KD11-Z 11/44 UBI MAP AH-F629B-MC
CKKUABO FICHE 1 OF 1

```
1                                   .REM    @
2
3
4
5
6
7                                                   IDENTIFICATION
8                                                   --------------
9
10
11                                          PRODUCT CODE:       AC-F627B-MC
12
13                                          PRODUCT NAME:       CKKUAB0 11/44 UBI MAP
14
15                                          DATE CREATED:       OCTOBER, 1980
16
17                                          MAINTAINER:         DIAGNOSTIC ENGINEERING
18
19                                          AUTHOR:             JOHN CIUKAJ
20
21
22
23
24                                  THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE
25                                  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
26                                  CORPORATION.  DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY
27                                  FOR ANY ERRORS THAT MAY APPEAR IN THIS MANUAL.
28
29                                  THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED TO THE PURCHASER
30                                  UNDER A LICENSE FOR USE ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED
31                                  (WITH INCLUSION OF DIGITAL'S COPYRIGHT NOTICE) ONLY FOR USE IN SUCH
32                                  SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.
33
34                                  DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OR
35                                  RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY
36                                  DIGITAL.
37
38                                       COPYRIGHT (C) 1979, 1980 BY DIGITAL EQUIPMENT CORPORATION
```

```
39
40
41
42                                  HISTORY SECTION
43
44
45                          CKKUAA0 WAS RELEASED OCTOBER 1979
46                          CKKUAB0 WAS RELEASED OCTOBER 1980
```

TABLE OF CONTENTS

1. ABSTRACT

THIS PROGRAM IS DESIGNED TO BE RUN ON A PDP11/44 ON WHICH
THE CPU, CACHE(IF APPLICABLE), AND MEMORY MANAGEMENT DIAGNOSTIC
PROGRAMS HAVE BEEN RUN. THE PROGRAM WILL DETECT ALL ERRORS
THAT ORIGINATE WITH THE MAP BOX AND PROVIDE LOOPING
CAPABILITIES SO THAT THE FIELD SERVICE ENGINEER CAN VERIFY
THE FAILURES. THERE MAY BE SOME CASES, SUCH AS THE CACHE
REGISTER DATA PATH, AND CACHE MEMORY DATA PATH, WHERE

```
104                 INTERACTION BETWEEN MODULES PROHIBITS CLOSE ISOLATION, BUT
105                 THE FAILING FUNCTION WILL BE CALLED OUT SO THE FIELD SERVICE
106                 ENGINEER CAN COMPLETE THE ISOLATION PROCESS.
107
108                 IF THE PROGRAM CATCHES AN ERROR IN AN EARLY TEST
109                 AND IS ALLOWED TO CONTINUE RUNNING THROUGH THE
110                 LATER TESTS THE ERROR INDICATIONS FROM THOSE LATER
111                 TESTS MAY BE INVALID.  THIS IS DUE TO THE STRUCTURE
112                 OF THE PROGRAM, WHICH ASSUMES THAT ALL AREAS TESTED
113                 PRIOR TO THE CURRENT TEST ARE FUNCTIONING PROPERLY.
114
115                 THE ERROR TYPE OUTS WILL BE IN TABLE FORMAT, WITH
116                 A MESSAGE INDICATING THE CLASS OF ERROR, A HEADER
117                 IDENTIFYING EACH COLUMN AND A REPORT OF ALL
118                 PERTINENT DATA.  WHEN THE TEST CAN PRODUCE
119                 MORE THAN ONE ERROR CONDITION, A SUMMARY OF
120                 ERRORS WILL BE GIVEN AT THE END OF THAT TEST
121                 CONSISTING OF: THE LOGICAL 'AND' AND 'OR' OF THE DATA
122                 PREVIOUSLY REPORTED AND THE NUMBER OF ERRORS IN THIS
123                 TEST.
124                 (SEE SECTION 6.3 FOR AN EXAMPLE OF THE ERROR TYPEOUTS.)
125
```

2. REQUIREMENTS

2.1 EQUIPMENT

THE BASIC PDP-11/44 COMPUTER, INCLUDING THE CPU, CACHE, MEMORY MANAGEMENT, AND AN LA-30 OR EQUIVALENT DEVICE FOR ERROR MESSAGES.

2.2 STORAGE

THIS PROGRAM WILL REQUIRE 8K TO LOAD BUT WILL UTILIZE ALL EXISTING CORE FOR A DUAL ADDRESSING TEST OF MEMORY FROM THE UNIBUS.

2.3 PRELIMINARY PROGRAMS

THE CPU, CACHE(IF APPLICABLE), AND MEMORY MANAGEMENT DIAGNOSTICS SHOULD BE RUN BEFORE THIS PROGRAM. THE MEMORY DIAGNOSTIC SHOULD AT LEAST MAKE A QUICK VERIFY OF THE AREA OF MEMORY THIS PROGRAM WILL LOAD AND RUN IN.

```
                                3.      LOADING PROCEDURE


                                3.1     METHOD.

                                        THIS PROGRAM CAN BE LOADED FROM ANY DEVICE THAT IS
                                        SUPPORTED BY XXDP AND SHOULD BE LOADED USING THE XXDP
                                        PROCEDURE FOR THAT DEVICE.
```

```
158                                     4.      STARTING PROCEDURE
159
160
161                                     4.1     STARTING ADDRESS
162
163                                             PROGRAM STARTS AT ADDRESS 200
164
165                                     4.2     PROGRAM AND/OR OPERATOR ACTION
166
167                                             THE PROGRAM WILL IDENTIFY ITSELF.  IT WILL THEN ASK IF
168                                             YOU WANT THE EOP MESSAGES PRINTED.  NO USER RESPONSE IS
169                                             REQUIRED, AS A WAIT LOOP, ENDING AFTER 2 SECONDS,
170                                             TERMINATES THE INPUT, AND THE DIAGNOSTIC ASSUMES YOU
171                                             WANT ALL EOP MESSAGES PRINTED.  IF YOU DO RESPOND WITH
172                                             AN 'N', NO EOP MESSAGE FOR A PARTICULAR PASS WILL BE
173                                             PRINTED AS LONG AS THERE ARE NO ERROR(S) IN THAT PASS.
174                                             IF AN ERROR DOES OCCUR IN THE PARTICULAR PASS, AN EOP
175                                             MESSAGE WILL PRINT SO YOU CAN IDENTIFY WHERE IN THE PASS
176                                             COUNT THE ERROR OCCURED.  IF AT ANY TIME YOU WISH TO HAVE
177                                             A PROGRESS REPORT, TYPE ANY CHARACTER ON THE TERMINAL,
178                                             AND AT THE END OF THE PASS IT IS IN, AN EOP MESSAGE WILL
179                                             PRINT FOR THAT PASS ONLY.  IT WILL RESUME NOT PRINTING
180                                             EOP MESSAGES AS LONG AS A CHARACTER HAS NOT BEEN ENTERED
181                                             AT THE TERMINAL.  IF YOU ELECT TO HAVE EOP MESSAGES PRINTED,
182                                             (ENTERING OTHER THAN AN 'N' OR DO NOT RESPOND IN THE 2
183                                             SECOND TIME LIMIT) THEY WILL BE AT THE END OF EACH PASS,
184                                             AND WILL INDICATE THE TOTAL NUMBER OF ERRORS OCCURRING ON
185                                             THAT PASS.
186
187                                     4.3     SPECIAL STARTING PROCEDURE
188
189                                             IF IT APPEARS THAT THE CACHE IS CAUSING SOME TROUBLE AND
190                                             YOU STILL WANT TO RUN THIS PROGRAM, IT IS POSSIBLE TO RUN
191                                             WITH THE CACHE DISABLED.  SIMPLY LOAD THE CACHE CONTROL
192                                             REGISTER (17777746) WITH THE DESIRED NUMBER.  THEN LOAD
193                                             THE PC (17777707) WITH THE STARTING ADDRESS (200) AND
194                                             PRESS "CONTINUE".  THE PROGRAM WILL NOW RUN NORMALLY EXCEPT
195                                             THAT CERTAIN TESTS WILL BE SKIPPED SINCE THE CACHE IS DISABLED.
196                                             THIS FACT IS INDICATED IN THE ABSTRACT OF EACH TEST THAT
197                                             CHECKS THE CACHE CONTROL REGISTER.
198
199                                             DEFINITION OF THE BITS IN THE CACHE CONTROL REGISTER:
200                                             BIT00   -DISABLE TRAPS
201                                             BIT02   -FORCE MISS ON READ,WHERE ADDRESS BIT 12 IS 0
202                                             BIT03   -FORCE MISS ON READ,WHERE ADDRESS BIT 12 IS 1
203                                             BIT09   -UNCONDITIONAL CACHE BYPASS
```

```
                                5.      OPERATING PROCEDURE


                                5.1     OPERATIONAL SWITCH SETTINGS

                                        SW15    1=      HALT ON ERROR
                                        SW14    1=      LOOP ON TEST
                                        SW13    1=      INHIBIT ERROR TYPEOUTS
                                        SW12    1=      INHIBIT TRACE TRAP
                                        SW11    1=      INHIBIT ITERATIONS
                                        SW10    1=      BELL ON ERROR
                                        SW09    1=      LOOP ON ERROR
                                        SW08    1=      LOOP ON TEST IN SWR<05:00>
                                        SW07    1=      INHIBIT MULTIPLE ERROR TYPE OUTS
                                        SW06    1=      SELECT CACHE TESTS. THIS IS USED FOR
                                                        MFG. QUICK VERIFY STATION AND CAN BE SELECTED
                                                        BY APT SCRIPTING. THESE TESTS ASSUME THAT
                                                        ALL MODULES EXCEPT UBI MODULE ARE KNOWN GOOD.


                                5.2     SUB-ROUTINE ABSTRACTS

                                        ALL SUBROUTINE ABSTRACTS APPEAR IN THE CODE BEFORE THEIR
                                        EXPANSION AND IN THE DOCUMENT THAT IMMEDIATELY FOLLOWS THIS.
                                        BELOW IS A LIST OF THE SUBROUTINE TITLES.

                                5.2.1   MACRO LIBRARY SUBROUTINES (FOUND IN MOST PROGRAMS)

                                        SCOPE HANDLER ROUTINE
                                        ERROR HANDLER ROUTINE
                                        ERROR MESSAGE TYPE OUT ROUTINE
                                        CONVERT 16-BIT VIRTUAL ADDRESSES TO 22-BIT PHYSICAL ADDRESSES
                                        SAVE AND RESTORE R0-R5 ROUTINES
                                        TYPE ROUTINE
                                        BINARY TO OCTAL (ASCII) AND TYPE
                                        CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
                                        TRAP DECODER
                                        POWER DOWN AND UP ROUTINES
                                        DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
                                        END OF PASS ROUTINE

                                5.2.2   SUBROUTINES UNIQUE TO THIS PROGRAM

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

                                5.2.3   TRAP AND ABORT HANDLER ROUTINES

                                        CPU TRAP HANDLER ROUTINE
                                        CACHE TRAPS AND ABORTS HANDLER ROUTINE
                                        MEMORY MANAGEMENT TRAPS AND ABORTS HANDLER ROUTINE
```

5.3 RUNNING UNDER APT

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

1. E TABLE 'A' IS USED FOR APT DUMP MODE.
   A. IN ADDITION TO NORMAL CPU DIAGNOSTIC TESTS THIS TABLE WILL SELECT THE OPTIONAL CACHE TESTS, ($SWREG=100) AND INHIBIT ITERATIONS($SWREG=4000)

2. E TABLE 'B' IS USED FOR APT QV MODE WHILE RUNNING ON A MANUFACTURING QV STATION. IT ACCOMPLISHES WHAT ETABLE 'A' DOES BUT ADDITIONALLY SUPRESSES TYPEOUTS.($ENVM=240)

3. ETABLE 'C' IS USED FOR APT QV OR RUNTIME MODES WHILE RUNNING ON SYSTEMS OTHER THAN MFG. QV STATIONS. THIS TABLE DESELECTS THE OPTIONAL CACHE TESTS.

1ST PASS RUN TIME	LONGEST TEST TIME	ADDITIONAL RUN TIME
10	5	0

.....	E TABLES	.....	
	A	B	C
E-MODE/S-MODE ($ENVM/$ENV)	200/000	240/001	240/001
SWITCH REGISTER 1 ($SWREG)	004100	0004100	004000
SWITCH REGISTER 2	000000	000000	000000
CPU TYPE/OPTIONS	00/0000	00/0000	00/0000

```
                        6.      ERRORS


                        6.1     ERROR HALTS AND DESCRIPTION

                                WHEN AN ERROR IS DETECTED AN 'ERROR' (EMT)
                                INSTRUCTION IS EXECUTED AND THE 'ERROR HANDLER ROUTINE'
                                CHECKS THE SWITCH REGISTER FOR MODE SELECTED.
                                THE PROGRAM WILL:
                                HALT ON ERROR           IF SW15=1
                                INHIBIT ERROR TYPE OUT  IF SW13=1
                                RING BELL ON ERROR      IF SW10=1
                                LOOP ON ERROR           IF SW9=1


                        6.2     ERROR RECOVERY

                                IF SW09=1, THE PROGRAM WILL LOOP BACK TO
                                THE POINT WHERE THE INSTRUCTION THAT CAUSED THE ERROR WAS
                                EXECUTED, WITHOUT ALLOWING ANY OF THE CONDITIONS TO CHANGE.
                                THIS WILL PROVIDE THE TIGHTEST POSSIBLE SCOPE LOOP.
                                IF SW09=0, EACH ERROR WILL BE REPORTED AND
                                LOGGED AND, AT THE END OF EACH TEST, A SUMMARY OF ALL ERRORS
                                OCCURRING IN THAT TEST WILL BE PROVIDED.  THE SUMMARY
                                CONSISTS OF THE LOGICAL AND AND OR OF THE ADDRESS AND/OR
                                DATA THAT WAS WRONG.


                        6.3     SAMPLE ERROR TYPE OUTS
                                SEE ''$ERRTB:'' FOR SAMPLE ERROR TYPEOUTS.

                        6.3.1   MULTIPLE TYPE ERRORS: AN EXAMPLE:

                                THE FOLLOWING REGISTERS TIMED OUT WHEN REFERENCED
                                REG.ADR TESTNO  ERRORPC
                                170210  000001  015226
                                170212  000001  015232
                                  .       .       .
                                  .       .       .
                                          TO
                                  .       .       .
                                  .       .       .
                                170372  000001  015232
                                170374  000001  015232
                                170376  000001  015232

                                SUMMARY OF MAP REGISTERS THAT TIMED OUT ON READ
                                REGADRS REGADRS
                                 ''OR''    ''AND''  #ERRORS TESTNO  ERRORPC
                                170376  170210       32  000001  010530
```

7. RESTRICTIONS

7.1 STARTING RESTRICTIONS

NONE

7.2 OPERATING RESTRICTIONS

NONE

```
353
354                    8.      MISCELLANEOUS
355
356
357                    8.1     EXECUTION TIME
358
359                            THE RUN TIME FOR ANY PASS IS APPROXIMATELY 3 SECONDS.
360
361                    8.2     ADDRESS GENERATION IN THE PDP-11/44
362
363                            THE FOLLOWING IS AN EXAMPLE OF HOW A MEMORY ADDRESS
364                            IS GENERATED BY THE UNIBUS MAP.  THIS ASSUMES THAT
365                            THE ADDRESS ORIGINATES IN THE CPU BUT THE PROCESS CAN
366                            APPLY TO ANY UNIBUS ADDRESS, STARTING AT LINE C2.
367
368
369                A.  VIRTUAL ADDRESS                          15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
370
371                A1. P.A.R. PAGE NUMBER (0-7)                 15 14 13
372
373                A2. OFFSET (FROM VIRTUAL ADDRESS)                     12 11 10 09 08 07 06 05 04 03 02 01 00
374
375                B.  P.A.R.[PAGE NO.] +  15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
376                                        ------------------------------------------------
377                C.  PHYS ADDRS (A2+B)   21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
378
379                C1. 17XXXXXX=> U.B.ADR. 21 20 19 18
380
381                C2. MAPPING REG.NO.(0-36)          17 16 15 14 13
382
383                C3. OFFSET                                       12 11 10 09 08 07 06 05 04 03 02 01 00
384
385                D.  MAP REG.[NO.]    +  21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01
386                                        ------------------------------------------------------------
387                E.  PHYS ADDR (C3+D)    21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
388
389
390                    DESCRIPTION OF LINES:
391                    A: VIRTUAL ADDRESS (16 BITS)
392                            A1: UPPER 3 BITS OF VIRTUAL ADDRESS, USED TO
393                                SELECT A PAGE ADDRESS REGISTER (PAR)
394                            A2: LOWER 13 BITS OF VIRTUAL ADDRESS, ADDED TO
395                                SELECTED PAR
396
397                    B: PAGE ADDRESS REGISTER (16 BITS), IN ADDITION PROCESS THIS GETS
398                       LEFT SHIFTED 6 BITS BEFORE ADDITION TO A2
399
400                    C: PHYSICAL ADDRESS CREATED BY MEMORY MANAGEMENT, (22 BITS)
401                            C1: IF UPPER 4 BITS ARE ALL ONES THEN BITS <17:00> GO OUT ON UNIBUS
402                            C2: IF MAP RELOCATION IS ENABLED THEN BITS <17:13> SELECT ONE
403                                OF THE 36 (OCTAL) MAP REGISTERS.
404                            C3: LOWER 13 BITS OF UNIBUS ADDRESS, ADDED TO SELECTED MAP REGISTER
405
406                    D: MAP REGISTER (22 BITS), ADDED TO BITS <12:00> OF UNIBUS ADDRESS
407
408                    E: PHYSICAL ADDRESS GENERATED BY UNIBUS MAP AND SENT TO THE CACHE.
```

```
409                                 9.      PROGRAM DESCRIPTION
410
411
412                                         THE ASSEMBLED LISTING,CKKUAB0.SEQ, HAS A PARAGRAPH DESCIRBING
413                                         EACH OF THE TESTS.  THE PARAGRAPH WILL INDICATE IF THE TEST IS
414                                         RUN CONDITIONALLY ON THE STATUS ON THE CACHE CONTROL REGISTER.@
```

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

```
559                             .SBTTL  OPERATIONAL SWITCH SETTINGS
                                ;*
                                ;*      SWITCH                  USE
                                ;*      ------          --------------------
                                ;*        15            HALT ON ERROR
                                ;*        14            LOOP ON TEST
                                ;*        13            INHIBIT ERROR TYPEOUTS
                                ;*        12            INHIBIT TRACE TRAP
                                ;*        11            INHIBIT ITERATIONS
                                ;*        10            BELL ON ERROR
                                ;*         9            LOOP ON ERROR
                                ;*         8            LOOP ON TEST IN SWR<4:0>
560                             ;*         7            INHIBIT MULTIPLE ERROR TYPEOUTS
                                ;*         6            SELECT CACHE-CIS TESTS
561                             ;*         5            SELECT MEMORY ON UNIBUS TEST
```

