       ;+
3251                                       ;+ THE FOLLOWING TESTS REQUIRE THE USE OF THE UNIBUS EXERCISER
3252                                       ;+ TO TEST OUT THE UNIBUS ADAPTER BOARD.SOME TESTS REQUIRE ONE
3253                                       ;+ UBE OTHER TESTS REQUIRE TWO UBE'S.
3254                                       ;+
3255                                       ;+ THE PROGRAM WILL AUTOSIZE TO SEE HOW MANY UBE'S ARE IN THE
3256                                       ;+ SYSTEM. DEPENDING ON THE RESULTS OF THE AUTOSIZING CERTAIN
3257                                       ;+ TESTS WILL BE SELECTED OR DESELECTED.
3258                                       ;+
3259                                       ;-----------------------------------------------------------------
3260
```

```
                                .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
                                ;;*************************************************************
    014034  000004              TST35:  SCOPE

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

```
                                        ;
                                        ; INITIALIZE POINTERS FOR AUTO-SIZING
                                        ;
014060  012700  170000                  MOV     #170000,R0      ; GET ADDRESS OF FIRST POSSIBLE UBE
014064  012702  000510                  MOV     #510,R2         ; GET VECTOR OF FIRST POSSIBLE UBE
014070  012701  002150                  MOV     #8E1D8,R1       ; TABLE HEADER FOR 1ST UBE
014074  012737  014106  000004          MOV     #1$,@#4         ; SET UP TIMEOUT VECTOR
014102  005710                  100$:   TST     (R0)            ; . IS THERE A UBE HERE ?
014104  000412                  BR      2$                      ; . . YES,THEN BRANCH AROUND TOMEOUT ROUTINE
                                        ;
                                        ; TIMEOUT ROUTINE
                                        ;
014106  062706  000004          1$:     ADD     #4,SP           ; . . READJUST THE STACK
014112  020027  170160                  CMP     R0,#170160      ; . . HAVE WE CHECKED ALL THE ADDRESSES
014116  001431                  BEQ     10$                     ; . . YES THEN GO SEE IF WE FOUND ANY UBE'S
014120  062700  000020                  ADD     #20,R0          ; . . NO,THEN GET THE NEXT UBE ADDRESS
014124  062702  000004                  ADD     #4,R2           ; . . GET THE NEXT UBE VECTOR
014130  000764                  BR      100$                    ; . . GO SEE IF THE NEXT UBE IS THERE
                                        ;
                                        ; ASSIGN UBE ADDRESSES TO THE CURRENT UBE TABLE
                                        ;
014132  012704  000005          2$:     MOV     #5,R4           ; . SET UP LOOP COUNTER
014136  010021                  3$:     MOV     R0,(R1)+        ; . . ASSIGN AN ADDRESS TO THE POINTER
014140  005720                  TST     (R0)+                   ; . . GET THE NEXT ADDRESS
014142  077403                  SOB     R4,3$                   ; . . ARE WE DONE ? NO THEN GO GET NEXT ADDRESS
014144  062700  000004                  ADD     #4,R0           ; . POINT R0 TO LAST UBE ADDRESS
014150  010021                  MOV     R0,(R1)+                ; . PUT THE ADDRESS INTO THE POINTER TABLE
014152  005720                  TST     (R0)+                   ; . POINT R0 TO NEXT UBE ADDRESS

                                        ;
                                        ; ASSIGN UBE VECTORS TO CURRENT UBE TABLE
                                        ;

014154  010221                  MOV     R2,(R1)+                ; . GET POINTER TO VECTOR ADDRESS
014156  005722                  TST     (R2)+                   ; . GET THE VECTOR PSW LOCATION
014160  010221                  MOV     R2,(R1)+                ; . GET POINTER TO VECTOR PSW
014162  005722                  TST     (R2)+                   ; . GET THE NEXT UBE'S VECTOR ADDRESS
014164  005237  001722                  INC     UBECT           ; . FLAG WE HAVE FOUND ANOTHER UBE

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

014170  022737  000002  001722          CMP     #2,UBECT        ; . HAVE WE FOUND TWO UBE'S
014176  001401                  BEQ     10$                     ; . GO DO THE UBE TESTS
014200  000740                  BR      100$

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

014202  012737  002234  000004  10$:     MOV     #TIMOUT,@#4     ; RESTORE TIMEOUT VECTOR
014210  005737  001722                  TST     UBECT           ; HAVE ANY UBE'S BEEN FOUND ?
014214  001002                  BNE     TST36                   ;; GO DO THE SELECTED TESTS FOR 1 UBE
014216  000137  022740                  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
                                ;;**************************************************************
     014222  000004             TST36:  SCOPE

                                        JSR     PC,IUBE                         ; INITIALIZE THE UBE
     014224  004737  002450             MOV     #340,PSW                        ; STORE PRIORITY 7
     014230  012737  000340  177776     MOV     #-1,@BE1CC              1$:     ; . DO FOR 1 CYCLE
     014236  012777  177777  165706     MOV     #$TMP0,@BE1BA                   ; . STORE ADDRESS FOR DATI
     014244  012777  001160  165702     MOV     #2041,@BE1CR1                   ; . DO 1 DATI AT NPG
     014252  012777  002041  165676     TSTB    @BE1CR1                 2$:     ; . . READY ON?
     014260  105777  165672             BPL     2$                              ; . . WAIT FOR READY
     014264  100375                     CMP     #-1,@BE1CC                      ; . CYCLE COUNT INCREMENTED?
     014266  022777  177777  165656     BNE     3$                              ; . IF YES, BRANCH
     014274  001001                     ERROR   +32                             ; . IF NO, ERROR IN NPG
     014276  104032                     SUB     #40,PSW                 3$:     ; . DECREMENT PRIORITY
     014300  162737  000040  177776     CMP     #0,PSW                          ; . LAST ONE?
     014306  022737  000000  177776     BLE     1$                              ; . IF NOT, BRANCH
     014314  003750                     MOV     #340,PSW                        ; RESTORE PRIORITY
     014316  012737  000340  177776
```

```
3413                            .SBTTL TEST - NO BUS GRANTS WITH PROCESSOR AT HIGHER PRIORITY
3414                            ;
3415                            ;* THIS TEST CHECKS THAT NO BUS REQUESTS ARE GOING TO BE HONORED
3416                            ;* WHEN THE PROCESSOR HAS HIGHER PRIORITY THAN THE REQUESTING DEVICE.
3417                            ;*
3418                            ;
3419                            ;  BGNTST
3420                            ;
3421                            ;       SET UP @BE1VEC TO POINT TO ERROR_CHECK_ROUTINE
3422                            ;       SET UP @BE1PSW TO #340
3423                            ;*
3424                            ;* TRY TO DO BR7 WITH CPU PRIORITY AT 7
3425                            ;*
3426                            ;       LET @BE1CR1 = #21 TO DO 1 DATI
3427                            ;       LET PSW = #340 TO SET PRIORITY AT 7
3428                            ;       DO "NOP"
3429                            ;       IF INTERUPT THEN
3430                            ;       . ERROR BG7 GRANTED WITH PROCESSOR AT HIGHER PRIORITY
3431                            ;       ENDIF
3432                            ;
3433                            ;*
3434                            ;* TRY TO DO BR6 WITH CPU PRIORITY AT 7 - 6
3435                            ;*
3436                            ;       DO FOR R3 FROM #340 DOWNTO #300 BY #40
3437                            ;       . LET @BE1CR1 = #11 TO DO 1 DATI
3438                            ;       . LET PSW = R3 TO CHANGE PRIORITY
3439                            ;       . DO "NOP"
3440                            ;       . IF INTERRUPT  THEN
3441                            ;       . . ERROR BG6 GRANTED WITH PROCESSOR AT HIGHER PRIORITY
3442                            ;       . ENDIF
3443                            ;       ENDDO
3444                            ;       LET PSW = #340 FOR THE NEXT PART
3445                            ;*
3446                            ;* TRY TO DO BR5 WITH CPU PRIORITY AT 7 - 5
3447                            ;*
3448                            ;       DO FOR R3 FROM #340 DOWNTO #240 BY #40
3449                            ;       . LET @BE1CR1 = #5 TO DO 1 DATI
3450                            ;       . LET PSW = R3 TO CHANGE PRIORITY
3451                            ;       . DO "NOP"
3452                            ;       . IF INTERRUPT THEN
3453                            ;       . . ERROR BG5 GRANTED WITH PROCESSOR AT HIGHER PRIORITY
3454                            ;       . ENDIF
3455                            ;       ENDDO
3456                            ;       LET PSW = #340
3457                            ;*
3458                            ;* TRY TO DO BR4 WITH CPU PRIORITY AT 7 - 4
3459                            ;*
3460                            ;       DO FOR R3 FROM #340 DOWNTO #200 BY #40
3461                            ;       . LET @BE1CR1 = #2003 TO DO 1 DATI
3462                            ;       . LET PSW = R3 TO CHANGE PRIORITY
3463                            ;       . DO "NOP"
3464                            ;       . IF INTERRUPT THEN
3465                            ;       . . ERROR BR4 GRANTED WITH PROCESSOR AT HIGHER PRIORITY
3466                            ;       . ENDIF
3467                            ;       ENDDO
3468                            ;
3469                            ;  ENDTST
```

TEST - NO BUS GRANTS WITH PROCESSOR AT HIGHER PRIORITY

```
                                        ;
                                        ;-------------------------------------------------------------
                                        ;
                                        ;;*************************************************************
                                        ;*TEST 37       NO BUS GRANTS WITH PROCESSOR AT HIGHER PRIORITY
                                        ;;*************************************************************
014324  000004                          TST37:  SCOPE
                                        ;
014326  004737  002450                          JSR     PC,IUBE                 ; INITIALIZE THE UBE
014332  012737  000340  177776                MOV     #340,PSW                ; SET CPU PRIORITY AT 7
014340  012777  000340  165620                MOV     #340,@BE1PSW            ; AT PRIORITY 7
                                        ;
                                        ; TRY TO DO BR7 WITH CPU PRIORITY AT 7
                                        ;
014346  012777  014366  165610                MOV     #1$,@BE1VEC             ; POINT UBE VECTOR TO PROGRAM
014354  012777  000021  165574                MOV     #21,@BE1CR1             ; BR7 WITH FUNO
014362  000240                               NOP                             ; JUST IN CASE
014364  000405                               BR      2$                      ; BRANCH AROUND IF NO INTERRUPT
014366  005077  165564          1$:         CLR     @BE1CR1                 ; CLEAR ANY OTHER REQUESTS
014372  062706  000004                       ADD     #4,SP                   ; ADJUST STACK POINTER
014376  104032                               ERROR   .32                     ; BG7 GRANTED WITH CPU AT 7
                                        ;
                                        ; TRY TO DO BR6 WITH CPU PRIORITY AT 7-6
                                        ;
014400  012777  014430  165556  2$:         MOV     #4$,@BE1VEC             ; POINT UBE VECTOR TO PROGRAM
014406  012703  000340                       MOV     #340,R3                 ; START WITH PRIORITY AT 7
014412  012777  000011  165536  3$:         MOV     #11,@BE1CR1             ; . BR6 WITH FUNO
014420  010337  177776                       MOV     R3,PSW                  ; . CHANGE PRIORITY
014424  000240                               NOP                             ; . JUST IN CASE OF INTERRUPS
014426  000405                               BR      5$                      ; . IF NO INTERRUPS, BRANCH
014430  005077  165522          4$:         CLR     @BE1CR1                 ; . CLEAR ANY OTHER REQUESTS
014434  062706  000004                       ADD     #4,SP                   ; . ADJUST STACK POINTER
014440  104032                               ERROR   .32                     ; . BG6 GRANTED WITH CPU AT 6-7
014442  162703  000040          5$:         SUB     #40,R3                  ; . LOWER PRIORITY
014446  020327  000300                       CMP     R3,#300                 ; . LAST ONE TO CHECK?
014452  002357                               BGE     3$                      ; . IF NOT, BRANCH
                                        ;
                                        ; TRY TO DO BR5 WITH CPU PRIORITY AT 7-5
                                        ;
014454  012777  014504  165502                MOV     #7$,@BE1VEC             ; POINT UBE VECTOR TO PROGRAM
014462  012703  000340                       MOV     #340,R3                 ; START WITH PRIORITY AT 7
014466  012777  000005  165462  6$:         MOV     #5,@BE1CR1              ; . BR5 WITH FUNO
014474  010337  177776                       MOV     R3,PSW                  ; . CHANGE PRIORITY
014500  000240                               NOP                             ; . JUST IN CASE OF INTERRUPS
014502  000405                               BR      8$                      ; . IF NO INTERRUPS, BRANCH
014504  005077  165446          7$:         CLR     @BE1CR1                 ; . CLEAR ANY OTHER REQUESTS
014510  062706  000004                       ADD     #4,SP                   ; . ADJUST STACK POINTER
014514  104032                               ERROR   .32                     ; . BG5 GRANTED WITH CPU AT 5-7
014516  162703  000040          8$:         SUB     #40,R3                  ; . LOWER PRIORITY
014522  020327  000240                       CMP     R3,#240                 ; . LAST ONE TO CHECK?
014526  002357                               BGE     6$                      ; . IF NOT, BRANCH
                                        ;
                                        ; TRY TO DO BR4 WITH CPU PRIORITY AT 7-4
                                        ;
014530  012777  014560  165426                MOV     #10$,@BE1VEC            ; POINT UBE VECTOR TO PROGRAM
014536  012703  000340                       MOV     #340,R3                 ; START WITH PRIORITY AT 7
014542  012777  000003  165406  9$:         MOV     #3,@BE1CR1              ; . BR6 WITH FUNO
```

T37 NO BUS GRANTS WITH PROCESSOR AT HIGHER PRIORITY

```
3524 014550 010337 177776          MOV     R3,PSW          ; . CHANGE PRIORITY
3525 014554 000240                 NOP                     ; . JUST IN CASE OF INTERRUPS
3526 014556 000405                 BR      11$             ; . IF NO INTERRUPS, BRANCH
3527 014560 005077 165372    10$:  CLR     @#BE1CR1        ; . CLEAR ANY OTHER REQUESTS
3528 014564 062706 000004          ADD     #4,SP           ; . ADJUST STACK POINTER
3529 014570 104032                 ERROR   ^32             ; . BG4 GRANTED WITH CPU AT 4-7
3530 014572 162703 000040    11$:  SUB     #40,R3          ; . LOWER PRIORITY
3531 014576 020327 000200          CMP     R3,#200         ; . LAST ONE TO CHECK?
3532 014602 002357                 BGE     9$              ; . IF NOT, BRANCH
3533
3534
```

```
3536                    .SBTTL TEST - BR7-BR4 ARBITRATION
3537
3538                    ;* THIS TEST CHECKS THAT BR7-BR4 INTERRUPTS CAN OCCUR WITH A PROCESSOR
3539                    ;* PRIORITY AT A LOWER LEVEL
3540                    ;*
3541                    ;
3542                    ;  BGNTST
3543                    ;
3544                    ;       SET UP @BE1PSW TO #340
3545                    ;*
3546                    ;* TRY TO DO BR7 WITH CPU PRIORITY AT 6 - 0
3547                    ;*
3548                    ;       DO FOR R3 FROM #300 DOWNTO #0 BY #40
3549                    ;       .  LET PSW = #340
3550                    ;       .  LET @BE1CR1 = #21 TO DO 1 DATI
3551                    ;       .  LET PSW = R3 TO CHANGE PRIORITY
3552                    ;       .  DO "NOP"
3553                    ;       .  IF NO INTERRUPT THEN
3554                    ;       .  .  ERROR BG7 DOESN'T  OCCUR
3555                    ;       .  ENDIF
3556                    ;       ENDDO
3557                    ;*
3558                    ;* TRY TO DO BR6 WITH CPU PRIORITY AT 5 - 0
3559                    ;*
3560                    ;       DO FOR R3 FROM #240 DOWNTO #0 BY #40
3561                    ;       .  LET PSW = #340
3562                    ;       .  LET @BE1CR1 = #11 TO DO 1 DATI
3563                    ;       .  LET PSW = R3 TO CHANGE PRIORITY
3564                    ;       .  DO "NOP"
3565                    ;       .  IF NO INTERUPT THEN
3566                    ;       .  .  ERROR BG6 DOESN'T  OCCUR
3567                    ;       .  ENDIF
3568                    ;       ENDDO
3569                    ;*
3570                    ;* TRY TO DO BR5 WITH CPU PRIORITY AT 4 - 0
3571                    ;*
3572                    ;       DO FOR R3 FROM #200 DOWNTO #0 BY #40
3573                    ;       .  LET PSW = #340
3574                    ;       .  LET @BE1CR1 = #5 TO DO 1 DATI
3575                    ;       .  LET PSW = R3 TO CHANGE PRIORITY
3576                    ;       .  DO "NOP"
3577                    ;       .  IF NO INTERRUPT THEN
3578                    ;       .  .  ERROR BG5 DOESN'T  OCCUR
3579                    ;       .  ENDIF
3580                    ;       ENDDO
3581                    ;*
3582                    ;* TRY TO DO BR4 WITH CPU PRIORITY AT 3 - 0
3583                    ;*
3584                    ;       DO FOR R3 FROM #140 DOWNTO #0 BY #40
3585                    ;       .  LET PSW = #340
3586                    ;       .  LET @BE1CR1 = #3 TO DO 1 DATI
3587                    ;       .  LET PSW = R3 TO CHANGE PRIORITY
3588                    ;       .  DO "NOP"
3589                    ;       .  IF NO INTERRUPT THEN
3590                    ;       .  .  ERROR BG4 DOESN'T  OCCUR
3591                    ;       .  ENDIF
3592                    ;       ENDDO
```

```
                                        ;
                                        ;  ENDTST
                                        ;
                                        ;-------------------------------------------------------------
                                        ;
                                        ;;*************************************************************
                                        ;*TEST 40        BR7-BR4 ARBITRATION
                                        ;;*************************************************************
    014604  000004                      TST40:  SCOPE
                                        ;
                                        ;
                                        ; TRY TO DO BR7 WITH CPU PRIORITY AT 6-0
                                        ;
                                        ;
                                        ; TRY TO DO BR6 WITH CPU PRIORITY AT 5-0
                                        ;
                                        ;
                                        ; TRY TO DO BR5 WITH CPU PRIORITY AT 4-0
                                        ;
```

```
                                                JSR     PC,IUBE                 ; INITIALIZE THE UBE
014606  004737  002450                          MOV     #340,@#BE1PSW           ; AT PRIORITY 7
014612  012777  000340  165346
                                                MOV     #2$,@#BE1VEC            ; POINT UBE VECTOR TO PROGRAM
014620  012777  014664  165336                  MOV     #300,R3                 ; START WITH PRIORITY AT 6
014626  012703  000300                  1$:     MOV     #340,PSW                ; . RAISE PRIORITY TO 7
014632  012737  000340  177776                  MOV     #21,@#BE1CR1            ; . BR7 WITH FUNO
014640  012777  000021  165310                  MOV     R3,PSW                  ; . LOWER PRIORITY
014646  010337  177776                          NOP                             ; . JUST IN CASE OF INTERRUPS
014652  000240                                  CLR     @#BE1CR1                ; . CLEAR ANY OTHER REQUESTS
014654  005077  165276                          ERROR   +32                     ; . BR7 NOT GRANTED WITH CPU AT 6-0
014660  104032                                  BR      3$                      ; . DON'T ADJUST STACK IF NO INTERRUPT
014662  000402                  2$:     ADD     #4,SP                   ; . ADJUST STACK POINTER
014664  062706  000004          3$:     SUB     #40,R3                  ; . LOWER PRIORITY
014670  162703  000040                  CMP     #0,R3                   ; . LAST ONE TO CHECK?
014674  022703  000000                  BGE     1$                      ; . IF NOT, BRANCH
014700  002354
                                        ;
                                        ; TRY TO DO BR6 WITH CPU PRIORITY AT 5-0
                                        ;
                                                JSR     PC,IUBE                 ; INITIALIZE THE UBE
014702  004737  002450                          MOV     #5$,@#BE1VEC            ; POINT UBE VECTOR TO PROGRAM
014706  012777  014752  165250                  MOV     #240,R3                 ; START WITH PRIORITY AT 5
014714  012703  000240                  4$:     MOV     #340,PSW                ; . RAISE PRIORITY TO 7
014720  012737  000340  177776                  MOV     #11,@#BE1CR1            ; . BR6 WITH FUNO
014726  012777  000011  165222                  MOV     R3,PSW                  ; . LOWER PRIORITY
014734  010337  177776                          NOP                             ; . JUST IN CASE OF INTERRUPS
014740  000240                                  CLR     @#BE1CR1                ; . CLEAR ANY OTHER REQUESTS
014742  005077  165210                          ERROR   +32                     ; . BR6 NOT GRANTED WITH CPU AT 5-0
014746  104032                                  BR      6$                      ; . DON'T ADJUST STACK
014750  000402                  5$:     ADD     #4,SP                   ; . ADJUST STACK POINTER
014752  062706  000004          6$:     SUB     #40,R3                  ; . LOWER PRIORITY
014756  162703  000040                  CMP     #0,R3                   ; . LAST ONE TO CHECK?
014762  022703  000000                  BGE     4$                      ; . IF NOT, BRANCH
014766  002354
                                        ;
                                        ; TRY TO DO BR5 WITH CPU PRIORITY AT 4-0
                                        ;
                                                JSR     PC,IUBE                 ; INITIALIZE THE UBE
014770  004737  002450                          MOV     #6$,@#BE1VEC            ; POINT UBE VECTOR TO PROGRAM
014774  012777  015040  165162                  MOV     #200,R3                 ; START WITH PRIORITY AT 4
015002  012703  000200                  7$:     MOV     #340,PSW                ; . RAISE PRIORITY TO 7
015006  012737  000340  177776                  MOV     #5,@#BE1CR1             ; . BR5 WITH FUNO
015014  012777  000005  165134                  MOV     R3,PSW                  ; . LOWER PRIORITY
015022  010337  177776                          NOP                             ; . JUST IN CASE OF INTERRUPS
015026  000240                                  CLR     @#BE1CR1                ; . CLEAR ANY OTHER REQUESTS
015030  005077  165122                          ERROR   +32                     ; . BR5 NOT GRANTED WITH CPU AT 4-0
015034  104032
```

T40 BR7-BR4 ARBITRATION

```
3647 015036  000402                        BR      9$                      ; . DON'T TOUCH STACK
3648 015040  062706  000004          8$:   ADD     #4,SP                   ; . ADJUST STACK POINTER
3649 015044  162703  000040          9$:   SUB     #40,R3                  ; . LOWER PRIORITY
3650 015050  022703  000000                CMP     #0,R3                   ; . LAST ONE TO CHECK?
3651 015054  002354                        BGE     7$                      ; . IF NOT, BRANCH
3652                                    ;
3653                                    ; TRY TO DO BR4 WITH CPU PRIORITY AT 3-0
3654                                    ;
3655 015056  004737  002450                JSR     PC,IUBE                 ; INITIALIZE THE UBE
3656 015062  012777  015126  165074        MOV     #11$,@BE1VEC            ; POINT UBE VECTOR TO PROGRAM
3657 015070  012703  000140                MOV     #140,R3                 ; START WITH PRIORITY AT 3
3658 015074  012737  000340  177776  10$:   MOV     #340,PSW                ; . RAISE PRIORITY TO 7
3659 015102  012777  000003  165046        MOV     #3,@BE1CR1              ; . BR4 WITH FUNO
3660 015110  010337  177776                MOV     R3,PSW                  ; . LOWER PRIORITY
3661 015114  000240                        NOP                             ; . JUST IN CASE OF INTERRUPS
3662 015116  005077  165034                CLR     @BE1CR1                 ; . CLEAR ANY OTHER REQUESTS
3663 015122  104032                        ERROR   +32                     ; . BR4 NOT GRANTED WITH CPU AT 3-0
3664 015124  000402                        BR      12$                     ; . DON'T ADJUST STACK
3665 015126  062706  000004          11$:  ADD     #4,SP                   ; . ADJUST STACK POINTER
3666 015132  162703  000040          12$:  SUB     #40,R3                  ; . LOWER PRIORITY
3667 015136  022703  000000                CMP     #0,R3                   ; . LAST ONE TO CHECK?
3668 015142  002354                        BGE     10$                     ; . IF NOT, BRANCH
3669
3670
```

```
                                .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
                                ;;*****************************************************************
 015144  000004                 TST41:  SCOPE
 015146  004737  002450                  JSR     PC,IUBE                 ; INITIALIZE THE UBE
 015152  012777  015242  164776          MOV     #2$,@BE1CR1             ; POINT UBE VECTOR TO PROGRAM
 015160  012777  000340  165000          MOV     #340,@BE1PSW            ; AT PRIORITY 7
 015166  012737  015252  000240          MOV     #3$,PIRQVEC             ; POINT PIRQ VECTOR TO PROGRAM
 015174  012737  000340  000242          MOV     #340,@#242              ; AT PRIORITY 7
 015202  012701  000020                  MOV     #20,R1                  ; STORE FOR BR7
 015206  012702  100000                  MOV     #100000,R2              ; STORE FOR PIRQ 7
 015212  012703  000300                  MOV     #300,R3                 ; START WITH PRIORITY 6
                                ;
                                ; DO ARBITRATION FOR LEVEL 7-4
                                ;
 015216  012737  000340  177776  1$:     MOV     #340,PSW                ; . START WITH PRIORITY 7
 015224  010177  164726                  MOV     R1,@BE1CR1              ; . CHANGE UBE PRIORITY
 015230  010237  177772                  MOV     R2,PIRQ                 ; . CHANGE PIRQ PRIORITY
 015234  010337  177776                  MOV     R3,PSW                  ; . LOWER CPU PRIORITY
 015240  000001                          WAIT                            ; . WAIT FOR INTERRUPT
 015242  062706  000004          2$:     ADD     #4,SP                   ; . CLEAN UP STACK
 015246  104032                          ERROR   +32                     ; . PIRQ'S DON'T TAKE PRIORITY
 015250  000402                          BR      4$                      ; . DON'T CLEAN UP STACK TWICE
 015252  062706  000004          3$:     ADD     #4,SP                   ; . CLEAN UP STACK
 015256  006001          4$:     ROR     R1                              ; . ADJUST BR FOR UBE
 015260  006002                          ROR     R2                      ; . ADJUST PIRQ'S
 015262  162703  000040                  SUB     #40,R3                  ; . LOWER FOR CPU PRIORITY
 015266  022703  000140                  CMP     #140,R3                 ; . PRIORITY 3?
 015272  001351                          BNE     1$                      ; . BRANCH IF NOT YET
 015274  005037  177772                  CLR     PIRQ                    ; CLEAR ANY REQUESTS
```

TEST - ARBITRATION BETWEEN INTERRUPTS FROM 2 UBE

```
                                .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 BETWEEN INTERRUPTS FROM 2 UBE
                                ;;*************************************************************************
    015300  000004              TST42:  SCOPE

015302  022737  000002  001722          CMP     #2,UBECT                        ; 2 UBE'S?
015310  001112                        BNE     TST43           ;;      EXIT, IF NO
015312  004737  002450                JSR     PC,IUBE                         ; INITIALIZE THE UBE
                                ;
                                ; INITIAL SETUP
                                ;
015316  012703  000240                MOV     #240,R3                         ; START AT PRIORITY 6
015322  012701  000020                MOV     #20,R1                          ; UBE AT LEVEL 7
015326  012777  015476  164630        MOV     #@BE1SV,@BE1VEC                 ; POINT UBE1 VECTOR TO PROGRAM
015334  012777  000340  164624        MOV     #340,@BE1PSW                    ; AT PRIORITY 7
015342  012777  015516  164634        MOV     #@BE2SV,@BE2VEC                 ; POINT UBE2 VECTOR TO PROGRAM
015350  012777  000340  164630        MOV     #340,@BE2PSW                    ; AT PRIORITY 7
                                ;
                                ; ARBITRATE BETWEEN 2 INTERRUPTS OF DIFFERENT LEVEL
                                ;
015356  012737  000340  177776  1$:   MOV     #340,PSW                        ; . RAISE PRIORITY TO 7
015364  010177  164566                MOV     R1,@BE1CR1                      ; . HIGHER PRIORITY TO UBE1
015370  006001                        ROR     R1                              ; . GET LOWER PRIORITY
015372  010177  164600                MOV     R1,@BE2CR1                      ; . LOWER PRIORITY TO UBE2
015376  052777  000001  164552        BIS     #BIT00,@BE1CR1                  ; . SET GO BIT OF UBE #1
```