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

```
020000          SW13=   20000
010000          SW12=   10000
004000          SW11=   4000
002000          SW10=   2000
001000          SW09=   1000
000400          SW08=   400
```

```
000200                  SW07=   200
000100                  SW06=   100
000040                  SW05=   40
000020                  SW04=   20
000010                  SW03=   10
000004                  SW02=   4
000002                  SW01=   2
000001                  SW00=   1
001000                          SW9=SW09                ;
000400                          SW8=SW08                ;
000200                          SW7=SW07                ;
000100                          SW6=SW06                ;
000040                          SW5=SW05                ;
000020                          SW4=SW04                ;
000010                          SW3=SW03                ;
000004                          SW2=SW02                ;
000002                          SW1=SW01                ;
000001                          SW0=SW00                ;
                        ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
100000                  BIT15=  100000
040000                  BIT14=  40000
020000                  BIT13=  20000
010000                  BIT12=  10000
004000                  BIT11=  4000
002000                  BIT10=  2000
001000                  BIT09=  1000
000400                  BIT08=  400
000200                  BIT07=  200
000100                  BIT06=  100
000040                  BIT05=  40
000020                  BIT04=  20
000010                  BIT03=  10
000004                  BIT02=  4
000002                  BIT01=  2
000001                  BIT00=  1
001000                          BIT9=BIT09              ;
000400                          BIT8=BIT08              ;
000200                          BIT7=BIT07              ;
000100                          BIT6=BIT06              ;
000040                          BIT5=BIT05              ;
000020                          BIT4=BIT04              ;
000010                          BIT3=BIT03              ;
000004                          BIT2=BIT02              ;
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
```

```
                                H  2
000114                  CACHVEC=114             ;;CACHE ERROR INTERRUPT VECTOR
000240                  PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
000250                  MMVEC=  250             ;;MEMORY MANAGEMENT VECTOR
```

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

```
                        .SBTTL  CPU REGISTER DEFINITIONS
177760                  SIZELO = 177760         ;;MEMORY SIZE REGISTER NUMBER TO PUT INTO A PAR
                                                ;;TO GET TO THE LAST 32 WORDS OF MEMORY
177762                  SIZEHI = 177762         ;;HIGH SIZE REGISTER, RESERVED FOR FUTURE USE
                                                ;;CURRENTLY ALL ZERO
177764                  SYSTID = 177764         ;;SYSTEM ID REGISTER
177766                  CPUERR = 177766         ;;CPU ERROR REGISTER HOLDS CONDITION THAT CAUSED
                                                ;;THE TRAP TO ERRVEC (000004)
```

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

```
172210          SIPDR4= 172210
172212          SIPDR5= 172212
172214          SIPDR6= 172214
172216          SIPDR7= 172216
                ;*SUPERVISOR ''D'' PAGE DESCRIPTOR REGISTERS
172220          SDPDR0= 172220
```

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

```
172356          KIPAR7= 172356
                ;*KERNEL "D" PAGE ADDRESS REGISTERS                    SEQ 0026
172360          KDPAR0= 172360
172362          KDPAR1= 172362
172364          KDPAR2= 172364
172366          KDPAR3= 172366
```

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

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

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

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

```
    >66                             .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
```