T42 ARBITRATION BETWEEN INTERRUPTS FROM 2 UBE

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

TEST - POWER DOWN TEST

```
                                .SBTTL TEST - POWER DOWN TEST

                                ;* THIS TEST INSURES THAT POWER DOWN CYCLE CAN BE INVOKED FROM THE
                                ;* UNIBUS SIDE OF UBA. IT WILL RUN ONLY IF POWER UP OPTION 11
                                ;* IS SELECTED (TRAP THRU 24/26) IN THE EAROM ON CPU BOARD.
                                ;*
                                ;
                                ;  BGNTST
                                ;
                                ;       LET BCSR<6,5>=<0,1> TO ACCESS EAROM
                                ;       IF @#165002<7,6> NE <1,1> FOR POWER UP CODE 00 THEN
                                ;       .  EXIT TEST
                                ;       ENDIF
                                ;       SAVE PWRVEC
                                ;       POINT PWRVEC TO 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
                                ;;*************************************************************
015536  000004                  TST43:  SCOPE
                                ;
                                ; RESTRICTIONS ON RUNNING THIS TEST
                                ;
015540  005737  001206                  TST     @#PASS                  ; FIRST PASS
015544  001102                  .       BNE     TST44           ;;IF NOT 1ST PASS, DON'T DO IT
015546  032737  000040  000052  .       BIT     #BIT05,@#52             ; UFD MODE?
015554  001076                  .       BNE     TST44           ;;EXIT TEST, IF SO
015556  122737  000001  001220  .       CMPB    @#PTENV,#ENV            ; APT?
015564  001472                  .       BEQ     TST44           ;;EXIT TEST IF APT
015566  032737  000010  177524  .       BIT     #BIT03,@#177524         ; FORCED CONSOLE MODE?
015574  001066                  .       BNE     TST44           ;;IF YES, EXIT TEST
015576  042737  000100  177520  .       BIC     #BIT06,@#BCSR           ; ENABLE ACCESS THRU 165000
015604  052737  000040  177520  .       BIS     #BIT05,@#BCSR           ; READ EAROM
015612  032737  000100  165000  .       BIT     #BIT06,@#165000         ; BATTERY OVERRIDE?
015620  001454                  .       BEQ     TST44           ;;IF SO, EXIT TEST
015622  032737  000200  165002  .       BIT     #BIT07,@#165002         ; POWER UP CODE 00?
015630  001450                  .       BEQ     TST44           ;;EXIT TEST, IF NOT
015632  032737  000100  165002  .       BIT     #BIT06,@#165002         ; POWER UP CODE 00?
015640  001444                  .       BEQ     TST44           ;;EXIT TEST, IF NOT
015642  004737  002450          .       JSR     PC,@#UBE                ; INITIALIZE THE UBE
                                ;
                                ; DO POWER DOWN SEQUENCE
                                ;
015646  013737  000024  001160  .       MOV     @#PWRVEC,@#TMP0         ; STORE POWER DOWN VECTOR
015654  012737  015706  000024  .       MOV     #1$,@#PWRVEC            ; POINT NEW ONE TO PROGRAM
015662  013737  000026  000026  .       MOV     @#PWRVEC+2,@#26         ; AT PRIORITY 7
015670  052777  000020  164264  .       BIS     #BIT04,@#BE1CR2         ; START POWER DOWN
```

T43 POWER DOWN TEST

```
 3865 015676  000240                          NOP
 3866 015700  000240                          NOP                     ; WAIT WHAT HAPPENED
 3867 015702  104032                          ERROR   .32             ; NO POWER DOWN FOR THRU UNIBUS
 3868 015704  000417                          BR      10$             ;
 3869 015706  042777  000020  164246  1$:     BIC     #BIT04,@BE1CR2  ; CLEAR POWER DOWN BIT
 3870 015714  062706  000004                  ADD     #4,SP           ; ADJUST STACK POINTER
 3871 015720  012737  015740  000024          MOV     #3$,PWRVEC      ; POINT POWER UP VECTOR
 3872 015726  012701  177777                  MOV     #177777,R1      ; TIMEOUT ROUTINE
 3873 015732  077101                  2$:     SOB     R1,2$           ; WAIT A WHILE
 3874 015734  104032                          ERROR   .32             ; NO POWER UP
 3875 015736  000402                          BR      10$             ;
 3876 015740  012706  001100          3$:     MOV     #1100,SP        ; ADJUST STACK
 3877 015744  013737  001160  000024  10$:     MOV     $TMP0,PWRVEC    ; RESTORE POWER DOWN VECTOR
 3878
```

```
                                .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
                                ;;*************************************************************
015752  000004                  TST44:  SCOPE

015754  004737  002450                  JSR     PC,IUBE                 ; INITIALIZE THE UBE
015760  013701  000114                  MOV     @@114,R1                ; SAVE PARITY TRAP
015764  012737  016032  000114          MOV     #1$,@@114               ; POINT TO PROGRAM AREA
015772  012737  000340  000116          MOV     @340,@@116              ; AT PRIORITY 7
016000  012777  177777  164144          MOV     @-1,@BE1CC              ; DO 1 CYCLE
016006  012777  001160  164140          MOV     @$TMP0,@BE1BA           ; SETUP ADDRESS REGISTER
016014  052777  010000  164140          BIS     @BIT12,@BE1CR2          ; SET WRONG PARITY BIT
016022  005777  164130                  TST     @BE1CR1                 ; READ A UBE REGISTER
016026  000240                          NOP                             ; SHOULD CAUSE A PARITY TRAP
016030  104032                          ERROR   .32                     ; NO PARITY TRAP
016032  012777  000000  164122  1$:     MOV     @0,@BE1CR2              ; CLEAR WRONG PARITY BIT
016040  062706  000004                  ADD     @4,SP                   ; ADJUST STACK POINTER
```

```
                                .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
                                ;;**************************************************************
016044  000004                  TST45:  SCOPE

016046  004737  002450                  JSR     PC,IUBE                 ; INITIALIZE THE UBE
016052  012777  000010  164102          MOV     @BIT03,@BE1CR2          ; INHIBIT SACK BIT
016060  005037  177776                  CLR     PSW                     ; LOWER PRIORITY
016064  012777  006003  164064          MOV     @6003,@BE1CR1           ; GIVE UP BUS AFTER BECOMING MASTER
016072  012701  007777                  MOV     @7777,R1                ; DELAY CONSTANT
016076  077101                  1$:     SOB     R1,1$                   ; WAIT IN A LOOP
016100  105777  164056                  TSTB    @BE1CR2                 ; NO NO SACK TIMEOUT?
016104  100001                          BPL     2$                      ; IF NO, BRANCH
016106  104032                          ERROR   .32                     ; NO NO SACK TIMEOUT
016110  012777  000000  164044  2$:     MOV     @0,@BE1CR2              ; CLEAR INHIBIT SACK BIT
016116  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
                                ;;*****************************************************************
     016124  000004             TST46:  SCOPE

                                        JSR     PC,IUBE                 ; INITIALISE UBE
                                        MOV     #-1,@BE1CC              ; DO 1 CYCLE
                                        MOV     #$TMP0,@BE1BA           ; AT ADDRESS $TMP0
                                        MOV     #10$,@BE1VEC            ; POINT VECTOR
                                        MOV     #3,PSW                  ; LOWER PRIORITY TO 3
                                        MOV     #2021,@BE1CR1           ; BR7, DATI
                                1$:     TSTB    @BE1CR1                 ; DONE?
                                        BPL     1$                      ; IF NOT, WAIT
                                        MOV     #340,PSW                ; RESTORE PRIORITY
                                        CMP     $TMP0,@BE1D8            ; DATA OK?
                                        BEQ     TST47           ;; IF OK, EXIT TEST
                                        ERROR   .32                     ; DATI ON BR7 IS WRONG
                                        BR      TST47           ;; GO TO NEXT TEST

                                10$:    ERROR   .32                     ; NO INTERRUPT LOGIC IS WRONG
                                        RTI
```

TEST - UNIBUS DEVICE DATO CYCLE

```
                                .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
                                ;;*************************************************************************
016224  000004                  TST47:  SCOPE

016226  004737  002450                  JSR     PC,IUBE         ; INITIALIZE THE UBE
                                ;
                                ; INITIALIZE POINTERS FOR THE TRANSFERS
                                ;
016232  042737  000040  172516          BIC     #BIT5,MMR3      ; MAKE SURE UNIBUS MAPPING IS DISABLED
016240  012700  001734                  MOV     #WRTBUF,R0      ; POINTER TO WRITE BUFFER
016244  012701  001774                  MOV     #PTRN16,R1      ; POINTER TO PATTERN TABLE

                                ;
                                ; EXECUTE THE LOOP TO DO THE TRANSFERS
                                ;
016250  012737  000002  177730          MOV     #BIT01,DCSR     ; SELECT UNIBUS LINES
```

T47 UNIBUS DEVICE DATO CYCLE TEST SEQ 0098

```
4045 016256  012704  000005              MOV     #5,R4           ; SET UP LOOP COUNT
4046 016262  012177  163662        1$:   MOV     (R1)+,@BE1DB    ; . GET CURRENT DATA
4047 016266  010077  163662              MOV     R0,@BE1BA       ; . GET ADDRESS IN THE WRITE BUFFER
4048 016272  005077  163664              CLR     @BE1CR2         ; . CLEAR ADDRESS BITS 16,17
4049 016276  012777  177777  163646      MOV     #-1.,@BE1CC     ; . SET UBE FOR 1 DATA TRANSFER
4050 016304  012777  003041  163644      MOV     #3041,@BE1CR1   ; . SET UBE FOR NPR-DATO-1 XFER
4051
4052                                     ;
4053                                     ; WAIT FOR THE TRANSFER TO BE COMPLETE
4054                                     ;
4055
4056 016312  032777  000200  163636  5$:   BIT     #BIT7,@BE1CR1   ; . . IS THE TRANSFER COMPLETE ?
4057 016320  001774                      BEQ     5$              ; . . NO,THEN GO WAIT FOR IT
4058 016322  005737  177732              TST     DDR             ; . . UNIBUS LINES 0?
4059 016326  001415                      BEQ     7$              ; . . IF SO, BRANCH
4060 016330  022704  000003              CMP     #3,R4           ; . . SELECTING CONTROL LINES
4061 016334  001004                      BNE     6$              ; . . IF NOT, BRANCH
4062 016336  032737  000373  177732      BIT     #373,DDR        ; . . ONLY USED LINES 0'S?
4063 016344  001406                      BEQ     7$              ; . . IF SO, BRANCH
4064 016346  013737  177732  001126  6$:   MOV     DDR,$BDDAT      ; . . RECIEVED DATA
4065 016354  005037  001124              CLR     $GDDAT          ; . . EXPECTED DATA
4066 016360  104016                      ERROR   +16             ; . . ERROR IN UNIBUS LINES
4067 016362  062737  000002  177730  7$:   ADD     #BIT01,DCSR     ; . . THRU ALL COMBINATIONS
4068 016370  032737  000006  177730      BIT     #6,DCSR         ; . . ALL DONE?
4069 016376  001003                      BNE     8$              ; . . IF NOT, BRANCH
4070 016400  012737  000002  177730      MOV     #BIT01,DCSR     ; . . OTHERWISE, START OVER
4071 016406  062700  000002        8$:   ADD     #2,R0           ; . . GET THE NEXT ADDRESS TO TRANSFER TO
4072 016412  077455                      SOB     R4,1$           ; . HAVE WE GONE THROUGH THE TABLE ?
4073
4074                                     ;
4075                                     ; CHECK IF THE TRANSFERS WERE COMPLETED CORRECTLY
4076                                     ;
4077
4078 016414  012700  001734              MOV     #WRTBUF,R0      ; POINTER TO WRITE BUFFER
4079 016420  012701  001774              MOV     #PTRN16,R1      ; POINTER TO PATTERN TABLE
4080 016424  012704  000005              MOV     #5,R4           ; SET UP LOOP COUNT
4081 016430  022021                10$:  CMP     (R0)+,(R1)+     ; . WAS THE TRANSFER CORRECT ?
4082 016432  001401                      BEQ     15$             ; . YES,THEN SKIP THE ERROR MESSAGE
4083 016434  104032                      ERROR   +32             ; . NO, THEN ERROR IN THE TRANSFER
4084 016436  077404                15$:  SOB     R4,10$          ; . HAVE WE CHECKED ALL 5 XFERS ?
4085
4086
4087
```

```
4089                                    .SBTTL TEST - UNIBUS DEVICE DATI CYCLE
4090
4091                                    ;* THIS TEST WILL SEE IF THE KTJ11-B MODULE CAN EXECUTE A
4092                                    ;* UNIBUS DEVICE DATI CYCLE.THE UNIBUS ADDRESS PATH ON THE
4093                                    ;* MODULE IS ALSO TESTED OUT.
4094                                    ;*
4095                                    ;
4096                                    ;  BGNTST
4097                                    ;
4098                                    ;*
4099                                    ;*  INITIALIZE THE POINTERS FOR THE TRANSFERS
4100                                    ;*
4101                                    ;       DISABLE UNIBUS MAPPING
4102                                    ;       LET R0 := POINTER TO 16 BIT PATTERNS
4103                                    ;       LET R1 := (R0)                          ;CURRENT PATTERN
4104                                    ;*
4105                                    ;*  GO DO THE TRANSFER THEN CHECK IF IT OCCURED CORRECTLY
4106                                    ;*
4107                                    ;       LET BE1DB := 0                          ;CLEAR DATA BUFFER
4108                                    ;       LET BE1BA := R0                         ;READ FROM DATA PATTERN
4109                                    ;       LET BE1CC := -1                         ;1 DATA XFER
4110                                    ;       SETUP UBE TO DO NPR-DATI-1 DATA XFER
4111                                    ;       SET OFF THE TRANSFER
4112                                    ;       IF BE1DB NEQ #377  THEN
4113                                    ;       .  ERROR DATA READ IN WAS INCORRECT
4114                                    ;       ENDIF
4115                                    ;
4116                                    ;  ENDTST
4117                                    ;
4118                                    ;-------------------------------------------------------------------------
4119
4120                                    ;;*************************************************************************
                                        ;*TEST 50        UNIBUS DEVICE DATI CYCLE TEST
                                        ;;*************************************************************************
    016440  000004                      TST50:  SCOPE
4121
4122 016442  004737  002450                     JSR     PC,IUBE                         ; INITIALIZE THE UBE
4123 016446  042737  000040  172516             BIC     #BIT5,MMR3      ; DISABLE UNIBUS MAPPING
4124
4125                                    ;
4126                                    ; INITIALIZE POINTERS FOR THE TRANSFERS
4127                                    ;
4128
4129 016454  012700  001774                     MOV     #PTRN16,R0      ; GET POINTER TO PATTERN TABLE
4130 016460  011001                             MOV     (R0),R1         ; GET CURRENT PATTERN
4131
4132                                    ;
4133                                    ; GO DO THE TRANSFER
4134                                    ;
4135
4136 016462  005077  163462             5$:     CLR     @BE1DB          ; . INITIALIZE THE DATA BUFFER
4137 016466  010077  163462                     MOV     R0,@BE1BA       ; . GET ADDRESS WE WANT TO READ
4138 016472  005077  163464                     CLR     @BE1CR2         ; . CLEAR ADDRESS BITS 16,17
4139 016476  012777  177777  163446             MOV     #-1,@BE1CC      ; . SET UBE FOR 1 TRANSFER
4140 016504  012777  002041  163444             MOV     #2041,@BE1CR1   ; . SET UBE FOR NPR-DATI-1 XFER
4141
4142                                    ;
```

J8


```
                                        ; WAIT FOR THE TRANSFER TO COMPLETE
                                        ;
016512  032777  000200  163436  10$:    BIT     #BIT7,@BE1CR1   ; . . IS THE TRANSFER COMPLETE ?
016520  001774                          BEQ     10$             ; . . WAIT FOR IT TO BE COMPLETE

                                        ;
                                        ; CHECK THE TRANSFER
                                        ;

016522  027720  163422                  CMP     @BE1DB,(R0)+    ; . WAS THE TRANSFER CORRECT ?
016526  001401                          BEQ     TST51           ;;      YES,THEN GO TO THE NEXT TEST
016530  104032                          ERROR   +32             ; . NO,THEN ERROR IN THE TRANSFER
```

```
4161                                    .SBTTL TEST - UNIBUS DEVICE DATO CYCLE WITH RELOCATION ENABLED
4162
4163                                    ;* THIS TEST WILL SEE IF THE KTJ11-B MODULE CAN EXECUTE
4164                                    ;* A UNIBUS DEVICE DATO CYCLE THROUGH THE UNIBUS MAP.
4165                                    ;*
4166                                    ;
4167                                    ;  BGNTST
4168                                    ;       IF ALL UNIBUS MEMORY THEN GO TO END OF PASS
4169                                    ;*
4170                                    ;*   SET UP UBE AND UNIBUS MAP REGISTERS FOR TRANSFER
4171                                    ;*
4172                                    ;       LET BE18A := 0                  ;POINT TO UMR0
4173                                    ;       LET BE1DB := 125252             ;DATA PATTERN IS 125252
4174                                    ;       LET BE1CC := -8.                ;DO 8 XFERS
4175                                    ;       SET UBE TO DO NPR-DATO-2 DATA XFERS
4176                                    ;       LET MAPL00 := ADDRESS OF WRITE BUFFER
4177                                    ;       LET MAPH00 := 0
4178                                    ;*
4179                                    ;*      GO DO THE TRANSFER
4180                                    ;*
4181                                    ;        SET OFF TRANSFER
4182                                    ;        WAIT TILL XFER IS DONE
4183                                    ;*
4184                                    ;*      CHECK IF THE TRANSFER OCCURED CORRECTLY
4185                                    ;*
4186                                    ;        LET R0 := POINTER TO WRITE BUFFER
4187                                    ;        DO FOR R1 := 1 TO 8
4188                                    ;        .  IF (R0)+ NEQ #125252 THEN
4189                                    ;        .  .  ERROR DATO DID NOT EXECUTE CORRECTLY
4190                                    ;        .  ENDIF
4191                                    ;        ENDDO
4192                                    ;
4193                                    ;  ENDTST
4194                                    ;
4195                                    ;-------------------------------------------------------------
4196
4197                                    ;;*************************************************************
                                        ;*TEST 51        UNIBUS DEVICE DATO CYCLE WITH RELOCATION ENABLED
                                        ;;*************************************************************
    016532  000004                      TST51:  SCOPE
4198
4199 016534  013701  177734                      MOV     KMCR,R1         ; STORE KMCR
4200 016540  042701  177700                      BIC     #177700,R1      ; LEAVE ONLY BITS <5-0>
4201 016544  022701  000077                      CMP     #77,R1          ; ALL UNIBUS MEMORY?
4202 016550  001002                              BNE     1$              ; IF NOT, BRANCH
4203 016552  000137  022766                      JMP     $EOP            ; IF ALL UNIBUS, SKIP TILL END OF PASS
4204 016556  004737  002450              1$:     JSR     PC,IUBE         ; INITIALIZE THE UBE
4205
4206                                    ;
4207                                    ; SET UP UBE AND UNIBUS MAP REGISTERS FOR TRANSFER
4208                                    ;
4209
4210 016562  005077  163366                      CLR     @BE1BA          ; POINT UBE ADDRESS TO UNIBUS MAP
4211 016566  005077  163370                      CLR     @BE1CR2         ; REGISTER #0
4212 016572  012777  125252  163350              MOV     #125252,@BE1DB  ; DATA PATTERN IS 125252
4213 016600  012777  177770  163344              MOV     #-8.,@BE1CC     ; DO 8. TRANSFERS
4214 016606  012737  001734  170200              MOV     #WRTBUF,MAPL00  ; SET UP LOW MAP REGISTER
```

T51 UNIBUS DEVICE DATO CYCLE WITH RELOCATION ENABLED

```
4215 016614  005037  170202                    CLR     MAPH00          ; SET UP HIGH MAP REGISTER
4216 016620  052737  000040  172516            BIS     #BIT5,MMR3      ; ENABLE UNIBUS MAPPING
4217
4218                                    ;
4219                                    ; GO DO THE TRANSFER
4220                                    ;
4221
4222 016626  012777  003041  163322            MOV     #3041,@BE1CR1   ; DO A NPR-DATO-1 XFER
4223 016634  032777  000200  163314  10$:       BIT     #BIT7,@BE1CR1   ; IS THE TRANSFER COMPLETE ?
4224 016642  001774                            BEQ     10$             ; WAIT FOR IT TO COMPLETE
4225
4226                                    ;
4227                                    ; CHECK IF THE TRANSFER OCCURED CORRECTLY
4228                                    ;
4229
4230 016644  042737  000040  172516            BIC     #BIT5,MMR3      ; DISABLE UNIBUS MAPPING
4231 016652  012700  001734                    MOV     #WRTBUF,R0      ; GET POINTER TO WRITE BUFFER
4232 016656  012701  000010                    MOV     #8.,R1          ; SET UP LOOP COUNTER
4233 016662  022027  125252          15$:      CMP     (R0)+,#125252   ; . SEE IF TRANSFER OCCURED CORRECTLY ?
4234 016666  001401                            BEQ     20$             ; . YES,THEN SKIP ERROR MESSAGE
4235 016670  104032                            ERROR   +32             ; . NO,THEN ERROR IN TRANSFER
4236 016672  077105                  20$:      SOB     R1,15$          ; . HAVE WE CHECKED ALL THE TRANSFERS
4237
4238
```

TEST - UNIBUS DEVICE DATI CYCLE WITH RELOCATION ENABLED

```
                                .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
                                ;;*************************************************************************
    016674  G00004              TST52:  SCOPE

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

    016702  005077  163246              CLR     @BE1BA          ; POINT UBE TO THE FIRST UNIBUS MAP
    016706  005077  163250              CLR     @BE1CR2         ; REGISTER
    016712  005077  163232              CLR     @BE1DB          ; INITIALIZE THE DATA BUFFER
    016716  012777  177777  163226      MOV     #-1,@BE1CC      ; SET FOR 1 TRANSFER
    016724  012737  001774  170200      MOV     #PTRN16,MAPL00  ; POINT MAP REGISTERS TO PATTERN TABLE
```

T52 UNIBUS DEVICE DATI CYCLE WITH RELOCATION ENABLED

```
4294 016732  005037  170202                    CLR     MAPH00          ;
4295 016736  052737  000040  172516            BIS     #BIT5,MMR3      ; ENABLE UNIBUS MAPPING
4296
4297                                ;
4298                                ; GO DO THE TRANSFER
4299                                ;
4300
4301 016744  012777  002041  163204            MOV     #2041,@BE1CR1   ; SET UBE FOR NPR-DATI-1 XFER
4302 016752  032777  000200  163176  5$:       BIT     #BIT7,@BE1CR1   ; IS THE TRANSFER COMPLETE
4303 016760  001774                            BEQ     5$              ; NO,THEN WAIT FOR IT TO BE COMPLETE
4304 016762  042737  000040  172516            BIC     #BIT5,@#MMR3    ; YES,THEN DISABLE UNIBUS MAPPING
4305
4306                                ;
4307                                ; CHECK THE TRANSFER
4308                                ;
4309
4310 016770  027727  163154  000377            CMP     @BE1DB,#377     ; DID THE TRANSFER HAPPEN CORRECTLY ?
4311 016776  001401                            BEQ     TST53           ;; GO TO THE NEXT TEST
4312 017000  104032                            ERROR   +32             ; TRANSFER DID NOT OCCUR CORRECTLY
4313
4314
4315
```

```
4317                            .SBTTL TEST - ALU TEST USING UBE
4318
4319                            ;* THIS TEST WILL CHECK THE FIRST 3 ALU'S EACH ONE AT A TIME. AT FIRST THE
4320                            ;* PATTERN THAT DOES NOT RESULT IN OVERFLOW WILL BE USED, THE SECOND PATTERN
4321                            ;* GENERATES OVERFLOW INTO THE NEXT ALU.
4322                            ;*
4323                            ;
4324                            ;  BGNTST
4325                            ;
4326                            ;       IN 18-BIT MODE DON'T RUN THIS TEST
4327                            ;       LET R1 := 1001 (PATTERN WITHOUT OVERFLOW FROM ALU)
4328                            ;       LET R2 := 0101 (MAP REGISTER PATTERN)
4329                            ;       LET R3 := @MAPL00
4330                            ;       LET R4 := 0110 (OVERFLOW)
4331                            ;       LET R5 := 1011
4332                            ;       DO FOR 3 PATTERNS
4333                            ;       . LET BE1BA := R1               ;POINT ADDRESS TO MAPX00
4334                            ;       . LET BE1CR2 := 0               ;
4335                            ;       . LET BE1DB := 0                ;INITIALIZE DATA BUFFER
4336                            ;       . LET BE1CC := -1               ;SET FOR 1 XFER
4337                            ;       . SET UBE FOR A NPR-DATI-1 XFER
4338                            ;*
4339                            ;*  SET UP THE MAP REGISTER FOR THE TRANSFER WITHOUT OVERFLOW FROM ALU
4340                            ;*
4341                            ;       . LET (R3) := R2                ;POINT UMR @0 TO DATA PATTERN
4342                            ;       . LET 2(R3) := 0
4343                            ;       . ENABLE UNIBUS MAPPING
4344                            ;*
4345                            ;*  GO DO THE TRANSFER
4346                            ;*
4347                            ;       . SET OFF THE TRANSFER
4348                            ;       . WAIT FOR TRANSFER TO COMPLETE
4349                            ;       . DISABLE UNIBUS MAPPING
4350                            ;*
4351                            ;*  CHECK IF THE TRANSFER OCCURED CORRECTLY
4352                            ;*
4353                            ;       . IF BE1DB NEQ (R1*R2) THEN
4354                            ;       . . ERROR DATI THROUGH UNIBUS MAP DID NOT EXECUTE CORRECTLY
4355                            ;       . ENDIF
4356                            ;*
4357                            ;* NOW CHECK CARRY OUT FROM ALU
4358                            ;*
4359                            ;       . LET BE1BA := R4               ;POINT ADDRESS TO MAPX00
4360                            ;       . LET BE1CR2 := 0               ;
4361                            ;       . LET BE1DB := 0                ;INITIALIZE DATA BUFFER
4362                            ;       . LET BE1CC := -1               ;SET FOR 1 XFER
4363                            ;       . SET UBE FOR A NPR-DATI-1 XFER
4364                            ;*
4365                            ;*  SET UP THE MAP REGISTER FOR THE TRANSFER
4366                            ;*
4367                            ;       . LET (R3) := R5                ;POINT UMR @0 TO DATA PATTERN
4368                            ;       . LET 2(R3) := 0
4369                            ;       . ENABLE UNIBUS MAPPING
4370                            ;*
4371                            ;*  GO DO THE TRANSFER
4372                            ;*
4373                            ;       . SET OFF THE TRANSFER
```

```
4374                                    ;       . WAIT FOR TRANSFER TO COMPLETE
4375                                    ;       . DISABLE UNIBUS MAPPING
4376                                    ;*
4377                                    ;*  CHECK IF THE TRANSFER OCCURED CORRECTLY
4378                                    ;*
4379                                    ;       . IF BE1DB NEQ (R4+R5) THEN
4380                                    ;       . . ERROR DATI THROUGH UNIBUS MAP DID NOT EXECUTE CORRECTLY
4381                                    ;       . ENDIF
4382                                    ;*
4383                                    ;* CHANGE THE PATTERN
4384                                    ;*
4385                                    ;       . SHIFT R1,R2,R4,R5 4 TIMES AND POINT TO MAPL03 AND MAPL06
4386                                    ;       . POINT R3 TO MAPL03 AND THEN TO MAPL06
4387                                    ;       ENDDO
4388                                    ;
4389                                    ;  ENDTST
4390                                    ;
4391                                    ;-------------------------------------------------------------------
4392
4393
4394                                    ;;*****************************************************************
                                        ;*TEST 53        ALU TEST USING UBE
                                        ;;*****************************************************************
     017002  000004                     TST53:  SCOPE
4395
4396 017004  032737  000040  177734             BIT     #BIT05,KMCR     ; 18 BIT MODE?
4397 017012  001402                             BEQ     10$             ; IF NOT, CONTINUE
4398 017014  000137  017432                     JMP     20$             ; OTHERWISE, EXIT
4399 017020  005037  172354             10$:    CLR     KIPAR6          ; FOR ERROR REPORTING
4400 017024  004737  002450                     JSR     PC,IUBE         ; INITIALIZE THE UBE
4401 017030  012700  000022                     MOV     #22,R0          ; INITIAL PATTERN FOR ADDRESS LINES
4402 017034  012701  000012                     MOV     #12,R1          ; INITIAL PATTERN FOR MAP REGISTER
4403 017040  012703  170200                     MOV     #MAPL00,R3      ; FIRST REGISTER PAIR TO BE USED
4404 017044  012704  000014                     MOV     #14,R4          ; PATTERN FOR THE OVERFLOW FOR A. LINES
4405 017050  012705  000026                     MOV     #26,R5          ; PATTERN FOR MAP PAIR
4406 017054  005037  001162                     CLR     $TMP1           ; LOCATION TO SELECT REGISTER PAIR
4407                                    ;
4408                                    ; SET UP UBE AND UNIBUS MAP REGISTERS FOR A DATI CYCLE
4409                                    ;
4410
4411 017060  010077  163070             1$:     MOV     R0,@BE1BA       ; POINT UBE TO THE PATTERN
4412 017064  005077  163072                     CLR     @BE1CR2         ; REGISTER
4413 017070  005077  163054                     CLR     @BE1DB          ; INITIALIZE THE DATA BUFFER
4414 017074  012777  177777  163050             MOV     #-1,@BE1CC      ; SET FOR 1 TRANSFER
4415 017102  010113                             MOV     R1,(R3)         ; POINT MAP REGISTERS TO PATTERN
4416 017104  005063  000002                     CLR     2(R3)           ; CLEAR HIGH MAP REGISTER
4417 017110  052737  000040  172516             BIS     #BIT5,MMR3      ; ENABLE UNIBUS MAPPING
4418
4419                                    ;
4420                                    ; GO DO THE TRANSFER
4421                                    ;
4422
4423 017116  012777  002041  163032             MOV     #2041,@BE1CR1   ; SET UBE FOR NPR-DATI-1 XFER
4424 017124  032777  000200  163024     2$:     BIT     #BIT7,@BE1CR1   ; IS THE TRANSFER COMPLETE
4425 017132  001774                             BEQ     2$              ; NO,THEN WAIT FOR IT TO BE COMPLETE
4426 017134  042737  000040  172516             BIC     #BIT5,@MMR3     ; YES,THEN DISABLE UNIBUS MAPPING
4427
```

```
                                        ;
                                        ; CHECK THE TRANSFER
                                        ;
017142  010037  001160                  MOV     R0,$TMP0        ; FIND ADDRESS ACCESSED
017146  042737  160000  001160          BIC     #160000,$TMP0   ; MAKE SURE THAT USING ALL 16 BITS FOR ADDRESS
017154  060137  001160                  ADD     R1,$TMP0        ;
017160  027777  162764  161772          CMP     @#E1DB,@$TMP0   ; DID THE TRANSFER HAPPEN CORRECTLY ?
017166  001412                          BEQ     3$              ; IF SO, BRANCH
017170  017737  162754  001126          MOV     @#E1DB,$BDDAT   ; WRONG DATA
017176  017737  161756  001124          MOV     @$TMP0,$GDDAT   ; EXPECTED DATA
017204  013737  001160  001122          MOV     $TMP0,$BDADR    ; ADDRESS USED
017212  104023                          ERROR   +23             ; TRANSFER DID NOT OCCUR CORRECTLY
                                        ;
                                        ; NOW CHECK THE SAME ALU FOR OVERFLOW
                                        ;
017214  010477  162734          3$:     MOV     R4,@#E1BA       ; POINT UBE TO THE PATTERN
017220  005077  162736                  CLR     @#E1CR2         ; REGISTER
017224  005077  162720                  CLR     @#E1DB          ; INITIALIZE THE DATA BUFFER
017230  012777  177777  162714          MOV     #-1,@#E1CC      ; SET FOR 1 TRANSFER
017236  010513                          MOV     R5,(R3)         ; POINT MAP REGISTERS TO PATTERN
017240  005063  000002                  CLR     2(R3)           ; CLEAR HIGH MAP REGISTER
017244  052737  000040  172516          BIS     #BIT5,@#MMR3    ; ENABLE UNIBUS MAPPING
                                        ;
                                        ; GO DO THE TRANSFER
                                        ;
017252  012777  002041  162676          MOV     #2041,@#E1CR1   ; SET UBE FOR NPR-DATI-1 XFER
017260  032777  000200  162670  4$:     BIT     #BIT7,@#E1CR1   ; IS THE TRANSFER COMPLETE
017266  001774                          BEQ     4$              ; NO,THEN WAIT FOR IT TO BE COMPLETE
017270  042737  000040  172516          BIC     #BIT5,@#MMR3    ; YES,THEN DISABLE UNIBUS MAPPING
                                        ;
                                        ; CHECK THE TRANSFER
                                        ;
017276  010437  001160                  MOV     R4,$TMP0        ; FIND ADDRESS ACCESSED
017302  042737  160000  001160          BIC     #160000,$TMP0   ; MAKE SURE THAT USING ALL 16 BITS FOR ADDRESS
017310  060537  001160                  ADD     R5,$TMP0        ;
017314  027777  162630  161636          CMP     @#E1DB,@$TMP0   ; DID THE TRANSFER HAPPEN CORRECTLY ?
017322  001412                          BEQ     5$              ; IF SO, BRANCH
017324  017737  162620  001126          MOV     @#E1DB,$BDDAT   ; WRONG DATA
017332  017737  161622  001124          MOV     @$TMP0,$GDDAT   ; EXPECTED DATA
017340  013737  001160  001122          MOV     $TMP0,$BDADR    ; ADDRESS USED
017346  104023                          ERROR   +23             ; TRANSFER DID NOT OCCUR CORRECTLY
                                        ;
                                        ; CHANGE THE PATTERN
                                        ;
017350  022737  002000  001720  5$:     CMP     #2000,PMIS      ; MORE THAN 32K OF PMI MEMORY?
017356  103025                          BHIS    TST54           ;; IF NOT, EXIT TEST
017360  072027  000004                  ASH     #4,R0           ; TO GET TO NEXT ALU
017364  072127  000004                  ASH     #4,R1           ; BY SHIFYING 4 TIMES
017370  072427  000004                  ASH     #4,R4           ;
017374  072527  000004                  ASH     #4,R5           ;
017400  062703  000014                  ADD     #12.,R3         ; GET NEXT REGISTER PAIR
017404  062737  060000  001162          ADD     #60000,$TMP1    ;
```

```
4485 017412  053700  001162              BIS     $TMP1,R0        ; SET TO SELECT NEXT REGISTER PAIR
4486 017416  053704  001162              BIS     $TMP1,R4        ;
4487 017422  022737  020000  001162      CMP     #20000,$TMP1    ; ALL 3 ALU'S DONE?
4488 017430  001213                      BNE     1$              ; IF NOT, BRANCH
4489 017432                        20$:
```