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

```
568                                 .SBTTL          ACT11 HOOKS
                                    ;;*****************************************************************
                                    ;*THE FOLLOWING LOCATIONS ARE SETUP TO BE USED WITH ACT11
                                    ;*
                                    ;*LOCATION 46 WILL CONTAIN THE ADDRESS OF THE LOCICAL
                                    ;*END OF THE PROGRAM.
                                    ;*LOCATION 52 IS USED TO SPECIFY PROGRAM OPERATING REQUIREMENTS
                                    ;*AND/OR RESTRICTIONS. THIS IS ACCOMPLISHED BY SETTING VARIOUS BITS
                                    ;*TO A ONE OR A ZERO. THE BITS USED AND THERE MEANING ARE:
                                    ;*
                                    ;*      BIT 15=1 PROGRAM SHOULD BE POWER FAILED WHILE RUNNING
                                    ;*            =0 NO POWER FAIL DESIRED
                                    ;*
                                    ;*      BIT 14=1 PROGRAM RUN TIME IS MEMORY SIZE DEPENDENT
                                    ;*            =0 RUN TIME IS NOT MEMORY SIZE DEPENDENT
                                    ;*
                                    ;*      BITS 13-0  MUST BE ZERO'S
        000204                      $SVPC=.                 ;;SAVE LOCATION COUNTER
        000046                      .=46                    ;;SET LOCATION COUNTER
000046  021534                      .WORD   $ENDAD          ;;SET LOC.46 TO ADDRESS $ENDAD
        000052                      .=52                    ;;SET LOCATION COUNTER
000052  000000                      .WORD   0               ;;SET LOC.52 TO ZERO
        000204                      .=$SVPC                 ;; RESTORE LOCATION COUNTER
```

```
 570                            .SBTTL  COMMON TAGS
                                ;;*************************************************************
                                ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
                                ;*USED IN THE PROGRAM.
        001100                          .=1100
001100                          $CMTAG:                              ;;START OF COMMON TAGS
                                ;$PASS: .WORD   0                    ;&& CONTAINS PASS COUNT (ELIMINATED FOR CKKUAB)
001100     000                  $TSTNM: .BYTE   0                    ;;CONTAINS THE TEST NUMBER
001101     000                  $ERFLG: .BYTE   0                    ;;CONTAINS ERROR FLAG
001102  000000                  $ICNT:  .WORD   0                    ;;CONTAINS SUBTEST ITERATION COUNT
001104  000000                  $LPADR: .WORD   0                    ;;CONTAINS SCOPE LOOP
001106  000000                  $LPERR: .WORD   0                    ;;CONTAINS SCOPE RETURN FOR ERRORS
001110  000000                  $ERTTL: .WORD   0                    ;;CONTAINS TOTAL ERRORS DETECTED
001112     000                  $ITEMB: .BYTE   0                    ;;CONTAINS ITEM CONTROL BYTE
001113     001                  $ERMAX: .BYTE   1                    ;;CONTAINS MAX. ERRORS PER TEST
001114  000000                  $ERRPC: .WORD   0                    ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001116  000000                  $GDADR: .WORD   0                    ;;CONTAINS  OF 'GOOD' DATA
001120  000000                  $BDADR: .WORD   0                    ;;CONTAINS  OF 'BAD' DATA
001122  000000                  $GDDAT: .WORD   0                    ;;CONTAINS 'GOOD' DATA
001124  000000                  $BDDAT: .WORD   0                    ;;CONTAINS 'BAD' DATA
001126  000000  000000  000000          .WORD   0,0,0                ;;RESERVED--NOT TO BE USED
001134  177560                  $TKS:   177560                       ;;TTY KBD STATUS
001136  177562                  $TKB:   177562                       ;;TTY KBD BUFFER
001140  177564                  $TPS:   177564                       ;;TTY PRINTER STATUS REG.
001142  177566                  $TPB:   177566                       ;;TTY PRINTER BUFFER REG.
001144     000                  $NULL:  .BYTE   0                    ;;CONTAINS NULL CHARACTER FOR FILLS
001145     002                  $FILLS: .BYTE   2                    ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001146     012                  $FILLC: .BYTE   12                   ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001147     000                  $TPFLG: .BYTE   0                    ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
001150  177570                  SWR:    .WORD   177570               ;&& LOCATION FOR STORING SWITCH REGISTER
001152  177570                  DISPLAY: .WORD  177570               ;&& DISPLAY REGISTER
001154  000000                  $REGAD: .WORD   0                    ;;CONTAINS THE  FROM
                                                                     ;;WHICH  ($REG0) WAS OBTAINED
        000006                          .REPT   $CM3
001156  000000                  $REG0:  .WORD   0                    ;;CONTAINS (($REGAD)+0)
001160  000000                  $REG1:  .WORD   0                    ;;CONTAINS (($REGAD)+2)
001162  000000                  $REG2:  .WORD   0                    ;;CONTAINS (($REGAD)+4)
001164  000000                  $REG3:  .WORD   0                    ;;CONTAINS (($REGAD)+6)
001166  000000                  $REG4:  .WORD   0                    ;;CONTAINS (($REGAD)+10)
001170  000000                  $REG5:  .WORD   0                    ;;CONTAINS (($REGAD)+12)
        000007                  .REPT   7
001172  000000                  $TMP0:  .WORD   0                    ;;USER DEFINED
001174  000000                  $TMP1:  .WORD   0                    ;;USER DEFINED
001176  000000                  $TMP2:  .WORD   0                    ;;USER DEFINED
001200  000000                  $TMP3:  .WORD   0                    ;;USER DEFINED
001202  000000                  $TMP4:  .WORD   0                    ;;USER DEFINED
001204  000000                  $TMP5:  .WORD   0                    ;;USER DEFINED
001206  000000                  $TMP6:  .WORD   0                    ;;USER DEFINED
001210  000000                  $TIMES: 0                            ;;MAX. NUMBER OF ITERATIONS
001212  000000                  $ESCAPE:0                            ;;ESCAPE ON ERROR
001214     207     377     377  $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
```

```
001220     077              $QUES:  .ASCII  /?/             ;;QUESTION MARK
001221     015              $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN                                   SEQ 0035
001222     012      000     $LF:    .ASCIZ  <12>            ;;LINE FEED
001224  000000              PADRSL: .WORD   0               ;HOLDS THE LOWER 16 BITS OF A 22 BIT ADDRESS
                                                            ;GENERATED FOR TYPE OUT.
001226  000000              PADRSH: .WORD   0               ;HOLDS THE UPPER 6 BITS OF A 22 BIT ADDRESS
```

```
001230  000000  000077          ADRAND: .WORD   0,77           ;GENERATED FOR TYPE OUT.
001234  000000  000077          ADDROR: .WORD   0,77           ;LOGICAL AND OF FAILING ADDRESSES
001240  000000  000077          DATAND: .WORD   0,77           ;LOGICAL OR OF FAILING ADDRESSES
001244  000000  000077          DATAOR: .WORD   0,77           ;LOGICAL AND OF BAD DATA
001250  000000                  PATAND: .WORD   0              ;LOGICAL OR OF BAD DATA
001252  000000                  PATTOR: .WORD   0              ;LOGICAL AND OF PATTERN LOADED
001254  000000                  LOWEST: .WORD   0              ;LOGICAL OR OF PATTERN LOADED
                                                               ;HOLDS NUMBER TO PUT IN PAR TO CAUSE THE
                                                               ;LOWEST USABLE MAP REGISTER TO RESPOND
001256  000000                  HIGEST: .WORD   0              ;HOLDS NUMBER TO PUT IN PAR TO CAUSE THE
                                                               ;HIGHEST USABLE MAP REGISTER TO RESPOND
001260  000000                  UBMLOW: .WORD   0              ;HOLDS NUMBER TO PUT IN PAR TO SIGNAL 1ST
                                                               ;ADDRESS OF UNIBUS MEMORY
001262  000000                  UBMHI:  .WORD   0              ;HOLDS NUMBER TO PUT IN PAR TO SIGNAL LAST
                                                               ;BLOCK OF 4K OF UNIBUS MEMORY
001264  000000                  MMRLOW: .WORD   0              ;HOLDS LOWEST MAP REGISTER NUMBER FROM 'LOWEST:'
001266  000000                  MMRHI:  .WORD   0              ;HOLDS HIGHEST MAP REGISTER NUMBER FROM 'HIGEST:'
001270  000000                  UBRLOW: .WORD   0              ;HOLDS LOWEST MAP REGISTER NUMBER FROM 'UBMLOW:'
001272  000000                  UBRHI:  .WORD   0              ;HOLDS HIGHEST MAP REGISTER NUMBER FROM 'UBMHI:'
001274  000000                  BUPWIN: .WORD   0              ;HOLDS LOWEST USEABLE PAR OF UPPER WINDOW
001276  000000                  LREGL:  .WORD   0              ;HOLDS I/O PAGE ADDR OF LOW 16 BITS OF
                                                               ;THE LOWEST USABLE MAP REGISTER
001300  000000                  LREGU:  .WORD   0              ;HOLDS I/O PAGE ADDR OF HIGH 6 BITS OF
                                                               ;OF THE LOWEST USABLE MAP REGISTER
001302  000000                  FRRCNT: .WORD   0              ;MULTIPLE ERROR ERROR COUNTER
001304  000000                  CNTR:   .WORD   0              ;AUXILIARY COUNTER
001306  000000                  FLAG:   .WORD   0              ;FLAG TO INDICATE TO LAST PROGRAM PASS N
001310  000000                  TESTNO: .WORD   0              ;HOLDS TEST NUMBER FOR ERROR TYPE OUTS
001312  000000                  CPUEXP: .WORD   0              ;HOLDS THE EXPECTED CPU ERROR CODE
001314  000000                  PCPUER: .WORD   0              ;HOLDS RECEIVED CPU ERROR CONDITION
001316  000000                  PPARER: .WORD   0              ;HOLDS RECEIVED PARITY ERROR CONDITION
001320  000000                  PCONTR: .WORD   0              ;HOLDS CONTENTS OF CONTROL REGISTER
001322  000000                  PMAINT: .WORD   0              ;HOLDS CONTENTS OF MAINTENENCE REGISTER
001324  000000                  BADPC:  .WORD   0              ;HOLDS PC OF INST THAT CAUSED TRAP
001326  000000                  OLDPC:  .WORD   0              ;HOLDS THE RETURN ADDRESS AFTER A TRAP
001330  000000                  OLDPS:  .WORD   0              ;HOLDS THE OLD PROCESSOR STATUS
001332  000000                  OLDPSW: .WORD   0              ;HOLDS OLD PSW FOR TBITRESTORE
001334  000000                  PMMR0:  .WORD   0              ;HOLDS CONTENTS OF MMR0 AFTER TRAP
001336  000000                  PMMR1:  .WORD   0              ;HOLDS CONTENTS OF MMR1 AFTER TRAP
001340  000000                  PMMR2:  .WORD   0              ;HOLDS CONTENTS OF MMR2 AFTER TRAP
001342  000000                  RSIZE:  .WORD   0              ;WILL HOLD P.A.R. DATA FOR TOP OF MEMORY
001344  000000                  RETRY:  .WORD   0              ;RETRY FLAG IN CASE OF PARITY ABORTS
001346  000000                  NXTTST: .WORD   0              ;LOCATION TO HOLD ESCAPE ADDRESS ON
                                                               ;PARITY ERRORS.
001350  000200                  DATA:   .WORD   200            ;PATTERN TO BE USED TO LOAD INTO MEMORY
```

```
                                .SBTTL  ERROR POINTER TABLE
                                ;;*****************************************************************
                                ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
                                ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
                                ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
                                ;*NOTE1:         IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
                                ;*NOTE2:         EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
                                ;*       EM              ;;POINTS TO THE ERROR MESSAGE
                                ;*       DH              ;;POINTS TO THE DATA HEADER
                                ;*       DT              ;;POINTS TO THE DATA
                                ;*       DF              ;;POINTS TO THE DATA FORMAT
    001352                      $ERRTB:
571
572 001352  022062              ;ITEM 1
573 001354  026052                      .WORD   EM1     ;NOT THE CORRECT CPU TRAP CONDITION THROUGH ERRVEC (#004)
574 001356  030262                      .WORD   DH1     ;RECEIVD EXPECTD TESTNO  PC AT ABORT
575 001360  030722                      .WORD   DT1     ;PCPUER,CPUEXP,TESTNO,BADPC,0
576                                     .WORD   DF1     ; 0, 0, 0, 0
577
578 001362  022147              ;ITEM 2
579 001364  026111                      .WORD   EM2     ;UNEXPECTED CPU TRAP THROUGH ERRVEC (#004)
580 001366  030274                      .WORD   DH2     ;RECEIVD TESTNO  PC AT ABORT
581 001370  030722                      .WORD   DT2     ;PCPUER,TESTNO,BADPC,0
582                                     .WORD   DF1     ; 0, 0, 0
583
584 001372  022221              ;ITEM 3
585 001374  026140                      .WORD   EM3     ;MEMORY MANAGEMENT TRAP, MEMORY MANAGEMENT STATUS REGISTERS
586                                     .WORD   DH3     ;STATUS  AUTOI/D VIRTADR
587 001376  030304                                      ;REGISTR REGISTR REGISTR TESTNO  PC AT ABORT
588 001400  030722                      .WORD   DT3     ;PMMR0,PMMR1,PMMR2,TESTNO,BADPC,0
589                                     .WORD   DF1     ; 0, 0, 0, 0, 0
590
591 001402  022327              ;ITEM 4
592 001404  026237                      .WORD   EM4     ;SUMMARY OF MAP REGISTERS THAT TIMED OUT ON READ
593                                     .WORD   DH4     ;REGADRS   REGADRS
594 001406  030320                                      ; "OR"      "AND"    #ERRORS TESTNO  ERR PC
595 001410  030727                      .WORD   DT4     ;ADDROR,ADRAND,ERRCNT,TESTNO,$ERRPC,0
596                                     .WORD   DF4     ; 2, 2, 1, 0, 0
597
598 001412  022407              ;ITEM 5
599 001414  026334                      .WORD   EM5     ;SUMMARY OF DUAL ADDRESSING ERRORS ON LOADING MAP REGISTERS
600                                     .WORD   DH5     ;REGLOAD   REGLOAD   REGDUAL   REGDUAL
601 001416  030334                                      ; "OR"      "AND"     "OR"      "AND"    #ERRORS TESTNO
602 001420  030734                      .WORD   DT5     ;ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO,0
603                                     .WORD   DF5     ; 2, 2, 2, 2, 1, 0
604
605 001422  022502              ;ITEM 6
606 001424  026471                      .WORD   EM6     ;SUMMARY OF BIT PATTERN FAILURES IN LOWER 16 BITS OF MAP REGISTERS
607                                     .WORD   DH6     ;MAPREG    MAPREG    EXPECTD EXPECTD RECEIVD RECEIVD
608 001426  030352                                      ; "OR"      "AND"     "OR"      "AND"    "OR"      "AND"  #ERRORS TESTNO
609 001430  030742                      .WORD   DT6     ;ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0
                                        .WORD   DF6     ; 2, 2, 0, 0, 0, 0, 1, 0
```

```
                                ;ITEM 7
001432  022604                          .WORD   EM7     ;SUMMARY OF BIT PATTERN FAILURES IN UPPER 6 BITS OF MAP REGISTERS
001434  026471                          .WORD   DH6     ;MAPREG    MAPREG    EXPECTD EXPECTD RECEIVD RECEIVD
                                                        ; "OR"      "AND"      "OR"    "AND"   "OR"    "AND"   #ERRORS TESTNO
001436  030352                          .WORD   DT6     ;ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0
001440  030742                          .WORD   DF6     ; 2, 2, 0, 0, 0, 0, 1, 0

                                ;ITEM 10
001442  022705                          .WORD   EM10    ;CAN'T GET TO MAIN MEMORY FROM UNIBUS WITH THE MAP OFF
                                                        ;SO I'LL JUMP TO THE SIZE JUMPER TEST FOR VERIFICATION
001444  026660                          .WORD   DH10    ;TESTNO  ERR PC
001446  030374                          .WORD   DT10    ;TESTNO,$ERRPC,0
001450  030722                          .WORD   DF1     ;0, 0

                                ;ITEM 11
001452  023061                          .WORD   EM11    ;SUMMARY OF COUNT PATTERN FAILURES ON THE UNIBUS DATA  PATH
001454  026677                          .WORD   DH11    ;EXPECTD EXPECTD RECEIVD RECEIVD
                                                        ; "OR"    "AND"   "OR"    "AND"  #ERRORS TESTNO
001456  030402                          .WORD   DT11    ;PATTOR,PATAND,DATAOR,DATAND,ERRCTN,TESTNO,0
001460  030752                          .WORD   DF11    ; 0, 0, 0, 0, 1, 0

                                ;ITEM 12
001462  023153                          .WORD   EM12    ;UNIBUS MAP IS RELOCATING WHEN NOT ENABLED
001464  026660                          .WORD   DH10    ;TESTNO  ERR PC
001466  030374                          .WORD   DT10    ;TESTNO,$ERRPC,0
001470  030722                          .WORD   DF1     ; 0, 0

                                ;ITEM 13
001472  023225                          .WORD   EM13    ;CANNOT USE ANY OF THE MAP REGISTERS OR PHYSICAL
                                                        ;ADDRESS BIT14 IS STUCK LOW, MUST RESTART PROGRAM
                                                        ;IF YOU DON'T LOOP ON THIS PROBLEM.
001474  026660                          .WORD   DH10    ;TESTNO  ERR PC
001476  030374                          .WORD   DT10    ;TESTNO,$ERRPC,0
001500  030722                          .WORD   DF1     ; 0, 0

                                ;ITEM 14
001502  023431                          .WORD   EM14    ;SUMMARY OF UNIBUS ADDRESS ERRORS, WITH MAP RELOCATION DISABLED
001504  026677                          .WORD   DH11    ;EXPECTD   EXPECTD   RECEIVD RECEIVD
                                                        ; "OR"      "AND"     "OR"    "AND"  #ERRORS TESTNO
001506  030420                          .WORD   DT14    ;ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO,0
001510  030760                          .WORD   DF14    ; 2, 2, 0, 0, 1, 0

                                ;ITEM 15
001512  023534                          .WORD   EM15    ;MAIN MEMORY TIME OUT OVER THE UNIBUS DID NOT OCCUR PROPERLY.
001514  027034                          .WORD   DH15    ;CONDITN CONDITN
                                                        ;EXPECTD RECEIVD TESTNO  ERR PC
001516  030436                          .WORD   DT15    ;CPUEXP,PCPUER,TESTNO,$ERRPC,0
001520  030722                          .WORD   DF1     ; 0, 0, 0, 0

                                ;ITEM 16
001522  023627                          .WORD   EM16    ;SUMMARY OF DUAL MAPPING ERRORS
001524  026677                          .WORD   DH11    ;EXPECTD   EXPECTD   RECEIVD   RECEIVD
                                                        ; "OR"      "AND"     "OR"      "AND"    #ERRORS TESTNO
001526  030420                          .WORD   DT14    ;ADDROR,ADRAND,DATAOR,DATAND,$ERRPC,TESTNO,0
001530  030734                          .WORD   DF5     ; 2, 2, 2, 2, 1, 0
```

```
                                ;ITEM 17
001532  023725                          .WORD   EM17    ;NO UNIBUS MEMORY EXISTS
001534  026660                          .WORD   DH10    ;TESTNO  ERR PC
001536  030374                          .WORD   DT10    ;TESTNO,$ERRPC,0
001540  030722                          .WORD   DF1     ; 0, 0

                                ;ITEM 20
001542  023755                          .WORD   EM20    ;INTERRUPT/ABORT LOGIC TESTS TRAP TO LOCATION 114 DID NOT OCCUR
001544  026660                          .WORD   DH10    ;TESTNO  ERR PC
001546  030374                          .WORD   DT10    ;TESTNO,$ERRPC,0
001550  030722                          .WORD   DF1     ; 0, 0

                                ;ITEM 21
001552  024054                          .WORD   EM21    ;INTERRUPT/ABORT TESTS R4 WAS OVERWRITTEN WITH
                                                        ;DATA INDICATING THAT INSTRUCTION WAS NOT ABORTED
001554  026660                          .WORD   DH10    ;TESTNO  ERR PC
001556  030374                          .WORD   DT10    ;TESTNO,$ERRPC,0
001560  030722                          .WORD   DF1     ; 0, 0

                                ;ITEM 22
001562  024132                          .WORD   EM22    ;INTERRUPT/ABORT TESTS TRAP DID NOT OCCUR DUE TO ABORT
001564  026660                          .WORD   DH10    ;TESTNO  ERR PC
001566  030374                          .WORD   DT10    ;TESTNO,$ERRPC,0
001570  030722                          .WORD   DF1     ; 0, 0

                                ;ITEM 23
001572  024220                          .WORD   EM23    ;LMA NOT LOADED PROPERLY
001574  027113                          .WORD   DH23    ;TESTNO  ERR PC  LMAEXP    LMARCV
001576  030450                          .WORD   DT23    ;TESTNO,$ERRPC,EADRES,EADRS2,0
001600  030766                          .WORD   DF23    ; 0, 0, 2, 2

                                ;ITEM 24
001602  024250                          .WORD   EM24    ;LMA FORCE JUMPER BIT NOT ZERO
001604  027154                          .WORD   DH24    ;TESTNO  ERR PC  LMAEXP  LMARCV
001606  030462                          .WORD   DT24    ;TESTNO,$ERRPC,$REG1,LMAHI,0
001610  030722                          .WORD   DF1     ; 0, 0, 0, 0

                                ;ITEM 25
001612  024306                          .WORD   EM25    ;LMA FORCE JUMPER BIT NOT SET
001614  027154                          .WORD   DH24    ;TESTNO  ERR PC  LMAEXP  LMARCV
001616  030462                          .WORD   DT24    ;TESTNO,$ERRPC,$REG1,LMAHI,0
001620  030722                          .WORD   DF1     ; 0, 0, 0, 0

                                ;ITEM 26
001622  024343                          .WORD   EM26    ;LMA CONTROL BITS INCORRECT
001624  027154                          .WORD   DH24    ;TESTNO  ERR PC  LMAEXP  LMARCV
001626  030474                          .WORD   DT26    ;TESTNO,$ERRPC,$TMP0,$REG2,0
001630  030722                          .WORD   DF1     ; 0, 0, 0, 0

                                ;ITEM 27
001632  024376                          .WORD   EM27    ;FORCE JUMPER BIT FAILS TO REVERT MAP REGISTER STATUS TO DEFAULT
001634  027213                          .WORD   DH27    ;TESTNO  ERR PC  LMARCV  KIPAR4
001636  030506                          .WORD   DT27    ;TESTNO,$ERRPC,$TMP0,KIPAR4,0
001640  030722                          .WORD   DF1     ; 0, 0, 0
```

```
719                                     ;ITEM 30
720 001642  024476                              .WORD   EM30    ;KIPAR5 NOT LOADED PROPERLY - SKIPPING NEXT TEST
721 001644  027252                              .WORD   DH30    ;TESTNO  ERR PC  PR5EXP  PR5RCV
722 001646  030520                              .WORD   DT30    ;TESTNO,$ERRPC,$TMP5,KIPAR5,0
723 001650  030722                              .WORD   DF1     ; 0, 0, 0, 0
```

```
                                ER200:                  ;THIS IS THE STARTING POINT FOR ERROR MESSAGES
                                                        ;201 THROUGH 377.  THEY ARE USED FOR MULTIPLE
                                                        ;ERROR MESSAGES.


                                ;ITEM 201
001652  024556                        .WORD   EM201   ;THE FOLLOWING REGISTERS TIMED OUT WHEN READ
001654  027311                        .WORD   DH201   ;REGADRS TESTNO  ERR PC
001656  030532                        .WORD   DT201   ;EADRES,TESTNO,$ERRPC,0
001660  030772                        .WORD   DF201   ; 2, 0, 0

                                ;ITEM 202
001662  024632                        .WORD   EM202   ;THE FOLLOWING ARE DUAL ADDRESSING ERRORS IN THE UNIBUS MAP
001664  027340                        .WORD   DH202   ;MAPREG    MAPREG    NON-ZER
                                                        ;TESTING   DUALED    CONTNTS TESTNO  ERR PC
001666  030542                        .WORD   DT202   ;EADRES,EADRS2,$TMP3,TESTNO,$ERRPC,0
001670  030742                        .WORD   DF6     ; 2, 2, 0, 0, 0

                                ;ITEM 203
001672  024725                        .WORD   EM203   ;THE BIT PATTERN THROUGH THE MAP REGISTERS FAILED
001674  027447                        .WORD   DH203   ;REGADRS   PATTRN  EXPCTD  RECEVD  TESTNO  ERR PC
001676  030556                        .WORD   DT203   ;EADRS2,$TMP0,$REG4,$REG3,TESTNO,$ERRPC,0
001700  030772                        .WORD   DF201   ; 2, 0, 0, 0, 0, 0

                                ;ITEM 204
001702  025006                        .WORD   EM204   ;UNIBUS DATA PATH COUNT PATTERN FAILURE
001704  027530                        .WORD   DH204   ;EXPECTD RECEIVD ADDRSLOAD TESTNO  ERR PC
001706  030574                        .WORD   DT204   ;$TMP0,$TMP1,$REG2,TESTNO,$ERRPC,0
001710  031000                        .WORD   DF204   ;0, 0, 3, 0, 0

                                ;ITEM 205
001712  025055                        .WORD   EM205   ;UNIBLS ADDRESSING ERRORS, MAP RELOCATION DISABLED
001714  027601                        .WORD   DH205   ;ADDRESS   ADDRESS
                                                        ;EXPECTD   RECEIVD   TESTNO ERR PC
001716  030610                        .WORD   DT205   ;EADRES,EADRS2,TESTNO,$ERRPC,0
001720  030760                        .WORD   DF14    ; 2, 2, 0, 0

                                ;ITEM 206
001722  025137                        .WORD   EM206   ;DATA PATTERN NOT CORRECT
001724  027666                        .WORD   DH206   ;ADDRESS  EXPCTD  RECVED  TESTNO  ERR PC
001726  030622                        .WORD   DT206   ;EADRES,$TMP4,$TMP5,TESTNO,$ERRPC,0
001730  030772                        .WORD   DF201   ; 2, 0, 0, 0, 0

                                ;ITEM 207
001732  025170                        .WORD   EM207   ;REFERENCED MAP REGISTER 0 WITH ADDRESS ONE BIT DIFFERENT THAN 17770
001734  027736                        .WORD   DH207   ;ADDRUSED  BITDIFF TESTNO  ERR PC
001736  030636                        .WORD   DT207   ;EADRES,$REG0,TESTNO,$ERRPC,0
001740  030772                        .WORD   DF201   ;2, 0, 0, 0

                                ;ITEM 210
001742  025277                        .WORD   EM210   ;MAP REGISTER UNDER TEST DID NOT RESPOND IN DUAL MAPPING TEST
001744  027777                        .WORD   DH210   ;TESTNO  ERR PC MAPREGADR
001746  030650                        .WORD   DT210   ;TESTNO,$ERRPC,EADRES,0
001750  031005                        .WORD   DF210   ; 0, 0, 2
```

```
                                        ;ITEM 211
001752  025377                                .WORD   EM211   ;RELOCATION THROUGH THE MAP WAS NOT CORRECT, CARRY PROPAGATION
                                                            ;TEST BEING RUN OVER UNIBUS
001754  030031                                .WORD   DH211   ;CORRECT   EXPECTD RECEIVD
                                                            ;ADDRESS   DATA    FROM UB TESTNO  ERR PC
001756  030660                                .WORD   DT211   ;EADRES,$REG3,$REG2,TESTNO,$ERRPC,0
001760  030772                                .WORD   DF201   ; 2, 0, 0, 0, 0

                                        ;ITEM 212
001762  025535                                .WORD   EM212   ;MAIN MEMORY TIME OUT OVER THE UNIBUS DID NOT OCCUR PROPERLY.
                                                            ;TEST BEING RUN OVER UNIBUS
001764  030031                                .WORD   DH211   ;CONDITN CONDITN
                                                            ;EXPECTD RECEIVD TESTNO  ERR PC
001766  030660                                .WORD   DT211   ;CPUEXP,PCPUER,TESTNO,$ERRPC,0
001770  030772                                .WORD   DF201   ; 0, 0, 0, 0

                                        ;ITEM 213
001772  025670                                .WORD   EM213   ;MAP REGISTER ENABLED WHEN DDW SAYS IT SHOULD BE DISABLED
001774  030213                                .WORD   DH213   ;TESTNO  ERR PC  REG NO  DDWDAT  DDWADR
001776  030706                                .WORD   DT213   ;TESTNO,$ERRPC,$TMP0,$TMP1,$REG5,0
002000  030722                                .WORD   DF1     ; 0, 0, 0, 0, 0

                                        ;ITEM 214
002002  025761                                .WORD   EM214   ;MAP REGISTER DISABLED WHEN DDW SAYS IT SHOULD BE ENABLED
002004  030213                                .WORD   DH213   ;TESTNO  ERR PC  REG NO  DDWDAT  DDWADR
002006  030706                                .WORD   DT213   ;TESTNO,$ERRPC,$TMP0,$TMP1,$REG5,0
002010  030722                                .WORD   DF1     ; 0, 0, 0, 0, 0
```

```
                                .SBTTL  SOFTWARE SWITCH REGISTER LOCATION
002012                          .$Y=.           ;SAVE ADDRESS LOCATION
000176                          .=176           ;ADDRESS TO SOFTWARE SWITCH REGISTER LOCATION
000176  000000          $SSWR:  .WORD   0       ;LOCATION FOR SOFTWARE SWITCH REGISTER
002012                          .=.$Y           ;RETURN TO PREVIOUS ADDRESS LOCATION
```

```
                        .SBTTL  ERROR MESSAGE TYPE OUT ROUTINE
                        ;;*****************************************************************
                        ;*
                        ;*      THIS SUBROUTINE IS CALLED BY THE ERROR HANDLER TO TYPE
                        ;*      THE ERROR MESSAGES.  IT PICKS UP THE ITEM BYTE ($ITEMB) NUMBER
                        ;*      AND USES THAT TO INDEX THROUGH THE ERROR TABLE.  THE ERROR
                        ;*      TABLE STARTS AT "$ERRTB" AND HAS FOUR (4) POINTERS FOR EACH
                        ;*      ENTRY, 'EM', 'DH', 'DT', 'DF'.  THE 'EM' POINTS TO THE ERROR
                        ;*      MESSAGE WHICH IS AN ASCIZ STRING.  THE 'DH' POINTS TO THE DATA
                        ;*      HEADER WHICH IS ANOTHER ASCIZ STRING.  THE 'DT' POINTS TO THE
                        ;*      DATA TABLE WHICH IS A GROUP OF WORDS CONTAINING THE ADDRESSES
                        ;*      OF THE DATA TO BY TYPED.  THE FORMAT OF THIS DATA IS
                        ;*      CONTROLLED BY THE 'DF' WHICH IS THE POINTER TO THE DATA FORMAT.
                        ;*      THE DATA FORMAT IS A GROUP OF BYTES WHICH CONTAIN NUMBERS
                        ;*      THAT CORRESPOND TO DIFFERENT TYPING FORMATS.
                        ;*      0       -16 BIT OCTAL FORMAT
                        ;*      1       -DECIMAL FORMAT
                        ;*      2       -22 BIT OCTAL FORMAT.  DATA IS LOWER 16 BITS OF THE
                        ;*               PHYSICAL ADDRESS, UPPER 6 BITS ARE ADJACENT TO LOWER 16
                        ;*      3       -22 BIT OCTAL FORMAT.  DATA IS THE 16 BIT VIRTUAL
                        ;*               ADDRESS IN KERNEL I-SPACE.
                        ;*      4       -18 BIT OCTAL FORMAT.  DATA IS A 16 BIT NUMBER THAT
                        ;*               WILL BE CONVERTED INTO A UNIBUS ADDRESS BY LEFT
                        ;*               SHIFTING IT 6 BITS.
                        ;*      5       -16 BIT OCTAL, SUPPRESS LEADING ZEROS
                        ;*      6       -16 BIT DECIMAL, SUPPRESS SPACES
                        ;*      IF YOU SHOULD HAVE A NEED TO JUST TYPE A STRING OF
                        ;*      NUMBERS, SET UP YOUR CODE THIS WAY:
                        ;*      MOV     #CONTINUE,-(SP) ;MOVE THE ADDRESS OF THE INSTRUCTION AFTER THE
                        ;*                              ;JUMP TO THE STACK
                        ;*      MOV     R0,-(SP)        ;SAVE R0
                        ;*      MOV     R1,-(SP)        ;AND R1 ON THE STACK
                        ;*      MOV     DTNAME,R0       ;MOVE THE ADDRESS OF THE DATA TABLE TO R0
                        ;*      JMP     TYPDAT          ;SUBROUTINE IDENTIFIED IN CENTER OF THIS ROUTINE
                        ;* CONTINUE: NEXT INSTRUCTION
                        ;*      AT A CONVENIENT SPOT, ALOCATE THE FOLLOWING.
                        ;*DTNAME: .WORD DTLIST,DFNAME   ;IDENTIFY THE LIST NAME AND DATA FORMAT BELOW
                        ;*DFNAME: .BYTE N,N,N,N,ETC.    ;CONSTRUCT YOUR OWN DATA FORMAT LINE
                        ;*      .EVEN
                        ;*DTLIST: VAR1,VAR2,VAR3,VAR4,.....,$CRLF,0      ;VARIABLES YOU WANT TYPED
                        ;*
                        ;;*****************************************************************
002012  010046          ERTYPE: MOV     R0,-(KSP)       ;SAVE R0 ON STACK
002014  005000                  CLR     R0              ;CLEAR R0
002016  113700  001112          MOVB    $ITEMB,R0       ;PUT ITEM NUMBER IN R0
002022  001004                  BNE     1$              ;BRANCH IF IT IS NON-ZERO
002024  013746  001114          MOV     $ERRPC,-(KSP)   ;PUT ERROR PC ON STACK FOR TYPING
002030  104402                  TYPOC                   ;TYPE FAILING PC
002032  000564                  BR      13$             ;GO TO RETURN
002034  005300          1$:     DEC     R0              ;ADJUST ITEM NUMBER TO BE A POINTER
002036  072027  000003          ASH     #3,R0           ;LEFT SHIFT ITEM NO. 3 PLACES
```

```
860 002042  100041                      BPL     22$             ;BRANCH IF ITEM NUMBER IS LESS THAN 200
861 002044  023727  001302  000020      CMP     ERRCNT,#20      ;;* SEE IF 20 (OCTAL) ERRORS HAVE PRINTED          SEQ 0045
862 002052  002410                      BLT     40$             ;;* BRANCH TO PRINT THE ERROR IF LESS
863 002054  001404                      BEQ     41$             ;;* BRANCH TO TYPE NO MORE DATA LINES IF EQUAL
864 002056  062766  000004  000002      ADD     #4,2(KSP)       ;;* CORRECT PC RETURN TO RETURN AFTER <CRLF> PRINT
865 002064  000547                      BR      13$             ;;* GO TO RETURN
```

```
 866 002066  104400  007260          41$:    TYPE    ,NOMORE         ;;* TYPE MESSAGE TO ANNOUNCE NO MORE PRINTING OF ERRORS
 867 002072  000544                  BR      13$             ;;* GO TO RETURN
 868 002074  022737  000001  001302  40$:    CMP     #1,ERRCNT       ;;* SEE IF THIS IS THE FIRST ERROR
 869 002102  001415                  BEQ     21$             ;;* BRANCH IF IT WAS AND GO TYPE ERROR MESSAGE
 870 002104  032777  000200  177036          BIT     #SW7,@SWR       ;SEE IF SWITCH 7 IS UP
 871 002112  001404                  BEQ     20$             ;BRANCH IF SWITCH NOT UP AND TYPE DATA
 872 002114  062766  000004  000002          ADD     #4,2(KSP)       ;SKIP 'TYPE  ,$CRLF' IF SW 7 IS UP
 873                                                         ;INHIBIT MULTIPLE ERROR TYPEOUTS
 874 002122  000530                  BR      13$             ;BRANCH TO EXIT
 875 002124  042700  177400          20$:    BIC     #177400,R0      ;CLEAR UPPER BYTE OF R0
 876 002130  062700  001656          ADD     #ER200+4,R0     ;POINT TO DATA TABLE ENTRY
 877 002134  000426                  BR      5$              ;GO TYPE DATA TABLE
 878 002136  042700  177000          21$:    BIC     #177000,R0      ;CLEAR UPPER BYTE OF R0
 879 002142  062700  000300          ADD     #<ER200-$ERRTB>,R0 ;ADD DIFFERENCE BETWEEN
 880                                                         ;ITEM 1 AND ITEM 201
 881                                                 ;;GET POINTER TO ERROR MESSAGE AND TYPE IT
 882                                                 ;;IF THE POINTER IS NOT ZERO
 883 002146  104400  001221          22$:    TYPE    ,$CRLF          ;TYPE A <CRLF>
 884 002152  062700  001352          ADD     #$ERRTB,R0      ;ADD BASE OF ERROR TABLE
 885 002156  012037  002166          MOV     (R0)+,2$        ;P M MESSAGE POINTER IN TYPE STATEMENT
 886 002162  001404                  BEQ     3$              ;BRANCH IF NO ERROR MESSAGE
 887 002164  104400                  TYPE                    ;TYPE ERROR MESSAGE
 888 002166  000000          2$:     .WORD   0               ;POINTER TO ERROR MESSAGE
 889 002170  104400  001221          TYPE    ,$CRLF          ;TYPE CRLF
 890                                                 ;;GET THE POINTER TO THE DATA HEADER AND
 891                                                 ;;TYPE IT IF THE POINTER IS NOT ZERO
 892 002174  012037  002204          3$:     MOV     (R0)+,4$        ;PUT HEADER POINTER IN TYPE STATEMENT
 893 002200  001404                  BEQ     5$              ;BRANCH IF NO DATA HEADER
 894 002202  104400                  TYPE                    ;TYPE THE DATA HEADER
 895 002204  000000          4$:     .WORD   0               ;POINTER TO DATA HEADER
 896 002206  104400  001221          TYPE    ,$CRLF          ;TYPE CRLF
 897                                                 ;;THIS IS THE START OF THE DATA OUTPUT IF THE
 898                                                 ;;DATA POINTER IS NOT ZERO.  R0 POINTS TO THE
 899                                                 ;;DATA FORMAT, R1 POINTS TO THE ADDRESS OF
 900                                                 ;;THE DATA WORDS.
 901 002212  010146          5$:     MOV     R1,-(KSP)       ;SAVE R1 ON THE STACK
 902         002214          TYPDAT=.
 903 002214  012001                  MOV     (R0)+,R1        ;PUT DATA TABLE POINTER IN R1
 904 002216  001471                  BEQ     12$             ;BRANCH IF NO DATA TABLE
 905 002220  012000                  MOV     (R0)+,R0        ;PICK UP DATA FORMAT POINTER
 906 002222  105710          6$:     TSTB    (R0)            ;IS THIS WORD OCTAL
 907 002224  001003                  BNE     7$              ;BRANCH IF NOT 16-BIT OCTAL
 908                                                 ;;WORD IS 16 BIT OCTAL FORMAT (DF = 0)
 909 002226  013146                  MOV     @(R1)+,-(KSP)   ;PUSH NEXT 16-BIT WORD ON STACK
 910 002230  104402                  TYPOC                   ;TYPE THE WORD ON STACK AS 16 BIT OCTAL
 911 002232  000456                  BR      11$             ;GET READY FOR NEXT WORD
 912 002234  122710  000001  7$:     CMPB    #1,(R0)         ;IS THE WORD DECIMAL
 913 002240  001003                  BNE     8$              ;BRANCH IF NOT DECIMAL
 914                                                 ;;WORD IS DECIMAL FORMAT (DF = 1)
 915 002242  013146                  MOV     @(R1)+,-(KSP)   ;PUSH NEXT 16-BIT WORD ON STACK
 916 002244  104410                  TYPDS                   ;TYPE THE WORD ON STACK AS DECIMAL
```

```
917 002246  000450                      BR      11$             ;GET READY FOR NEXT WORD
918 002250  122710  000002      8$:     CMPB    #2,(R0)         ;IS WORD 22-BIT PHYSICAL ADDRESS                SEQ 0047
919 002254  001012                      BNE     9$              ;BRANCH IF NOT 22-BIT PHYSICAL ADDR
920                                                     ;:WORD IS 22-BIT PHYSICAL FORMAT (DF = 2)
921 002256  012146                      MOV     (R1)+,-(KSP)    ;PUSH NEXT 16-BIT WORD ON STACK
922 002260  004737  004166              JSR     PC,$DB2O        ;CONVERT NUMBER TO OCTAL ASCIZ
```

```
                                                                ;ONLY WANT 8 DIGITS
                                                                ;PUT POINTER AFTER 'TYPE' CALL
                                                                ;TYPE ASCIZ STRING
                                                                ;WORD HOLDS POINTER TO ASCIZ STRING
                                                                ;GET READY FOR NEXT WORD
                                                                ;IS THIS A 16-BIT VIRTUAL ADDRESS
                                                                ;BRANCH IF NOT 16-BIT VIRT. ADDR.
                                                        ;;WORD IS 22-BIT VIRTUAL ADDRESS FORMAT
                                                        ;;KERNEL I-SPACE ASSUMED.  (DF = 3)
                                                                ;PUSH NEXT 16-BIT WORD ON STACK
                                                                ;GO TYPE 22-BIT ADDRESS FROM 16-BIT V.A.
                                                                ;GET READY FOR NEXT WORD
                                                                ;IS THIS A 16 BIT NUMBER TO BE CONVERTED TO
                                                                ;AN 18 BIT UNIBUS ADDRESS LEFT SHIFTED 6?
                                                                ;SKIP OVER FORMAT 4 ROUTINE IF NOT
                                                        ;;WORD IS FORMAT 4.  DATA WORD IS A UNIBUS
                                                        ;;ADDRESS OUTPUT WILL BE 18-BITS WORD LEFT SHIFTED 6.

                                                                ;CONVERT TO 18-BIT UNIBUS ADDR AND TYPE
                                                                ;GET READY FOR NEXT WORD
                                                                ;IS THIS A 16 BIT NUMBER TO BE PRINTED AS
                                                                ;OCTAL WITH LEADING ZEROS SUPPRESSED?
                                                                ;BRANCH TO DECIMAL LEADING SPACES SUPPRESS ROUTINE
                                                        ;;WORD IS FORMAT 5.  DATA WORD IS TO BE
                                                        ;;PRINTED IN OCTAL, LEADING ZEROS SUPPRESSED.
                                                                ;PUSH NEXT 16-BIT WORD ON STACK
                                                                ;GO TYPE OCTAL SUPPRESS LEADING ZEROS
                                                                ;TYPE 6 DIGITS AND
                                                                ;SUPPRESS LEADING ZEROS
                                                                ;GET READY FOR NEXT WORD
                                                        ;;WORD IS FORMAT 6. DATA WORD IS TO BE
                                                        ;;PRINTED IN DECIMAL, LEADING SPACES SUPPRESSED.
                                                                ;PUSH NEXT 16-BIT WORD ON STACK
                                                                ;SET FLAG TO SUPPRESS LEADING SPACES

                                                                ;CLEAR THE LEADING SPACES SUPPRESS FLAG
                                                                ;POINT TO NEXT FORMAT BYTE
                                                                ;TYPE TWO SPACES
                                                                ;IS THERE ANOTHER WORD?
                                                                ;BRANCH IF NOT ALL DONE
                                                                ;RESTORE R1
                                                                ;RESTORE R0
                                                                ;RETURN TO ERROR ROUTINE
                                                                ;TWO SPACES
                                                                ;LEADING ZEROS SUPPRESS FLAG LOCATION
```

```
                                .SBTTL  CONVERT 16-BIT VIRTUAL ADDRESS TO 22-BIT PHYSICAL ADDRESS
                                ;;*****************************************************************
                                ;*
                                ;*      THIS ROUTINE IS CALLED BY A 'JSR PC' AFTER THE VIRTUAL ADDRESS
                                ;*      IS PUSHED ON THE KERNEL STACK.  THE V.A. IS THEN LOADED INTO
                                ;*      R1 AND THE UPPER 3 BITS ARE SHIFTED INTO R0 TO SELECT THE
                                ;*      CORRECT KERNEL I-SPACE PAR.  THE LOWER 12 BITS OF THE VIRTUAL
                                ;*      ADDRESS ARE ADDED TO THE PAR AS THEY ARE BY MEMORY MANAGEMENT
                                ;*      AND THE PHYSICAL ADDRESS IS SAVED IN MEMORY TO BE CONVERTED
                                ;*      TO ASCIZ AND TYPED.
                                ;*
                                ;;*****************************************************************
002414  104412                  TYPVAD: SAVREG                          ;SAVE ALL REGISTERS
002416  016601  000002                  MOV     2(KSP),R1               ;PUT VIRTUAL ADDR IN R1
002422  005000                          CLR     R0                      ;CLEAR R0 FOR CALCULATIONS
002424  073027  000003                  ASHC    #3,R0                   ;LEFT SHIFT R0,R1 3 PLACES
002430  006300                          ASL     R0                      ;LEFT SHIFT R0 ONE MORE PLACE
002432  006001                          ROR     R1                      ;RIGHT SHIFT R1 SO OFFSET IS CORRECT
002434  006001                          ROR     R1                      ;RIGHT SHIFT R1
002436  006001                          ROR     R1                      ;RIGHT SHIFT R1
002440  062700  172340                  ADD     #KIPAR0,R0              ;FORM DESIRED PAR ADDR IN R0
002444  011003                          MOV     (R0),R3                 ;PUT CONTENTS OF PAR IN R3
002446  005002                          CLR     R2                      ;CLEAR R2 FOR PHYSICAL ADDR CALCULATIONS
002450  073227  000006                  ASHC    #6,R2                   ;LEFT SHIFT <R2,R3> 6 PLACES
002454  060103                          ADD     R1,R3                   ;ADD OFFSET IN R1 TO BASE IN R3
002456  005502                          ADC     R2                      ;ADD ANY POSSIBLE CARRY TO UPPER 6 BITS
002460  010237  001226                  MOV     R2,PADRSH               ;PUT UPPER 6 BITS OF ADDR IN CORE
002464  010337  001224                  MOV     R3,PADRSL               ;PUT LOWER 16 BITS OF ADDR IN CORE
002470  012746  001224                  MOV     #PADRSL,-(KSP)          ;PUT POINTER TO LOWER 16 BITS ON STACK
002474  004737  004166                  JSR     PC,$DB2O                ;CONVERT NUMBER TO OCTAL ASCIZ
002500  062716  000003                  ADD     #3,(KSP)                ;ONLY TYPE 8 DIGITS
002504  012637  002512                  MOV     (KSP)+,3$               ;PUT POINTER AFTER TYPE INST
002510  104400                          TYPE                            ;TYPE THE 22-BIT VIRTUAL ADDRESS
002512  000000                  3$:     .WORD   0                       ;THIS WORD HOLDS THE POINTER TO
                                                                        ;THE ASCIZ STRING
002514  104414                          RESREG                          ;RESTORE ALL THE REGISTERS
002516  012616                          MOV     (KSP)+,(KSP)            ;LEAVE ONLY RETURN ADDR ON STACK
002520  000207                          RTS     PC                      ;RETURN TO ERROR HANDLER
```

```
1006                                    .SBTTL  SUBROUTINE TO CONVERT WORD TO A UNIBUS ADDRESS AND TYPE
1007                                    ;;**************************************************************
1008                                    ;*THIS SUBROUTINE IS USED TO CONVERT THE A WORD PUSHED
1009                                    ;*ON THE STACK INTO A UNIBUS ADDRESS AND TYPE IT AS A
1010                                    ;*6 DIGIT NUMBER.  IT USES R1 & R0 AND LEAVES
1011                                    ;*ALL OTHER REGISTERS UNCHANGED.
1012                                    ;;**************************************************************
1013 002522  104412                    UBADDR: SAVREG
1014 002524  016601  000002                    MOV     2(KSP),R1       ;LOAD 16 BIT ADDRESS INTO R1
1015 002530  005000                            CLR     R0              ;CLEAR R0 FOR CALCULATIONS
1016 002532  073027  000006                    ASHC    #6,R0           ;LEFT SHIFT <R0:R1> 6 PLACES
1017 002536  010137  001224                    MOV     R1,PADRSL       ;PUT LOWER 16 BITS IN PADRSL
1018 002542  010037  001226                    MOV     R0,PADRSH       ;PUT UPPER 6 BITS IN PADRSH
1019 002546  012746  001224                    MOV     #PADRSL,-(KSP)  ;PUSH POINTER TO WORDS ON STACK
1020 002552  004737  004166                    JSR     PC,$DB2O        ;JUMP TO CONVERT ROUTINE
1021 002556  062716  000005                    ADD     #5,(KSP)        ;ONLY USE LOWER 6 CHARS.
1022 002562  012637  002570                    MOV     (KSP)+,3$       ;PUT POINTER AFTER TYPE CALL.
1023 002566  104400                            TYPE
1024 002570  000000                    3$:     .WORD   0               ;HOLDS POINTER TO FIRST CHAR.
1025 002572  104414                            RESREG
1026 002574  012616                            MOV     (KSP)+,(KSP)    ;LEAVE ONLY RETURN ADDRESS ON STACK
1027 002576  000207                            RTS     PC              ;RETURN TO ERROR TYPE ROUTINE.
```

```
   1028                                 .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES
                                        ;;*****************************************************************
                                        ;*SAVE R0-R5
                                        ;*CALL:
                                        ;*      SAVREG
                                        ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
                                        ;*
                                        ;*TOP---(+16)
                                        ;* +2---(+18)
                                        ;* +4---R5
                                        ;* +6---R4
                                        ;* +8---R3
                                        ;*+10---R2
                                        ;*+12---R1
                                        ;*+14---R0
   002600                               $SAVREG:
   002600  010046                               MOV     R0,-(SP)        ;;PUSH R0 ON STACK
   002602  010146                               MOV     R1,-(SP)        ;;PUSH R1 ON STACK
   002604  010246                               MOV     R2,-(SP)        ;;PUSH R2 ON STACK
   002606  010346                               MOV     R3,-(SP)        ;;PUSH R3 ON STACK
   002610  010446                               MOV     R4,-(SP)        ;;PUSH R4 ON STACK
   002612  010546                               MOV     R5,-(SP)        ;;PUSH R5 ON STACK
   002614  016646  000022                       MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
   002620  016646  000022                       MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
   002624  016646  000022                       MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
   002630  016646  000022                       MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
   002634  000002                              RTI
                                        ;*RESTORE R0-R5
                                        ;*CALL:
                                        ;*      RESREG
   002636                               $RFSREG:
   002636  012666  000022                       MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
   002642  012666  000022                       MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
   002646  012666  000022                       MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
   002652  012666  000022                       MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
   002656  012605                               MOV     (SP)+,R5        ;;POP STACK INTO R5
   002660  012604                               MOV     (SP)+,R4        ;;POP STACK INTO R4
   002662  012603                               MOV     (SP)+,R3        ;;POP STACK INTO R3
   002664  012602                               MOV     (SP)+,R2        ;;POP STACK INTO R2
   002666  012601                               MOV     (SP)+,R1        ;;POP STACK INTO R1
   002670  012600                               MOV     (SP)+,R0        ;;POP STACK INTO R0
   002672  000002                              RTI
```

```
1029                                    .SBTTL  TYPE ROUTINE
                                        ;;*****************************************************************
                                        ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
                                        ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
                                        ;*NOTE1:         $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
                                        ;*NOTE2:         $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
                                        ;*NOTE3:         $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
                                        ;*
                                        ;*CALL:
                                        ;*1) USING A TRAP INSTRUCTION
                                        ;*       TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
                                        ;*OR
                                        ;*       TYPE
                                        ;*       MESADR
                                        ;*
                                        ;*2) USING A JSR INSTRUCTION
                                        ;*       MOV     PS,-(SP)        ;;PUSH PROCESSOR STATUS WORD ON THE STACK
                                        ;*       JSR     PC,$TYPE        ;;CALL TYPE ROUTINE
                                        ;*       MESADDR                 ;;FIRST ADRESS OF MESSAGE
002674  105737  001147          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
002700  100002                          BPL     1$              ;;BR IF YES
002702  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
002704  000413                          BR      3$              ;;LEAVE
002706  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
002710  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
002714  132737  000040  020035          BITB    #40,$ENVM       ;&& SUPRESS TYPEOUTS?
002722  001003                          BNE     60$             ;&& YES
002724  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
002726  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
002730  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
002732  012600                  60$:    MOV     (SP)+,R0        ;&& RESTORE R0
002734  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
002740  000002                          RTI                     ;;RETURN
002742  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
002746  001426                          BEQ     8$
002750  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT
002754  001004                          BNE     5$
002756  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
002760  104400  001221                  TYPE    ,$CRLF
002764  000757                          BR      2$              ;;GET NEXT CHARACTER
002766  004737  003050          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
002772  123726  001146          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
002776  001352                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
003000  013746  001144                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                ;;AND THE NULL CHAR.
003004  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
003010  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
003012  004737  003050                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
003016  105337  003200                  DECB    $CHARCNT        ;;DON'T COUNT THE NULL AS A CHARACTER
003022  000770                          BR      7$              ;;LOOP
                                        ;;HORIZONTAL TAB PROCESSOR
```

```
003024  112716  000040          8$:     MOVB    #'  ,(SP)          ;;REPLACE TAB WITH SPACE
003030  004737  003050          9$:     JSR     PC,$TYPEC          ;;TYPE A SPACE                    SEQ 0053
003034  132737  000007  003200          BITB    #7,$CHARCNT        ;;BRANCH IF NOT AT
003042  001372                          BNE     9$                 ;;TAB STOP
003044  005726                          TST     (SP)+              ;;POP SPACE OFF STACK
003046  000726                          BR      2$                 ;;GET NEXT CHARACTER
```

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

```
1030                                    .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
                                        ;;*****************************************************************
                                        ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
                                        ;*OCTAL (ASCII) NUMBER AND TYPE IT.
                                        ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPOS                   ;;CALL FOR TYPEOUT
                                        ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
                                        ;*      .BYTE   M               ;;M=1 OR 0
                                        ;*                                      ;;1=TYPE LEADING ZEROS
                                        ;*                                      ;;0=SUPPRESS LEADING ZEROS
                                        ;*
                                        ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
                                        ;*$TYPOS OR $TYPOC
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPON                   ;;CALL FOR TYPEOUT
                                        ;*
                                        ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPOC                   ;;CALL FOR TYPEOUT
003204  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
003210  116637  000001  003427          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
003216  112637  003431                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
003222  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
003226  000406                          BR      $TYPON
003230  112737  000001  003427  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
003236  112737  000006  003431          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
003244  112737  000005  003426  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
003252  010346                          MOV     R3,-(SP)        ;;SAVE R3
003254  010446                          MOV     R4,-(SP)        ;;SAVE R4
003256  010546                          MOV     R5,-(SP)        ;;SAVE R5
003260  113704  003431                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
003264  005404                          NEG     R4
003266  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
003272  110437  003430                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
003276  113704  003427                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
003302  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
003306  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
003310  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
003312  000404                          BR      3$              ;;GO DO MSB
003314  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
003316  006105                          ROL     R5
003320  006105                          ROL     R5                              .
003322  010503                          MOV     R5,R3
003324  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
003326  105337  003430                  DECB    $OMODE          ;;TYPE THIS DIGIT?
003332  100016                          BPL     7$              ;;BR IF NO
003334  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
```

```
003340  001002                  BNE     4$              ;;TEST FOR 0                                    SEQ 0056
003342  005704                  TST     R4              ;;SUPPRESS THIS 0?
003344  001403                  BEQ     5$              ;;BR IF YES
003346  005204          4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
003350  052703  000060          BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
003354  052703  000040          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
```