TEST - CARRY PROPOGATION TEST USING UBE

```
                                .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
                                ;;*************************************************************************
     017432  000004             TST54:  SCOPE

                                        CLR     KIPAR6          ; CLEAR FOR ERROR REPORTS
     017434  005037  172354             JSR     PC,IUBE         ; INITIALIZE THE UBE
     017440  004737  002450

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

     017444  012737  052525  000000     MOV     #52525,@#0      ; TEST LOCATION
     017452  012777  000002  162474     MOV     #2,@BE1BA       ; POINT UBE TO THE FIRST UNIBUS MAP
     017460  005077  162476             CLR     @BE1CR2         ; REGISTER
     017464  005077  162460             CLR     @BE1DB          ; INITIALIZE THE DATA BUFFER
     017470  012777  177777  162454     MOV     #-1,@BE1CC      ; SET FOR 1 TRANSFER
     017476  012737  177777  170200     MOV     #177777,MAPLO0  ; POINT MAP REGISTERS TO OVERFLOW
     017504  012737  000077  170202     MOV     #77,MAPHO0      ;
```

T54 CARRY PROPOGATION TEST USING UBE

```
4545 017512  052737  000040  172516          BIS     #BIT5,MMR3      ; ENABLE UNIBUS MAPPING
4546
4547                                 ;
4548                                 ; GO DO THE TRANSFER
4549                                 ;
4550
4551 017520  012777  002041  162430          MOV     #2041,@BE1CR1   ; SET UBE FOR NPR-DATI-1 XFER
4552 017526  032777  000200  162422  5$:     BIT     #BIT7,@BE1CR1   ; IS THE TRANSFER COMPLETE
4553 017534  001774                          BEQ     5$              ; NO,THEN WAIT FOR IT TO BE COMPLETE
4554 017536  042737  000040  172516          BIC     #BIT5,@#MMR3    ; YES,THEN DISABLE UNIBUS MAPPING
4555
4556                                 ;
4557                                 ; CHECK THE TRANSFER
4558                                 ;
4559
4560 017544  027727  162400  052525          CMP     @BE1DB,#52525   ; DID THE TRANSFER HAPPEN CORRECTLY ?
4561 017552  001411                          BEQ     TST55           ;; GO TO THE NEXT TEST
4562 017554  017737  162370  001126          MOV     @BE1DB,$BDDAT   ; WRONG DATA
4563 017562  012737  052525  001124          MOV     #52525,$GDDAT   ; EXPECTED DATA
4564 017570  005037  001122                  CLR     $BDADR          ; ADDRESS USED
4565 017574  104023                          ERROR   .23             ; TRANSFER DID NOT OCCUR CORRECTLY
4566
```

```
                                .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
                                ;;*****************************************************************
    017576  000004              TST55:  SCOPE
4604 017600  004737  002450             JSR     PC,IUBE         ; INITIALIZE THE UBE
4605
4606                            ;
4607                            ; SET UP UBE AND UNIBUS MAP REGISTERS FOR A DATI CYCLE
4608                            ;
4609 017604  012777  017734  162352     MOV     #10$,@BE1VEC    ; POINT INTERUPT VECTOR
4610 017612  012777  000340  162346     MOV     #340,@BE1PSW    ;
4611 017620  005077  162330             CLR     @BE1BA          ; POINT UBE TO THE FIRST UNIBUS MAP
4612 017624  005077  162332             CLR     @BE1CR2         ; REGISTER
4613 017630  005077  162314             CLR     @BE1DB          ; INITIALIZE THE DATA BUFFER
4614 017634  012777  177777  162310     MOV     #-1,@BE1CC      ; SET FOR 1 TRANSFER
4615 017642  012737  160000  170200     MOV     #160000,MAPL00  ; POINT MAP REGISTERS TO NXM 17760000
4616 017650  012737  000077  170202     MOV     #77,MAPH00      ;
4617 017656  052737  000040  172516     BIS     #BIT5,MMR3      ; ENABLE UNIBUS MAPPING
4618                            ;
4619                            ; GO DO THE TRANSFER
4620                            ;
4621 017664  012777  002041  162264     MOV     #2041,@BE1CR1   ; SET UBE FOR NPR-DATI-1 XFER
```

T55 NXM TEST USING UBE WITH RELOCATION ENABLED

```
 4622 017672  032777  000200  162256  5$:     BIT     @BIT7,@BE1CR1   ; IS THE TRANSFER COMPLETE
 4623 017700  001774                          BEQ     5$              ; NO,THEN WAIT FOR IT TO BE COMPLETE
 4624 017702  032777  000400  162252          BIT     @BIT08,@BE1CR2  ; NXM ?
 4625 017710  001001                          BNE     6$              ; IF SO, BRANCH
 4626 017712  104022                          ERROR   .22             ; NXM NOT SET
 4627 017714  106427  000140          6$:     MTPS    @140            ; ALLOW INTERRUPTS
 4628 017720  000240                          NOP
 4629 017722  000240                          NOP
 4630 017724  104022                          ERROR   .22             ; NXM WITHOUT INTERRUPT
 4631 017726  000404                          BR      12$             ;
 4632 017730  106427  000340                  MTPS    @340            ; RESTORE PRIORITY
 4633                                ;
 4634                                ; CHECK THE TRANSFER
 4635                                ;
 4636 017734  062706  000004          10$:    ADD     @4,SP           ; ADJUST STACK
 4637 017740  042737  000040  172516  12$:    BIC     @BIT5,@#MR3     ; YES,THEN DISABLE UNIBUS MAPPING
 4638
```

```
                                .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
                                ;;*********************************************************************
    017746  000004              TST56:  SCOPE
                                ;
                                ; SETUP MAPPING REGISTERS FOR INITIAL CONDITIONS
                                ;
                                ; SET UP UBE AND UNIBUS MAP REGISTERS FOR A DATI CYCLE
                                ;
```

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

```
4694 020060  010277  162070                  MOV     R2,@BE1BA       ; . . POINT UBE TO FIRST MAP REGISTER
4695 020064  005077  162072                  CLR     @BE1CR2         ; . . REGISTER
4696 020070  010377  162066                  MOV     R3,@BE1CR2      ; . . LOAD ADDRESS <17-16>
4697 020074  005077  162050                  CLR     @BE1DB          ; . . INITIALIZE THE DATA BUFFER
4698 020100  012777  177777  162044          MOV     #-1,@BE1CC      ; . . SET FOR 1 TRANSFER
4699 020106  005021                          CLR     (R1)+           ; . . CLEAR LOW MAP REGISTER
4700 020110  005021                          CLR     (R1)+           ; . . AND HIGH MAP TOO
4701 020112  012777  020200  162044          MOV     #10$,@BE1VEC    ; . . LOAD INTERRUPT VECTOR
4702 020120  012777  000340  162040          MOV     #340,@BE1PSW    ;
4703                                 ;
4704                                 ; GO DO THE TRANSFER
4705                                 ;
4706
4707 020126  012777  002041  162022          MOV     #2041,@BE1CR1   ; . . SET UBE FOR NPR-DATI-1 XFER
4708 020134  032777  000200  162014  5$:     BIT     #BIT7,@BE1CR1   ; . . IS THE TRANSFER COMPLETE
4709 020142  001774                          BEQ     5$              ; . . NO,THEN WAIT FOR IT TO BE COMPLETE
4710 020144  032777  000400  162010          BIT     #BIT08,@BE1CR2  ; . . NXM SET?
4711 020152  001002                          BNE     6$              ; . . IF SO, BRANCH
4712 020154  104022                          ERROR   +22             ; . . NO NXM
4713 020156  000412                          BR      11$             ;
4714 020160  106427  000140          6$:     MTPS    #140            ; . . LOWER PRIORITY
4715 020164  000240                          NOP                     ; . . WAIT FOR INTERRUPT
4716 020166  000240                          NOP                     ;
4717 020170  104022                          ERROR   +22             ; . . NO INTERRUPT ON NXM
4718 020172  106427  000340                  MTPS    #340            ; . . RAISE PRIORITY
4719 020176  000402                          BR      11$             ;
4720 020200  062706  000004          10$:    ADD     #4,SP           ; . . ADJUST STACK
4721 020204  062702  020000          11$:    ADD     #20000,R2       ; . . SELECT NEXT MAP PAIR REGISTER
4722 020210  103001                          BCC     12$             ; . . IF NO CARRY, BRANCH
4723 020212  005203                          INC     R3              ; . . OTHERWISE INCREMENT ADDRESS <17-16>
4724 020214  023701  001160          12$:    CMP     $TMP0,R1        ; . . ALL DONE?
4725 020220  101315                          BHI     2$              ; . . IF LESS THAN, BRANCH
4726                                 ;
4727                                 ; CHECK IF DONE WITH 22-BIT MODE AND 18-BIT
4728                                 ;
4729 020222  032737  000040  177734          BIT     #BIT05,KMCR     ; . 22-BIT MODE DONE?
4730 020230  001412                          BEQ     13$             ; . IF YES, BRANCH TO EXIT
4731 020232  052737  000400  177730          BIS     #BIT08,DCSR     ; . ENABLE DIAGN. MODE
4732 020240  012737  000027  177734          MOV     #27,KMCR        ; . 22-BIT, 32K PMI
4733 020246  042737  000400  177730          BIC     #BIT08,DCSR     ; . DISABLE DIAGN. MODE
4734 020254  000672                          BR      1$              ; . DO AGAIN IN 22-BIT MODE
4735                                 ;
4736                                 ; IF ANY UNIBUS MEMORY PRESENT, DO RELOCATION THRU IT
4737                                 ;
4738 020256  022737  170374  001160  13$:    CMP     @#MAPL37,$TMP0  ; ANY UNIBUS MEMORY?
4739 020264  001432                          BEQ     200$            ; IF NO, EXIT
4740 020266  012777  140000  161660          MOV     #140000,@BE1BA  ; TRY TO DO DATI THRU
4741 020274  012777  000003  161660          MOV     #3,@BE1CR2      ;  THRU MAP 36 REGISTER
4742 020302  005037  170370                  CLR     @#MAPL36        ; CLEAR MAPPING REGISTER
4743 020306  005037  170372                  CLR     @#MAPH36        ;  PAIR
4744 020312  012737  123456  000000          MOV     #123456,@#0     ; SET UP PATTERN AT 0
4745 020320  005077  161624                  CLR     @BE1DB          ; CLEAR DATA REGISTER
4746 020324  012777  002071  161624          MOV     #2071,@BE1CR1   ; START DATI
4747 020332  105777  161620          20$:    TSTB    @BE1CR1         ; WAIT TILL
4748 020336  100375                          BPL     20$             ;  DONE
4749 020340  022777  123456  161602          CMP     #123456,@BE1DB  ; DATA CAME FROM @#0?
4750 020346  001001                          BNE     200$            ; IF NOT, BRANCH
```

L9


```
 4751 020350  104033                        ERROR   +33             ; RELOCATED UNIBUS MEMORY
 4752 020352  052737  000400  177730  200$:  BIS     #BIT08,DCSR     ; ENABLE DIAGN. MODE
 4753 020360  013737  001162  177734         MOV     $TMP1,KMCR      ; RESTORE KMCR
 4754 020366  042737  000400  177730         BIC     #BIT08,DCSR     ; DISABLE DIAGN. MODE
```

```
                        .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
                        ;;*************************************************************
020374  000004          TST57:  SCOPE

020376  005737  001720          TST     PMIS                    ; ANY PMI MEMORY?
020402  001561          BEQ     TST60           ;; IF NONE, EXIT TEST
                        ;
                        ; ENABLE MMU AND SIZE MEMORY
                        ;
020404  004737  002244          JSR     PC,MAPPR        ; REMAP PROGRAM AREA
020410  052737  000001  177572  BIS     #BIT00,MMR0     ; ENABLE MMU
020416  052737  000020  172516  BIS     #BIT04,MMR3     ; ENABLE 22-BIT MODE
020424  013702  001720          MOV     PMIS,R2         ; STORE HIGHEST PAGE
020430  042737  000001  177572  BIC     #BIT00,MMR0     ; DISABLE MMU
                        ;
                        ; SETUP MAPPING REGISTERS AND KMCR
                        ;
020436  052737  000040  172516  BIS     #BIT5,@#MMR3    ; ENABLE UNIBUS MAPPING
020444  013737  177734  001160  MOV     KMCR,$TMP0      ; SAVE KMCR
020452  052737  000400  177730  BIS     #BIT08,DCSR     ; ENABLE DIAGNOSTIC MODE
020460  012737  000067  177734  MOV     #67,KMCR        ; 18-BIT, >32K DISABLED
020466  042737  000400  177730  BIC     #BIT08,DCSR     ; DISABLE DIAGNOSTIC MODE
020474  005037  170200          CLR     MAPL00          ; CLEAR LOW MAP REGISTER
020500  012737  000001  170202  MOV     #1,MAPH00       ; POINT TO 32K
020506  012777  020620  161450  MOV     #20$,@BE1VEC    ; . LOAD INTERRUPT VECTOR
020514  012777  000340  161444  MOV     #340,@BE1PSW    ;
020522  012737  002000  172354  MOV     #2000,KIPAR6    ; INITIAL POINTER TO ABOVE 32K
                        ;
                        ; DO DATI TO DISABLED MEMORY
                        ;
020530  004737  002450  10$:    JSR     PC,IUBE         ; INTIALISE UBE
020534  005077  161414          CLR     @BE1BA          ; . ADDRESS TO ACCESS THRU MAP 0
020540  012777  177777  161404  MOV     #-1,@BE1CC      ; . ONE CYCLE
020546  012777  002041  161402  MOV     #2041,@BE1CR1   ; . SET UBE FOR NPR-DATI-1 XFER
```

T57 MAIN MEMORY DISABLE THRU UBE

```
4810 020554  032777  000200  161374  14$:   BIT     @#BIT7,@#BE1CR1   ; . IS THE TRANSFER COMPLETE
4811 020562  001774                         BEQ     14$               ; . NO,THEN WAIT FOR IT TO BE COMPLETE
4812 020564  032777  000400  161370         BIT     @#BIT08,@#BE1CR2  ; . NXM SET?
4813 020572  001002                         BNE     15$               ; . IF SO, BRANCH
4814 020574  104022                         ERROR   .22               ; . NO NXM
4815 020576  000412                         BR      25$               ;
4816 020600  106427  000140         15$:    MTPS    @#140             ; . LOWER PRIORITY
4817 020604  000240                         NOP                       ; . WAIT FOR INTERRUPT
4818 020606  000240                         NOP                       ;
4819 020610  104022                         ERROR   .22               ; . NO INTERRUPT ON NXM
4820 020612  106427  000340                 MTPS    @#340             ; . RAISE PRIORITY
4821 020616  000402                         BR      25$               ; . BRANCH
4822 020620  062706  000004         20$:    ADD     @#4,SP            ; . ADJUST STACK
4823                                 ;
4824                                 ; CHANGE PAGE ACCESSED AND CHECK END CONDITIONS
4825                                 ;
4826 020624  023702  172354         25$:    CMP     KIPAR6,R2         ; . ALL PAGES DONE?
4827 020630  103032                         BHIS    30$               ; . IF YES, EXIT
4828 020632  062737  004000  170200         ADD     @#4000,MAPL00     ; . INCREMENT IN 1K
4829 020640  103002                         BCC     26$               ; . IF NO CARRY, BRANCH
4830 020642  005237  170202                 INC     MAPH00            ; . OTHERWISE INCR. HIGH MAP
4831 020646  032737  017777  170200  26$:   BIT     @#17777,MAPL00    ; . IS IT ON 4K BOUNDARY?
4832 020654  001017                         BNE     27$               ; . IF NOT, BRANCH
4833 020656  062737  000200  172354         ADD     @#200,KIPAR6      ; . INCREMENT COUNTER
4834 020664  052737  000400  177730         BIS     @#BIT08,DCSR      ; . ENABLE DIAG. MODE
4835 020672  005337  177734                 DEC     KMCR              ; . ACCESS NEXT HIGHER LOCATION
4836 020676  032737  000040  177734         BIT     @#BIT05,KMCR      ; . ALL 128K DONE?
4837 020704  001404                         BEQ     30$               ; . IF SO, BRANCH
4838 020706  042737  000400  177730         BIC     @#BIT08,DCSR      ; . DISABLE DIAG. MODE
4839 020714  000705                 27$:    BR      10$               ; . DO FOR THE NEXT PAGE
4840 020716  042737  000040  172516  30$:   BIC     @#BIT05,MMR3      ; DISABLE MAPPING
4841 020724  052737  000400  177730         BIS     @#BIT08,DCSR      ; ENABLE DIAG. MODE
4842 020732  013737  001160  177734         MOV     $TMP0,KMCR        ; RESTORE KMCR
4843 020740  042737  000400  177730         BIC     @#BIT08,DCSR      ; DISABLE DIAG. MODE
4844
```

```
4846                                    .SBTTL TEST - UNIBUS DEVICE DATOB CYCLE
4847
4848                                    ;* THIS TEST WILL SEE IF THE KTJ11-B MODULE CAN EXECUTE
4849                                    ;* A UNIBUS DEVICE DATOB .
4850                                    ;*
4851                                    ;
4852                                    ;  BGNTST
4853                                    ;
4854                                    ;*
4855                                    ;*  SET UP UBE FOR A DATOB CYCLE
4856                                    ;*
4857                                    ;       LET R1 := ADDRESS OF THE WRITE BUFFER
4858                                    ;       LET (R1) := 0                   ;CLEAR OUT THE LOCATION
4859                                    ;       LET BE1DB := 77777              ;DATA PATTERN IS 77777
4860                                    ;       LET BE1CC := -1                 ;SET FOR 1 DATA XFER
4861                                    ;       LET BE1BA := R1                 ;ADDRESS IS FIRST LOCATION IN WRITE BUFFER
4862                                    ;       SET UBE FOR NPR-DATOB-1 XFER
4863                                    ;       SET OFF THE TRANSFER
4864                                    ;       WAIT FOR THE TRANSFER TO COMPLETE
4865                                    ;*
4866                                    ;*  CHECK IF THE TRANSFER OCCURED CORRECTLY
4867                                    ;*
4868                                    ;       IF (R1) NEQ #377 THEN
4869                                    ;       . ERROR TRANSFER DID NOT EXECUTE CORRECTLY
4870                                    ;       ENDIF
4871                                    ;
4872                                    ;  ENDTST
4873                                    ;
4874                                    ;-------------------------------------------------------------------------
4875
4876                                    ;;**********************************************************************
                                        ;*TEST 60        UNIBUS DEVICE DATOB CYCLE TEST
                                        ;;**********************************************************************
    020746  000004                      TST60:  SCOPE
4877
4878 020750  004737  002450                    JSR     PC,IUBE         ; INITIALIZE THE UBE
4879 020754  012701  001734                    MOV     #WRTBUF,R1      ; GET POINTER TO WRITE BUFFER
4880 020760  005011                            CLR     (R1)            ; CLEAR OUT THE LOCATION
4881 020762  012777  077777  161160            MOV     #77777,@BE1DB   ; SET DATA BUFFER UP
4882 020770  012777  177777  161154            MOV     #-1,@BE1CC      ; SET FOR 1 DATA TRANSFER
4883 020776  010177  161152                    MOV     R1,@BE1BA       ; POINT UBE XFER TO WRITE BUFFER
4884                                    ;
4885                                    ; DO THE TRANSFER
4886                                    ;
4887 021002  012777  003441  161146            MOV     #3441,@BE1CR1   ; SET UP UBE FOR NPR-DATOB-1 XFER
4888 021010  032777  000200  161140    5$:     BIT     #BIT7,@BE1CR1   ; IS THE TRANSFER COMPLETE ?
4889 021016  001774                            BEQ     5$              ; NO,THEN WAIT FOR IT TO COMPLETE
4890                                    ;
4891                                    ; CHECK THE TRANSFER
4892                                    ;
4893 021020  021127  000377                    CMP     (R1),#377       ; WAS THE TRANSFER CORRECT ?
4894 021024  001401                            BEQ     TST61           ;; YES GO TO THE NEXT TEST
4895 021026  104033                            ERROR   .33             ; ERROR TRANSFER WAS INCORRECT
4896
4897
```

```
                                .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)+ := 052525
                                ;       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
                                ;;*****************************************************************
021030  000004                  TST61:  SCOPE

021032  004737  002450                  JSR     PC,IUBE         ; INITIALIZE THE UBE

                                ;
                                ; INITIALIZE WRITE BUFFER FOR DATIP CYCLE
                                ;

021036  012701  001734                  MOV     #WRTBUF,R1      ; GET POINTER TO WRITE BUFFER
021042  012702  000010                  MOV     #8.,R2          ; SET UP LOOP COUNTER
021046  012721  052525          5$:     MOV     #052525,(R1)+   ; . INITIALIZE WRITE BUFFER LOCATION
021052  077203                          SOB     R2,5$           ; . HAVE WE DONE IT 8 TIMES ?

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

021054  012777  001734  161072          MOV     #WRTBUF,@BE1BA  ; SET UP UBE TO WRITE BUFFER
```

T61 UNIBUS DEVICE DATIP CYCLE TEST

```
 4953 021062  005077  161074                CLR     @BE1CR2          ;
 4954 021066  012777  177770  161056        MOV     #-8.,@BE1CC      ; SET UP FOR 8 TRANSFERS
 4955 021074  012777  002441  161054        MOV     #2441,@BE1CR1    ; SET UP UBE FOR NPR-DATIP-1 XFER
 4956 021102  032777  000200  161046  10$:   BIT     #BIT7,@BE1CR1    ; IS THE TRANSFER COMPLETE ?
 4957 021110  001774                        BEQ     10$              ; NO,THEN WAIT FOR IT TO BE COMPLETE
 4958
 4959                                ;
 4960                                ; CHECK IF DATIP WAS EXECUTED CORRECTLY
 4961                                ;
 4962
 4963 021112  012701  001734                MOV     #WRTBUF,R1       ; GET POINTER TO WRITE BUFFER
 4964 021116  012702  000010                MOV     #8.,R2           ; SET UP LOOP COUNTER
 4965 021122  022127  125252          15$:  CMP     (R1)+,#125252    ; . WAS THE TRANSFER CORRECT ?
 4966 021126  001401                        BEQ     20$              ; . YES,THEN GO SEE IF WE ARE DONE CHECKING
 4967 021130  104033                        ERROR   +33              ; . NO,THEN ERROR IN THE TRANSFER
 4968 021132  077205                20$:   SOB     R2,15$           ; . HAVE WE CHECKED ALL THE XFERS
 4969
 4970
```

TEST - UNIBUS DEVICE I/O PAGE READ CYCLE

```
4972                            .SBTTL TEST - UNIBUS DEVICE I/O PAGE READ CYCLE
4973
4974                            ;* THIS TEST  WILL SEE IF THE KTJ11-B MODULE CAN EXECUTE
4975                            ;* A UNIBUS DEVICE I/O PAGE READ CYCLE. THIS CONSISTS OF
4976                            ;* THREE DIFFERENT TYPES OF READS :
4977                            ;*
4978                            ;*      1. UNIBUS DEVICE PMI I/O PAGE READ CYCLE
4979                            ;*      2. UNIBUS DEVICE BOOT ROM READ CYCLE
4980                            ;*      3. UNIBUS DEVICE MAP REGISTER READ CYCLE.
4981                            ;*
4982                            ;* IT ALSO WILL CHECK THAT A UNIBUS DEVICE READ CYCLE NOT
4983                            ;* OF THE ABOVE THREE TYPES SHOULD CAUSE A TIMEOUT.
4984                            ;*
4985                            ;
4986                            ;  BGNTST
4987                            ;*
4988                            ;*  DO A UNIBUS DEVICE PMI I/O PAGE READ CYCLE
4989                            ;*
4990                            ;       LET BE1BA := ADDRESS OF CSR OF MEMORY BOARD
4991                            ;       LET BE1CC := -1                         ;SET FOR 1 TRANSFER
4992                            ;       SET UBE FOR A NPR-DATI-1 DATA XFER
4993                            ;       SET OFF THE TRANSFER
4994                            ;       WAIT FOR THE TRANSFER TO BE COMPLETE
4995                            ;       IF BE1DB NEQ CONTENTS OF MEMORY CSR THEN
4996                            ;       .  ERROR UNIBUS DEVICE PMI I/O PAGE READ CYCLE DID NOT EXECUTE CORRECTLY
4997                            ;       ENDIF
4998                            ;*
4999                            ;*  DO A UNIBUS DEVICE BOOT ROM READ CYCLE
5000                            ;*
5001                            ;       LET BE1BA := 773000
5002                            ;       LET BE1CC := -1                         ;SET FOR 1 TRANSFER
5003                            ;       SET UBE FOR A NPR-DATI-1 DATA XFER
5004                            ;       SET OFF THE TRANSFER
5005                            ;       WAIT FOR THE TRANSFER TO BE COMPLETE
5006                            ;       IF BE1DB NEQ @#173000 THEN
5007                            ;       .  ERROR UNIBUS DEVICE BOOT ROM READ CYCLE DID NOT EXECUTE CORRECTLY
5008                            ;       ENDIF
5009                            ;*
5010                            ;*  DO A UNIBUS DEVICE MAP REGISTER READ CYCLE
5011                            ;*
5012                            ;       LET BE1BA := 770200                     ;UNIBUS MAP REGISTER #0
5013                            ;       LET BE1CC := -1                         ;SET FOR 1 TRANSFER
5014                            ;       SET UBE FOR A NPR-DATI-1 DATA XFER
5015                            ;       SET OFF THE TRANSFER
5016                            ;       WAIT FOR THE TRANSFER TO BE COMPLETE
5017                            ;       IF BE1DB NEQ @#770200 THEN
5018                            ;       .  ERROR UNIBUS DEVICE MAP REGISTER READ CYCLE DID NOT EXECUTE CORRECTLY
5019                            ;       ENDIF
5020                            ;*
5021                            ;*    DO AN ILLEGAL UNIBUS DEVICE I/O PAGE READ CYCLE
5022                            ;*
5023                            ;       LET BE1BA := 777730,777732,777734       ;DCSR,DDR,KMCR ADDRESS
5024                            ;       LET BE1CC := -1                         ;1 DATA TRANSFER
5025                            ;       SET UBE FOR NPR-DATI-1 DATA XFER
5026                            ;       SET OFF THE TRANSFER
5027                            ;       WAIT FOR THE TRANSFER TO BE COMPLETE
5028                            ;       IF TRANSFER DID NOT TIMEOUT THEN
```

```
5029                                    ;       . ERROR AN ILLEGAL UNIBUS DEVICE CYCLE HAS OCCURED
5030                                    ;       ENDIF
5031                                    ;
5032                                    ;  ENDTST
5033                                    ;
5034                                    ;-------------------------------------------------------------------
5035
5036                                    ;;*****************************************************************
                                        ;*TEST 62       UNIBUS DEVICE I/O PAGE READ CYCLE
                                        ;;*****************************************************************
     021134  000004                     TST62:  SCOPE
5037
5038 021136  004737  002450                     JSR     PC,IUBE         ; INITIALIZE THE UBE
5039
5040                                    ;
5041                                    ; DO A UNIBUS DEVICE PMI I/O PAGE CYCLE
5042                                    ;
5043 021142  013777  001732  161004             MOV     MCSR,@#BE1BA    ; GET ADDRESS OF MEMORY CSR ADDRESS
5044 021150  012777  000003  161004             MOV     #3,@#BE1CR2     ; SET THE UPPER TWO ADDRESS BITS
5045 021156  012777  177777  160766             MOV     #-1,@#BE1CC     ; SET FOR 1 TRANSFER
5046 021164  012777  002041  160764             MOV     #2041,@#BE1CR1  ; SET UBE FOR NPR-DATI-1 XFER
5047 021172  032777  000200  160756  5$:        BIT     #BIT7,@#BE1CR1  ; ARE WE DONE THE TRANSFER ?
5048 021200  001774                             BEQ     5$              ; NO,THEN FOR WAIT FOR IT TO
5049                                    ;
5050                                    ; CHECK OUT THE TRANSFER
5051                                    ;
5052 021202  027777  160524  160740             CMP     @MCSR,@#BE1DB   ; WAS THE TRANSFER CORRECT
5053 021210  001401                             BEQ     10$             ; YES,THEN GO DO BOOT ROM READ
5054 021212  104033                             ERROR   .33             ; NO,THEN ERROR IN READ CYCLE
5055                                    ;
5056                                    ; DO A UNIBUS DEVICE BOOT ROM READ CYCLE
5057                                    ;
5058 021214  012777  173000  160732  10$:        MOV     #173000,@#BE1BA ; GET ADDRESS OF OF THE BOOT ROM
5059 021222  012777  000003  160732             MOV     #3,@#BE1CR2     ; SET THE UPPER TWO ADDRESS BITS
5060 021230  012777  177777  160714             MOV     #-1,@#BE1CC     ; SET FOR 1 TRANSFER
5061 021236  012777  002041  160712             MOV     #2041,@#BE1CR1  ; SET UBE FOR NPR-DATI-1 XFER
5062 021244  032777  000200  160704  15$:        BIT     #BIT7,@#BE1CR1  ; ARE WE DONE THE TRANSFER ?
5063 021252  001774                             BEQ     15$             ; NO,THEN FOR WAIT FOR IT TO
5064                                    ;
5065                                    ; CHECK OUT THE TRANSFER
5066                                    ;
5067 021254  052737  000200  177520             BIS     #BIT07,BCSR     ; SELECT RIGHT ROM
5068 021262  023777  173000  160660             CMP     @#173000,@#BE1DB ; WAS THE TRANSFER CORRECT
5069 021270  001401                             BEQ     20$             ; YES,THEN GO DO UNIBUS DEVICE MAP REGISTER READ
5070 021272  104033                             ERROR   .33             ; NO,THEN ERROR IN READ CYCLE
5071                                    ;
5072                                    ; DO A UNIBUS DEVICE MAP REGISTER READ
5073                                    ;
5074 021274  012777  170200  160652  20$:        MOV     #170200,@#BE1BA ; GET ADDRESS OF THE MAP REGISTER
5075 021302  012777  000003  160652             MOV     #3,@#BE1CR2     ; SET THE UPPER TWO ADDRESS BITS
5076 021310  012777  177777  160634             MOV     #-1,@#BE1CC     ; SET FOR 1 TRANSFER
5077 021316  012777  002041  160632             MOV     #2041,@#BE1CR1  ; SET UBE FOR NPR-DATI-1 XFER
5078 021324  032777  000200  160624  25$:        BIT     #BIT7,@#BE1CR1  ; ARE WE DONE THE TRANSFER ?
5079 021332  001774                             BEQ     25$             ; NO,THEN FOR WAIT FOR IT TO
5080                                    ;
5081                                    ; CHECK OUT THE TRANSFER
5082                                    ;
```

T62 UNIBUS DEVICE I/O PAGE READ CYCLE

```
5083 021334  023777  170200  160606          CMP     @#170200,@BE1DB ; WAS THE TRANSFER CORRECT
5084 021342  001401                          BEQ     30$             ; YES,THEN GO DO ILLEGAL UNIBUS DEVICE I/O READ
5085 021344  104033                          ERROR   .33             ; NO,THEN ERROR IN READ CYCLE
5086                                 ;
5087                                 ; DO AN ILLEGAL UNIBUS DEVICE I/O PAGE READ
5088                                 ;
5089 021346  012777  021450  160610  30$:     MOV     #40$,@BE1VEC    ; SET UP INTERRUPT VECTOR
5090 021354  012777  000340  160604          MOV     #340,@BE1PSW    ;
5091 021362  012701  000003                  MOV     #3,R1           ; DO FOR 3 REGISTERS
5092 021366  012702  177730                  MOV     #177730,R2      ; STARTING WITH DCSR (THEN DDR, KMCR)
5093 021372  010277  160556          31$:    MOV     R2,@BE1BA       ; GET ADDRESS OF THE DIAGNOSTIC REGISTER
5094 021376  012777  000003  160556          MOV     #3,@BE1CR2      ; SET THE UPPER TWO ADDRESS BITS
5095 021404  012777  177777  160540          MOV     #-1,@BE1CC      ; SET FOR 1 TRANSFER
5096 021412  012777  002041  160536          MOV     #2041,@BE1CR1   ; SET UBE FOR NPR-DATI-1 XFER
5097 021420  106427  000140                  MTPS    #140            ; LOWER PRIORITY
5098 021424  032777  000200  160524  35$:     BIT     #BIT7,@BE1CR1   ; ARE WE DONE THE TRANSFER ?
5099 021432  001774                          BEQ     35$             ; NO,THEN FOR WAIT FOR IT TO
5100 021434  000240                          NOP                     ; WAIT FOR INTERRUPT
5101 021436  000240                          NOP                     ;
5102 021440  104033                          ERROR   .33             ; ERROR KTJ11 RESPONDED TO AN ILLEGAL ADDRESS
5103 021442  106427  000340                  MTPS    #340            ; RAISE PRIORITY BACK
5104 021446  000407                          BR      45$             ;
5105 021450  062706  000004          40$:    ADD     #4,SP           ; CLEAN UP THE STACK AND THEN GO DO NEXT TEST
5106 021454  032777  000400  160500          BIT     #BIT08,@BE1CR2  ; NXM SET?
5107 021462  001001                          BNE     45$             ; IF SET, BRANCH
5108 021464  104033                          ERROR   .33             ; NXM NOT SET ON ILLEGAL REFERENCE
5109 021466  005722                  45$:    TST     (R2)+           ; GET ADDRESS OF NEXT REGISTER
5110 021470  004737  002450                  JSR     PC,IUBE         ; CLEAN UP UBE
5111 021474  077142                          SOB     R1,31$          ; DO FOR ALL 3 REGISTERS
5112
```

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

                        ;* THIS TEST  WILL SEE IF THE KTJ11-B MODULE CAN EXECUTE
                        ;* A UNIBUS DEVICE I/O PAGE WRITE CYCLE. THIS CONSISTS OF
                        ;* TWO DIFFERENT TYPES OF WRITES :
                        ;*
                        ;*      1. UNIBUS DEVICE PMI I/O PAGE WRITE CYCLE
                        ;*      2. UNIBUS DEVICE MAP REGISTER WRITE CYCLE.
                        ;*
                        ;* IT ALSO WILL CHECK THAT A UNIBUS DEVICE WRITE CYCLE NOT
                        ;* OF THE ABOVE TWO TYPES SHOULD CAUSE A TIMEOUT.
                        ;*
                        ;
                        ;  BGNTST
                        ;
                        ;*
                        ;*  DO A UNIBUS DEVICE PMI I/O PAGE WRITE CYCLE
                        ;*
                        ;       LET BE1DB := 52525
                        ;       LET BE1BA := ADDRESS OF CSR OF MEMORY BOARD
                        ;       LET BE1CC := -1                         ;SET FOR 1 TRANSFER
                        ;       SET UBE FOR A NPR-DATO-1 DATA XFER
                        ;       SET OFF THE TRANSFER
                        ;       WAIT FOR THE TRANSFER TO BE COMPLETE
                        ;       IF CONTENTS OF MEMORY CSR NEQ 52525 THEN
                        ;       .  ERROR UNIBUS DEVICE PMI I/O PAGE WRITE CYCLE DID NOT EXECUTE CORRECTLY
                        ;       ENDIF
                        ;*
                        ;*  DO A UNIBUS DEVICE MAP REGISTER WRITE CYCLE
                        ;*
                        ;       LET BE1DB := 52525                      ;DATA PATTERN
                        ;       LET BE1BA := 770200                     ;UNIBUS MAP REGISTER #0
                        ;       LET BE1CC := -1                         ;SET FOR 1 TRANSFER
                        ;       SET UBE FOR A NPR-DATO-1 DATA XFER
                        ;       SET OFF THE TRANSFER
                        ;       WAIT FOR THE TRANSFER TO BE COMPLETE
                        ;       IF MAP REGISTER 0 NEQ #52525 THEN
                        ;       .  ERROR UNIBUS DEVICE MAP REGISTER READ CYCLE DID NOT EXECUTE CORRECTLY
                        ;       ENDIF
                        ;*
                        ;*  DO A UNIBUS DEVICE TO BOOT ROM WRITE CYCLE
                        ;*
                        ;       LET BE1BA := 773000
                        ;       LET BE1CC := -1                         ;SET FOR 1 TRANSFER
                        ;       SET UBE FOR A NPR-DATO-1 DATA XFER
                        ;       SET OFF THE TRANSFER
                        ;       WAIT FOR THE TRANSFER TO BE COMPLETE
                        ;       IF NO TIMEOUT THEN
                        ;       .  ERROR UNIBUS DEVICE BOOT ROM WRITE CYCLE DID NOT EXECUTE CORRECTLY
                        ;       ENDIF
                        ;*
                        ;*  DO AN ILLEGAL UNIBUS DEVICE I/O PAGE WRITE CYCLE
                        ;*
                        ;       LET BE1DB := 52525                      ;DATA PATTERN
                        ;       LET BE1BA := 777730,777732,777734       ;DCSR, DDR, KMCR ADDRESSES
                        ;       LET BE1CC := -1                         ;1 DATA TRANSFER
                        ;       SET UBE FOR NPR-DATO-1 DATA XFER
```

TEST - UNIBUS DEVICE I/O PAGE WRITE CYCLE

```
                                        ;       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
                                        ;;*****************************************************************
021476  000004                          TST63:  SCOPE

021500  004737  002450                          JSR     PC,IUBE         ; INITIALIZE THE UBE
                                        ;
                                        ; DO A UNIBUS DEVICE PMI I/O PAGE CYCLE
                                        ;
021504  017737  160222  001160                  MOV     @MCSR,$TMP0     ; STORE MEMORY CSR
021512  013777  001732  160434                  MOV     MCSR,@BE18A     ; GET ADDRESS OF MEMORY CSR
021520  012777  000003  160434                  MOV     #3,@BE1CR2      ; SET UPPER TWO ADDRESS BITS
021526  012777  040000  160414                  MOV     #BIT14,@BE1DB   ; GET WRITE PATTERN
021534  012777  177777  160410                  MOV     #-1,@BE1CC      ; SET FOR 1 XFER
021542  012777  003041  160406                  MOV     #3041,@BE1CR1   ; SET UBE FOR NPR-DATO-1 XFER
021550  032777  000200  160400  5$:     BIT     #BIT7,@BE1CR1   ; IS THE TRANSFER COMPLETE ?
021556  001774                          BEQ     5$              ; NO,THEN WAIT FOR IT TO COMPLETE
                                        ;
                                        ; CHECK THE TRANSFER
                                        ;
021560  032777  040000  160144                  BIT     #BIT14,@MCSR    ; WAS THE TRANSFER CORRECT ?
021566  001001                          BNE     10$             ; YES,THEN GO DO A MAP REGISTER WRITE
021570  104033                          ERROR   +33             ; NO,THEN ERROR IN THE TRANSFER
                                        ;
                                        ; DO A UNIBUS DEVICE MAP REGISTER WRITE CYCLE
                                        ;
021572  013777  001160  160132  10$:    MOV     $TMP0,@MCSR     ; RESTORE MEMORY CSR
021600  005037  170200                  CLR     MAPL00          ; CLEAR MAP REGISTER PAIR
021604  005037  170202                  CLR     MAPH00          ;
021610  012777  170200  160336          MOV     #MAPL00,@BE18A  ; GET ADDRESS OF MAP REGISTER
021616  012777  000003  160336          MOV     #3,@BE1CR2      ; SET UPPER TWO ADDRESS BITS
021624  012777  052525  160316          MOV     #52525,@BE1DB   ; GET WRITE PATTERN
021632  012777  177776  160312          MOV     #-2,@BE1CC      ; SET FOR 2 XFER
021640  012777  003041  160310          MOV     #3041,@BE1CR1   ; SET UBE FOR NPR-DATO-1 XFER
021646  032777  000200  160302  15$:    BIT     #BIT7,@BE1CR1   ; IS THE TRANSFER COMPLETE ?
021654  001774                          BEQ     15$             ; NO,THEN WAIT FOR IT TO COMPLETE
                                        ;
                                        ; CHECK THE TRANSFER
                                        ;
021656  023727  170200  052524                  CMP     MAPL00,#52524   ; WAS THE TRANSFER CORRECT ?
021664  001401                          BEQ     16$             ; YES,THEN GO DO A MAP REGISTER WRITE
021666  104033                          ERROR   +33             ; NO,THEN ERROR IN THE TRANSFER
021670  022737  000025  170202  16$:    CMP     #25,MAPH00      ; SECOND WORD OK TOO?
021676  001401                          BEQ     17$             ; IF SO, BRANCH
021700  104033                          ERROR   +33             ; NO,THEN ERROR IN THE TRANSFER
                                        ;
```

T63 UNIBUS DEVICE I/O PAGE WRITE CYCLE

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

```
                                .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
                                ;;*********************************************************************
022152  000004                  TST64:  SCOPE
022154  004737  002450                  JSR     PC,IUBE         ; CLEAN UP UBE
                                ;
                                ; DO DATO TO ALL 100 REGISTERS WITH 125252 AS PATTERN
                                ;
022160  012777  170200  157766          MOV     #MAPL00,@BE1BA  ; THE STARTING ADDRESS
022166  012777  177700  157756          MOV     #-100,@BE1CC    ; 80 WORDS TO WRITE TO
022174  012777  125252  157746          MOV     #125252,@BE1DB  ; THE PATTERN TO BE WRITTEN
022202  012777  000003  157752          MOV     #3,@BE1CR2      ; ADDRESS <17-16>
022210  012777  003041  157740          MOV     #3041,@BE1CR1   ; GO DO DATO'S
022216  105777  157734          1$:     TSTB    @BE1CR1         ; DONE BIT SET?
022222  100375                          BPL     1$              ; WAIT
                                ;
                                ; CHECK THAT ALL REGISTERS WRITTEN OK
                                ;
022224  012701  170200                  MOV     #MAPL00,R1      ; START WITH MAP REGISTER 1
022230  010177  157720                  MOV     R1,@BE1BA       ; STARTING ADDRESS
022234  012777  177777  157710  2$:     MOV     #-1,@BE1CC      ; . DO JUST 1 CYCLE
022242  012777  002041  157706          MOV     #2041,@BE1CR1   ; . DATI FROM LOW MAP REGISTER
022250  105777  157702          3$:     TSTB    @BE1CR1         ; . DONE BIT SET?
022254  100375                          BPL     3$              ; . IF NOT, WAIT
022256  022777  125252  157664          CMP     #125252,@BE1DB  ; . READ OK?
022264  001401                          BEQ     4$              ; . IF SO, BRANCH
022266  104033                          ERROR   +33             ; . ERROR IN WRITING AND READING MAP REGISTERS
022270  012777  177777  157654  4$:     MOV     #-1,@BE1CC      ; . NOW DO HIGH MAP REGISTER
022276  012777  002041  157652          MOV     #2041,@BE1CR1   ; . DATI FROM HIGH REGISTER
022304  105777  157646          5$:     TSTB    @BE1CR1         ; . DONE BIT SET?
022310  100375                          BPL     5$              ; . IF NOT, WAIT
022312  022777  000052  157630          CMP     #52,@BE1DB      ; . READ OK?
022320  001401                          BEQ     6$              ; . IF SO, BRANCH
022322  104033                          ERROR   +33             ; . ERROR IN WRITING AND READING MAP REGISTERS
022324  005021          6$:     CLR     (R1)+           ; . CLEAR REGISTER JUST WRITTEN
022326  005021                  CLR     (R1)+           ;
022330  022701  170376                  CMP     #MAPH37,R1      ; . ALL DONE?
022334  101337                          BHI     2$              ; . IF NOT, BRANCH
```

TEST - UNIBUS DEVICE DATI CYCLE WITH CACHE ENABLED

```
5327                            .SBTTL TEST - UNIBUS DEVICE DATI CYCLE WITH CACHE ENABLED
5328
5329                            ;* THIS TEST WILL SEE IF THE CACHE OPERATES CORRECTLY
5330                            ;* WHEN A UNIBUS DEVICE READ CYCLE IS DONE
5331                            ;*
5332                            ;
5333                            ;  BGNTST
5334                            ;
5335                            ;       ENABLE THE CACHE
5336                            ;       ENABLE UNIBUS MAPPING
5337                            ;*
5338                            ;*  CHECK INITIAL SETTING OF THE CACHE VALID BITS
5339                            ;*
5340                            ;       SELECT THE CACHE VALID BITS
5341                            ;       IF KMCR <15-9> NEQ ^B0000000 THEN
5342                            ;       . ERROR IN THE CACHE
5343                            ;       ENDIF
5344                            ;*
5345                            ;*  CHECK INITIAL SETTING OF THE CACHE AVAILABILITY BITS
5346                            ;*
5347                            ;       SELECT THE CACHE AVAILABLE SET BITS
5348                            ;       IF KMCR <14-9> NEQ ^B111111 THEN
5349                            ;       . ERROR IN THE CACHE
5350                            ;       ENDIF
5351                            ;*
5352                            ;*  GO DO TWO CONSECUTIVE READS
5353                            ;*  THE FIRST WILL ALLOCATE SET A
5354                            ;*  THE SECOND WILL CAUSE A CACHE HIT IN SET A
5355                            ;*
5356                            ;       LET MAP REGISTER 1 POINT TO LOCATION 2000
5357                            ;       LET BE1BA := 20000                      ;OCTAL BOUNDARY READ
5358                            ;       LET BE1CR2 := 0                         ;CLEAR UPPER 2 ADDRESS BITS
5359                            ;       LET BE1CC := -2                         ;SET FOR TWO TRANSFERS
5360                            ;       SET UBE FOR NPR-DATI-1 XFER
5361                            ;       SET OFF THE TRANSFER
5362                            ;       WAIT FOR THE TRANSFER TO COMPLETE
5363                            ;       DISABLE UNIBUS MAPPING
5364                            ;       TURN OFF THE CACHE
5365                            ;       SELECT THE CACHE VALID BITS
5366                            ;*
5367                            ;*  CHECK IF THE CACHE OPERATED CORRECTLY
5368                            ;*
5369                            ;       IF KMCR <15-9> NEQ ^B1001000 THEN
5370                            ;       . ERROR CACHE DID NOT GET UPDATED CORRECTLY
5371                            ;       ENDIF
5372                            ;       SELECT THE CACHE AVAILBILITY BITS
5373                            ;       IF KMCR <15-9> NEQ ^B1000111 THEN
5374                            ;       . ERROR CACHE DID NOT OPERATE CORRECTLY
5375                            ;       ENDIF
5376                            ;
5377                            ;  ENDTST
5378                            ;
5379                            ;-----------------------------------------------
5380
5381                            ;;*****************************************************************
                                ;*TEST 65       UNIBUS DEVICE CYCLE WITH CACHE ENABLED
                                ;;*****************************************************************
```

T65 UNIBUS DEVICE CYCLE WITH CACHE ENABLED

```
                022336  000004                  TST65:  SCOPE
 5382
 5383 022340  004737  002450                          JSR     PC,IUBE         ; INITIALIZE THE UBE
 5384 022344  052737  000100  177734                  BIS     #BIT6,KMCR      ; ENABLE THE DMA CACHE
 5385 022352  032737  000100  177734                  BIT     #BIT6,KMCR      ; IS THE CACHE PRESENT ?
 5386 022360  001002                                  BNE     1$              ; IF YES, CONTINUE WITH TEST
 5387 022362  000137  022766                          JMP     $EOP            ; OTHERWISE, DO END OF PASS
 5388 022366  012737  160000  170374  1$:             MOV     #160000,MAPL37  ; INITIALIZE MAP REGISTERS
 5389 022374  012737  000077  170376                  MOV     #77,MAPH37      ;
 5390 022402  005037  170200                          CLR     MAPL00          ;
 5391 022406  005037  170202                          CLR     MAPH00          ;
 5392 022412  052737  000040  172516                  BIS     #BIT5,MMR3      ; ENABLE UNIBUS MAPPING
 5393 022420  052737  000400  177730                  BIS     #BIT08,DCSR     ; ENABLE DIAGNOSTIC MODE
 5394 022426  042737  000400  177734                  BIC     #BIT08,KMCR     ; MAKE SURE THAT VALID BITS SET
 5395 022434  042737  000400  177730                  BIC     #BIT08,DCSR     ; DISABLE DIAGNOSTIC MODE
 5396 022442                                  5$:
 5397                                         ;
 5398                                         ; GO DO TWO CONSECUTIVE READS STARTING AT A OCTAL BOUNDARY.
 5399                                         ; THE FIRST READ WILL ALLOCATE CACHE SET A. THE SECOND READ
 5400                                         ; WILL CAUSE A CACHE HIT IN SET A
 5401                                         ;
 5402 022442  012737  000200  170204  10$:     MOV     #200,MAPL01     ; POINT MAP REGISTER TO LOCATION 200
 5403 022450  005037  170206                          CLR     MAPH01
 5404 022454  012777  020000  157472                  MOV     #20000,@BE1BA   ; SET TRANSFER ADDRESS TO OCTAL BOUNDARY
 5405 022462  012777  000000  157472                  MOV     #0,@BE1CR2      ; CLEAR UPPER 2 ADDRESS BITS
 5406 022470  012777  177776  157454                  MOV     #-2,@BE1CC      ; SET UBE FOR TWO TRANSFERS
 5407                                         ;
 5408                                         ; GO DO THE TRANSFER
 5409                                         ;
 5410 022476  012777  002041  157452                  MOV     #2041,@BE1CR1   ; SET UBE FOR NPR-DATI-1 XFER
 5411 022504  032777  000200  157444  15$:     BIT     #BIT7,@BE1CR1   ; IS THE TRANSFER COMPLETE ?
 5412 022512  001774                                  BEQ     15$             ; NO,THEN WAIT FOR IT TO BE COMPLETE
 5413                                         ;
 5414                                         ; CHECK THE VALID BITS IN THE KMCR
 5415                                         ;
 5416 022514  013703  177734                  MOV     KMCR,R3         ; PUT KMCR INTO R3 FOR MASKING
 5417 022520  042703  000777                  BIC     #777,R3         ; MASK OUT BITS 0-8
 5418 022524  022703  110000                  CMP     #110000,R3      ; ARE THE BITS SET CORRECTLY ?
 5419 022530  001401                          BEQ     20$             ; YES,THEN GO CHECK THE AVAILIBILITY BITS
 5420 022532  104033                          ERROR   .33             ; NO,THEN ERROR IN DMA CACHE
 5421 022534  005037  172516          20$:    CLR     MMR3            ; DISABLE MAPPING
 5422 022540  042737  000100  177734                  BIC     #BIT06,KMCR     ; AND CACHE
 5423 022546  023777  000202  157374                  CMP     @#202,@BE1DB    ; DATA READ OK?
 5424 022554  001401                          BEQ     TST66           ;;IF OK, EXIT TEST
 5425 022556  104033                          ERROR   .33             ; OTHERWISE ERROR IN CACHE
 5426
```

```
                                .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
                                ;;*************************************************************************
     022560  000004             TST66:  SCOPE