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

```
1031                                    .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
                                        ;;*****************************************************************
                                        ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
                                        ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
                                        ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
                                        ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
                                        ;*REPLACED WITH SPACES.
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
                                        ;*      TYPDS                   ;;GO TO THE ROUTINE
003432                                  $TYPDS:
003432  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
003434  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
003436  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
003440  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
003442  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
003444  012746  020200                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
003450  016605  000020                  MOV     20(SP),R5       ;;GET THE INPUT NUMBER
003454  100004                          BPL     1$              ;;BR IF INPUT IS POS.
003456  005405                          NEG     R5              ;;MAKE THE BINARY NUMBER POS.
003460  112766  000055  000001          MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
003466  005000                  1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
003470  012703  003714                  MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
003474  112723  000040                  MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
003500  005002                  2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
003502  016001  003704                  MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
003506  160105                  3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
003510  002402                          BLT     4$              ;;BR IF DONE
003512  005202                          INC     R2              ;;INCREASE THE BCD DIGIT BY 1
003514  000774                          BR      3$
003516  060105                  4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
003520  005702                          TST     R2              ;;CHECK IF BCD DIGIT=0
003522  001002                          BNE     5$              ;;FALL THROUGH IF 0
003524  105716                          TSTB    (SP)            ;;STILL DOING LEADING 0'S?
003526  100407                          BMI     7$              ;;BR IF YES
003530  106316                  5$:     ASLB    (SP)            ;;MSD?
003532  103003                          BCC     6$              ;;BR IF NO
003534  116663  000001  177777          MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
003542  052702  000060          6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
003546  052702  000040          7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
003552  110223                          MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
003554  005720                          TST     (R0)+           ;;JUST INCREMENTING
003556  020027  000010                  CMP     R0,#10          ;;CHECK THE TABLE INDEX
003562  002746                          BLT     2$              ;;GO DO THE NEXT DIGIT
003564  003002                          BGT     8$              ;;GO TO EXIT
003566  010502                          MOV     R5,R2           ;;GET THE LSD
003570  000764                          BR      6$              ;;GO CHANGE TO ASCII
003572  105726                  8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
003574  100003                          BPL     9$              ;;BR IF NO
003576  116663  177777  177776          MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
003604  105013                  9$:     CLRB    (R3)            ;;SET THE TERMINATOR
```

```
003606  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
003610  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3                                    SEQ 0059
003612  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
003614  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
003616  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
003620  105737  002413                  TSTB    SPSUPP          ;. SEE IF LEADING SPACES ARE TO BE SUPPRESSED
```