5446 022562  004737  002450             JSR     PC,IUBE                 ; CLEAR UBE
5447 022566  052737  000005  172100     BIS     #BIT02!BIT00,@#172100   ; SET WRONG PARITY IN MEMORY CSR
5448 022574  042737  003740  172100     BIC     #3740,@#172100          ; CLEAR CHECK BITS
5449 022602  005037  000000             CLR     @#0                     ; WRITE LOCATION 0 WITH WRONG PARITY
5450 022606  042737  000004  172100     BIC     #BIT02,@#172100         ; CLEAR WRONG PARITY
5451 022614  052737  000040  172516     BIS     #BIT05,MMR3             ; ENABLE MEMORY MAPPING
5452 022622  042737  000100  177734     BIC     #BIT06,KMCR             ; DISABLE JUST TO MAKE SURE
5453 022630  052737  000100  177734     BIS     #BIT06,KMCR             ; ENABLE CACHE
5454 022636  012737  000000  170204     MOV     #0,MAPL01               ; POINT MAP REGISTER TO LOCATION 200
5455 022644  005037  170206             CLR     MAPH01
5456 022650  012777  020000  157276     MOV     #20000,@BE1BA           ; SET TRANSFER ADDRESS TO OCTAL BOUNDARY
5457 022656  012777  000000  157276     MOV     #0,@BE1CR2              ; CLEAR UPPER 2 ADDRESS BITS
5458 022664  012777  177777  157260     MOV     #-1,@BE1CC              ; SET UBE FOR TWO TRANSFERS
5459                                    ;
5460                                    ; GO DO THE TRANSFER
5461                                    ;
5462 022672  012777  002041  157256     MOV     #2041,@BE1CR1           ; SET UBE FOR NPR-DATI-1 XFER
5463 022700  032777  000200  157250     15$:    BIT     #BIT7,@BE1CR1   ; IS THE TRANSFER COMPLETE ?
5464 022706  001774                     BEQ     15$                     ; NO,THEN WAIT FOR IT TO BE COMPLETE
5465 022710  032737  177000  177734     BIT     #177000,KMCR            ; WAS ALLOCATED?
5466 022716  001401                     BEQ     16$                     ; IF NOT, BRANCH
5467 022720  104032                     ERROR   +32                     ; IF WAS, REPORT ERROR
5468 022722  005037  000000             16$:    CLR     @#0             ; REWRITE 0 WITH RIGHT PARITY
5469 022726  005037  172516             CLR     @#MMR3                  ; CLEAR MAPPING ENABLED
5470 022732  042737  000001  172100     BIC     #BIT00,@#172100
5471
5472
5473 022740                             UBEM:
                                ;;*************************************************************************
                                ;*TEST 67        DUMMY TEST
                                ;;*************************************************************************
     022740  000004             TST67:  SCOPE
5474 022742  022737  000001  001206     CMP     #1,$PASS                        ; SECOND PASS?
5475 022750  001006                     BNE     $EOP                            ; IF NOT, GOTO EOP
5476 022752  012737  000012  023014     MOV     #12,$EOPCT                      ; AFTER 1ST PASS
5477 022760  012737  000012  023022     MOV     #12,$ENDCT                      ; PRINT EVERY 10TH
```

```
                                        ;
                                        ; END OF PASS ROUTINE
                                        ;
                                        .SBTTL  END OF PASS ROUTINE
                                        ;;**********************************************************************
                                        ;*INCREMENT THE PASS NUMBER ($PASS)
                                        ;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
                                        ;*IF THERES A MONITOR GO TO IT
                                        ;*IF THERE ISN'T JUMP TO RESTART
022766                                  $EOP:
022766  000004                                SCOPE
022770  005037  001102                      CLR     $TSTNM          ;;ZERO THE TEST NUMBER
022774  005037  001164                      CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
023000  005237  001206                      INC     $PASS           ;;INCREMENT THE PASS NUMBER
023004  042737  100000  001206              BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
023012  005327                              DEC     (PC)+           ;;LOOP?
023014  000001                  $EOPCT: .WORD   1
023016  003055                          BGT     $DOAGN          ;;YES
023020  012737                          MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
023022  000001                  $ENDCT: .WORD   1
023024  023014                          $EOPCT
023026  005737  002574                  TST     UQUIET          ;NO MESSAGE IF RUNNING UFD QUIET MODE
023032  001037                          BNE     1$
023034  104401  023042                  TYPE    .65$            ;;TYPE ASCIZ STRING
023040  000407                          BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <12><15>/END PASS #/
023060                                  64$:
023060  013746  001206                  MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
                                                                ;;TYPE PASS NUMBER
023064  104405                          TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
023066  104401  023074                  TYPE    .67$            ;;TYPE ASCIZ STRING
023072  000412                          BR      66$             ;;GET OVER THE ASCIZ
                                ;;67$:  .ASCIZ  /     TOTAL ERRORS:/
023120                                  66$:
023120  013746  001112                  MOV     $ERTTL,-(SP)    ;;SAVE $ERTTL FOR TYPEOUT
                                                                ;;TOTAL NUMBER OF ERRORS
023124  104405                          TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
023126  104401  001175                  TYPE    .$CRLF          ;;TYPE CARRIAGE RETURN, LINE FEED
023132                                  1$:
023132  013700  000042          $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
023136  001405                          BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
023140  000005                          RESET                   ;;CLEAR THE WORLD
023142  004710                  $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
023144  000240                          NOP                     ;;SAVE ROOM
023146  000240                          NOP                     ;;FOP
023150  000240                          NOP                     ;;ACT11
023152                                  $DOAGN:
023152  000137                          JMP     @(PC)+          ;;RETURN
023154  003270                  $RTNAD: .WORD   RESTART
023156    377     377     000   $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
                                        .EVEN
                                        ;
                                        ; SYSMAC ROUTINES
                                        ;
```

END OF PASS ROUTINE

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

                                .SBTTL  SCOPE HANDLER ROUTINE
                                ;;**********************************************************************
                                ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
                                ;*AND LOAD THE TEST NUMBER(@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
023162                          @SCOPE:
023162  104407                          CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
023164  052737  001000  177520          BIS     #BIT09,@CSR     ;ENABLE
023172  032777  040000  155740  1@:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
023200  001117                          BNE     @OVER           ;;YES IF SW14=1
                                ;@@@@@START OF CODE FOR THE XOR TESTER@@@@@
023202  000416                  @XTSTR: BR      6@              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                                                ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
023204  013746  000004                  MOV     @@ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
023210  012737  023230  000004          MOV     #5@,@@ERRVEC    ;;SET FOR TIMEOUT
023216  005737  177060                  TST     @@177060        ;;TIME OUT ON XOR?
023222  012637  000004                  MOV     (SP)+,@@ERRVEC  ;;RESTORE THE ERROR VECTOR
023226  000466                          BR      @SVLAD          ;;GO TO THE NEXT TEST
023230  022626                  5@:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
023232  012637  000004                  MOV     (SP)+,@@ERRVEC  ;;RESTORE THE ERROR VECTOR
023236  000426                          BR      7@              ;;LOOP ON THE PRESENT TEST
023240                          6@:;@@@@@END OF CODE FOR THE XOR TESTER@@@@@
023240  032777  00040^  155672          BIT     @BIT08,@SWR     ;;LOOP ON SPEC. TEST?
023246  001407                          BEQ     2@              ;;BR IF NO
023250  017746  155664                  MOV     @SWR,-(SP)      ;;SET DESIRED TEST NUM. FROM SWR
023254  042716  000300                  BIC     #@SWMSK,(SP)    ;;STRIP AWAY UNDESIRED BITS
023260  122637  001102                  CMPB    (SP)+,@TSTNM    ;;ON THE RIGHT TEST?
023264  001465                          BEQ     @OVER           ;;BR IF YES
023266  105737  001103          2@:     TSTB    @ERFLG          ;;HAS AN ERROR OCCURRED?
023272  001421                          BEQ     3@              ;;BR IF NO
023274  123737  001115  001103          CMPB    @ERMAX,@ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
023302  101015                          BHI     3@              ;;BR IF NO
023304  032777  001000  155626          BIT     @BIT09,@SWR     ;;LOOP ON ERROR?
023312  001404                          BEQ     4@              ;;BR IF NO
023314  013737  001110  001106  7@:     MOV     @LPERR,@LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
023322  000446                          BR      @OVER
023324  105037  001103          4@:     CLRB    @ERFLG          ;;ZERO THE ERROR FLAG
023330  005037  001164                  CLR     @TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
023334  000415                          BR      1@              ;;ESCAPE TO THE NEXT TEST
023336  032777  004000  155574  3@:     BIT     @BIT11,@SWR     ;;INHIBIT ITERATIONS?
023344  001011                          BNE     1@              ;;BR IF YES
023346  005737  001206                  TST     @PASS           ;;IF FIRST PASS OF PROGRAM
023352  001406                          BEQ     1@              ;;      INHIBIT ITERATIONS
023354  005237  001104                  INC     @ICNT           ;;INCREMENT ITERATION COUNT
023360  023737  001164  001104          CMP     @TIMES,@ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
023366  002024                          BGE     @OVER           ;;BR IF MORE ITERATION REQUIRED
```

SCOPE HANDLER ROUTINE

```
023370  012737  000001  001104  1$:     MOV     #1,$ICNT          ;;REINITIALIZE THE ITERATION COUNTER
023376  013737  023462  001164          MOV     $MXCNT,$TIMES     ;;SET NUMBER OF ITERATIONS TO DO
023404  105237  001102          $SVLAD: INCB    $TSTNM            ;;COUNT TEST NUMBERS
023410  113737  001102  001204          MOVB    $TSTNM,$TESTN     ;;SET TEST NUMBER IN APT MAILBOX
023416  011637  001106                  MOV     (SP),$LPADR       ;;SAVE SCOPE LOOP ADDRESS
023422  011637  001110                  MOV     (SP),$LPERR       ;;SAVE ERROR LOOP ADDRESS
023426  005037  001166                  CLR     $ESCAPE           ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
023432  112737  000001  001115          MOVB    #1,$ERMAX         ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
023440  013777  001102  155474  $OVER:  MOV     $TSTNM,@DISPLAY   ;;DISPLAY TEST NUMBER
023446  013716  001106                  MOV     $LPADR,(SP)       ;;FUDGE RETURN ADDRESS
023452  042737  001000  177520          BIC     #BIT09,BCSR       ;DIS HOB
023460  000002                          RTI
023462  000001          $MXCNT: 1                                 ;;MAX. NUMBER OF ITERATIONS
  5493                  .SBTTL  TYPE ROUTINE
                        ;;*****************************************************************
                        ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
                        ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
                        ;*NOTE1:         $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
                        ;*NOTE2:         $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
                        ;*NOTE3:         $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
                        ;*
                        ;*CALL:
                        ;*1) USING A TRAP INSTRUCTION
                        ;*      TYPE    ,MESADR                   ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
                        ;*OR
                        ;*      TYPE
                        ;*      MESADR
                        ;*
023464  105737  001157  $TYPE:  TSTB    $TPFLG            ;;IS THERE A TERMINAL?
023470  100002                  BPL     1$                ;;BR IF YES
023472  000000                  HALT                      ;;HALT HERE IF NO TERMINAL
023474  000430                  BR      3$                ;;LEAVE
023476  010046          1$:     MOV     R0,-(SP)          ;;SAVE R0
023500  017600  000002          MOV     @2(SP),R0         ;;GET ADDRESS OF ASCIZ STRING
023504  122737  000001  001220  CMPB    @APTENV,$ENV      ;;RUNNING IN APT MODE
023512  001011                  BNE     62$               ;;NO,GO CHECK FOR APT CONSOLE
023514  132737  000100  001221  BITB    @APTSPOOL,$ENVM   ;;SPOOL MESSAGE TO APT
023522  001405                  BEQ     62$               ;;NO,GO CHECK FOR CONSOLE
023524  010037  023534          MOV     R0,61$            ;;SETUP MESSAGE ADDRESS FOR APT
023530  004737  026150          JSR     PC,$ATY3          ;;SPOOL MESSAGE TO APT
023534  000000          61$:    .WORD   0                 ;;MESSAGE ADDRESS
023536  132737  000040  001221  62$:    BITB    @APTCSUP,$ENVM    ;;APT CONSOLE SUPPRESSED
023544  001003                  BNE     60$               ;;YES,SKIP TYPE OUT
023546  112046          2$:     MOVB    (R0)+,-(SP)       ;;PUSH CHARACTER TO BE TYPED ONTO STACK
023550  001005                  BNE     4$                ;;BR IF IT ISN'T THE TERMINATOR
023552  005726                  TST     (SP)+             ;;IF TERMINATOR POP IT OFF THE STACK
023554  012600          60$:    MOV     (SP)+,R0          ;;RESTORE R0
023556  062716  000002  3$:     ADD     #2,(SP)           ;;ADJUST RETURN PC
023562  000002                  RTI                       ;;RETURN
023564  122716  000011  4$:     CMPB    #HT,(SP)          ;;BRANCH IF <HT>
023570  001430                  BEQ     8$
023572  122716  000200          CMPB    #CRLF,(SP)        ;;BRANCH IF NOT <CRLF>
023576  001006                  BNE     5$
023600  005726                  TST     (SP)+             ;;POP  <CR><LF> EQUIV
023602  104401                  TYPE                      ;;TYPE A CR AND LF
023604  001175                  $CRLF
023606  105037  024014          CLRB    $CHARCNT          ;;CLEAR CHARACTER COUNT
```

TYPE ROUTINE

```
023612  000755                          BR      2$              ;;GET NEXT CHARACTER
023614  004737  023676          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
023620  123726  001156          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
023624  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
023626  013746  001154                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                ;;AND THE NULL CHAR.
023632  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
023636  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
023640  004737  023676                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
023644  105337  024014                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
023650  000770                          BR      7$              ;;LOOP
                                ;HORIZONTAL TAB PROCESSOR
023652  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
023656  004737  023676          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
023662  132737  000007  024014          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
023670  001372                          BNE     9$              ;;TAB STOP
023672  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
023674  000724                          BR      2$              ;;GET NEXT CHARACTER
023676                          $TYPEC:
023676  105777  155242                  TSTB    @#TKS           ;;CHAR IN KYBD BUFFER?          ;MJD001
023702  100022                          BPL     10$             ;;BR IF NOT                     ;MJD001
023704  017746  155236                  MOV     @#TKB,-(SP)     ;;GET CHAR                      ;MJD001
023710  042716  177600                  BIC     #177600,(SP)    ;;STRIP EXTRANEOUS BITS         ;MJD001
023714  122716  000023                  CMPB    #$XOFF,(SP)     ;;WAS CHAR XOFF                 ;MJD001
023720  001012                          BNE     102$            ;;BR IF NOT                     ;MJD001
023722                          101$:                                                           ;MJD001
023722  105777  155216                  TSTB    @#TKS           ;;WAIT FOR CHAR                 ;MJD001
023726  100375                          BPL     101$                                            ;MJD001
023730  117716  155212                  MOVB    @#TKB,(SP)      ;;GET CHAR                      ;MJD001
023734  042716  177600                  BIC     #177600,(SP)    ;;STRIP IT                      ;MJD001
023740  122716  000021                  CMPB    #$XON,(SP)      ;;WAS IT XON?                   ;MJD001
023744  001366                          BNE     101$            ;;BR IF NOT                     ;MJD001
023746                          102$:                                                           ;MJD001
023746  005726                          TST     (SP)+           ;;FIX STACK                     ;MJD001
023750                          10$:                                                            ;MJD001
023750  105777  155174                  TSTB    @#TPS           ;;WAIT UNTIL PRINTER IS READY
023754  100375                          BPL     10$                                             ;MJD001
023756  116677  000002  155166          MOVB    2(SP),@#TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
023764  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
023772  001003                          BNE     1$              ;;BRANCH IF NO
023774  105037  024014                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
024000  000406                          BR      $TYPEX          ;;EXIT
024002  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
024010  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
024012  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
024014  000000                  $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
024016  000207                  $TYPEX: RTS     PC
5494                            .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
                                ;;**************************************************************
                                ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
                                ;*OCTAL (ASCII) NUMBER AND TYPE IT.
                                ;*$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
024020  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
024024  116637  000001  024243          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
024032  112637  024245                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
024036  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
024042  000406                          BR      $TYPON
024044  112737  000001  024243  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
024052  112737  000006  024245          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
024060  112737  000005  024242  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
024066  010346                          MOV     R3,-(SP)        ;;SAVE R3
024070  010446                          MOV     R4,-(SP)        ;;SAVE R4
024072  010546                          MOV     R5,-(SP)        ;;SAVE R5
024074  113704  024245                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
024100  005404                          NEG     R4
024102  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
024106  110437  024244                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
024112  113704  024243                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
024116  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
024122  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
024124  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
024126  000404                          BR      3$              ;;GO DO MSB
024130  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
024132  006105                          ROL     R5
024134  006105                          ROL     R5
024136  010503                          MOV     R5,R3
024140  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
024142  105337  024244                  DECB    $OMODE          ;;TYPE THIS DIGIT?
024146  100016                          BPL     7$              ;;BR IF NO
024150  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
024154  001002                          BNE     4$              ;;TEST FOR 0
024156  005704                          TST     R4              ;;SUPPRESS THIS 0?
024160  001403                          BEQ     5$              ;;BR IF YES
024162  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
024164  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
024170  052703  000040          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
024174  110337  024240                  MOVB    R3,8$           ;;SAVE FOR TYPING
024200  104401  024240                  TYPE    ,8$             ;;GO TYPE THIS DIGIT
024204  105337  024242          7$:     DECB    $OCNT           ;;COUNT BY 1
024210  003347                          BGT     2$              ;;BR IF MORE TO DO
024212  002402                          BLT     6$              ;;BR IF DONE
024214  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
024216  000744                          BR      2$              ;;GO DO THE LAST DIGIT
024220  012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
024222  012604                          MOV     (SP)+,R4        ;;RESTORE R4
024224  012603                          MOV     (SP)+,R3        ;;RESTORE R3
```

```
024226  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
024234  012616                          MOV     (SP)+,(SP)
024236  000002                          RTI                     ;;RETURN
024240     000                  8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
024241     000                          .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
024242     000                  $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
024243     000                  $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
024244  000000                  $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
  5495                          .SBTTL  COMVERT 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
024246                        $TYPDS:
024246  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
024250  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
024252  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
024254  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
024256  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
024260  012746  020200                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
024264  016605  000020                  MOV     20(SP),R5       ;;GET THE INPUT NUMBER
024270  100004                          BPL     1$              ;;BR IF INPUT IS POS.
024272  005405                          NEG     R5              ;;MAKE THE BINARY NUMBER POS.
024274  112766  000055  000001          MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
024302  005000                  1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
024304  012703  024462                  MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
024310  112723  000040                  MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
024314  005002                  2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
024316  016001  024452                  MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
024322  160105                  3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
024324  002402                          BLT     4$              ;;BR IF DONE
024326  005202                          INC     R2              ;;INCREASE THE BCD DIGIT BY 1
024330  000774                          BR      3$
024332  060105                  4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
024334  005702                          TST     R2              ;;CHECK IF BCD DIGIT=0
024336  001002                          BNE     5$              ;;FALL THROUGH IF 0
024340  105716                          TSTB    (SP)            ;;STILL DOING LEADING 0'S?
024342  100407                          BMI     7$              ;;BR IF YES
024344  106316                  5$:     ASLB    (SP)            ;;MSD?
024346  103003                          BCC     6$              ;;BR IF NO
024350  116663  000001  177777          MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
024356  052702  000060          6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
024362  052702  000040          7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
024366  110223                          MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
024370  005720                          TST     (R0)+           ;;JUST INCREMENTING
024372  020027  000010                  CMP     R0,#10          ;;CHECK THE TABLE INDEX
024376  002746                          BLT     2$              ;;GO DO THE NEXT DIGIT
024400  003002                          BGT     8$              ;;GO TO EXIT
024402  010502                          MOV     R5,R2           ;;GET THE LSD
024404  000764                          BR      6$              ;;GO CHANGE TO ASCII
024406  105726                  8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
024410  100003                          BPL     9$              ;;BR IF NO
```

CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```
        024412  116663  177777  177776          MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
        024420  105013                  9$:     CLRB    (R3)            ;;SET THE TERMINATOR
        024422  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
        024424  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
        024426  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
        024430  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
        024432  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
        024434  104401  024462                  TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
        024440  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
        024446  012616                          MOV     (SP)+,(SP)
        024450  000002                          RTI                     ;;RETURN TO USER
        024452  023420                  $DTBL:  10000.
        024454  001750                          1000.
        024456  000144                          100.
        024460  000012                          10.
        024462                          $DBLK:  .BLKW   4
   5496                                 .SBTTL  TTY INPUT ROUTINE
                                        ;;**********************************************************************
                                        .ENABL  LSB
                                        ;;**********************************************************************
                                        ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
                                        ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
                                        ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
                                        ;*WHEN OPERATING IN TTY FLAG MODE.
        024472  022737  000176  001140  $CKSWR: CMP     @#SWREG,SWR     ;;IS THE SOFT-SWR SELECTED?
        024500  001074                          BNE     15$             ;;BRANCH IF NO
        024502  105777  154436                  TSTB    @#TKS           ;;CHAR THERE?
        024506  100071                          BPL     15$             ;;IF NO, DON'T WAIT AROUND
        024510  117746  154432                  MOVB    @#TKB,-(SP)     ;;SAVE THE CHAR
        024514  042716  177600                  BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
        024520  022726  000007                  CMP     #7,(SP)+        ;;IS IT A CONTROL G?
        024524  001062                          BNE     15$             ;;NO, RETURN TO USER
        024526  123727  001134  000001          CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
        024534  001456                          BEQ     15$             ;;BRANCH IF YES
        024536  104401  025227                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
        024542  104401  025234          $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
        024546  013746  000176                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
        024552  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
        024554  104401  025245                  TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
        024560  005046                  19$:    CLR     -(SP)           ;;CLEAR COUNTER
        024562  005046                          CLR     -(SP)           ;;THE NEW SWR
        024564  105777  154354          7$:     TSTB    @#TKS           ;;CHAR THERE?
        024570  100375                          BPL     7$              ;;IF NOT TRY AGAIN
        024572  117746  154350                  MOVB    @#TKB,-(SP)     ;;PICK UP CHAR
        024576  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
        024602  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
        024606  001005                          BNE     10$             ;;BRANCH IF NOT
        024610  104401  025222                  TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
        024614  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
        024620  000757                          BR      19$             ;;LET'S TRY IT AGAIN
        024622  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
        024626  001022                          BNE     16$             ;;BRANCH IF NO
        024630  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
        024634  001403                          BEQ     11$             ;;BRANCH IF YES
        024636  016677  000002  154274          MOV     2(SP),@#SWR     ;;SAVE NEW SWR
        024644  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
        024650  104401  001175          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
```

TTY INPUT ROUTINE

```
024654  123727  001135  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
024662  001003                          BNE     15$             ;;BRANCH IF NOT
024664  012777  000100  154252          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
024672  000002                  15$:    RTI                     ;;RETURN
024674  004737  023676          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
024700  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
024704  002420                          BLT     18$             ;;BRANCH IF YES
024706  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
024712  003015                          BGT     18$             ;;BRANCH IF YES
024714  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
024720  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
024724  001403                          BEQ     17$             ;;BRANCH IF YES
024726  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
024730  006316                          ASL     (SP)            ;;   CHAR OVER TO MAKE
024732  006316                          ASL     (SP)            ;;   ROOM FOR NEW ONE.
024734  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
024740  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
024744  000707                          BR      7$              ;;GET THE NEXT ONE
024746  104401  001174          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
024752  000720                          BR      20$             ;;SIMULATE CONTROL-U
                                .DSABL  LSB
                                ;;**********************************************************************
                                ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
                                ;*CALL:
                                ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
                                ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
                                ;*                              ;;WITH PARITY BIT STRIPPED OFF
                                ;
024754  011646          $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
024756  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
024764  105777  154154          1$:     TSTB    @$TKS           ;;WAIT FOR
024770  100375                          BPL     1$              ;;A CHARACTER
024772  117766  154150  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
025000  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
025006  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
025014  001013                          BNE     3$              ;;BRANCH IF NO
025016  105777  154122          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
025022  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
025024  117746  154116                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
025030  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
025034  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
025040  001366                          BNE     2$              ;;IF NOT DISCARD IT
025042  000750                          BR      1$              ;;YES, RESUME
025044  026627  000004  000021  3$:     CMP     4(SP),#@$XON    ;;IS IT A RANDOM XON?                   ;RAN001
025052  001744                          BEQ     1$              ;;BRANCH IF YES                         ;RAN001
025054  026627  000004  000140          CMP     4(SP),#140      ;;IS IT UPPER CASE?
025062  002407                          BLT     4$              ;;BRANCH IF YES
025064  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
025072  003003                          BGT     4$              ;;BRANCH IF YES
025074  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
025102  000002                  4$:     RTI                     ;;GO BACK TO USER
                                ;;**********************************************************************
                                ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
                                ;*CALL:
                                ;*      RDLIN                   ;;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
```

```
025104  010346                  $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
025106  012703  025212          1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
025112  022703  025222          2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
025116  101405                          BLOS    4$              ;;BR IF YES
025120  104410                          RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
025122  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
025124  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
025130  001003                          BNE     3$              ;;SKIP IF NOT
025132  104401  001174          4$:     TYPE    ,$QUES          ;;TYPE A '?'
025136  000763                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
025140  111337  025210          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
025144  104401  025210                  TYPE    ,9$
025150  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
025154  001356                          BNE     2$              ;;LOOP IF NOT RETURN
025156  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
025162  104401  001176                  TYPE    ,$LF            ;;TYPE A LINE FEED
025166  012603                          MOV     (SP)+,R3        ;;RESTORE R3
025170  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
025172  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
025200  012766  025212  000004          MOV     #$TTYIN,4(SP)
025206  000002                          RTI                     ;;RETURN
025210     000                  9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
025211     000                          .BYTE   0               ;;TERMINATOR
025212                          $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
025222     136     125     015  $CNTLU: .ASCIZ  /↑U/<15><12>    ;;CONTROL "U"
025225     012     000
025227     136     107     015  $CNTLG: .ASCIZ  /↑G/<15><12>    ;;CONTROL "G"
025232     012     000
025234     015     012     123  $MSWR:  .ASCIZ  <15><12>/SWR = /
025237     127     122     040
025242     075     040     000
025245     040     040     116  $MNEW:  .ASCIZ  /  NEW = /
025250     105     127     040
025253     075     040     000
5497                            .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
                                ;;**********************************************************************
                                ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
                                ;*CHANGE IT TO BINARY.
                                ;*CALL:
                                ;*      RDOCT                   ;;READ AN OCTAL NUMBER
                                ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
                                ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT
025256  011646                  $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
025260  016666  000004  000002          MOV     4(SP),2(SP)     ;;INPUT NUMBER
025266  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
025270  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
025272  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
025274  104411                  1$:     RDLIN                   ;;READ AN ASCIZ LINE
025276  012600                          MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
025300  005001                          CLR     R1              ;;CLEAR DATA WORD
025302  005002                          CLR     R2
025304  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
025306  001412                          BEQ     3$              ;;IF ZERO GET OUT
025310  006301                          ASL     R1              ;;*2
025312  006102                          ROL     R2
025314  006301                          ASL     R1              ;;*4
025316  006102                          ROL     R2
```

READ AN OCTAL NUMBER FROM THE TTY

```
025320  006301                          ASL     R1              ;;*8
025322  006102                          ROL     R2
025324  042716  177770                  BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
025330  062601                          ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
025332  000764                          BR      2$              ;;LOOP
025334  005726                  3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
025336  010166  000012                  MOV     R1,12(SP)       ;;SAVE THE RESULT
025342  010237  025356                  MOV     R2,$HIOCT
025346  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
025350  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
025352  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
025354  000002                          RTI                     ;;RETURN
025356  000000                  $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
  5498                          .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
                                ;
025360  011646                  $RDDEC: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR
025362  016666  000004  000002          MOV     4(SP),2(SP)     ;;THE INPUT NUMBER
025370  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
025372  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
025374  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
025376  104411                  1$:     RDLIN                   ;;READ AN ASCIZ LINE
025400  012600                          MOV     (SP)+,R0        ;;ADDRESS OF 1ST CHAR.
025402  010037  025526                  MOV     R0,6$           ;;SAVE INCASE OF BAD INPUT
025406  005046                          CLR     -(SP)           ;;CLEAR DATA WORD
025410  005002                          CLR     R2              ;;SIGN SET POSITIVE
025412  122710  000055                  CMPB    #'-,(R0)        ;;SEE IF A MINUS SIGN WAS TYPED
025416  001001                          BNE     2$              ;;BR IF NO MINUS SIGN
025420  112002                          MOVB    (R0)+,R2        ;;SAVE FOR LATER USE
025422  112001                  2$:     MOVB    (R0)+,R1        ;;PICKUP THIS CHARACTER
025424  001424                          BEQ     3$              ;;GET OUT IF ZERO
025426  122701  000060                  CMPB    #'0,R1          ;;MAKE SURE THIS CHARACTER
025432  003032                          BGT     5$              ;;IS A DIGIT BETWEEN 0 & 9
025434  122701  000071                  CMPB    #'9,R1
025440  002427                          BLT     5$
025442  032716  170000                  BIT     #^C7777,(SP)    ;;DON'T LET NUMBER GET TO BIG
025446  001024                          BNE     5$              ;;BR IF NUMBER WOULD OVERFLOW
025450  006316                          ASL     (SP)            ;;*2
025452  011646                          MOV     (SP),-(SP)      ;;SAVE FOR LATER
025454  006316                          ASL     (SP)            ;;*4
025456  006316                          ASL     (SP)            ;;*8.
025460  062616                          ADD     (SP)+,(SP)      ;;*10.
025462  102416                          BVS     5$              ;;OVERFLOW ISN'T ALLOWED
025464  162701  000060                  SUB     #'0,R1          ;;STRIP AWAY THE ASCII JUNK
025470  060116                          ADD     R1,(SP)         ;;ADD IN THIS DIGIT
025472  102412                          BVS     5$              ;;OVERFLOW ISN'T ALLOWED
025474  000752                          BR      2$              ;;LOOP
025476  005702                  3$:     TST     R2              ;;CHECK IF NUMBER IS NEG
```

READ A DECIMAL NUMBER FROM THE TTY

```
025500  001401                          BEQ     4$              ;;BR IF NO
025502  005416                          NEG     (SP)            ;;YES--NEGATE THE NUMBER
025504  012666  000012          4$:     MOV     (SP)+,12(SP)    ;;SAVE THE RESULT
025510  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
025512  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
025514  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
025516  000002                          RTI                     ;;RETURN
025520  005726                  5$:     TST     (SP)+           ;;CLEAN PARTIAL NUMBER FROM STACK
025522  105010                          CLRB    (R0)            ;;SET A TERMINATOR
025524  104401                          TYPE                    ;;TYPE THE INPUT UP TO BAD CHAR.
025526  000000                  6$:     .WORD   0               ;;POINTER GOES HERE
025530  104401  001174                  TYPE    .$QUES          ;;"?" "CR" &"LF"
025534  000720                          BR      1$              ;;TRY AGAIN
 5499
 5500                           ;ENABLE MOB IN ERROR ROUTINE SINCE MOST LIKELY IT IS CURRENTLY DISABLED.
 5501
 5502                           .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
025536                          $ERROR:
025536  005737  002574          $ERROR: TST     UQUIET          ;;TEST FOR USER-QUIET MODE
025542  001403                          BEQ     9$              ;;BRANCH IF FIELD-SERVICE MODE
025544  005000                          CLR     R0              ;;IN CASE R0 HAS A #3 IN IT (^C)
025546  004737  025760                  JSR     PC,ABORT        ;;TEST FOR ABORT CONDITION
025552                          9$:
025552  104407                  9$:     CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
025554  052737  001000  177520          BIS     #BIT09,BCSR     ;ENABLE
025562  105237  001103          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
025566  001775                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
025570  013777  001102  153344          MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
025576  032777  002000  153334          BIT     #BIT10,@SWR     ;;BELL ON ERROR?
025604  001402                          BEQ     1$              ;;NO - SKIP
025606  104401  001170                  TYPE    .$BELL          ;;RING BELL
025612  005237  001112          1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
025616  011637  001116                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
025622  162737  000002  001116          SUB     #2,$ERRPC
025630  117737  153262  001114          MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
025636  032777  020000  153274          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
025644  001004                          BNE     20$             ;;SKIP TYPEOUTS
025646  004737  026120                  JSR     PC,ERTYPE       ;;GO TO USER ERROR ROUTINE
025652  104401  001175                  TYPE    .$CRLF
025656                          20$:
025656  122737  000001  001220  20$:    CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
025664  001007                          BNE     2$              ;;NO,SKIP APT ERROR REPORT
025666  113737  001114  025700          MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
025674  004737  026160                  JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
025700     000                  21$:    .BYTE   0
025701     000                          .BYTE   0
```

ERROR HANDLER ROUTINE

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

```
026166                          $ATYC:
026166  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
026170  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
026172  105737  026404                  TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
026176  001450                          BEQ     5$              ;;IF NOT:  BR
026200  122737  000001  001220          CMPB    @APTENV,$ENV    ;;OPERATING UNDER APT?
026206  001031                          BNE     3$              ;;IF NOT:  BR
026210  132737  000100  001221          BITB    @APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
026216  001425                          BEQ     3$              ;;IF NOT:  BR
026220  017600  000004                  MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
026224  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
026232  005737  001200          1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
026236  001375                          BNE     1$              ;;IF NOT:  WAIT
026240  010037  001214                  MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
026244  105720                  2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
026246  001376                          BNE     2$
026250  163700  001214                  SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
026254  006200                          ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
026256  010037  001216                  MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
026262  012737  000004  001200          MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
026270  000413                          BR      5$
026272  017637  000004  026316  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
026300  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
026306  013746  177776                  MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
026312  004737  023464                  JSR     PC,$TYPE        ;;CALL TYPE MACRO
026316  000000                  4$:     .WORD   0
026320                          5$:
026320  105737  026406          10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
026324  001416                          BEQ     12$             ;;IF NOT:  BR
026326  005737  001220                  TST     $ENV            ;;RUNNING UNDER APT?
026332  001413                          BEQ     12$             ;;IF NOT:  BR
026334  005737  001200          11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
026340  001375                          BNE     11$             ;;IF NOT:  WAIT
026342  017637  000004  001202          MOV     @4(SP),$FATAL   ;;GET ERROR #
026350  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
026356  005237  001200                  INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
026362  105037  026406          12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
026366  105037  026405                  CLRB    $LFLG           ;;CLEAR LOG FLAG
026372  105037  026404                  CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
026376  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
026400  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
026402  000207                          RTS     PC              ;;RETURN
026404     000          $MFLG:  .BYTE   0               ;;MESSG. FLAG
026405     000          $LFLG:  .BYTE   0               ;;LOG FLAG
026406     000          $FFLG:  .BYTE   0               ;;FATAL FLAG
                                .EVEN
            000200              APTSIZE=200
            000001              APTENV=001
            000100              APTSPOOL=100
            000040              APTCSUP=040
5509                            .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
                                ;;**************************************************************************
                                ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
                                ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
                                ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
026410                          $ERRTYP:
026410  104401  001175                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
```

ERROR MESSAGE TYPEOUT ROUTINE

```
026414  010046                          MOV     R0,-(SP)        ;;SAVE R0
026416  005000                          CLR     R0              ;;PICKUP THE ITEM INDEX
026420  153700  001114                  BISB    @#ITEMB,R0
026424  001004                          BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
                                                                ;;TYPE THE PC OF THE ERROR
026426  013746  001116                  MOV     #ERRPC,-(SP)    ;;SAVE #ERRPC FOR TYPEOUT
                                                                ;;ERROR ADDRESS
026432  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
026434  000426                          BR      6$              ;;GET OUT
026436  005300                  1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
026440  006300                          ASL     R0              ;;      WORK FOR THE ERROR TABLE
026442  006300                          ASL     R0
026444  006300                          ASL     R0
026446  062700  001324                  ADD     #ERRTB,R0       ;;FORM TABLE POINTER
026452  012037  026462                  MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
026456  001404                          BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
026460  104401                          TYPE                    ;;TYPE THE "ERROR MESSAGE"
026462  000000                  2$:     .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
026464  104401  001175                  TYPE    ,#CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
026470  012037  026500          3$:     MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
026474  001404                          BEQ     5$              ;;SKIP TYPEOUT IF 0
026476  104401                          TYPE                    ;;TYPE THE "DATA HEADER"
026500  000000                  4$:     .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
026502  104401  001175                  TYPE    ,#CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
026506  011000                  5$:     MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
026510  001004                          BNE     7$              ;;GO TYPE THE DATA
026512  012600                  6$:     MOV     (SP)+,R0        ;;RESTORE R0
026514  104401  001175                  TYPE    ,#CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
026520  000207                          RTS     PC              ;;RETURN
026522                          7$:
026522  013046                          MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
026524  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
026526  005710                          TST     (R0)            ;;IS THERE ANOTHER NUMBER?
026530  001770                          BEQ     6$              ;;BR IF NO
026532  104401  026540                  TYPE    ,8$             ;;TYPE TWO(2) SPACES
026536  000771                          BR      7$              ;;LOOP
026540     040     040     000  8$:     .ASCIZ  /  /            ;;TWO(2) SPACES
                                        .EVEN
5510                                    .SBTTL  TRAP DECODER
                                        ;;**************************************************************************
                                        ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
                                        ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
                                        ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
                                        ;*GO TO THAT ROUTINE.
026544  010046                  #TRAP:  MOV     R0,-(SP)        ;;SAVE R0
026546  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
026552  005740                          TST     -(R0)           ;;BACKUP BY 2
026554  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
026556  006300                          ASL     R0              ;;POSITION FOR INDEXING
026560  016000  026600                  MOV     #TRPAD(R0),R0   ;;INDEX TO TABLE
026564  000200                          RTS     R0              ;;GO TO ROUTINE
                                        ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
026566  011646                  #TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
026570  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
026576  000002                          RTI                     ;;RESTORE THE PSW
                                        .SBTTL  TRAP TABLE
                                        ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
```

```
                                ;*BY THE "TRAP" INSTRUCTION.
                                ;       ROUTINE
                                ;       -------
026600  026566                  $TRPAD: .WORD   $TRAP2
026602  023464                          $TYPE   ;;CALL=TYPE     TRAP*1(104401)  TTY TYPEOUT ROUTINE
026604  024044                          $TYPOC  ;;CALL=TYPOC    TRAP*2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
026606  024020                          $TYPOS  ;;CALL=TYPOS    TRAP*3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
026610  024060                          $TYPON  ;;CALL=TYPON    TRAP*4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
026612  024246                          $TYPDS  ;;CALL=TYPDS    TRAP*5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
026614  024542                          $GTSWR  ;;CALL=GTSWR    TRAP*6(104406)  GET SOFT-SWR SETTING
026616  024472                          $CKSWR  ;;CALL=CKSWR    TRAP*7(104407)  TEST FOR CHANGE IN SOFT-SWR
026620  024754                          $RDCHR  ;;CALL=RDCHR    TRAP*10(104410) TTY TYPEIN CHARACTER ROUTINE
026622  025104                          $RDLIN  ;;CALL=RDLIN    TRAP*11(104411) TTY TYPEIN STRING ROUTINE
026624  025256                          $RDOCT  ;;CALL=RDOCT    TRAP*12(104412) READ AN OCTAL NUMBER FROM TTY
026626  025360                          $RDDEC  ;;CALL=RDDEC    TRAP*13(104413) READ A DECIMAL NUMBER FROM TTY
5511                            .SBTTL  POWER DOWN AND UP ROUTINES
                                ;;**********************************************************************
                                ;POWER DOWN ROUTINE
026630  012737  026770  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
026636  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
026644  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
026646  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
026650  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
026652  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
026654  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
026656  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
026660  017746  152254                  MOV     @$WR,-(SP)      ;;PUSH @$WR ON STACK
026664  010637  026774                  MOV     SP,$SAVR6       ;;SAVE SP
026670  012737  026702  000024          MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
026676  000000                          HALT
026700  000776                          BR      .-2             ;;HANG UP
                                ;;**********************************************************************
                                ;POWER UP ROUTINE
026702  012737  026770  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
026710  013706  026774                  MOV     $SAVR6,SP       ;;GET SP
026714  005037  026774                  CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
026720  005237  026774          1$:     INC     $SAVR6          ;;WAIT FOR THE INC
026724  001375                          BNE     1$              ;;OF  WORD
026726  012677  152206                  MOV     (SP)+,@$WR      ;;POP STACK INTO @$WR
026732  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
026734  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
026736  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
026740  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
026742  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
026744  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
026746  012737  026630  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
026754  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
026762  104401                          TYPE                    ;REPORT THE POWER FAILURE
026764  026776          $PWRMG: .WORD   $POWER          ;;POWER FAIL MESSAGE POINTER
026766  000002                          RTI
026770  000000          $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
026772  000776                  BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
026774  000000          $SAVR6: 0                       ;;PUT THE SP HERE
026776    015     012     120   $POWER: .ASCIZ  <15><12>"POWER"
027001    117     127     105
027004    122     000

                                        .EVEN
```

POWER DOWN AND UP ROUTINES

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

```
5521 027236  124   110   105  EM5:   .ASCIZ  /THERE WAS NO DIFFERENCE FOUND BETWEEN HI AND LO MAP REGISTERS/
     027241  122   105   040
     027244  127   101   123
     027247  040   116   117
     027252  040   104   111
     027255  106   106   105
     027260  122   105   116
     027263  103   105   040
     027266  106   117   125
     027271  116   104   040
     027274  102   105   124
     027277  127   105   105
     027302  116   040   110
     027305  111   040   101
     027310  116   104   040
     027313  114   117   040
     027316  115   101   120
     027321  040   122   105
     027324  107   111   123
     027327  124   105   122
     027332  123   000
5522 027334  105   122   122  EM6:   .ASCIZ  /ERROR IN BITS 3-6,9-14 IN THE DCSR/
     027337  117   122   040
     027342  111   116   040
     027345  102   111   124
     027350  123   040   063
     027353  055   066   054
     027356  071   055   061
     027361  064   040   111
     027364  116   040   124
     027367  110   105   040
     027372  104   103   123
     027375  122   000
5523 027377  104   103   123  EM7:   .ASCIZ  /DCSR DID NOT RESPOND PROPORLY ON RESET/
     027402  122   040   104
     027405  111   104   040
     027410  116   117   124
     027413  040   122   105
     027416  123   120   117
     027421  116   104   040
     027424  120   122   117
     027427  120   117   122
     027432  114   131   040
     027435  117   116   040
     027440  122   105   123
     027443  105   124   000
5524 027446  124   111   115  EM10:  .ASCIZ  /TIMEOUT HAS OCCURED ON ACCESS TO THE DCSR/
     027451  105   117   125
     027454  124   040   110
     027457  101   123   040
     027462  117   103   103
     027465  125   122   105
     027470  104   040   117
     027473  116   040   101
     027476  103   103   105
     027501  123   123   040
     027504  124   117   040
```

POWER DOWN AND UP ROUTINES

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

```
                027762  116     040     103
                027765  131     103     114
                027770  105     000
   5530         027772  104     104     122     EM16:   .ASCIZ  /DDR IS NOT 0, WHEN DCRS<2-1> SELECTS UNIBUS LINES/
                027775  040     111     123
                030000  040     116     117
                030003  124     040     060
                030006  054     040     127
                030011  110     105     116
                030014  040     104     103
                030017  122     123     074
                030022  062     055     061
                030025  076     040     123
                030030  105     114     105
                030033  103     124     123
                030036  040     125     116
                030041  111     102     125
                030044  123     040     114
                030047  111     116     105
                030052  123     000
   5531         030054  104     103     123     EM17:   .ASCIZ  /DCSR<07> DOES NOT BECOME 1/
                030057  122     074     060
                030062  067     076     040
                030065  104     117     105
                030070  123     040     116
                030073  117     124     040
                030076  102     105     103
                030101  117     115     105
                030104  040     061     000
   5532         030107  111     116     104     EM20:   .ASCIZ  /INDIRECT ADDRESSING OF MAPPING REGISTER PAIRS/
                030112  111     122     105
                030115  103     124     040
                030120  101     104     104
                030123  122     105     123
                030126  123     111     116
                030131  107     040     117
                030134  106     040     115
                030137  101     120     120
                030142  111     116     107
                030145  040     122     105
                030150  107     111     123
                030153  124     105     122
                030156  040     120     101
                030161  111     122     123
                030164  000
   5533         030165  122     105     107     EM21:   .ASCIZ  /REGISTER PAIR 40 PERFORMS RELOCATION/
                030170  111     123     124
                030173  105     122     040
                030176  120     101     111
                030201  122     040     064
                030204  060     040     120
                030207  105     122     106
                030212  117     122     115
                030215  123     040     122
                030220  105     114     117
                030223  103     101     124
                030226  111     117     116
```

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

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

```
                030743  123  040  124
                030746  110  122  125
                030751  040  125  102
                030754  105  000
     5544       030756  105  122  122  EM34:   .ASCIZ  /ERROR IN THE UNIBUS MEMORY TEST/
                030761  117  122  040
                030764  111  116  040
                030767  124  110  105
                030772  040  125  116
                030775  111  102  125
                031000  123  040  115
                031003  105  115  117
                031006  122  131  040
                031011  124  105  123
                031014  124  000
     5545       031016  125  116  105  EM35:   .ASCIZ  /UNEXPECTED TIMEOUT HAS OCCURED/
                031021  130  120  105
                031024  103  124  105
                031027  104  040  124
                031032  111  115  105
                031035  117  125  124
                031040  040  110  101
                031043  123  040  117
                031046  103  103  125
                031051  122  105  104
                031054  000
     5546       031055  125  116  111  EM36:   .ASCIZ  /UNIBUS TIMEOUT DID NOT WORK CORRECTLY/
                031060  102  125  123
                031063  040  124  111
                031066  115  105  117
                031071  125  124  040
                031074  104  111  104
                031077  040  116  117
                031102  124  040  127
                031105  117  122  113
                031110  040  103  117
                031113  122  122  105
                031116  103  124  114
                031121  131  000
     5547       031123  104  103  123  EM37:   .ASCIZ  /DCSR<3> DID NOT DISABLE UBA ROM RESPONSE/
                031126  122  074  063
                031131  076  040  104
                031134  111  104  040
                031137  116  117  124
                031142  040  104  111
                031145  123  101  102
                031150  114  105  040
                031153  125  102  101
                031156  040  122  117
                031161  115  040  122
                031164  105  123  120
                031167  117  116  123
                031172  105  000
     5548
     5549       031174  124  105  123  DH1:    .ASCII  /TEST   ERROR   ERROR   ADDRESS/<12><15>
                031177  124  011  105
                031202  122  122  117
```

POWER DOWN AND UP ROUTINES

```
        031205  122  011  105
        031210  122  122  117
        031213  122  011  101
        031216  104  104  122
        031221  105  123  123
        031224  012  015
 5550   031226  040  040  043          .ASCIZ  /  #     PC      #/
        031231  011  040  040
        031234  120  103  011
        031237  040  040  043
        031242  000
 5551   031243  124  105  123  DH2:    .ASCII  /TEST   ERROR   ERROR   GOOD    BAD/<12><15>
        031246  124  011  105
        031251  122  122  117
        031254  122  011  105
        031257  122  122  117
        031262  122  011  107
        031265  117  117  104
        031270  011  102  101
        031273  104  012  015
 5552   031276  040  040  043          .ASCIZ  /  #     PC      #     DATA    DATA    ADDRESS/
        031301  011  040  040
        031304  120  103  011
        031307  040  040  043
        031312  011  104  101
        031315  124  101  011
        031320  104  101  124
        031323  101  011  101
        031326  104  104  122
        031331  105  123  123
        031334  000
 5553   031335  124  105  123  DH3:    .ASCII  /TEST   ERROR   ERROR   GOOD    BAD/<12><15>
        031340  124  011  105
        031343  122  122  117
        031346  122  011  105
        031351  122  122  117
        031354  122  011  107
        031357  117  117  104
        031362  011  102  101
        031365  104  012  015
 5554   031370  040  040  043          .ASCIZ  /  #     PC      #     ADDRESS ADDRESS/
        031373  011  040  040
        031376  120  103  011
        031401  040  040  043
        031404  011  101  104
        031407  104  122  105
        031412  123  123  011
        031415  101  104  104
        031420  122  105  123
        031423  123  000
 5555   031425  124  105  123  DH4:    .ASCII  /TEST   ERROR   ERROR   HI      LOW/<12><15>
        031430  124  011  105
        031433  122  122  117
        031436  122  011  105
        031441  122  122  117
        031444  122  011  110
        031447  111  011  114
```

POWER DOWN AND UP ROUTINES

```
        031452  117  127  012
        031455  015
  5556  031456  040  040  043          .ASCIZ  /  #     PC     #     MAP    MAP/
        031461  011  040  040
        031464  120  103  011
        031467  040  040  043
        031472  011  115  101
        031475  120  011  115
        031500  101  120  000
  5557  031503  124  105  123  DH5:    .ASCII  /TEST   ERROR  ERROR  GOOD   BAD/<12><15>
        031506  124  011  105
        031511  122  122  117
        031514  122  011  105
        031517  122  122  117
        031522  122  011  107
        031525  117  117  104
        031530  011  102  101
        031533  104  012  015
  5558  031536  040  040  043          .ASCIZ  /  #     PC     #     DATA   DATA/
        031541  011  040  040
        031544  120  103  011
        031547  040  040  043
        031552  011  104  101
        031555  124  101  011
        031560  104  101  124
        031563  101  000
  5559  031565  124  105  123  DH13:   .ASCII  /TEST   ERROR  ERROR  DDR    PATTERN/<12><15>
        031570  124  011  105
        031573  122  122  117
        031576  122  011  105
        031601  122  122  117
        031604  122  011  104
        031607  104  122  011
        031612  120  101  124
        031615  124  105  122
        031620  116  012  015
  5560  031623  040  040  043          .ASCIZ  /  #     PC     #/
        031626  011  040  040
        031631  120  103  011
        031634  040  040  043
        031637  000
  5561  031640  124  105  123  DH16:   .ASCII  /TEST   ERROR  ERROR/<12><15>
        031643  124  011  105
        031646  122  122  117
        031651  122  011  105
        031654  122  122  117
        031657  122  012  015
  5562  031662  040  040  043          .ASCIZ  /  #     PC     #/
        031665  011  040  040
        031670  120  103  011
        031673  040  040  043
        031676  000
  5563  031677  124  105  123  DH20:   .ASCII  /TEST   ERROR  ERROR  KMCR PAIR/<12><15>
        031702  124  011  105
        031705  122  122  117
        031710  122  011  105
        031713  122  122  117
```

```
                031716  122  011  113
                031721  115  103  122
                031724  040  120  101
                031727  111  122  012
                031732  015
 5564           031733  040  040  043          .ASCIZ  /  #     PC     #     FAILED/
                031736  011  040  040
                031741  120  103  011
                031744  040  040  043
                031747  011  106  101
                031752  111  114  105
                031755  104  000
 5565           031757  124  105  123  DH23:   .ASCII  /TEST   ERROR  ERROR  GOOD   BAD    ADDRESS/<12><15>
                031762  124  011  105
                031765  122  122  117
                031770  122  011  105
                031773  122  122  117
                031776  122  011  107
                032001  117  117  104
                032004  011  102  101
                032007  104  011  101
                032012  104  104  122
                032015  105  123  123
                032020  012  015
 5566           032022  040  040  043          .ASCIZ  /  #     PC     #     DATA   DATA   <22-16> <13-0>/
                032025  011  040  040
                032030  120  103  011
                032033  040  040  043
                032036  011  104  101
                032041  124  101  011
                032044  104  101  124
                032047  101  011  074
                032052  062  062  055
                032055  061  066  076
                032060  011  074  061
                032063  063  055  060
                032066  076  000
 5567           032070  124  105  123  DH27:   .ASCII  /TEST   ERROR  ERROR  ADDRESS ADDRESS/<12><15>
                032073  124  011  105
                032076  122  122  117
                032101  122  011  105
                032104  122  122  117
                032107  122  011  101
                032112  104  104  122
                032115  105  123  123
                032120  011  101  104
                032123  104  122  105
                032126  123  123  012
                032131  015
 5568           032132  040  040  043          .ASCIZ  /  #     PC     #     <22-16> <15-0>/
                032135  011  040  040
                032140  120  103  011
                032143  040  040  043
                032146  011  074  062
                032151  062  055  061
                032154  066  076  011
                032157  074  061  065
```

POWER DOWN AND UP ROUTINES

```
        032162  055     060     076
        032165  000
 5569                                   .EVEN
 5570 032166  001716  001116  001714  DT1:    .WORD   TEST,$ERRPC,ERRNUM,$BDADR,0
      032174  001122  000000
 5571 032200  001716  001116  001714  DT2:    .WORD   TEST,$ERRPC,ERRNUM,$GDDAT,$BDDAT,$BDADR,0
      032206  001124  001126  001122
      032214  000000
 5572 032216  001716  001116  001714  DT3:    .WORD   TEST,$ERRPC,ERRNUM,$GDADR,$BDADR,0
      032224  001120  001122  000000
 5573 032232  001716  001116  001714  DT4:    .WORD   TEST,$ERRPC,ERRNUM,$GDDAT,$BDDAT,0
      032240  001124  001126  000000
 5574 032246  001716  001116  001714  DT16:   .WORD   TEST,$ERRPC,ERRNUM,0
      032254  000000
 5575 032256  001716  001116  001714  DT20:   .WORD   TEST,$ERRPC,ERRNUM,KMCR,$BDADR,0
      032264  177734  001122  000000
 5576 032272  001716  001122  001714  DT21:   .WORD   TEST,$BDADR,ERRNUM,0
      032300  000000
 5577 032302  001716  001116  001714  DT23:   .WORD   TEST,$ERRPC,ERRNUM,$GDDAT,$BDDAT,KIPAR6,$BDADR,0
      032310  001124  001126  172354
      032316  001122  000000
 5578 032322  001716  001116  001714  DT27:   .WORD   TEST,$ERRPC,ERRNUM,MAPH01,MAPL01,0
      032330  170206  170204  000000
 5579 032336  001716  001116  001714  DT30:   .WORD   TEST,$ERRPC,ERRNUM,MAPL00,0
      032344  170200  000000
 5580         000001                     .END
```

SYMBOL TABLE

```
ABASE = 000000        BE1BA   002154        DH1     031174        ERROR = 104000        MAPH16= 170272
ABORT   025760        BE1CC   002152        DH13    031565        ERRVEC= 000004        MAPH17= 170276
ABORTC  026012        BE1CLR  002160        DH16    031640        ERTYPE  026120        MAPH2 = 170212
ABORTE  026056        BE1CR1  002156        DH2     031243        GTSWR = 104406        MAPH20= 170302
ABORTZ  026102        BE1CR2  002162        DH20    031677        HT    = 000011        MAPH21= 170306
ACDM1 = 000000        BE1DB   002150        DH23    031757        IOTVEC= 000020        MAPH22= 170312
ACDM2 = 000000        BE1INT  001724        DH27    032070        IUBE    002450        MAPH23= 170316
ACPUOP= 000000        BE1PSW  002166        DH3     031335        KDPAR0= 172360        MAPH24= 170322
ADDM0 = 000000        BE1SV   015476        DH4     031425        KDPAR1= 172362        MAPH25= 170326
ADDM1 = 000000        BE1VEC  002164        DH5     031503        KDPAR2= 172364        MAPH26= 170332
ADDM10= 000000        BE2BA   002174        DISPLA  001142        KDPAR3= 172366        MAPH27= 170336
ADDM11= 000000        BE2CC   002172        DISPRE  000174        KDPAR4= 172370        MAPH3 = 170216
ADDM12= 000000        BE2CLR  002200        DSWR  = 177570        KDPAR5= 172372        MAPH30= 170342
ADDM13= 000000        BE2CR1  002176        DT1     032166        KDPAR6= 172374        MAPH31= 170346
ADDM14= 000000        BE2CR2  002202        DT16    032246        KDPAR7= 172376        MAPH32= 170352
ADDM15= 000000        BE2DB   002170        DT2     032200        KDPDR0= 172320        MAPH33= 170356
ADDM2 = 000000        BE2INT  001726        DT20    032256        KDPDR1= 172322        MAPH34= 170362
ADDM3 = 000000        BE2PSW  002206        DT21    032272        KDPDR2= 172324        MAPH35= 170366
ADDM4 = 000000        BE2SV   015516        DT23    032302        KDPDR3= 172326        MAPH36= 170372
ADDM5 = 000000        BE2VEC  002204        DT27    032322        KDPDR4= 172330        MAPH37= 170376
ADDM6 = 000000        BIT0  = 000001        DT3     032216        KDPDR5= 172332        MAPH4 = 170222
ADDM7 = 000000        BIT00 = 000001        DT30    032336        KDPDR6= 172334        MAPH5 = 170226
ADDM8 = 000000        BIT01 = 000002        DT4     032232        KDPDR7= 172336        MAPH6 = 170232
ADDM9 = 000000        BIT02 = 000004        EMTSAV  002500        KIPAR0= 172340        MAPH7 = 170236
ADEVCT= 000000        BIT03 = 000010        EMTVEC= 000030        KIPAR1= 172342        MAPL0 = 170200
ADEVM = 000000        BIT04 = 000020        EM1     027006        KIPAR2= 172344        MAPL00= 170200
AENV  = 000000        BIT05 = 000040        EM10    027446        KIPAR3= 172346        MAPL01= 170204
AENVM = 000000        BIT06 = 000100        EM11    027520        KIPAR4= 172350        MAPL02= 170210
AFATAL= 000000        BIT07 = 000200        EM12    027572        KIPAR5= 172352        MAPL03= 170214
AMADR1= 000000        BIT08 = 000400        EM13    027644        KIPAR6= 172354        MAPL04= 170220
AMADR2= 000000        BIT09 = 001000        EM14    027713        KIPAR7= 172356        MAPL05= 170224
AMADR3= 000000        BIT1  = 000002        EM15    027743        KIPDR0= 172300        MAPL06= 170230
AMADR4= 000000        BIT10 = 002000        EM16    027772        KIPDR1= 172302        MAPL07= 170234
AMAMS1= 000000        BIT11 = 004000        EM17    030054        KIPDR2= 172304        MAPL1 = 170204
AMAMS2= 000000        BIT12 = 010000        EM2     027052        KIPDR3= 172306        MAPL10= 170240
AMAMS3= 000000        BIT13 = 020000        EM20    030107        KIPDR4= 172310        MAPL11= 170244
AMAMS4= 000000        BIT14 = 040000        EM21    030165        KIPDR5= 172312        MAPL12= 170250
AMSGAD= 000000        BIT15 = 100000        EM22    030232        KIPDR6= 172314        MAPL13= 170254
AMSGLG= 000000        BIT2  = 000004        EM23    030306        KIPDR7= 172316        MAPL14= 170260
AMSGTY= 000000        BIT3  = 000010        EM24    030320        KMCR  = 177734        MAPL15= 170264
AMTYP1= 000000        BIT4  = 000020        EM25    030340        LF    = 000012        MAPL16= 170270
AMTYP2= 000000        BIT5  = 000040        EM26    030410        MAPH0 = 170202        MAPL17= 170274
AMTYP3= 000000        BIT6  = 000100        EM27    030473        MAPH00= 170202        MAPL2 = 170210
AMTYP4= 000000        BIT7  = 000200        EM3     027114        MAPH01= 170206        MAPL20= 170300
APASS = 000000        BIT8  = 000400        EM30    030530        MAPH02= 170212        MAPL21= 170304
APRIOR= 000000        BIT9  = 001000        EM31    030571        MAPH03= 170216        MAPL22= 170310
APTCSU= 000040        BPTVEC= 000014        EM32    030635        MAPH04= 170222        MAPL23= 170314
APTENV= 000001        CCR   = 177746        EM33    030702        MAPH05= 170226        MAPL24= 170320
APTSIZ= 000200        CKSWR = 104407        EM34    030756        MAPH06= 170232        MAPL25= 170324
APTSPO= 000100        CR    = 000015        EM35    031016        MAPH07= 170236        MAPL26= 170330
ASWREG= 000000        CRLF  = 000200        EM36    031055        MAPH1 = 170206        MAPL27= 170334
ATESTN= 000000        CTBLE   002022        EM37    031123        MAPH10= 170242        MAPL3 = 170214
AUNIT = 000000        DCSR  = 177750        EM4     027160        MAPH11= 170246        MAPL30= 170340
AUSWR = 000000        DDIN    002222        EM5     027236        MAPH12= 170252        MAPL31= 170344
AVECT1= 000000        DDISP = 177570        EM6     027334        MAPH13= 170256        MAPL32= 170350
AVECT2= 000000        DDOUT   002210        EM7     027377        MAPH14= 170262        MAPL33= 170354
BCSR  = 177520        DDR   = 177752        ERRNUM  001714        MAPH15= 170266        MAPL34= 170360
```

SYMBOL TABLE