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

```
1032                                    .SBTTL  TRAP DECODER
                                        ;;**********************************************************************
                                        ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
                                        ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
                                        ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
                                        ;*GO TO THAT ROUTINE.
003724  010046                          $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
003726  016600  000002                          MOV     2(SP),R0        ;;GET TRAP ADDRESS
003732  005740                                  TST     -(R0)           ;;BACKUP BY 2
003734  111000                                  MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
003736  016000  003744                          MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
003742  000200                                  RTS     R0              ;;GO TO ROUTINE
```

```
                                .SBTTL  TRAP TABLE
                                ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
                                ;*BY THE "TRAP" INSTRUCTION.
                                ;       ROUTINE
                                ;       -------
                                $TRPAD:
     003744
     003744  002674                     $TYPE   ;;CALL=TYPE     TRAP+0(104400)  TTY TYPEOUT ROUTINE
     003746  003230                     $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
     003750  003204                     $TYPOS  ;;CALL=TYPOS    TRAP+4(104404)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
     003752  003244                     $TYPON  ;;CALL=TYPON    TRAP+6(104406)  TYPE OCTAL NUMBER (AS PER LAST CALL)
     003754  003432                     $TYPDS  ;;CALL=TYPDS    TRAP+10(104410) TYPE DECIMAL NUMBER (WITH SIGN)
     003756  002600                     $SAVREG ;;CALL=SAVREG   TRAP+12(104412) SAVE R0-R5 ROUTINE
     003760  002636                     $RESREG ;;CALL=RESREG   TRAP+14(104414) RESTORE R0-R5 ROUTINE
1033 003762  004306                     TBITOFF ;;CALL=TBITO    TRAP+16(104416) THIS WILL TURN OFF T BIT TRAPPING
1034 003764  004334                     TBITRESTORE     ;;CALL=TBITR    TRAP+20(104420) THIS WILL RETURN THE T BIT TO PREVIO
1035
```

```
1036                                    .SBTTL  POWER DOWN AND UP ROUTINES
                                        ;;*****************************************************************
                                        ;POWER DOWN ROUTINE
003766  012737  004114  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
003774  012737  000340  000026         MOV     #340,@#PWRVEC+2 ;;PRIO:7
004002  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
004004  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
004006  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
004010  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
004012  010446                        MOV     R4,-(SP)        ;;PUSH R4 ON STACK
004014  010546                        MOV     R5,-(SP)        ;;PUSH R5 ON STACK
004016  010637  004120                MOV     SP,$SAVR6       ;;SAVE SP
004022  012737  004034  000024        MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
004030  000000                        HALT
004032  000776                        BR      .-2             ;;HANG UP
                                    ;POWER UP ROUTINE
004034  013706  004120          $PWRUP: MOV     $SAVR6,SP       ;;GET SP
004040  005037  004120                CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
004044  005237  004120          1$:   INC     $SAVR6          ;;WAIT FOR THE INC
004050  001375                        BNE     1$              ;;OF  WORD
004052  012605                        MOV     (SP)+,R5        ;;POP STACK INTO R5
004054  012604                        MOV     (SP)+,R4        ;;POP STACK INTO R4
004056  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
004060  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
004062  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
004064  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
004066  012737  003766  000024        MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
004074  012737  000340  000026        MOV     #340,@#PWRVEC+2 ;;PRIO:7
004102  104400                        TYPE                    ;;REPORT THE POWER FAILURE
004104  004122                $PWRMG: .WORD   PWRMSG          ;;POWER FAIL MESSAGE POINTER
004106  012716                        MOV     (PC)+,(SP)      ;;RESTART AT START
004110  010000                $PWRAD: .WORD   START           ;;RESTART ADDRESS
004112  000002                        RTI
004114  000000                $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
004116  000776                        BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
004120  000000                $SAVR6: 0                       ;;PUT THE SP HERE
1037 004122     012     015     120  PWRMSG: .ASCIZ  <12><15>?POWER FAILURE, RESTARTING PROGRAM?
1038                                    .EVEN
```

```
1039                                    .SBTTL  DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
                                        ;;**********************************************************
                                        ;*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
                                        ;*UNSIGNED OCTAL ASCIZ NUMBER.
                                        ;*CALL
                                        ;*      MOV     #PNTR,-(SP)     ;;POINTER TO LOW WORD OF BINARY NUMBER
                                        ;*      JSR     PC,@#$DB20      ;;CALL THE ROUTINE
                                        ;*      RETURN                  ;;THE ADDRESS OF THE FIRST ASCIZ CHAR. IS ON THE STACK
004166  104412                          $DB20:  SAVREG                  ;;SAVE ALL REGISTERS
004170  016601  000002                          MOV     2(SP),R1        ;;PICKUP THE POINTER TO LOW WORD
004174  012705  004305                          MOV     #$OCTVL+13.,R5  ;;POINTER TO DATA TABLE
004200  012704  000014                          MOV     #12.,R4         ;;DO ELEVEN CHARACTERS
004204  012703  177770                          MOV     #^C7,R3         ;;MASK
004210  012100                                  MOV     (R1)+,R0        ;;LOWER WORD
004212  012101                                  MOV     (R1)+,R1        ;;HIGH WORD
004214  005002                                  CLR     R2              ;;TERMINATOR
004216  110245                          1$:     MOVB    R2,-(R5)        ;;PUT CHARACTER IN DATA TABLE
004220  010002                                  MOV     R0,R2           ;;GET THIS DIGIT
004222  005304                                  DEC     R4              ;;COUNT THIS CHARACTER
004224  003007                                  BGT     3$              ;;BR IF NOT THE LAST DIGIT
004226  001405                                  BEQ     2$              ;;BR IF IT IS THE LAST DIGIT
004230  005205                                  INC     R5              ;;ALL DIGITS DONE-ADJUST POINTER FOR FIRST
004232  010566  000002                          MOV     R5,2(SP)        ;;ASCIZ CHAR. & PUT IT ON THE STACK
004236  104414                                  RESREG                  ;;RESTORE ALL REGISTERS
004240  000207                                  RTS     PC              ;;RETURN TO USER
004242  006203                          2$:     ASR     R3              ;;POSITION THE MASK FOR THE LAST DIGIT
004244  006001                          3$:     ROR     R1              ;;POSITION THE BINARY NUMBER FOR
004246  006000                                  ROR     R0              ;;      THE NEXT OCTAL DIGIT
004250  006001                                  ROR     R1
004252  006000                                  ROR     R0
004254  006001                                  ROR     R1
004256  006000                                  ROR     R0
004260  040302                                  BIC     R3,R2           ;;MASK OUT ALL JUNK
004262  062702  000060                          ADD     #'0,R2          ;;MAKE THIS CHAR. ASCII
004266  000753                                  BR      1$              ;;GO PUT IT IN THE DATA TABLE
004270                                  $OCTVL: .BLKB   14.             ;;RESERVE DATA TABLE
```

```
                                        .SBTTL  TURN OFF AND SAVE T-BIT
                                        ;;*****************************************************************
                                        ;;*****************************************************************
                                        ;*               **SUBROUTINES UNIQUE TO THIS PROGRAM**
                                        ;;*****************************************************************
                                        ;;*****************************************************************

                                        ;;*****************************************************************
                                        ;*
                                        ;*      THIS SUBROUTINE IS REACHED BY THE TRAP CALL 'TBITO', IT IS
                                        ;*      USED TO TURN OFF THE T-BIT IF IT IS ON.  THE PROCESSOR STATUS
                                        ;*      IS SAVED IN ''OLDPSW'' SO THAT THE T-BIT CAN BE RESTORED TO ITS
                                        ;*      PREVIOUS STATUS WHEN CONDITIONS WARRANT.
                                        ;*
                                        ;;*****************************************************************
            000020                      TBIT=BIT4               ;T-BIT IS BIT04 IN PROC. STATUS
004306  032766  000020  000002  TBITOF: BIT     #TBIT,2(KSP)    ;IS THE T-BIT ON?
004314  001406                        BEQ     1$              ;BRANCH TO EXIT IF IT IS NOT ON
004316  016637  000002  001332        MOV     2(KSP),OLDPSW   ;SAVE OLD PSW FOR RESTORING T BIT
004324  042766  000020  000002        BIC     #TBIT,2(KSP)    ;CLEAR T BIT
004332  000006                  1$:   RTT                     ;RETURN TO PROGRAM
```

```
                                        .SBTTL  RESTORE T-BIT TO ITS PREVIOUS CONDITION
                                        ;;*****************************************************************
                                        ;*
                                        ;*      THIS SUBROUTINE CAN BE REACHED BY THE TRAP CALL 'TBITR', IT IS
                                        ;*      USED TO RESTORE THE T-BIT AFTER A PARTICULAR TEST THAT CANNOT
                                        ;*      BE RUN WITH THE T-BIT ON.  IT USES THE PROCESSOR STATUS STORED
                                        ;*      IN "OLDPSW" BY 'TBITO', REPLACES THE PS ON THE STACK WITH IT
                                        ;*      AND DOES AN 'RTT'.
                                        ;*
                                        ;;*****************************************************************
004334  013766  001332  000002  TBITRE: MOV     OLDPSW,2(KSP)   ;PUT OLD PSW ON STACK
004342  042737  000020  001332          BIC     #TBIT,OLDPSW    ;CLEAR T-BIT IN "OLDPSW"
                                                                ;SO THAT IT WON'T BE TURNED ON BY ACCIDENT
004350  000006                          RTT     ;RETURN TO PROGRAM AND INHIBIT T-BIT TRAP AFTER THIS INSTRUCTION
```

```
                                        .SBTTL  SUBROUTINE TO CLEAR ALL OF THE MAP REGISTERS
                                        ;;*****************************************************************
                                        ;*
                                        ;*      THIS SUBROUTINE CLEARS ALL OF THE MAP REGISTERS IF MAPPING IS
                                        ;*      DISABLED BY LOADING THE ADDRESS OF MAPL00 INTO R3 AND THEN
                                        ;*      CLEARING THE REGISTER POINTED TO BY R3 UNTIL R3 POINTS ABOVE
                                        ;*      MAPH37.  IF MAPPING IS ENABLED, ALL REGISTERS EXCEPT MAPL1
                                        ;*      IS CLEARED.  THE LOWER WORD OF MAPL1 RECEIVES 20000.  THIS IS
                                        ;*      SO APT CAN PROPERLY MONITOR THE PROGRESS OF THE DIAGNOSTIC.
                                        ;*
                                        ;;*****************************************************************
004352  012703  170200          CLRMAP: MOV     #MAPL0,R3       ;PUT FIRST MAP ADDR IN R3
004356  005023                          CLR     (R3)+           ;CLEAR MAPL0
004360  005023                          CLR     (R3)+           ;CLEAR MAPL0+2
004362  032737  000040  172516          BIT     #BIT5,MMR3      ;SEE IF MAPPING IS ENABLED
004370  001402                          BEQ     1$              ;BRANCH TO CLEAR ALL IF NOT ENABLED
004372  012723  020000                  MOV     #20000,(R3)+    ;LOAD 20000 INTO MAPL1 FOR POSSIBLE APT USE
004376  005023                  1$:     CLR     (R3)+           ;CLEAR MAP REGISTERS
004400  022703  170400                  CMP     #MAPH37+2,R3    ;SEE IF LAST ADDR+2 IS IN R3
004404  001374                          BNE     1$              ;BRANCH IF NOT DONE YET
004406  000207                          RTS     PC              ;RETURN TO MAIN PROGRAM
```

```
                                        .SBTTL  SUBROUTINE TO LOG AND REPORT TIMEOUTS OF MAP REGISTERS
                                        ;;**************************************************************
                                        ;*
                                        ;*      THIS SUBROUTINE IS USED TO LOG AND REPORT THE FACT THAT A
                                        ;*      REFERENCE TO A MAPPING REGISTER TIMED OUT ON THE UNIBUS.  IT
                                        ;*      KEEPS A "LOGICAL AND" AND A "LOGICAL OR" OF EACH ADDRESS THAT
                                        ;*      TIMES OUT.
                                        ;*
                                        ;;**************************************************************
004410  005227                  TIMEOUT:INC     (PC)+           ;INCREMENT ONE TIME GATE
004412  177777                  TOFLAG: .WORD   -1              ;ONE TIME ENTANCE FLAG
004414  001403                          BEQ     10$             ;BRANCH IF FLAG IS NOW ZERO
004416  005237  020014                  INC     $MSGTY          ;INDICATE TO APT A FATAL ERROR OCCURED
004422  000000                          HALT    ;I HAVE ENTERED THIS ROUTINE BEFORE I FINISHED REPORTING THE FIRST ERROR.
                                                                ;THE SECOND ENTRY ADDRESS IS ON THE STACK AND THE
                                                                ;FIRST ERROR CONDITION IS PROBABLY STILL LOCKED UP.
004424  012637  001326          10$:    MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS
004430  012637  001330                  MOV     (KSP)+,OLDPS    ;SAVE OLD PSW
004434  105737  007774                  TSTB    CPUTYP          ;SEE IF THIS IS AN 11/44
004440  001406                          BEQ     1$              ;BRANCH TO CONTINUE IF IT IS
004442  005237  001314                  INC     PCPUER          ;INCREMENT PCPUER TO SHOW A TIMEOUT OCCURED
004446  005737  001312                  TST     CPUEXP          ;SEE IF THERE WAS AN EXPECTED ERROR
004452  001435                          BEQ     3$              ;GO REPORT ERROR IF NONE EXPECTED
004454  000442                          BR      4$              ;BRANCH TO EXIT IF TIMEOUT WAS EXPECTED
004456  013737  177766  001314  1$:     MOV     CPUERR,PCPUER   ;SAVE CPU ERROR REGISTER
004464  013737  001314  177766          MOV     PCPUER,CPUERR   ;CLEAR CPU ERROR REGISTER
004472  023737  001314  001312          CMP     PCPUER,CPUEXP   ;SEE IF EXPECTED CONDITION CAME UP.
004500  001405                          BEQ     2$              ;BRANCH IF IT WAS A TIMEOUT
004502  012737  177777  004412          MOV     #-1,TOFLAG      ;RESET ONE TIME GATE
004510  104001                          ERROR   +1              ;NOT THE CORRECT CPU TRAP THROUGH 4
004512  000423                          BR      4$              ;BRANCH TO EXIT
004514  105737  007774          2$:     TSTB    CPUTYP          ;IS THIS AN 11/24?
004520  001403                          BEQ     25$             ;BRANCH IF NOT
004522  005237  001302                  INC     ERRCNT          ;COUNT THIS AS A TIMEOUT
004526  000415                          BR      4$              ;GO TO EXIT
004530  022737  000020  001312  25$:    CMP     #TIMOUT,CPUEXP  ;SEE IF A TIMEOUT WAS EXPECTED
004536  001411                          BEQ     4$              ;BRANCH TO EXIT THIS ROUTINE IF IT WAS
004540  013746  172356                  MOV     KIPAR7,-(SP)    ;PUT PAR ON STACK FOR ADREXT SUBROUTINE USE
004544  010046                          MOV     R0,-(SP)        ;PUT VIRTUAL ADDRESS ON STACK FOR ADREXT SUBROUTINE USE
004546  004737  005250          3$:     JSR     PC,ADREXT       ;GO SET DATA IN THE 4 WORDS OF ADDROR AND ADRAND
004552  012737  177777  004412          MOV     #-1,TOFLAG      ;RESET ONE TIME GATE
004560  104201                          ERROR   +201            ;THE FOLLOWING REGISTERS TIMED OUT WHEN READ
004562  012737  177777  004412  4$:     MOV     #-1,TOFLAG      ;RESET ONE TIME GATE
004570  013746  001330                  MOV     OLDPS,-(KSP)    ;RESTORE OLD PSW
004574  013746  001326                  MOV     OLDPC,-(KSP)    ;PUSH RETURN ADDRESS BACK ON THE STACK
004600  000006                          RTT                     ;RETURN TO THE TEST
```

```
                                .SBTTL  CPU TRAP HANDLER ROUTINES
                                ;;*****************************************************************
                                ;*                          **TRAP HANDLING ROUTINES**
                                ;;*****************************************************************

                                ;;*****************************************************************
                                ;*
                                ;*      THIS SUBROUTINE WILL HANDLE ALL CPU TRAPS AND ABORTS, THROUGH
                                ;*      "ERRVEC" (000004).  IF THIS SUBROUTINE IS ENTERED BY A SECOND
                                ;*      TRAP BEFORE THE FIRST HAS BEEN PROCESSED A HALT IS EXECUTED.
                                ;*      IF THE WORD 'CPUEXP' IS ZERO, NO TRAP WAS EXPECTED AND AN
                                ;*      UNEXPECTED ERROR MESSAGE IS GIVEN.  IF THE WORD 'CPUEXP' IS
                                ;*      NOT ZERO THEN THE CPU ERROR REGISTER 'CPUERR' IS COMPARED WITH
                                ;*      'CPUEXP' TO SEE IF THE PROPER CONDITION OCCURRED.  'PCPUER' CAN
                                ;*      BE USED AS A FLAG TO INDICATE THAT A TRAP HAS OCCURRED SINCE IT
                                ;*      IS LOADED WITH THE ERROR REGISTER IF A TRAP VECTORS HERE
                                ;*
                                ;;*****************************************************************
004602  005227                  CPUER:  INC     (PC)+           ;MAKE FLAG ZERO IF FIRST TIME
004604  177777                  CPFLAG: .WORD   -1              ;NEGATIVE ONE FOR A FLAG
004606  001403                          BEQ     10$             ;BRANCH IF FIRST TIME IN
004610  005237  020014                  INC     $MSGTY          ;INDICATE TO APT A FATAL ERROR OCCURED
004614  000000                          HALT                    ;I HAVE ENTERED THIS ROUTINE BEFORE
                                ;I FINISHED REPORTING THE FIRST ERROR.  THE SECOND ENTRY ADDRESS IS ON
                                ;THE STACK, AND THE FIRST ERROR CONDITION IS PROBABLY STILL LOCKED UP.
004616  012637  001326          10$:    MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS IN CASE OF LOOP
004622  012637  001330                  MOV     (KSP)+,OLDPS    ;SAVE OLD PSW IN CASE OF LOOP
004626  013737  177766  001314          MOV     CPUERR,PCPUER   ;SAVE CPU ERROR REGISTER
004634  013737  001326  001324          MOV     OLDPC,BADPC     ;SAVE PC+2 AT TIME OF ABORT
004642  005737  001312                  TST     CPUEXP          ;SEE IF ANY CONDITION WAS EXPECTED
004646  001414                          BEQ     1$              ;BRANCH IF NO TRAP WAS EXPECTED
004650  105737  007774                  TSTB    CPUTYP          ;SEE IF THIS WAS AN 11/44
004654  001016                          BNE     2$              ;BRANCH TO CONTINUE IF AN 11/24
004656  023737  001314  001312          CMP     PCPUER,CPUEXP   ;SEE IF EXPECTED ERROR OCCURED
004664  001414                          BEQ     3$              ;BRANCH IF ERROR CODES MATCH
004666  012737  177777  004604          MOV     #-1,CPFLAG      ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
004674  104001                          ERROR   +1              ;NOT THE CORRECT CPU TRAP THROUGH 4
004676  000407                          BR      3$              ;SKIP NEXT INSTRUCTION
004700  012737  177777  004604  1$:     MOV     #-1,CPFLAG      ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
004706  104002                          ERROR   +2              ;UNEXPECTED CPU TRAP THROUGH 4
004710  000402                          BR      3$              ;SKIP NEXT INSTRUCTION
004712  005237  001302          2$:     INC     ERRCNT          ;INCREMENT ERRCNT TO SHOW AN ERROR FOR 11/24
004716  012737  177777  004604  3$:     MOV     #-1,CPFLAG      ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
004724  013737  001314  177766          MOV     PCPUER,CPUERR   ;CLEAR CPU ERROR REGISTER
004732  013746  001330                  MOV     OLDPS,-(KSP)    ;PUSH OLD PSW BACK ON STACK
004736  013746  001326                  MOV     OLDPC,-(KSP)    ;PUSH RETURN ADDRESS BACK ON STACK
004742  000006                          RTT                     ;RETURN FROM INTERRUPT OR ABORT
```

```
                                        .SBTTL  MEMORY MANAGEMENT TRAPS AND ABORTS HANDLER ROUTINE
                                        ;;*****************************************************************
                                        ;*
                                        ;*      THIS  ROUTINE WILL HANDLE ALL SPURIOUS MEMORY MANAGEMENT TRAPS
                                        ;*      AND ABORTS.  IT WILL REPORT THE CONDITION OF ALL THE MEMORY
                                        ;*      MAMAGEMENT STATUS REGISTERS, AND THEN RETURN TO THE TEST AND
                                        ;*      TRY TO CONTINUE RUNNING.
                                        ;*
                                        ;;*****************************************************************
004744  005227                  MMTRAP: INC     (PC)+           ;MAKE FLAG ZERO IF FIRST TIME
004746  177777                  MMFLAG: .WORD   -1              ;FLAG SHOULD BE NEG ONE
004750  001403                          BEQ     10$             ;BRANCH IF FIRST TIME INTO ROUTINE
004752  005237  020014                  INC     $MSGTY          ;INDICATE TO APT A FATAL ERROR OCCURED
004756  000000                          HALT    ;I HAVE ENTERED THIS ROUTINE BEFORE I FINISHED REPORTING THE
                                        ;FIRST ERROR.  THE SECOND ENTRY ADDRESS IS ON THE STACK AND THE FIRST ERROR
                                        ;CONDITION IS PROBABLY STILL LOCKED UP .
004760  011637  001324          10$:    MOV     (KSP),BADPC     ;SAVE PC AT TIME OF ABORT OR TRAP
004764  012637  001326                  MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS IN CASE OF LOOP
004770  012637  001330                  MOV     (KSP)+,OLDPS    ;SAVE OLD PSW IN CASE OF LOOP
004774  013737  177572  001334          MOV     MMR0,PMMR0      ;SAVE STATUS REGISTER
005002  013737  177574  001336          MOV     MMR1,PMMR1      ;SAVE AUTO INC/DEC REGISTER
005010  013737  177576  001340          MOV     MMR2,PMMR2      ;SAVE VIRTUAL ADDRESS REGISTER
005016  104003                          ERROR   +3              ;UNEXPECTED M.M. ABORT OR TRAP
005020  042737  177776  177572  1$:     BIC     #177776,MMR0    ;CLEAR ALL BITS EXCEPT 0
005026  012737  177777  004746          MOV     #-1,MMFLAG      ;RESTORE A NEGATIVE ONE TO FLAG
005034  013746  001330                  MOV     OLDPS,-(KSP)    ;PUSH OLD PSW ONTO STACK
005040  013746  001326                  MOV     OLDPC,-(KSP)    ;PUSH RETURN ADDRESS ON STACK
005044  000006                          RTT                     ;RETURN TO MAIN PROGRAM
```

```
                                        .SBTTL  SUBROUTINE TO TEST A LOCATION FOR WRITEABILITY
                                        ;;*****************************************************************
                                        ;*
                                        ;*      THIS SUBROUTINE CLEARS A TEST LOCATION, LOADS THE LOCATION USING
                                        ;*      THE MAP REGISTER, AND DETERMINES IF THE LOCATION WAS LOADED.  IF
                                        ;*      IT WAS, RETURN IS NORMAL TO THE TEST.  IF NOT, THE PC ON THE
                                        ;*      STACK IS UPDATED BY 2 AND THEN A RETURN IS EXECUTED.
                                        ;*
                                        ;;*****************************************************************
005046  062737  000200  172350  TSTLOC: ADD     #200,KIPAR4     ;MAP TO NEXT REGISTER
005054  005237  001172                  INC     $TMP0           ;INCREMENT REGISTER COUNTER
005060  105737  020034                  TSTB    $ENV            ;TEST APT STATUS
005064  001[illegible]                  BEQ     NEXT            ;BRANCH IF NOT UNDER APT
005066  005737  006154                   ST     FLOATR          ;SEE IF BIT 15 OF FLOATR IS SET
005072  100011                          BPL     1$              ;BRANCH IF STILL PLUS
005074  022705  020102                  CMP     #$DDW1,R5       ;SEE IF R5 IS POINTING TO UPPER DDW
005100  001417                          BEQ     NEXT            ;BRANCH IF SO - ALL DONE
005102  012705  020102                  MOV     #$DDW1,R5       ;MOVE ADDRESS OF DDW1 TO R5 AND
005106  012737  000001  006154          MOV     #BIT0,FLOATR    ;RESET BIT 0 IN FLOATR
005114  000411                          BR      NEXT            ;BRANCH OVER ASL
005116  005227                  1$:     INC     (PC)+           ;INCREMENT NEXT LOCATION FOR FIRST TIME THROUGH CHECK
005120  177777                  FTTHRU: .WORD   -1              ;FIRST TIME ENTRANCE FLAG
005122  001004                          BNE     1$              ;BRANCH IF NOT FIRST TIME
005124  012737  000001  006154          MOV     #BIT0,FLOATR    ;MOVE BIT 0 TO LOCATION FLOATR
005132  000402                          BR      NEXT            ;BRANCH OVER THE ASL
005134  006337  006154          1$:     ASL     FLOATR          ;ROTATE THE TEST BIT TO THE LEFT
005140  005037  037776          NEXT:   CLR     37776           ;CLEAR TEST LOCATION
005144  005037  001314                  CLR     PCPUER          ;CLEAR ERROR LOCATION
005150  010210                          MOV     R2,(R0)         ;TRY TO LOAD TEST CELL THROUGH MAP
005152  023702  037776                  CMP     37776,R2        ;SEE IF TEST LOCATION WAS LOADED
005156  001407                          BEQ     2$              ;BRANCH IF IT WAS LOADED
005160  005737  001314                  TST     PCPUER          ;SEE IF A TIMEOUT OCCURED
005164  001402                          BEQ     1$              ;BRANCH OVER SPECIAL STACK PUSH IF NOT
005166  013743  001172                  MOV     $TMP0,-(R3)     ;PUSH REGISTER NUMBER THAT TIMED OUT ON SPECIAL STACK
005172  062716  000002          1$:     ADD     #2,(SP)         ;CORRECT PC RETURN FOR LOAD FAILURE INDICATION
005176  000207                  2$:     RTS     PC              ;RETURN FROM THIS SUBROUTINE
```

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

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

```
                                        .SBTTL  SUBROUTINE TO TAKE PAR AND LOAD 2 WORDS EACH OF ADDROR & ADRAND
                                        ;*****************************************************************
                                        ;*
                                        ;*      THIS SUBROUTINE ASSUMES THE CONTENTS OF THE PAR, AND THE VIRTUAL
                                        ;*      ADDRESS HAVE BEEN PUT ON THE STACK.  IT TAKES THE PAR, SHIFTS IT
                                        ;*      TO EXPOSE THE UPPER ADDRESS BITS, BIT SETS THEM TO ADDROR+2, COM-
                                        ;*      PLIMENTS THE CONTENTS AND BIT CLEARS ADRAND+2.  AFTER RELOADING
                                        ;*      THE PAR IN R5, IT SHIFTS TO GET THE LOWER 16 BIT EQUIVALENT AND
                                        ;*      ADDS THE VIRTUAL ADDRESS TO CREATE THE PHYSICAL LOWER 16 BITS.
                                        ;*      THEN IT BIT SETS THEM TO ADDROR, COMPLIMENTS THE CONTENTS AND
                                        ;*      BIT CLEARS ADRAND.  ANOTHER COMPLIMENT BRINGS THE STATE BACK TO
                                        ;*      ITS ORIGINAL STATE.  THIS SUBROUTINE LEAVES WITH THE LOWER 16
                                        ;*      BITS AND THE UPPER 6 BITS ON THE STACK, AND ARE TO BE REMOVED IN
                                        ;*      THAT ORDER, AND MUST BE REMOVED AFTER RETURN.
                                        ;*
                                        ;*****************************************************************
005250  010546                  ADREXT: MOV     R5,-(SP)        ;SAVE R5
005252  016605  000006                  MOV     6(SP),R5        ;MOVE PAR CONTENTS TO R5 FOR SHIFTING
005256  072527  177766                  ASH     #-10.,R5        ;SHIFT R5 TO THE RIGHT 10 PLACES
005262  042705  177700                  BIC     #177700,R5      ;CLEAR BITS 15 TO 6
005266  010537  001204                  MOV     R5,$TMP5        ;MOVE OBTAINED UPPER 6 BITS TO $TMP5 FOR FUTURE TRANSFER
005272  050537  001236                  BIS     R5,ADDROR+2     ;SET THE 'OR' PATTERN OF UPPER 6 BITS TO ADDROR+2
005276  005105                          COM     R5              ;COMPLIMENT R5
005300  040537  001232                  BIC     R5,ADRAND+2     ;CLEAR THE 'AND' PATTERN OF UPPER 6 BITS TO ADRAND+2
005304  016605  000006                  MOV     6(SP),R5        ;PUT PAR CONTENTS BACK IN R5
005310  013766  001204  000006          MOV     $TMP5,6(SP)     ;MOVE UPPER 6 BITS OF PHYSICAL ADDRESS ON STACK
005316  042705  176000                  BIC     #176000,R5      ;STRIP OFF UPPER 6 BITS OF PAR CONTENTS
005322  072527  000006                  ASH     #6,R5           ;SHIFT REMAINING BITS 6 PLACES TO THE LEFT
005326  042766  160000  000004          BIC     #160000,4(SP)   ;STRIP OFF PAR PAGE BITS FROM ADDRESS TO FORM OFFSET
005334  060566  000004                  ADD     R5,4(SP)        ;FORM LOWER 16 BITS OF ADDRESS
005340  012605                          MOV     (SP)+,R5        ;RESTORE R5
005342  056637  000002  001234          BIS     2(SP),ADDROR    ;SET THE 'OR' PATTERN TO ADDROR
005350  005166  000002                  COM     2(SP)           ;COMPLIMENT THE ADDRESS
005354  046637  000002  001230          BIC     2(SP),ADRAND    ;CLEAR THE 'AND' PATTERN TO ADRAND
005362  005166  000002                  COM     2(SP)           ;RETURN ADDRESS TO ITS ORIGINAL STATE
005366  012605                          MOV     (SP)+,R5        ;SAVE RETURN ADDRESS
005370  012637  006144                  MOV     (SP)+,EADRES    ;POP STACK INTO EADRES
005374  012637  006146                  MOV     (SP)+,EADRES+2  ;POP STACK INTO EADRES+2
005400  010516                          MOV     R5,(SP)         ;RESTORE RETURN ADDRESS
005402  000207                          RTS     PC              ;RETURN
```