```
MAPL35= 170364   SWREG   000176   TST30   012120   $BASE   001254   $GDADR  001120
MAPL36= 170370   SW0   = 000001   TST31   012334   $BDADR  001122   $GDDAT  001124
MAPL37= 170374   SW00  = 000001   TST32   012756   $BDDAT  001126   $GET42  023132
MAPL4 = 170220   SW01  = 000002   TST33   013210   $BELL   001170   $GTSWR  024542
MAPL5 = 170224   SW02  = 000004   TST34   013634   $CDW1   001260   $HD   = 000001
MAPL6 = 170230   SW03  = 000010   TST35   014034   $CDW2   001262   $HIBTS  000232
MAPL7 = 170234   SW04  = 000020   TST36   014222   $CHARC  024014   $HIOCT  025356
MAPPR   002244   SW05  = 000040   TST37   014324   $CKSWR  024472   $ICNT   001104
MCSR    001732   SW06  = 000100   TST4    003762   $CMTAG  001100   $ILLUP  026770
MEMSIZ  002332   SW07  = 000200   TST40   014604   $CM3   = 000000   $INTAG  001135
MMR0  = 177572   SW08  = 000400   TST41   015144   $CM4   = 000002   $ITEMB  001114
MMR1  = 177574   SW09  = 001000   TST42   015300   $CNTLG  025227   $LF     001176
MMR2  = 177576   SW1   = 000002   TST43   015536   $CNTLU  025222   $LFLG   026405
MMR3  = 172516   SW10  = 002000   TST44   015752   $CPUOP  001226   $LPADR  001106
MMVEC = 000250   SW11  = 004000   TST45   016044   $CRLF   001175   $LPERR  001110
NOABRT  026116   SW12  = 010000   TST46   016124   $DBLK   024462   $MADR1  001232
NOCAH   007664   SW13  = 020000   TST47   016224   $DDW0   001264   $MADR2  001236
NROM    013576   SW14  = 040000   TST5    004042   $DDW1   001266   $MADR3  001242
OBADR   002140   SW15  = 100000   TST50   016440   $DDW10  001310   $MADR4  001246
PCR   = 177522   SW2   = 000004   TST51   016532   $DDW11  001312   $MAIL   001200
PIRQ  = 177772   SW3   = 000010   TST52   016674   $DDW12  001314   $MAMS1  001230
PIRQVE= 000240   SW4   = 000020   TST53   017002   $DDW13  001316   $MAMS2  001234
PMIS    001720   SW5   = 000040   TST54   017432   $DDW14  001320   $MAMS3  001240
POLY  = 120001   SW6   = 000100   TST55   017576   $DDW15  001322   $MAMS4  001244
PR0   = 000000   SW7   = 000200   TST56   017746   $DDW2   001270   $MBADR  000234
PR1   = 000040   SW8   = 000400   TST57   020374   $DDW3   001272   $MFLG   026404
PR2   = 000100   SW9   = 001000   TST6    004226   $DDW4   001274   $MNEW   025245
PR3   = 000140   TBITVE= 000014   TST60   020746   $DDW5   001276   $MSGAD  001214
PR4   = 000200   TBLMH   012716   TST61   021030   $DDW6   001300   $MSGLG  001216
PR5   = 000240   TBLML   012656   TST62   021134   $DDW7   001302   $MSGTY  001200
PR6   = 000300   TEST    001716   TST63   021476   $DDW8   001304   $MSWR   025234
PR7   = 000340   TIMOUT  002234   TST64   022152   $DDW9   001306   $MTYP1  001231
PS    = 177776   TKVEC = 000060   TST65   022336   $DEVCT  001210   $MTYP2  001235
PSW   = 177776   TOPTBL  010542   TST66   022360   $DEVM   001256   $MTYP3  001241
PTRN16  001774   TOUT    001730   TST67   022740   $DOAGN  023152   $MTYP4  001245
PTRN6   002010   TPVEC = 000064   TST7    004404   $DTBL   024452   $MXCNT  023462
PWRVEC= 000024   TRAPVE= 000034   TYPDS = 104405   $ENDAD  023142   $NULL   001154
RDCHR = 104410   TRTVEC= 000014   TYPE  = 104401   $ENDCT  023022   $NWTST= 000001
RDDEC = 104413   TST1    003274   TYPOC = 104402   $ENULL  023156   $OCNT   024242
RDLIN = 104411   TST10   004466   TYPON = 104404   $ENV    001220   $OMODE  024244
RDOCT = 104412   TST11   004624   TYPOS = 104403   $ENVM   001221   $OVER   023440
RESTAR  003270   TST12   004722   UBECT   001722   $EOP    022766   $PASS   001206
RESVEC= 000010   TST13   005044   UBEM    022740   $EOPCT  023014   $PASTM  000240
R6    =%000006   TST14   005450   UBETST  013210   $ERFLG  001103   $POWER  026776
R7    =%000007   TST15   005646   UFDFLG  002572   $ERMAX  001115   $PWRDN  026630
SAV30   002566   TST16   006052   UFDSET= 000001   $ERROR  025536   $PWRMG  026764
SAV32   002570   TST17   006242   UMSIZ   002424   $ERRPC  001116   $PWRUP  026702
SCOPE = 000004   TST2    003420   UQUIET  002574   $ERRTB  001324   $QUES   001174
SIMLGO= 170014   TST20   006514   VALTBL  010526   $ERRTY  026410   $RDCHR  024754
SR0   = 177572   TST21   007114   VMKOR   002576   $ERTTL  001112   $RDDEC  025360
SR1   = 177574   TST22   007574   WRTBUF  001734   $ESCAP  001166   $RDLIN  025104
SR2   = 177576   TST23   007704   $APTHD  000232   $ETABL  001220   $RDOCT  025256
SR3   = 172516   TST24   010076   $ATYC   026166   $ETEND  001324   $RDSZ = 000010
STACK = 001100   TST25   010556   $ATY1   026142   $FATAL  001202   $RTNAD  023154
START   002500   TST26   011266   $ATY3   026150   $FFLG   026406   $SAVR6  026774
STKLMT= 177774   TST27   011436   $ATY4   026160   $FILLC  001156   $SCOPE  023162
SWR     001140   TST3    003640   $AUTOB  001134   $FILLS  001155   $SETUP= 000137
```

D13

SEQ 0159

SYMBOL TABLE

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

. ABS.  032350      000
        000000      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  56592 WORDS  ( 222 PAGES)
DYNAMIC MEMORY:  19748 WORDS  ( 75 PAGES)
ELAPSED TIME:  00:02:40
OKTAB0.BIC,OKTAB0/CR/NL:TOC/-SP=ORION.MLB/ML,OKTAB0
```

COKTABO CREATED BY MACRO ON 5-OCT-84 AT 09:26 PAGE 1

SEQ 0160

SYMBOL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES							
ABASE	= 000000	9-381	9-381						
ABORT	025760	64-5502	#64-5502						
ABORTC	026012	64-5502	#64-5502						
ABORTE	026056	64-5502	#64-5502						
ABORTZ	026102	64-5502	64-5502	#64-5502					
ACDW1	= 000000	9-381	9-381						
ACDW2	= 000000	9-381	9-381						
ACPUOP	= 000000	9-381	9-381						
ADDW0	= 000000	9-381	9-381						
ADDW1	= 000000	9-381	9-381						
ADDW10	= 000000	9-381	9-381						
ADDW11	= 000000	9-381	9-381						
ADDW12	= 000000	9-381	9-381						
ADDW13	= 000000	9-381	9-381						
ADDW14	= 000000	9-381	9-381						
ADDW15	= 000000	9-381	9-381						
ADDW2	= 000000	9-381	9-381						
ADDW3	= 000000	9-381	9-381						
ADDW4	= 000000	9-381	9-381						
ADDW5	= 000000	9-381	9-381						
ADDW6	= 000000	9-381	9-381						
ADDW7	= 000000	9-381	9-381						
ADDW8	= 000000	9-381	9-381						
ADDW9	= 00000C	9-381	9-381						
ADEVCT	= 000000	9-381	9-381						
ADEVM	= 000000	9-381	9-381						
AENV	= 000000	9-381	9-381						
AENVM	= 000000	9-381	9-381						
AFATAL	= 000000	9-381	9-381						
AMADR1	= 000000	9-381	9-381						
AMADR2	= 000000	9-381	9-381						
AMADR3	= 000000	9-381	9-381						
AMADR4	= 000000	9-381	9-381						
AMAMS1	= 000000	9-381	9-381						
AMAMS2	= 000000	9-381	9-381						
AMAMS3	= 000000	9-381	9-381						
AMAMS4	= 000000	9-381	9-381						
AMSGAD	= 000000	9-381	9-381						
AMSGLG	= 000000	9-381	9-381						
AMSGTY	= 000000	9-381	9-381						
AMTYP1	= 000000	9-381	9-381						
AMTYP2	= 000000	9-381	9-381						
AMTYP3	= 000000	9-381	9-381						
AMTYP4	= 000000	9-381	9-381						
APASS	= 000000	9-381	9-381						
APRIOR	= 000000	9-381							
APTCSU	= 000040	64-5493	#64-5508						
APTENV	= 000001	10-819	28-2175	37-3075	45-3845	64-5493	64-5502	64-5508	#64-5508
APTSIZ	= 000200	10-816	#64-5508						
APTSP0	= 000100	64-5493	64-5508	#64-5508					
ASMREG	= 000000	9-381	9-381						
ATESTN	= 000000	9-381	9-381						

F13

SYMBOL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES								
AUNIT	= 000000	9-381	9-381							
AUSWR	= 000000	9-381	9-381							
AVECT1	= 000000	9-381	9-381							
AVECT2	= 000000	9-381	9-381							
BCSR	= 177520	#8-362	*37-3057	*45-3849	*45-3850	*60-5067	*64-5492	*64-5492	*64-5502	
BE1BA	002154	#10-618	40-3400	46-3910	48-3975	49-4047	50-4137	51-4210	52-4289	53-4411
		53-4444	54-4539	55-4611	56-4694	56-4740	57-4807	58-4883	59-4952	60-5043
		60-5058	60-5074	60-5093	61-5189	61-5208	61-5227	61-5251	62-5296	62-5307
		63-5404	64-5456							
BE1CC	002152	#10-617	40-3399	40-3404	46-3909	48-3974	49-4049	50-4139	51-4213	52-4292
		53-4414	53-4447	54-4542	55-4614	56-4698	57-4808	58-4882	59-4954	60-5045
		60-5060	60-5076	60-5095	61-5192	61-5211	61-5228	61-5254	62-5297	62-5308
		62-5315	63-5406	64-5458						
BE1CLR	002160	#10-620								
BE1CR1	002156	#10-619	40-3401	40-3402	41-3482	41-3485	41-3493	41-3497	41-3508	41-3512
		41-3523	41-3527	42-3608	42-3611	42-3625	42-3628	42-3642	42-3645	42-3659
		42-3662	43-3700	43-3711	44-3776	44-3779	44-3805	46-3912	47-3946	48-3978
		48-3979	49-4050	49-4056	50-4140	50-4146	51-4222	51-4223	52-4301	52-4302
		53-4423	53-4424	53-4456	53-4457	54-4551	54-4552	55-4621	55-4622	56-4707
		56-4708	56-4746	56-4747	57-4809	57-4810	58-4887	58-4888	59-4955	59-4956
		60-5046	60-5047	60-5061	60-5062	60-5077	60-5078	60-5096	60-5098	61-5193
		61-5194	61-5212	61-5213	61-5231	61-5233	61-5256	61-5258	62-5300	62-5301
		62-5309	62-5310	62-5316	62-5317	63-5410	63-5411	64-5462	64-5463	
BE1CR2	002162	#10-621	45-3864	45-3869	46-3911	46-3915	47-3944	47-3949	47-3952	49-4048
		50-4138	51-4211	52-4290	53-4412	53-4445	54-4540	55-4612	55-4624	56-4695
		56-4696	56-4710	56-4741	57-4812	59-4953	60-5044	60-5059	60-5075	60-5094
		60-5106	61-5190	61-5209	61-5240	61-5252	61-5266	62-5299	63-5405	64-5457
BE1DB	002150	#10-616	10-803	39-3321	48-3982	49-4046	50-4136	50-4153	51-4212	52-4291
		52-4310	53-4413	53-4435	53-4437	53-4446	53-4468	53-4470	54-4541	54-4560
		54-4562	55-4613	56-4697	56-4745	56-4749	58-4881	60-5052	60-5068	60-5083
		61-5191	61-5210	61-5253	62-5298	62-5312	62-5319	63-5423		
BE1INT	001724	#10-587	44-3785	*44-3792	*44-3801	*44-3807				
BE1PSW	002166	#10-623	41-3477	42-3601	43-3701	44-3769	55-4610	56-4702	57-4801	60-5090
		61-5230	61-5250							
BE1SV	015476	44-3768	#44-3800							
BE1VEC	002164	#10-622	41-3481	41-3491	41-3506	41-3521	42-3605	42-3622	42-3639	42-3656
		44-3768	48-3976	55-4609	56-4701	57-4800	60-5089	61-5229	61-5249	
BE2BA	002174	#10-631								
BE2CC	002172	#10-630								
BE2CLR	002200	#10-633	44-3797							
BE2CR1	002176	#10-632	44-3778	44-3780	44-3800					
BE2CR2	002202	#10-634								
BE2DB	002170	#10-629								
BE2INT	001726	#10-588	44-3789	*44-3793	*44-3802	*44-3806				
BE2PSW	002206	#10-636	44-3771							
BE2SV	015516	44-3770	#44-3805							
BE2VEC	002204	#10-635	44-3770							
BIT0	= 000001	#8-265	12-974							
BIT00	= 000001	#8-265	8-265	10-682	19-1417	44-3779	44-3780	57-4786	57-4789	64-5447
		64-5470								
BIT01	= 000002	#8-265	8-265	18-1369	19-1415	20-1451	20-1465	21-1539	22-1652	24-1796
		25-1862	26-1947	49-4044	49-4067	49-4070				

C13

SYMBOL CROSS REFERENCE CREF V02

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

SYMBOL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES								
CRLF	= 000200	#8-265	10-821	10-821	64-5493	64-5493				
CTBLE	002022	#10-603	36-2954							
DCSR	= 177730	#8-363	*10-682	15-1191	*15-1196	15-1197	15-1199	15-1208	*15-1213	15-1215
		15-1218	15-1220	15-1225	*16-1271	*17-1317	*17-1334	*18-1368	*18-1369	18-1373
		*18-1388	*19-1414	*19-1415	*19-1417	*19-1424	*20-1450	*20-1451	*20-1465	*20-1467
		*21-1538	*21-1539	*21-1605	*22-1651	*22-1652	*22-1675	*23-1717	23-1741	*23-1749
		*24-1795	*24-1796	*24-1812	*25-1861	*25-1862	*25-1904	*26-1946	*26-1947	*26-2005
		*27-2054	27-2128	*27-2148	*29-2215	*29-2237	*30-2303	*30-2374	*31-2474	*31-2552
		*32-2588	*32-2605	*33-2638	*33-2705	*34-2735	*34-2764	*35-2818	*35-2875	*36-2953
		*36-2971	*36-3008	*37-3056	37-3132	*37-3138	*37-3140	*37-3147	*39-3312	*49-4044
		*49-4067	49-4068	*49-4070	*56-4678	*56-4680	*56-4731	*56-4733	*56-4752	*56-4754
		*57-4795	*57-4797	*57-4834	*57-4838	*57-4841	*57-4843	*63-5393	*63-5395	
DDIN	002222	#10-682	20-1455	25-1881	26-1966	27-2127	30-2316	30-2339	30-2355	31-2484
		31-2493	31-2510	31-2527	31-2544	33-2671	33-2673	34-2741	34-2750	35-2851
		35-2868	36-2988	36-2998						
DDISP	= 177570	#8-265	9-381	10-816						
DDOUT	002210	#10-657	18-1377	21-1580	24-1809	26-1976	33-2652	33-2676		
DDR	= 177732	#8-364	*10-657	18-1378	18-1380	19-1422	19-1426	20-1458	20-1461	20-1463
		*22-1666	*23-1739	25-1884	25-1887	26-1968	26-1970	36-3002	49-4058	49-4062
		49-4064								
DH1	031174	10-387	10-435	10-448	10-536	#64-5549				
DH13	031565	10-454	10-467	#64-5559						
DH16	031640	10-480	10-492	10-498	10-511	10-517	10-542	10-548	10-554	10-560
		10-566	10-572	10-578	#64-5561					
DH2	031243	10-393	10-400	10-441	#64-5551					
DH20	031677	10-486	#64-5563							
DH23	031757	10-504	#64-5565							
DH27	032070	10-530	#64-5567							
DH3	031335	10-407	#64-5553							
DH4	031425	10-414	#64-5555							
DH5	031503	10-421	10-428	10-460	10-473	10-523	#64-5557			
DISPLA	001142	#9-381	*10-816	*10-816	64-5492	64-5502				
DISPRE	000174	#8-371	10-816							
DSWR	= 177570	#8-265	9-381	10-816						
DT1	032166	10-388	10-436	10-449	#64-5570					
DT16	032246	10-481	10-493	10-499	10-512	10-518	10-543	10-549	10-555	10-561
		10-573	10-579	#64-5574						
DT2	032200	10-395	10-402	10-443	#64-5571					
DT20	032256	10-487	#64-5575							
DT21	032272	10-567	#64-5576							
DT23	032302	10-506	#64-5577							
DT27	032322	10-531	#64-5578							
DT3	032216	10-409	#64-5572							
DT30	032336	10-537	#64-5579							
DT4	032232	10-416	10-423	10-430	10-455	10-462	10-468	10-475	10-525	#64-5573
EMTSAV	002500	#10-815								
EMTVEC	= 000030	#8-265	*10-816	*10-816						
EM1	027006	10-386	#64-5517							
EM10	027446	10-434	#64-5524							
EM11	027520	10-440	#64-5525							
EM12	027572	10-447	#64-5526							
EM13	027644	10-453	#64-5527							

SEQ 0164

SYMBOL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES
EM14	027713	10-459 #64-5528
EM15	027743	10-466 #64-5529
EM16	027772	10-472 #64-5530
EM17	030054	10-479 #64-5531
EM2	027052	10-392 #64-5518
EM20	030107	10-485 #64-5532
EM21	030165	10-491 #64-5533
EM22	030232	10-497 #64-5534
EM23	030306	10-503 #64-5535
EM24	030320	10-510 #64-5536
EM25	030340	10-516 #64-5537
EM26	030410	10-522 #64-5538
EM27	030473	10-529 #64-5539
EM3	027114	10-399 #64-5519
EM30	030530	10-535 #64-5540
EM31	030571	10-541 #64-5541
EM32	030635	10-547 #64-5542
EM33	030702	10-553 #64-5543
EM34	030756	10-559 #64-5544
EM35	031016	10-565 #64-5545
EM36	031055	10-571 #64-5546
EM37	031123	10-577 #64-5547
EM4	027160	10-406 #64-5520
EM5	027236	10-413 #64-5521
EM6	027334	10-420 #64-5522
EM7	027377	10-427 #64-5523
ERRNUM	001714	#10-583 *64-5505 64-5570 64-5571 64-5572 64-5573 64-5574 64-5575 64-5576 64-5577 64-5578 64-5579
ERROR =	104000	#8-265 10-701 11-902 12-970 12-981 12-994 12-1005 13-1060 13-1072 14-1128 15-1203 15-1217 15-1221 15-1226 16-1280 16-1286 17-1324 18-1375 18-1381 18-1385 19-1427 20-1464 21-1587 21-1591 22-1668 23-1743 24-1817 25-1889 26-1973 26-1982 27-2074 27-2099 27-2120 27-2130 29-2226 29-2234 30-2323 30-2330 30-2345 30-2352 30-2361 30-2368 31-2498 31-2504 31-2515 31-2521 31-2532 31-2538 31-2549 31-2556 32-2597 32-2603 33-2658 33-2664 33-2682 33-2689 33-2696 33-2702 34-2755 34-2761 35-2857 35-2871 36-3001 36-3004 37-3095 37-3120 37-3128 37-3143 38-3235 38-3239 40-3406 41-3487 41-3499 41-3514 41-3529 42-3612 42-3629 42-3646 42-3663 43-3716 44-3787 44-3791 45-3867 45-3874 46-3914 47-3951 48-3984 48-3987 49-4066 49-4083 50-4155 51-4235 52-4312 53-4440 53-4473 54-4565 55-4626 55-4630 56-4712 56-4717 56-4751 57-4814 57-4819 58-4895 59-4967 60-5054 60-5070 60-5085 60-5102 60-5108 61-5201 61-5220 61-5223 61-5236 61-5242 61-5262 61-5268 62-5314 62-5321 63-5420 63-5425 64-5467
ERRVEC =	000004	#8-265 10-752 *10-753 *10-754 *10-768 10-816 *10-816 *10-816 37-3125 *37-3126 *37-3141 *37-3148 64-5492 *64-5492 *64-5492 *64-5492
ERTYPE	026120	64-5502 #64-5504
GNS =	******	8-371 8-371 10-821 64-5482 64-5482 64-5510
GTSWR =	104406	10-821 #64-5510
HT =	000011	#8-265 64-5493 64-5493
IOTVEC =	000020	#8-265 *10-816 *10-816
IUBE	002450	#10-802 40-3397 41-3475 42-3600 42-3621 42-3638 42-3655 43-3699 44-3762

SYMBOL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES								
		45-3857	46-3905	47-3943	48-3973	49-4031	50-4122	51-4204	52-4283	53-4400
		54-4532	55-4604	56-4688	57-4806	58-4878	59-4937	60-5038	60-5110	61-5184
		61-5246	61-5270	62-5292	63-5383	64-5446				
KDPAR0	= 172360	#8-266								
KDPAR1	= 172362	#8-266								
KDPAR2	= 172364	#8-266								
KDPAR3	= 172366	#8-266								
KDPAR4	= 172370	#8-266								
KDPAR5	= 172372	#8-266								
KDPAR6	= 172374	#8-266								
KDPAR7	= 172376	#8-266								
KDPDR0	= 172320	#8-266								
KDPDR1	= 172322	#8-266								
KDPDR2	= 172324	#8-266								
KDPDR3	= 172326	#8-266								
KDPDR4	= 172330	#8-266								
KDPDR5	= 172332	#8-266								
KDPDR6	= 172334	#8-266								
KDPDR7	= 172336	#8-266								
KIPAR0	= 172340	#8-266								
KIPAR1	= 172342	#8-266								
KIPAR2	= 172344	#8-266								
KIPAR3	= 172346	#8-266								
KIPAR4	= 172350	#8-266								
KIPAR5	= 172352	#8-266								
KIPAR6	= 172354	#8-266	*10-758	*10-760	10-762	10-832	*21-1556	21-1583	*21-1596	*22-1661
		*23-1728	*23-1732	*24-1803	*25-1877	25-1893	*25-1898	*25-1900	25-1901	*26-1960
		*26-1967	*26-1977	*26-1986	*27-2064	*27-2066	*27-2078	*27-2114	27-2134	*27-2142
		*38-3230	*38-3243	38-3244	*53-4399	*54-4531	*57-4802	57-4826	*57-4833	64-5577
KIPAR7	= 172356	#8-266								
KIPDR0	= 172300	#8-266								
KIPDR1	= 172302	#8-266								
KIPDR2	= 172304	#8-266								
KIPDR3	= 172306	#8-266								
KIPDR4	= 172310	#8-266								
KIPDR5	= 172312	#8-266								
KIPDR6	= 172314	#8-266								
KIPDR7	= 172316	#8-266								
KMCR	= 177734	#8-365	10-776	16-1258	16-1268	*16-1273	16-1274	16-1276	16-1278	*16-1279
		16-1285	21-1535	21-1565	*21-1566	21-1569	*21-1602	22-1646	23-1726	23-1730
		*23-1731	*23-1748	24-1790	25-1856	26-1941	27-2057	27-2062	27-2086	*27-2088
		27-2089	27-2093	27-2096	27-2097	*27-2098	*27-2115	*27-2139	27-2140	*27-2145
		*28-2170	28-2171	*29-2219	*29-2220	29-2221	29-2223	29-2224	*29-2227	29-2228
		29-2231	29-2232	*30-2304	30-2310	*30-2317	30-2318	*30-2324	30-2325	*30-2338
		30-2340	*30-2346	30-2347	*30-2354	30-2356	*30-2362	30-2363	*30-2373	*31-2476
		31-2479	*31-2492	31-2494	*31-2499	31-2500	*31-2509	31-2511	*31-2516	31-2517
		*31-2526	31-2528	*31-2533	31-2534	*31-2543	31-2545	*31-2550	31-2551	*32-2587
		*32-2590	*32-2591	32-2592	32-2594	32-2596	*32-2598	32-2599	*32-2604	*33-2639
		*33-2643	33-2644	*33-2645	33-2646	*33-2653	33-2654	33-2657	*33-2659	33-2660
		33-2663	*33-2668	*33-2669	33-2677	33-2679	33-2681	*33-2683	33-2684	33-2686
		33-2688	33-2690	33-2693	33-2695	33-2697	33-2699	33-2701	*33-2703	*34-2737
		*34-2745	34-2746	*34-2747	34-2748	*34-2751	34-2752	34-2754	*34-2756	34-2757

K13

SYMBOL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES								
		34-2760	*34-2762	35-2816	*35-2838	*35-2839	*35-2840	35-2853	35-2869	*36-2972
		36-2999	37-3134	38-3202	38-3207	38-3220	51-4199	53-4396	56-4670	56-4671
		*56-4679	56-4729	*56-4732	*56-4753	57-4794	*57-4796	*57-4835	57-4836	*57-4842
		*63-5384	63-5385	*63-5394	63-5416	*63-5422	*64-5452	*64-5453	64-5465	64-5575
LF	= 000012	*8-265	64-5493	64-5493						
MAPH0	= 170202	*8-339								
MAPH00	= 170202	*8-274	8-339	*23-1734	*24-1806	*26-1962	*26-1989	*26-1990	*27-2110	*27-2138
		*30-2309	*31-2478	*33-2642	*34-2739	*36-2970	*51-4215	*52-4294	*54-4544	*55-4616
		*57-4799	*57-4830	*61-5207	61-5221	*63-5391				
MAPH01	= 170206	*8-276	8-341	*35-2849	*35-2850	*35-2866	*35-2867	*63-5403	*64-5455	64-5578
MAPH02	= 170212	*8-278	8-343							
MAPH03	= 170216	*8-280	8-345							
MAPH04	= 170222	*8-282	8-347							
MAPH05	= 170226	*8-284	8-349							
MAPH06	= 170232	*8-286	8-351							
MAPH07	= 170236	*8-288	8-353							
MAPH1	= 170206	*8-341								
MAPH10	= 170242	*8-290								
MAPH11	= 170246	*8-292								
MAPH12	= 170252	*8-294								
MAPH13	= 170256	*8-296								
MAPH14	= 170262	*8-298								
MAPH15	= 170266	*8-300								
MAPH16	= 170272	*8-302								
MAPH17	= 170276	*8-304								
MAPH2	= 170212	*8-343								
MAPH20	= 170302	*8-306								
MAPH21	= 170306	*8-308								
MAPH22	= 170312	*8-310								
MAPH23	= 170316	*8-312								
MAPH24	= 170322	*8-314								
MAPH25	= 170326	*8-316								
MAPH26	= 170332	*8-318								
MAPH27	= 170336	*8-320								
MAPH3	= 170216	*8-345								
MAPH30	= 170342	*8-322								
MAPH31	= 170346	*8-324								
MAPH32	= 170352	*8-326								
MAPH33	= 170356	*8-328								
MAPH34	= 170362	*8-330								
MAPH35	= 170366	*8-332								
MAPH36	= 170372	*8-334	*56-4743							
MAPH37	= 170376	*8-336	*22-1665	62-5324	*63-5389					
MAPH4	= 170222	*8-347								
MAPH5	= 170226	*8-349								
MAPH6	= 170232	*8-351								
MAPH7	= 170236	*8-353								
MAPL0	= 170200	*8-338								
MAPL00	= 170200	*8-273	8-338	11-882	12-956	12-964	13-1044	13-1045	13-1053	13-1054
		14-1112	14-1119	21-1544	21-1555	*23-1733	*24-1805	25-1869	*26-1961	*26-1988
		26-1992	*27-2109	*27-2136	*30-2308	*31-2477	*33-2641	*34-2738	*36-2987	*36-2997
		*51-4214	*52-4293	53-4403	*54-4543	*55-4615	*57-4798	*57-4828	57-4831	*61-5205