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

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

```
1427 006070  005257  001350                INC     DATA            ;CHANGE DATA PATTERN FOR NEXT PASS
1428 006074  042737  177700  001350        BIC     #177700,DATA    ;CLEAR UPPER 10 BITS OF DATA PATTERN                SEQ 0077
1429 006102  052737  000300  001350        BIS     #300,DATA       ;START WITH 3XX IN DATA PATTERN
1430 006110  005037  001312                CLR     CPUEXP          ;NO CPU TRAPS EXPECTED FOR AWHILE
1431 006114  005726                        TST     (SP)+           ;POP STACK EXPOSING FINAL RETURN ADDRESS
1432 006116  000207                        RTS     PC              ;EXIT
```

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

```
                                .SBTTL  PRE-TESTING SETUP
                                ;*******************
                                ;START OF TEST CODE
                                ;*******************

        010000                          .=10000                 ;START TEST CODE AT ADDRESS 10000 (2K)
010000                          START:
010000  012737  000340  177776          MOV     #340,@#PS       ;;LOCK OUT ALL INTERRUPTS
010006  012706  001100                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
010012  005026                  200$:   CLR     (R6)+           ;;CLEAR MEMORY LOCATION
010014  022706  001134                  CMP     #$TKS,R6        ;;DONE?
010020  001374                          BNE     200$            ;;LOOP BACK IF NO
010022  005037  020022                  CLR     $PASS           ;;& INITIALIZE PASS COUNT
010026  012706  001100                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
010032  012737  020140  000020          MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
010040  012737  000340  000022          MOV     #340,@#IOTVEC+2 ;;LEVEL 7
010046  012737  020520  000030          MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
010054  012737  000340  000032          MOV     #340,@#EMTVEC+2 ;;LEVEL 7
010062  012737  003724  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
010070  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
010076  012737  003766  000024          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
010104  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
010112  013737  021354  021346          MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
010120  005037  001210                  CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
010124  005037  001212                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
010130  112737  000001  001113          MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
010136  012737  021602  000014          MOV     #$RTRN,@#TBITVEC ;;SET "T" BIT VECTOR TO $RTRN
010144  012737  000340  000016          MOV     #340,@#TBITVEC+2 ;;LEVEL 7
010152  012737  000002  021602          MOV     #RTI,$RTRN      ;;SET $RTRN TO A RTI
010160  012737  010206  000010          MOV     #65$,@#RESVEC   ;;TRY TO DO A RTT
010166  005046                          CLR     -(SP)           ;;DUMMY PS
010170  012746  010176                  MOV     #64$,-(SP)      ;;AND PC
010174  000006                          RTT                     ;;TRY THE RTT
010176  012737  000006  021602  64$:    MOV     #RTT,$RTRN      ;;RTT IS LEGAL--SET $RTRN TO A RTT
010204  000402                          BR      66$
010206  062706  000010          65$:    ADD     #10,SP          ;;RTT ILLEGAL--CLEAN OFF THE STACK
010212  012737  000012  000010  66$:    MOV     #RESVEC+2,@#RESVEC ;;RESTORE TRAP CATCHER
010220  005037  021610                  CLR     $TBIT           ;;CLEAR "T" BIT SWITCH
010224  012737  010602  001104          MOV     #TST1+2,$LPADR  ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
010232  012737  010602  001106          MOV     #TST1+2,$LPERR  ;;SETUP THE ERROR LOOP ADDRESS
1469 010240  005227  177777             INC     #-1             ;;FIRST TIME?
010244  001020                          BNE     67$             ;;BRANCH IF NO
010246  022737  021534  000042          CMP     #$ENDAD,@#42    ;;ACT-11?
010254  001414                          BEQ     67$             ;;BRANCH IF YES
010256  104400  010264                  TYPE    ,68$            ;;TYPE ASCIZ STRING
010262  000411                          BR      67$             ;;GET OVER THE ASCIZ
010264     200     103     113  68$:    .ASCIZ  <CRLF>?CKKUAB0 UBI MAP?<CRLF>
                                        .EVEN
010306                          67$:
1470 010306  005737  020022             TST     $PASS           ;IS THIS THE FIRST PASS?
1471 010312  001067                     BNE     9$              ;BRANCH IF NOT
```

```
1472 010314  013746  000004                MOV     4,-(SP)         ;SAVE TIMEOUT VECTOR
1473 010320  012737  010410  000004        MOV     #2$,4           ;TIMEOUTS TO 104$                    SEQ 0080
1474 010326  013746  000006                MOV     6,-(SP)         ;SAVE PS VECTOR
1475 010332  012737  000340  000006        MOV     #340,6          ;PRIORITY 7
1476 010340  000007                        MFPT                    ;DETERMINE PROCESSOR TYPE
1477 010342  110037  007774                MOVB    R0,CPUTYP       ;MOVE THE CPU NUMBER TO CPUTYP
```

```
1478 010346  105337  007774                    DECB    CPUTYP          ;MAKE THE 11/44 VALUE ZERO
1479 010352  001005                            BNE     1$              ;BRANCH TO NEXT TEST IF NOT AN 11/44
1480 010354  104400  021775                    TYPE    ,CPUMSG         ;TYPE THE CPU TYPE HEADER
1481 010360  104400  022056                    TYPE    ,ELEV44         ;TYPE THE 11/44 NUMBER
1482 010364  000421                            BR      5$              ;CONTINUE
1483 010366  122737  000002  007774  1$:       CMPB    #2,CPUTYP       ;SEE IF THIS IS AN 11/24
1484 010374  001007                            BNE     3$              ;BRANCH TO FATAL ERROR MESSAGE PRINTING IF NOT
1485 010376  104400  021775                    TYPE    ,CPUMSG         ;TYPE THE CPU TYPE HEADER
1486 010402  104400  022052                    TYPE    ,ELEV24         ;TYPE THE 11/24 NUMBER
1487 010406  000410                            BR      5$              ;CONTINUE
1488 010410  062706  000004          2$:       ADD     #4,SP           ;CLEAN STACK AFTER TIMEOUT
1489 010414  005237  020014          3$:       INC     $MSGTY          ;TELL APT THIS IS A FATAL ERROR
1490 010420  104400  007446          4$:       TYPE    ,BADCPU         ;TYPE THE BAD CPU MESSAGE
1491 010424  000000                            HALT                    ;FATAL ERROR - THIS DIAGNOSTIC IS WRITTEN
1492                                                                    ;FOR 11/24 AND 11/44 PROCESSORS ONLY
1493 010426  000774                            BR      4$              ;DON'T ALLOW CONTINUE
1494 010430  012737  010450  000004  5$:       MOV     #6$,4           ;TIMEOUTS TO 120$
1495 010436  022777  177777  170504            CMP     #-1,@SWR        ;SEE IF HARDWARE SWITCH REGISTER EXISTS OR CONTAINS -1
1496 010444  001006                            BNE     8$              ;BRANCH TO CONTINUE IF IT EXISTS AND DOESN'T CONTAIN -1
1497 010446  000402                            BR      7$              ;BRANCH TO SET UP FOR SOFTWARE SWITCH REGISTER IF SO
1498 010450  062706  000004          6$:       ADD     #4,SP           ;CLEAN STACK AFTER TIMEOUT
1499 010454  012737  000176  001150  7$:       MOV     #$SSWR,SWR      ;MOVE ADDRESS OF SOFTWARE SWITCH REGISTER TO SWR
1500 010462  012637  000006          8$:       MOV     (SP)+,6         ;RESTORE TIMEOUT PS
1501 010466  012637  000004                    MOV     (SP)+,4         ;RESTORE TIMEOUT VECTOR
1502 010472  132737  000200  020035  9$:       BITB    #200,$ENVM      ;IS APT SIZING
1503 010500  001403                            BEQ     LOOP            ;BRANCH IF NOT
1504 010502  012737  020036  001150            MOV     #$SWREG,SWR     ;USE APT SWITCH REGISTER
1505 010510  012737  004744  000250  LOOP:     MOV     #MMTRAP,MMVEC   ;LOAD MEMORY MANAGEMENT TRAP SERVICE ROUTINE ADDRESS
1506 010516  012737  000340  000252            MOV     #340,MMVEC+2    ;SET PRIORITY SEVEN
1507 010524  012737  004602  000004            MOV     #CPUER,ERRVEC   ;LOAD CPU TRAP SERVICE ROUTINE ADDR
1508 010532  012737  000340  000006            MOV     #340,ERRVEC+2   ;SET PRIORITY SEVEN
1509 010540  005037  001312                    CLR     CPUEXP          ;NOT EXPECTING ANY CPU ERRORS
1510 010544  005037  177572                    CLR     MMR0            ;START IN 16 BIT MAPPING
1511 010550  005037  172516                    CLR     MMR3            ;DISABLE MAP AND 22-BIT MAPPING
1512 010554  012700  177777                    MOV     #-1,R0          ;NEGATIVE ONE USED TO INITIALIZE FLAGS
1513 010560  010037  004604                    MOV     R0,CPFLAG       ;INITIALIZE FLAGS
1514 010564  010037  004412                    MOV     R0,TOFLAG       ;INITIALIZE FLAGS
1515 010570  010037  004746                    MOV     R0,MMFLAG       ;INITIALIZE FLAGS
1516 010574  010037  177766                    MOV     R0,CPUERR       ;CLEAR CPU ERROR REGISTER
```

```
                        .SBTTL  TEST 1 - MAP REGISTER RESPONSE TEST
                        ;;*****************************************************************
                        ;*TEST 1         MAP REGISTER RESPONSE TEST
                        ;*
                        ;*       THIS TEST IS USED TO ENSURE THAT ALL THE UNIBUS MAP REGISTERS
                        ;*       CAN BE REFERENCED UNDER PROGRAM CONTROL, WITHOUT TIMING OUT.
                        ;*       THE ADDRESSES OF ANY MAP REGISTERS THAT TIME OUT WILL BE REPORTED
                        ;*       AND, AT THE END OF THE TEST, A SUMMARY OF THOSE REGISTERS WILL
                        ;*       BE GIVEN.
                        ;*
                        ;;*****************************************************************
                        TST1:
 010600                         SCOPE
 010600  000004
 010602  004737  017662          JSR     PC,PRETST       ;GO SET UP PRETEST DATA
 010606  011022  010634  000001  .WORD   TST2,20$,1       ;DATA USED BY PRETST
1527 010614  012737  004410  000004  MOV     #TIMEOUT,ERRVEC ;LOAD ERRVEC WITH ROUTINE ADDRESS
1528 010622  105737  007774          TSTB    CPUTYP          ;TEST TO SEE WHICH CPU IS RUNNING THIS DIAGNOSTIC
1529 010626  001460                  BEQ     5$              ;BRANCH AROUND MAP REGISTER EXISTENCE CHECK IF 11/44
1530 010630  005037  001302          CLR     ERRCNT          ;CLEAR THE ERROR COUNTER
1531 010634  017702  157340  20$:    MOV     @MAPLO,R2       ;READ FIRST MAP REGISTER TO R2
1532 010640  005737  001302          TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
1533 010644  001451                  BEQ     5$              ;BRANCH TO CHECK THEM ALL IF NONE, THEY ARE IN THIS 11/24
1534 010646  005037  001302          CLR     ERRCNT          ;CLEAR THE ERROR COUNTER
1535 010652  105037  006162          CLRB    SPECST          ;CLEAR THE CHARACTER RECEIVER LOCATION
1536 010656  105737  020034          TSTB    $ENV            ;ARE WE RUNNING UNDER APT
1537 010662  001036                  BNE     4$              ;BRANCH IF SO
1538 010664  104400  007613          TYPE    ,MRQUES         ;ASK USER IF THERE ARE MAP REGISTERS IN THIS 11/24
1539 010670  105737  006162          TSTB    SPECST          ;SEE IF A CHARACTER WAS INPUTED DURING THE PRINTING
1540 010674  001006                  BNE     2$              ;BRANCH TO CHECK INPUT IF SO
1541 010676  105777  170232  1$:     TSTB    @$TKS           ;SEE IF A CHARACTER IS INPUTED
1542 010702  100375                  BPL     1$              ;BRANCH BACK IF NOT
1543 010704  117737  170226  006162  MOVB    @$TKB,SPECST    ;MOVE THE CHARACTER TO THE SPECST LOCATION
1544 010712  142737  000200  006162  2$:     BICB    #200,SPECST     ;CLEAR THE PARITY BIT
1545 010720  105777  170214  3$:     TSTB    @$TPS           ;SEE IF PRINTER IS READY
1546 010724  100375                  BPL     3$              ;BRANCH BACK UNTIL READY
1547 010726  113777  006162  170206  MOVB    SPECST,@$TPB    ;PRINT THE CHARACTER
1548 010734  104400  001221          TYPE    ,$CRLF          ;TYPE A <CRLF>
1549 010740  122737  000131  006162  CMPB    #'Y,SPECST      ;SEE IF THIS WAS A 'Y'
1550 010746  001410                  BEQ     5$              ;BRANCH AROUND FATAL MESSAGE IF EQUAL TO A 'Y'
1551 010750  104400  007702          TYPE    ,GMRMOD         ;TYPE:  'DIAGNOSTIC CHECKS THIS MODULE - INSERT
1552                                                          ;       'BEFORE RE-RUNNING'
1553 010754  000000                  HALT                    ;FATAL ERROR - WAIT FOR USER ACTION
1554 010756  000726                  BR      20$             ;TRY AGAIN - RESTART REQUESTED
1555 010760  005237  020014  4$:     INC     $MSGTY          ;TELL APT THIS IS A FATAL ERROR
1556 010764  000000                  HALT                    ;HALT - FATAL ERROR
1557 010766  000722                  BR      20$             ;TRY AGAIN - RESTART REQUESTED
1558 010770  012700  170200  5$:     MOV     #MAPLO,R0       ;PUT FIRST MAP REGISTER ADDR IN R0
1559 010774  012737  011006  001106  MOV     #6$,$LPERR      ;SET LOOP ON ERROR POINTER TO 6$
1560 011002  012703  000100          MOV     #100,R3         ;TEST ALL MAP REGISTERS
1561 011006  012002  6$:     MOV     (R0)+,R2        ;READ MAP REGISTERS TO R2
1562 011010  077302                  SOB     R3,6$           ;DO ALL OF THEM
1563 011012  005737  001302          TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
1564 011016  001401                  BEQ     TST2            ;;GO TO NEXT TEST IF NO ERRORS
1565 011020  104004                  ERROR   +4              ;SUMMARY OF MAP REGISTERS THAT TIMED OUT ON READ
```

```
1580                                    .SBTTL  TEST 2 - BIT PATTERN AND CLEAR TEST OF 40 MAP REGISTERS
                                        ;;*****************************************************************
                                        ;*TEST 2        BIT PATTERN AND CLEAR TEST OF 40 MAP REGISTERS
                                        ;*
                                        ;*      THIS TEST WILL RUN 7 BIT PATTERNS THROUGH BOTH WORDS OF 40 UNIBUS
                                        ;*      MAP REGISTER LOCATIONS MAPL00 - MAPL37, USING TWO MAJOR PASSES.
                                        ;*      IT WILL TEST THE LOWER 16 BITS ON THE FIRST PASS, AND THE UPPER
                                        ;*      6 BITS ON THE SECOND.  THIS TEST WILL MAKE SURE THE REGISTER CAN
                                        ;*      CLEAR, AND THEN RUN 6 BIT PATTERNS THROUGH EACH UNIBUS MAP REGISTER.
                                        ;*      IF THE DATA PATTERN RECEIVED DOES NOT MATCH THE EXPECTED PATTERN,
                                        ;*      THEN THE MAP REGISTER ADDRESS, DATA RECEIVED, PATTERN LOADED, AND
                                        ;*      PATTERN EXPECTED ARE REPORTED.  AT THE END OF EACH OF THE TWO MAJOR
                                        ;*      PASSES, A SUMMARY OF ALL ERRORS IS GIVEN SO THAT YOU CAN DETERMINE
                                        ;*      IF THE ERROR IS BIT SENSITIVE OR REGISTER SENSITIVE.
                                        ;*
                                        ;;*****************************************************************
     011022                             TST2:
     011022  000004                             SCOPE
     011024  004737  017662                     JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     011030  011400  011044  000002             .WORD   TST3,20$,2      ;DATA USED BY PRETST
1581 011036  012737  004602  000004             MOV     #CPUER,ERRVEC   ;LOAD CPU TRAP SERVICE ROUTINE ADDRESS
1582 011044  005037  001302          20$:       CLR     ERRCNT          ;CLEAR THE ERROR COUNT LOCATION
1583 011050  012700  170200                     MOV     #MAPL00,R0      ;MOVE STARTING ADDRESS OF LOWER 16 BITS REGISTER TO R0
1584 011054  005037  006156                     CLR     MASK1           ;MOVE '0' TO MASK1
1585 011060  012737  000001  006160             MOV     #1,MASK2        ;MOVE '1' TO MASK2
1586 011066  012701  000040          1$:        MOV     #40,R1          ;DO 40 REGISTERS LOOP COUNTER
1587 011072  012702  000007          2$:        MOV     #7,R2           ;MOVE PATTERN LOOP COUNTER TO R2
1588 011076  012705  021712                     MOV     #PATRNS,R5      ;MOVE PATTERN START TO R5
1589 011102  011504                  3$:        MOV     (R5),R4         ;MOVE NEXT COUNT PATTERN TO R2
1590 011104  043704  006156                     BIC     MASK1,R4        ;USE MASK1 PRE-LOADED FOR PROPER LOADING
1591 011110  010410                             MOV     R4,(R0)         ;LOAD MAP REGISTER WITH COUNT PATTERN
1592 011112  011504                             MOV     (R5),R4         ;RELOAD PATTERN
1593 011114  043704  006160                     BIC     MASK2,R4        ;USE THE 2ND MASK TO CONSTRUCT EXPECTED VALUE
1594 011120  020410                             CMP     R4,(R0)         ;COMPARE EXPECTED WITH RECEIVED
1595 011122  001450                             BEQ     7$              ;BRANCH IF DATA IS OK
1596 011124  011003                  4$:        MOV     (R0),R3         ;READ BAD MAP REGISTER DATA INTO R3
1597 011126  011537  001172                     MOV     (R5),$TMP0      ;MOVE PATTERN TO $TMP0
1598 011132  043737  006156  001172             BIC     MASK1,$TMP0     ;FORM INPUT PATTERN FOR ERROR PRINTING
1599 011140  013746  172356                     MOV     KIPAR7,-(SP)    ;PUT PAR CONTENTS ON STACK FOR ADREXT SUBROUTINE
1600 011144  010046                             MOV     R0,-(SP)        ;PUT VIRTUAL ADDRESS ON STACK FOR ADREXT SUBROUTINE
1601 011146  004737  005250                     JSR     PC,ADREXT       ;GO SET DATA IN THE 4 WORDS OF ADDROR AND ADRAND
1602                                                                     ;AND FORM A PHYSICAL 22-BIT ADDRESS
1603 011152  011546                             MOV     (R5),-(SP)      ;PUT PATTERN ON STACK FOR PATEXT SUBROUTINE USE
1604 011154  004737  005224                     JSR     PC,PATEXT       ;GO SET DATA INTO PATTOR AND PATAND
1605 011160  011046                             MOV     (R0),-(SP)      ;PUT DATA ON STACK FOR DATEXT SUBROUTINE
1606 011162  004737  005200                     JSR     PC,DATEXT       ;SUBROUTINE TO LOAD DATAOR AND DATAND
1607 011166  010037  006150                     MOV     R0,EADRS2       ;MOVE ADDRESS IN R0 TO EADRES FOR ERROR CALL
1608 011172  012737  011202  001106             MOV     #5$,$LPERR      ;RESET LOOP ON ERROR FOR TIGHT LOOP
1609 011200  000411                             BR      6$              ;BRANCH OVER LOOP ON ERROR SECTION
1610 011202  011504                  5$:        MOV     (R5),R4         ;RELOAD PATTERN
1611 011204  043704  006156                     BIC     MASK1,R4        ;PREPARE PATTERN FOR LOAD
```

```
1612 011210  010410                    MOV     R4,(R0)         ;RELOAD MAP REGISTER WITH COUNT PATTERN
1613 011212  011504                    MOV     (R5),R4         ;RELOAD PATTERN                                  SEQ 0084
1614 011214  043704  006160            BIC     MASK2,R4        ;PREPARE PATTERN FOR EXPECTED
1615 011220  020410                    CMP     R4,(R0)         ;SEE IF PATTERN IS OK NOW
1616 011222  001401                    BEQ     65$             ;BRANCH IF SO
1617 011224  104203            6$:     ERROR   +203            ;THE BIT PATTERN THROUGH THE MAP REGISTERS FAILED
```