SYMBOL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES								
		61-5208	61-5218	62-5296	62-5306	*63-5390	64-5579			
MAPL01	= 170204	#8-275	8-340	*35-2848	*35-2865	*63-5402	*64-5454	64-5578		
MAPL02	= 170210	#8-277	8-342							
MAPL03	= 170214	#8-279	8-344							
MAPL04	= 170220	#8-281	8-346							
MAPL05	= 170224	#8-283	8-348							
MAPL06	= 170230	#8-285	8-350							
MAPL07	= 170234	#8-287	8-352							
MAPL1	= 170204	#8-340								
MAPL10	= 170240	#8-289	56-4685							
MAPL11	= 170244	#8-291								
MAPL12	= 170250	#8-293								
MAPL13	= 170254	#8-295								
MAPL14	= 170260	#8-297								
MAPL15	= 170264	#8-299								
MAPL16	= 170270	#8-301								
MAPL17	= 170274	#8-303								
MAPL2	= 170210	#8-342								
MAPL20	= 170300	#8-305								
MAPL21	= 170304	#8-307								
MAPL22	= 170310	#8-309								
MAPL23	= 170314	#8-311								
MAPL24	= 170320	#8-313								
MAPL25	= 170324	#8-315								
MAPL26	= 170330	#8-317								
MAPL27	= 170334	#8-319								
MAPL3	= 170214	#8-344								
MAPL30	= 170340	#8-321								
MAPL31	= 170344	#8-323								
MAPL32	= 170350	#8-325								
MAPL33	= 170354	#8-327								
MAPL34	= 170360	#8-329								
MAPL35	= 170364	#8-331								
MAPL36	= 170370	#8-333	*56-4742							
MAPL37	= 170374	#8-335	*22-1664	56-4673	56-4738	*63-5388				
MAPL4	= 170220	#8-346								
MAPL5	= 170224	#8-348								
MAPL6	= 170230	#8-350								
MAPL7	= 170234	#8-352								
MAPPR	002244	#10-719	10-826	21-1552	22-1653	23-1718	24-1797	25-1863	26-1948	27-2055
		35-2819	38-3228	57-4785						
MCSR	001732	#10-590	60-5043	60-5052	61-5188	61-5189	61-5199	61-5205		
MEMSIZ	002332	#10-752	10-829							
MMR0	= 177572	#8-357	*10-828	*10-830	*21-1553	*21-1603	*22-1654	*22-1672	*23-1719	*23-1744
		*24-1798	*24-1810	*25-1865	*25-1903	*26-1950	*26-2004	*27-2056	*27-2147	*35-2820
		*35-2874	*38-3229	*38-3246	*57-4786	*57-4789				
MMR1	= 177574	#8-358								
MMR2	= 177576	#8-359								
MMR3	= 172516	#8-360	*10-827	*10-831	*21-1554	*21-1604	*22-1655	*22-1673	*23-1735	*23-1745
		*24-1799	*24-1811	*25-1864	*26-1949	*27-2107	*27-2111	*27-2146	*29-2216	*29-2218
		29-2235	*30-2307	*30-2372	*31-2475	*32-2589	*33-2640	*33-2704	*34-2736	*34-2763
		*35-2821	*35-2841	*36-2969	*38-3224	*39-3313	*49-4036	*50-4123	*51-4216	*51-4230

SYMBOL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES								
		#52-4295	#52-4304	#53-4417	#53-4426	#53-4450	#53-4459	#54-4545	#54-4554	#55-4617
		#55-4637	#56-4684	#57-4767	#57-4793	#57-4840	#61-5255	#63-5392	#63-5421	#64-5451
		#64-5469								
MMVEC	= 000250	#8-266								
NOABRT	026116	64-5502	64-5502	#64-5502						
NOCAM	007664	28-2179	#28-2183							
NROM	013576	37-3080	#37-3151							
OBADR	002140	#10-605	36-2961	36-2974	36-2984	36-2994				
PCR	= 177522	#8-367	#37-3058							
PIRQ	= 177772	#8-265	#43-3712	#43-3724						
PIRQVE	= 000240	#8-265	#43-3702							
PMIS	001720	#10-585	#10-832	#16-1263	17-1313	18-1366	19-1412	20-1448	21-1531	22-1642
		23-1715	24-1786	25-1852	26-1939	26-1951	27-2051	27-2108	28-2168	29-2213
		30-2300	31-2471	32-2585	33-2635	34-2729	35-2813	35-2822	35-2828	35-2830
		36-2948	53-4477	57-4780	57-4788					
POLY	= 120001	#8-368	37-3107	37-3109						
PR0	= 000000	#8-265								
PR1	= 000040	#8-265								
PR2	= 000100	#8-265								
PR3	= 000140	#8-265								
PR4	= 000200	#8-265								
PR5	= 000240	#8-265								
PR6	= 000300	#8-265								
PR7	= 000340	#8-265								
PS	= 177776	#8-265	8-265							
PSW	= 177776	#8-265	#40-3398	#40-3407	40-3408	#40-3410	#41-3476	#41-3494	#41-3509	#41-3524
		#42-3607	#42-3609	#42-3624	#42-3626	#42-3641	#42-3643	#42-3658	#42-3660	#43-3710
		#43-3713	#44-3775	#44-3781	#47-3945	#47-3953	#48-3977	#48-3981	#61-5257	
PTRN16	001774	#10-592	12-971	18-1371	36-2973	49-4038	49-4079	50-4129	52-4293	
PTRN6	002010	#10-593	12-995							
PWRVEC	= 000024	#8-265	#10-816	#10-816	45-3861	#45-3862	45-3863	#45-3871	#45-3877	#64-5511
		#64-5511	#64-5511	#64-5511	#64-5511	#64-5511				
RDCHR	= 104410	64-5496	#64-5510							
RDDEC	= 104413	#64-5510								
RDLIN	= 104411	64-5497	64 5498	#64-5510						
RDOCT	= 104412	#64-5510								
RESTAR	003270	#10-833	64-5482							
RESVEC	= 000010	#8-265								
R6	=%000006	#8-265	#10-816	#10-816	10-816					
R7	=%000007	#8-265								
SAV30	002566	10-815	#10-815	#10-815	64-5502	64-5502				
SAV32	002570	#10-815	#10-815	64-5502	64-5502					
SCOPE	= 000004	#8-265	11-866	12-951	13-1039	14-1107	15-1181	16-1250	17-1308	18-1364
		19-1410	20-1447	21-1529	22-1640	23-1713	24-1784	25-1850	26-1937	27-2050
		28-2166	29-2211	30-2298	31-2469	32-2583	33-2633	34-2727	35-2811	36-2947
		37-3051	38-3197	39-3310	40-3395	41-3473	42-3598	43-3698	44-3758	45-3837
		46-3903	47-3941	48-3971	49-4028	50-4120	51-4197	52-4281	53-4394	54-4529
		55-4603	56-4668	57-4778	58-4876	59-4935	60-5036	61-5182	62-5291	63-5381
		64-5444	64-5473	64-5482						
SIMLGO	= 170014	#8-369								
SR0	= 177572	#8-266								
SR1	= 177574	#8-266								

SYMBOL CROSS REFERENCE CREF V02

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

SYMBOL CROSS REFERENCE CREF V02 SEQ 0170

SYMBOL	VALUE	REFERENCES							
TST13	005044	20-1449	#21-1529						
TST14	005450	21-1532	#22-1640						
TST15	005646	22-1643	22-1647	#23-1713					
TST16	006052	23-1716	23-1747	#24-1784					
TST17	006242	24-1787	24-1791	24-1814	#25-1850				
TST2	003420	11-890	#12-951						
TST20	006514	25-1853	25-1857	#26-1937					
TST21	007114	26-1940	26-1942	#27-2050					
TST22	007574	27-2085	#28-2166						
TST23	007704	28-2169	28-2172	#29-2211					
TST24	010076	29-2214	#30-2298						
TST25	010556	30-2375	#31-2469						
TST26	011266	31-2554	#32-2583						
TST27	011436	32-2586	#33-2633						
TST3	003640	#13-1039							
TST30	012120	#34-2727							
TST31	012334	34-2730	#35-2811						
TST32	012756	35-2817	35-2876	#36-2947					
TST33	013210	36-2949	#37-3051						
TST34	013634	37-3149	#38-3197						
TST35	014034	38-3203	#39-3310						
TST36	014222	39-3371	#40-3395						
TST37	014324	#41-3473							
TST4	003762	#14-1107							
TST40	014604	#42-3598							
TST41	015144	#43-3698							
TST42	015300	#44-3758							
TST43	015536	44-3761	44-3798	#45-3837					
TST44	015752	45-3842	45-3844	45-3846	45-3848	45-3852	45-3854	45-3856	#46-3903
TST45	016044	#47-3941							
TST46	016124	#48-3971							
TST47	016224	48-3983	48-3985	#49-4028					
TST5	004042	#15-1181							
TST50	016440	#50-4120							
TST51	016532	50-4154	#51-4197						
TST52	016674	#52-4281							
TST53	017002	52-4311	#53-4394						
TST54	017432	53-4478	#54-4529						
TST55	017576	54-4561	#55-4603						
TST56	017746	#56-4668							
TST57	020374	#57-4778							
TST6	004226	15-1219	#16-1250						
TST60	020746	57-4781	#58-4876						
TST61	021030	58-4894	#59-4935						
TST62	021134	#60-5036							
TST63	021476	#61-5182							
TST64	022152	#62-5291							
TST65	022336	#63-5381							
TST66	022560	63-5424	#64-5444						
TST67	022740	#64-5473							
TST7	004404	16-1264	16-1281	#17-1308					
TYPDS	104405	64-5482	64-5482	#64-5510					

SYMBOL CROSS REFERENCE CREF V02 SEQ 0171

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

SYMBOL CROSS REFERENCE CREF V02 SEQ 0172

SYMBOL	VALUE	REFERENCES
$DBLK	024462	64-5495 64-5495 #64-5495
$DDW0	001264	#9-381
$DDW1	001266	#9-381
$DDW10	001310	#9-381
$DDW11	001312	#9-381
$DDW12	001314	#9-381
$DDW13	001316	#9-381
$DDW14	001320	#9-381
$DDW15	001322	#9-381
$DDW2	001270	#9-381
$DDW3	001272	#9-381
$DDW4	001274	#9-381
$DDW5	001276	#9-381
$DDW6	001300	#9-381
$DDW7	001302	#9-381
$DDW8	001304	#9-381
$DDW9	001306	#9-381
$DEVCT	001210	#9-381
$DEVM	001256	#9-381
$DOAGN	023152	64-5482 64-5482 #64-5482
$DTBL	024452	64-5495 #64-5495
$ENDAD	023142	8-379 10-821 #64-5482 64-5502 64-5502
$ENDCT	023022	10-816 #64-5477 #64-5482
$ENULL	023156	#64-5482
$ENV	001220	#9-381 10-819 10-821 28-2175 37-3075 45-3845 64-5493 64-5502 64-5508 64-5508
$ENVM	001221	#9-381 10-816 64-5493 64-5493 64-5508
$EOP	022766	51-4203 63-5387 64-5475 #64-5482
$EOPCT	023014	#10-816 #64-5476 #64-5482 64-5482
$ERFLG	001103	#9-381 64-5492 64-5492 64-5492 #64-5492 64-5492 64-5492 #64-5502 64-5502 64-5502
$ERMAX	001115	#9-381 #10-816 64-5492 #64-5492 64-5492 64-5492
$ERROR	025536	10-816 #64-5502
$ERRPC	001116	#9-381 #64-5502 #64-5502 64-5502 64-5502 64-5502 64-5509 64-5570 64-5571 64-5572 64-5573 64-5574 64-5575 64-5577 64-5578 64-5579
$ERRTB	001324	#10-381 64-5509
$ERRTY	026410	64-5506 #64-5509
$ERTTL	001112	#9-381 64-5482 #64-5502 64-5502 64-5502
$ESCAP	001166	#9-381 #10-816 #64-5492 64-5502 64-5502 64-5502 64-5502
$ETABL	001220	#9-381
$ETEND	001324	8-380 #9-381
$FATAL	001202	#9-381 #64-5508
$FFLG	026406	#64-5508 #64-5508 64-5508 #64-5508 #64-5508
$FILLC	001156	#9-381 64-5493 64-5493 64-5493
$FILLS	001155	#9-381 64-5493 64-5493
$GDADR	001120	#9-381 #13-1058 #13-1070 64-5572
$GDDAT	001124	#9-381 10-657 #12-968 #12-979 #12-992 #12-1003 #14-1126 #15-1202 #15-1207 #15-1208 #16-1276 #18-1376 18-1378 18-1382 #19-1416 19-1418 19-1422 #20-1453 #20-1460 #21-1579 21-1588 #22-1662 22-1666 #24-1807 #25-1886 #26-1965 #26-1971 #26-1974 26-1978 #26-2001 #27-2091 #27-2092 27-2093 #27-2095 #29-2224 #29-2225 #29-2232 #29-2233 #30-2310 #30-2311 #30-2322 #30-2329 #30-2344 #30-2351 #30-2360 #30-2367 #31-2479 #31-2497 #31-2503 #31-2514 #31-2520 #31-2531 #31-2537 #31-2548

SYMBOL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES								
		*31-2555	*32-2592	*32-2593	*32-2602	*33-2651	*33-2656	*33-2662	*33-2675	*33-2679
		*33-2680	*33-2686	*33-2687	*33-2690	*33-2691	*33-2699	*33-2700	*34-2746	34-2752
		*34-2759	*49-4065	*53-4438	*53-4471	*54-4563	64-5571	64-5573	64-5577	
$GET42	023132	#64-5482								
$GTSWR	024542	#64-5496	64-5510	64-5510						
$HD	= 000001	8-262	8-262	8-262						
$HIBTS	000232	#8-380								
$HIOCT	025356	*64-5497	#64-5497							
$ICNT	001104	#9-381	*64-5492	64-5492	*64-5492	64-5492	64-5492			
$ILLUP	026770	64-5511	64-5511	#64-5511						
$INTAG	001135	#9-381	64-5496	64-5496	64-5496					
$ITEMB	001114	#9-381	*64-5502	64-5502	64-5502	64-5502	64-5505	64-5509		
$LF	001176	#9-381	64-5493	64-5493	64-5496	64-5496	64-5496	64-5498	64-5498	64-5502
		64-5502								
$LFLG	026405	*64-5508	#64-5508							
$LPADR	001106	#9-381	*10-816	*64-5492	*64-5492	64-5492	64-5492	64-5492		
$LPERR	001110	#9-381	*10-816	64-5492	*64-5492	64-5492	64-5492	64-5502		
$MADR1	001232	#9-381								
$MADR2	001236	#9-381								
$MADR3	001242	#9-381								
$MADR4	001246	#9-381								
$MAIL	001200	8-380	8-380	#9-381	10-816	10-821	64-5492	64-5493	64-5502	
$MAMS1	001230	#9-381								
$MAMS2	001234	#9-381								
$MAMS3	001240	#9-381								
$MAMS4	001244	#9-381								
$MBADR	000234	#8-380								
$MFLG	026404	*64-5508	64-5508	*64-5508	#64-5508					
$MNEW	025245	64-5496	#64-5496							
$MSGAD	001214	#9-381	*64-5508	64-5508						
$MSGLG	001216	#9-381	*64-5508							
$MSGTY	001200	#9-381	64-5508	*64-5508	64-5508	*64-5508				
$MSWR	025234	64-5496	#64-5496							
$MTYP1	001231	#9-381								
$MTYP2	001235	#9-381								
$MTYP3	001241	#9-381								
$MTYP4	001245	#9-381								
$MXCNT	023462	64-5492	64-5492	64-5492	#64-5492					
$NULL	001154	#9-381	64-5493	64-5493	64-5493					
$NWTST	= 000001	#11-866	11-866	#11-866	#12-951	12-951	#12-951	#13-1039	13-1039	#13-1039
		#14-1107	14-1107	#14-1107	#15-1181	15-1181	#15-1181	#16-1250	16-1250	#16-1250
		#17-1308	17-1308	#17-1308	#18-1364	18-1364	#18-1364	#19-1410	19-1410	#19-1410
		#20-1447	20-1447	#20-1447	#21-1529	21-1529	#21-1529	#22-1640	22-1640	#22-1640
		#23-1713	23-1713	#23-1713	#24-1784	24-1784	#24-1784	#25-1850	25-1850	#25-1850
		#26-1937	26-1937	#26-1937	#27-2050	27-2050	#27-2050	#28-2166	28-2166	#28-2166
		#29-2211	29-2211	#29-2211	#30-2298	30-2298	#30-2298	#31-2469	31-2469	#31-2469
		#32-2583	32-2583	#32-2583	#33-2633	33-2633	#33-2633	#34-2727	34-2727	#34-2727
		#35-2811	35-2811	#35-2811	#36-2947	36-2947	#36-2947	#37-3051	37-3051	#37-3051
		#38-3197	38-3197	#38-3197	#39-3310	39-3310	#39-3310	#40-3395	40-3395	#40-3395
		#41-3473	41-3473	#41-3473	#42-3598	42-3598	#42-3598	#43-3698	43-3698	#43-3698
		#44-3758	44-3758	#44-3758	#45-3837	45-3837	#45-3837	#46-3903	46-3903	#46-3903
		#47-3941	47-3941	#47-3941	#48-3971	48-3971	#48-3971	#49-4028	49-4028	#49-4028

COKTABO CREATED BY MACRO ON 5-OCT-84 AT 09:26 PAGE 15

SYMBOL CROSS REFERENCE CREF V02 SEQ 0174

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

SYMBOL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES
		8-263 64-5492 64-5492 64-5492 64-5492 64-5492 64-5492 64-5492 64-5492 64-5492 64-5492 64-5492 64-5492
$TESTN	001204	#9-381 *64-5492
$TIMES	001164	#9-381 *10-816 *64-5482 *64-5492 64-5492 *64-5492 64-5492 64-5492
$TKB	001146	#9-381 64-5493 64-5493 64-5493 64-5493 64-5496 64-5496 64-5496 64-5496 64-5496 64-5496
$TKS	001144	#9-381 64-5493 64-5493 64-5493 64-5493 64-5496 64-5496 64-5496 64-5496 64-5496 64-5496
$TMP0	001160	*8-373 *8-376 #9-381 *16-1258 *16-1260 16-1261 16-1279 *18-1370 18-1372 18-1382 18-1384 *19-1418 *26-1952 26-2002 27-2098 *29-2228 29-2229 *35-2844 35-2850 35-2867 *37-3060 *37-3085 *37-3125 37-3148 40-3400 *45-3861 45-3877 46-3910 48-3975 48-3982 *53-4432 *53-4433 *53-4434 53-4435 53-4438 53-4439 *53-4465 *53-4466 *53-4467 53-4468 53-4471 53-4472 *56-4673 *56-4676 56-4724 56-4738 *57-4794 57-4842 *61-5188 61-5205
$TMP1	001162	#9-381 *21-1565 21-1602 *23-1730 23-1748 *26-1963 26-1967 26-1977 *26-1997 *26-1998 26-1999 26-2002 *27-2086 27-2145 *53-4406 *53-4484 53-4485 53-4486 53-4487 *56-4670 56-4753
$TN	= 000070	8-262 #8-262 11-866 11-866 #11-866 11-890 12-951 12-951 #12-951 13-1039 13-1039 #13-1039 14-1107 14-1107 #14-1107 15-1181 15-1181 #15-1181 15-1219 16-1250 16-1250 #16-1250 16-1264 16-1281 17-1308 17-1308 #17-1308 17-1314 18-1364 18-1364 #18-1364 18-1367 19-1410 19-1410 #19-1410 19-1413 19-1425 20-1447 20-1447 #20-1447 20-1449 21-1529 21-1529 #21-1529 21-1532 22-1640 22-1640 #22-1640 22-1643 22-1647 23-1713 23-1713 #23-1713 23-1716 23-1747 24-1784 24-1784 #24-1784 24-1787 24-1791 24-1814 25-1850 25-1850 #25-1850 25-1853 25-1857 26-1937 26-1937 #26-1937 26-1940 26-1942 27-2050 27-2050 #27-2050 27-2085 28-2166 28-2166 #28-2166 28-2169 28-2172 29-2211 29-2211 #29-2211 29-2214 30-2298 30-2298 #30-2298 30-2375 31-2469 31-2469 #31-2469 31-2554 32-2583 32-2583 #32-2583 32-2586 33-2633 33-2633 #33-2633 34-2727 34-2727 #34-2727 34-2730 35-2811 35-2811 #35-2811 35-2817 35-2876 36-2947 36-2947 #36-2947 36-2949 37-3051 37-3051 #37-3051 37-3149 38-3197 38-3197 #38-3197 38-3203 39-3310 39-3310 #39-3310 39-3371 40-3395 40-3395 #40-3395 41-3473 41-3473 #41-3473 42-3598 42-3598 #42-3598 43-3698 43-3698 #43-3698 44-3758 44-3758 #44-3758 44-3761 44-3798 45-3837 45-3837 #45-3837 45-3842 45-3844 45-3846 45-3848 45-3852 45-3854 45-3856 46-3903 46-3903 #46-3903 47-3941 47-3941 #47-3941 48-3971 48-3971 #48-3971 48-3983 48-3985 49-4028 49-4028 #49-4028 50-4120 50-4120 #50-4120 50-4154 51-4197 51-4197 #51-4197 52-4281 52-4281 #52-4281 52-4311 53-4394 53-4394 #53-4394 53-4478 54-4529 54-4529 #54-4529 54-4561 55-4603 55-4603 #55-4603 56-4668 56-4668 #56-4668 57-4778 57-4778 #57-4778 57-4781 58-4876 58-4876 #58-4876 58-4894 59-4935 59-4935 #59-4935 60-5036 60-5036 #60-5036 61-5182 61-5182 #61-5182 62-5291 62-5291 #62-5291 63-5381 63-5381 #63-5381 63-5424 64-5444 64-5444 #64-5444 64-5473 64-5473 #64-5473
$TPB	001152	#9-381 64-5493 64-5493 64-5493
$TPFLG	001157	#9-381 64-5493 64-5493 64-5493
$TPS	001150	#9-381 64-5493 64-5493 64-5493
$TRAP	026544	10-816 #64-5510
$TRAP2	026566	#64-5510 64-5510
$TRP	= 000014	#64-5510 64-5510 64-5510 64-5510 64-5510 #64-5510 64-5510 64-5510 64-5510 64-5510 64-5510 #64-5510 64-5510 64-5510 64-5510 64-5510 64-5510 #64-5510 64-5510 64-5510 64-5510 64-5510 64-5510 #64-5510 64-5510 64-5510 64-5510 64-5510 64-5510 #64-5510 64-5510 64-5510 64-5510 64-5510 64-5510 #64-5510 64-5510 64-5510 64-5510 64-5510 64-5510

SYMBOL CROSS REFERENCE CREF V02 SEQ 0176

SYMBOL	VALUE	REFERENCES
		#64-5510 64-5510 64-5510 64-5510 64-5510 #64-5510 64-5510 64-5510 64-5510
		64-5510 #64-5510
$TRPAD	G26600	64-5510 #64-5510
$TSTM	000236	#8-380
$TSTNM	001102	#9-381 *64-5482 64-5492 64-5492 *64-5492 64-5492 64-5492 64-5492 64-5492
		64-5502 64-5502 64-5502 64-5504
$TTYIN	025212	64-5496 64-5496 64-5496 #64-5496
$TYPBN	= ******	64-5510
$TYPDS	024246	#64-5495 64-5510 64-5510
$TYPE	023464	#64-5493 64-5508 64-5510 64-5510
$TYPEC	023676	64-5493 64-5493 64-5493 #64-5493 64-5496
$TYPEX	024016	64-5493 64-5493 #64-5493
$TYPOC	024044	#64-5494 64-5510 64-5510
$TYPON	024060	64-5494 #64-5494 64-5510
$TYPOS	024020	#64-5494 64-5510
$UNIT	001212	#9-381
$UNITM	000242	#8-380
$USWR	001224	#9-381
$VECT1	001250	#9-381
$VECT2	001252	#9-381
$XOFF	= 000023	64-5493 64-5493
$XON	= 000021	64-5493 64-5493 64-5496
$XTSTR	023202	#64-5492
$$GET4	= 000000	#64-5482 64-5482
$OFILL	024243	*64-5494 *64-5494 64-5494 #64-5494
$40CAT	= **^***	64-5492 64-5502
.$ASTA	= ******	64-5508 64-5508
.$X	= 000232	#8-380 8-380

SEQ 0177

MACRO CROSS REFERENCE CREF V02

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

MACRO CROSS REFERENCE CREF V02

MACRO NAME	REFERENCES									
TYPTXT	#8-265	64-5482	64-5482							
$ABORT	#64-5502	64-5502								
$$CMRE	#8-381									
$$CMTM	#8-381	9-381	9-381							
$$ESCA	#8-265									
$$NEWT	#8-265	11-866	12-951	13-1039	14-1107	15-1181	16-1250	17-1308	18-1364	19-1410
	20-1447	21-1529	22-1640	23-1713	24-1784	25-1850	26-1937	27-2050	28-2166	29-2211
	30-2298	31-2469	32-2583	33-2633	34-2727	35-2811	36-2947	37-3051	38-3197	39-3310
	40-3395	41-3473	42-3598	43-3698	44-3758	45-3837	46-3903	47-3941	48-3971	49-4028
	50-4120	51-4197	52-4281	53-4394	54-4529	55-4603	56-4668	57-4778	58-4876	59-4935
	60-5036	61-5182	62-5291	63-5381	64-5444	64-5473				
$$SET	#64-5510	64-5510	64-5510	64-5510	64-5510	64-5510	64-5510	64-5510	64-5510	64-5510
	64-5510	64-5510								
$$SETM	#10-816	10-816								
$$SKIP	#8-265	11-890	15-1219	16-1264	16-1281	17-1314	18-1367	19-1413	19-1425	20-1449
	21-1532	22-1643	22-1647	23-1716	23-1747	24-1787	24-1791	24-1814	25-1853	25-1857
	26-1940	26-1942	27-2085	28-2169	28-2172	29-2214	30-2375	31-2554	32-2586	34-2730
	35-2817	35-2876	36-2949	37-3149	38-3203	39-3371	44-3761	44-3798	45-3842	45-3844
	45-3846	45-3848	45-3852	45-3854	45-3856	48-3983	48-3985	50-4154	52-4311	53-4478
	54-4561	57-4781	58-4894	63-5424						
.EQUAT	#8-244	8-265								
.HEADE	#8-244	8-262								
.KT11	#8-247	8-266								
.SETUP	#8-244	10-815								
.SWRHI	#8-244	8-263								
.SWRLO	#8-244	#8-263	8-264							
.$ACT1	#8-243	8-379								
.$APTB	#9-381	9-381								
.$APTH	#8-243	8-380								
.$APTY	#8-243	64-5508								
.$CATC	#8-244	8-371								
.$CMTA	#8-245	8-381								
.$EOP	#8-245	64-5482								
.$ERRO	#8-245	64-5502								
.$ERRT	#8-245	64-5509								
.$POWE	#8-246	64-5511								
.$RDDE	#8-247	64-5498								
.$RDOC	#8-247	64-5497								
.$READ	#8-246	64-5496								
.$SAVE	#8-246									
.$SCOP	#8-245	64-5492								
.$SIZE	#8-247									
.$TRAP	#8-246	64-5510								
.$TYPD	#8-246	64-5495								
.$TYPE	#8-245	64-5493								
.$TYPO	#8-246	64-5494								