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

```
1660                                    .SBTTL  TEST 3 - DUAL ADDRESS LOADS & READS MAP REG'S
                                        ;;*****************************************************
                                        ;*TEST 3        DUAL ADDRESS LOADS & READS MAP REG'S
                                        ;*
                                        ;*      THIS TEST ENSURES THAT ONLY ONE UNIBUS MAP REGISTER IS LOADED
                                        ;*      DURING A "MOV   #DATA,MAPREG" INSTRUCTION.  ALL MAP REGISTERS
                                        ;*      ARE CLEARED AND ONE REGISTER AT A TIME, STARTING WITH MAPL00,
                                        ;*      IS LOADED WITH A -1.  THEN, ALL MAP REGISTERS ARE READ,
                                        ;*      STARTING WITH MAPH37, AND VERIFIED TO BE ZERO.  ANY REGISTER
                                        ;*      THAT IS NOT ZERO AND WHOSE UNIBUS ADDRESS DOES NOT MATCH THAT
                                        ;*      OF THE REGISTER UNDER TEST IS REPORTED TO BE IN ERROR.  AT THE
                                        ;*      END OF THE TEST A SUMMARY OF ALL DUALED REGISTERS IS GIVEN.
                                        ;*
                                        ;;*****************************************************
     011400                             TST3:
     011400  000004                             SCOPE
     011402  004737  017662                     JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     011406  011710  011470  000003             .WORD   TST4,20$,3      ;DATA USED BY PRETST
1661 011414  042737  000001  177572             BIC     #1,MMR0         ;TURN OFF MEMORY MANAGEMENT
1662 011422  012700  170200                     MOV     #MAPL00,R0      ;LOAD ADDRESS OF MAPL00 IN R0
1663 011426  012702  177700                     MOV     #177700,R2      ;SET UP XOR DATA
1664 011432  012701  170400             1$:     MOV     #MAPH37+2,R1    ;PUT ADDRESS OF MAPH37+2 IN R1
1665 011436  012737  000077  001172             MOV     #77,$TMP0       ;SET UP TEST PATTERN IN $TMP0
1666 011444  004737  004352                     JSR     PC,CLRMAP       ;CLEAR ALL MAP REGISTERS FOR TEST
1667 011450  012703  000100                     MOV     #100,R3         ;SET UP LOOP COUNTER FOR TESTING 40 REGISTERS, 2 WORDS EACH
1668 011454  005037  172516                     CLR     MMR3            ;CLEAR MMR3
1669 011460  012704  000100             2$:     MOV     #100,R4         ;SET UP LOOP COUNTER FOR CHECKING 40 REGISTERS, 2 WORDS EACH
1670 011464  074237  001172                     XOR     R2,$TMP0        ;REVERSE BIT STATES OF $TMP0 IN BITS 7 TO 15
1671 011470  013710  001172             20$:    MOV     $TMP0,(R0)      ;LOAD MAP REGISTER UNDER TEST
1672 011474  012701  170400                     MOV     #MAPH37+2,R1    ;PUT ADDRESS OF MAPH37+2 IN R1
1673 011500  005741                     3$:     TST     -(R1)           ;SEE IF MAP REGISTER IS ZERO
1674 011502  001424                             BEQ     6$              ;GO SEE IF MORE REGISTERS TO TEST IF = 0
1675 011504  005011                             CLR     (R1)            ;CLEAR THE FAILED LOCATION
1676 011506  020100                             CMP     R1,R0           ;SEE IF NON-ZERO REGISTER ADDRESS MATCHES ADDRESS LOADED
1677 011510  001421                             BEQ     6$              ;GO SEE IF MORE REGISTERS TO TEST IF SO
1678 011512  011137  001200                     MOV     (R1),$TMP3      ;MOVE FAULTY DATA TO $TMP3
1679 011516  062701  000002                     ADD     #2,R1           ;PREPARE FOR POSSIBLE ERROR LOOPING
1680 011522  013746  172356                     MOV     KIPAR7,-(SP)    ;PUT PAR ON STACK FOR ADREXT SUBROUTINE USE
1681 011526  010146                             MOV     R1,-(SP)        ;PUT VIRTUAL ADDRESS ON STACK FOR ADREXT SUBROUTINE USE
1682 011530  004737  005250                     JSR     PC,ADREXT       ;GO SET DATA IN THE 4 WORDS OF ADDROR AND ADRAND
1683                                                                     ;AND FORM A PHYSICAL 22-BIT ADDRESS
1684 011534  010046                             MOV     R0,-(SP)        ;PUT ADDRESS ON STACK FOR DATEXT SUBROUTINE USE
1685 011536  004737  005200                     JSR     PC,DATEXT       ;SUBROUTINE TO LOAD DATAOR AND DATAND
1686 011542  010037  006150                     MOV     R0,EADRS2       ;MOVE ADDRESS IN R0 TO EADRS2 FOR ERROR CALL
1687 011546  104202                             ERROR   +202            ;DUAL ADDRESSING ERROR IN THE UNIBUS MAP
1688 011550  162701  000002             5$:     SUB     #2,R1           ;UNDO ERROR LOOPING PREPARATION - NO LOOPING
1689 011554  077427                     6$:     SOB     R4,3$           ;BRANCH IF MORE REGISTERS TO CHECK
1690 011556  005020                             CLR     (R0)+           ;CLEAR THE REGISTER JUST TESTED AND POINT TO NEXT REGISTER
1691 011560  005303                             DEC     R3              ;DECREMENT LOOP COUNTER AND
1692 011562  001336                             BNE     2$              ;BRANCH IF MORE REGISTERS TO TEST
1693 011564  005737  001302                     TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
```

```
1694 011570  001403                    BEQ     RELC22          ;GO TO NEXT SECTION IF NO ERRORS
1695 011572  005337  001110              DEC     $ERTTL          ;DON'T COUNT ERROR +5 AS ANOTHER ERROR                SEQ 0087
1696 011576  104005                     ERROR   +5              ;SUMMARY OF DUAL ADDRESSING ERRORS ON LOADING MAP REGISTERS
1697 011600  104420           RELC22:   TBITR                   ;RESTORE THE T BIT TO ITS CONDITION
1698                                                             ;BEFORE THE LAST TEST
1699 011602  012700  077406              MOV     #77406,R0       ;MAKE THE KERNEL I-SPACE PAGES ALL
```

```
1700                                                                    ;4K, UPWARD EXPANDABLE, READ/WRITE
1701 011606  012701  172300                    MOV     #KIPDR0,R1      ;MOVE ADDRESS OF KIPDR0 TO R1
1702 011612  010021                            MOV     R0,(R1)+        ;KERNEL I-SPACE PAGE 0 KIPDR0
1703 011614  010021                            MOV     R0,(R1)+        ;KERNEL I-SPACE PAGE 1 KIPDR1
1704 011616  010021                            MOV     R0,(R1)+        ;KERNEL I-SPACE PAGE 2 KIPDR2
1705 011620  010021                            MOV     R0,(R1)+        ;KERNEL I-SPACE PAGE 3 KIPDR3
1706 011622  010021                            MOV     R0,(R1)+        ;KERNEL I-SPACE PAGE 4 KIPDR4
1707 011624  010021                            MOV     R0,(R1)+        ;KERNEL I-SPACE PAGE 5 KIPDR5
1708 011626  010021                            MOV     R0,(R1)+        ;KERNEL I-SPACE PAGE 6 KIPDR6
1709 011630  010021                            MOV     R0,(R1)+        ;KERNEL I-SPACE PAGE 7 KIPDR7
1710 011632  012701  172340                    MOV     #KIPAR0,R1      ;MOVE ADDRESS OF KIPAR0 TO R1
1711 011636  005021                            CLR     (R1)+           ;MAP KIPAR0 TO PHYSICAL 0
1712 011640  012721  000200                    MOV     #200,(R1)+      ;MAP KIPAR1 TO PHYSICAL 4K - 8K
1713 011644  012721  000400                    MOV     #400,(R1)+      ;MAP KIPAR2 TO PHYSICAL 8K - 12K
1714 011650  012721  000600                    MOV     #600,(R1)+      ;MAP KIPAR3 TO PHYSICAL 12K - 16K
1715 011654  012721  001000                    MOV     #1000,(R1)+     ;MAP KIPAR4 TO PHYSICAL 16K - 20K
1716 011660  012721  001200                    MOV     #1200,(R1)+     ;MAP KIPAR5 TO PHYSICAL 20K - 24K
1717 011664  012721  170000                    MOV     #170000,(R1)+   ;MAP KIPAR6 TO UNIBUS
1718 011670  012721  177600                    MOV     #177600,(R1)+   ;MAP KIPAR7 TO I/O PAGE
1719 011674  012737  000001  177572           MOV     #BIT0,MMR0      ;ENABLE FULL 18-BIT MAPPING
1720 011702  012737  000020  172516           MOV     #BIT4,MMR3      ;ENABLE 22-BIT MAPPING
1721                                 ;*        AT THIS POINT 22-BIT RELOCATION FROM MEMORY MANAGEMENT
1722                                 ;*        IS ENABLED, WITH THE KIPAR'S MAPPED TO PHYSICAL 0-24K.
1723                                 ;*        KIPAR6 IS MAPPED TO THE UNIBUS (170000) AND
1724                                 ;*        KIPAR7 IS MAPPED TO THE I/O PAGE (177600).
```

```
1735                                    .SBTTL  TEST 4 - MAP REGISTER ADDRESS DECODE TEST
                                        ;;*****************************************************************
                                        ;*TEST 4        MAP REGISTER ADDRESS DECODE TEST
                                        ;*
                                        ;*      THIS TEST TRIES TO VERIFY THAT NONE OF THE INPUTS TO THE ADDRESS
                                        ;*      DECODER FOR THE UNIBUS MAP REGISTERS IS STUCK TRUE.  KIPAR6 IS
                                        ;*      SET UP TO HOLD 177702 AND R4 HAS THE VIRTUAL ADDRESS TO SELECT
                                        ;*      MAPL00, THROUGH KIPAR6.  THE TEST THEN CHANGES ONE BIT AT A TIME
                                        ;*      IN PAR6 SO THAT IT SHOULD NEVER REFERENCE MAPL00.  IF IT DOES, AN
                                        ;*      ERROR IS REPORTED.
                                        ;*
                                        ;;*****************************************************************
     011710                             TST4:
     011710  000004                             SCOPE
     011712  004737  017662                     JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     011716  012130  011724  000004             .WORD   TST5,20$,4      ;DATA USED BY PRETST
1736 011724  012737  000020  001312  20$:       MOV     #TIMOUT,CPUEXP  ;EXPECTING CPU TIME OUT ON UNIBUS
1737 011732  013737  172354  001176             MOV     KIPAR6,$TMP2    ;SAVE KIPAR6 FOR RESTORATION LATER
1738 011740  012737  177702  172354             MOV     #177702,KIPAR6  ;PUT MAP REGISTER 0 ADDR IN PAR6
1739 011746  012702  175254                     MOV     #175254,R2      ;PATTERN FOR TESTING.
1740 011752  010237  170200                     MOV     R2,MAPL0        ;LOAD MAP REGISTER 0
1741 011756  012700  004000                     MOV     #BIT11,R0       ;SET BIT 11 TO FLOAT THROUGH PAR6
1742 011762  012704  140000                     MOV     #140000,R4      ;VIRT.ADDR. TO SELECT PAR6
1743 011766  074037  172354             1$:     XOR     R0,KIPAR6       ;CHANGE A BIT OF MAP REGISTER 0'S ADDR
1744 011772  010046                             MOV     R0,-(SP)        ;SAVE R0 ON STACK
1745 011774  013746  172354                     MOV     KIPAR6,-(SP)    ;SAVE KIPAR6 ON STACK
1746 012000  011401                             MOV     (R4),R1         ;READ LOCATION POINTED TO BY PAR6
1747 012002  020201                             CMP     R2,R1           ;SEE IF DATA FETCHED MATCHES PATTERN
1748 012004  001035                             BNE     3$              ;BRANCH IF NOT SAME
1749 012006  005014                             CLR     (R4)            ;TRY TO CLEAR THIS LOCATION
1750 012010  005737  170200                     TST     MAPL0           ;SEE IF MAP REGISTER 0 GOT CLEARED
1751 012014  001030                             BNE     2$              ;BRANCH IF MAP REGISTER NOT ZERO
1752 012016  010237  170200                     MOV     R2,MAPL0        ;RESTORE MAPL0
1753 012022  013746  172354                     MOV     KIPAR6,-(SP)    ;PUT PAR ON STACK FOR ADREXT SUBROUTINE USE
1754 012026  010446                             MOV     R4,-(SP)        ;PUT VIRTUAL ADDRESS ON STACK FOR ADREXT SUBROUTINE USE
1755 012030  004737  005250                     JSR     PC,ADREXT       ;GO SET DATA IN THE 4 WORDS OF ADDROR AND ADRAND
1756 012034  012737  012044  001106             MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
1757 012042  000411                             BR      11$             ;GO CALL ERROR
1758 012044  011401                     10$:    MOV     (R4),R1         ;READ LOCATION POINTED TO BY PAR6
1759 012046  020201                             CMP     R2,R1           ;SEE IF DATA FETCHED MATCHES PATTERN
1760 012050  001013                             BNE     3$              ;BRANCH IF NOT SAME
1761 012052  005014                             CLR     (R4)            ;TRY TO CLEAR THIS LOCATION
1762 012054  005737  170200                     TST     MAPL0           ;SEE IF MAP REGISTER 0 GOT CLEARED
1763 012060  001006                             BNE     2$              ;BRANCH IF MAP REGISTER NOT ZERO
1764 012062  010237  170200                     MOV     R2,MAPL0        ;RESTORE MAPL0
1765 012066  104207                     11$:    ERROR   +207            ;GOT TO MAPL0 WITH ONE BIT DIFFERENT IN ADDRESS FROM 1777020
1766 012070  012737  000077  006146             MOV     #77,EADRES+2    ;RESTORE 77 TO EADRES+2
1767 012076  010114                     2$:     MOV     R1,(R4)         ;RELOAD THE LOCATION
1768 012100  012637  172354             3$:     MOV     (SP)+,KIPAR6    ;RESTORE KIPAR6
1769 012104  012600                             MOV     (SP)+,R0        ;RESTORE R0
1770 012106  074037  172354                     XOR     R0,KIPAR6       ;RESTORE BIT TO ORIGINAL STATUS
1771 012112  006200                             ASR     R0              ;RIGHT SHIFT ONE PLACE
1772 012114  001324                             BNE     1$              ;GO CONTINUE TEST IF BIT NOT SHIFTED OUT YET
1773 012116  005037  001312             4$:     CLR     CPUEXP          ;ZERO EXPECTED CPU TRAP FLAG
1774 012122  013737  001176  172354             MOV     $TMP2,KIPAR6    ;RESTORE KIPAR6
```

```
                                .SBTTL  TEST 5 - DATA PATH, UNIBUS TO MAIN MEMORY
                                ;;*****************************************************************
                                ;*TEST 5        DATA PATH, UNIBUS TO MAIN MEMORY
                                ;*
                                ;*      THIS TEST RUNS A COUNT PATTERN THROUGH A MEMORY LOCATION VIA
                                ;*      THE UNIBUS.  THE UNIBUS MAP IS LEFT OFF DURING THIS TEST SO
                                ;*      THAT THE ADDRESS IS NOT RELOCATED.  THE TEST TRIES TO LOAD THE
                                ;*      PATTERN INTO ADDRESS 040000 (8K) BUT IF THE MAP JUMPERS ARE
                                ;*      SET NOT TO RESPOND TO THAT ADDRESS THE NEXT 4K IS TRIED UNTIL
                                ;*      THE TEST GETS TO MAIN MEMORY FROM THE UNIBUS.  IF THIS TEST
                                ;*      DETERMINES THAT IT CANNOT GET TO MAIN MEMORY FROM THE UNIBUS
                                ;*      IT REPORTS THE FACT AND SKIPS THE NEXT TEST FOR VERIFICATION.
                                ;*
                                ;;*****************************************************************
      012130                    TST5:
      012130  000004                    SCOPE
      012132  004737  017662            JSR     PC,PRETST       ;GO SET UP PRETEST DATA
      012136  012470  012162  000005    .WORD   TST6,20$,5      ;DATA USED BY PRETST
 1788 012144  004737  004352            JSR     PC,CLRMAP       ;CLEAR ALL MAP REGISTERS
 1789 012150  012737  000020  001312    MOV     #TIMOUT,CPUEXP  ;TIMEOUTS MIGHT OCCUR IN THIS TEST.
 1790 012156  012704  000035            MOV     #35,R4          ;DO 35 ACCESSES
 1791 012162  012737  170400  172354  20$:  MOV     #170400,KIPAR6  ;START WITH ADDRESS 8K FROM UNIBUS
 1792 012170  012737  170400  001206        MOV     #170400,$TMP6   ;MOVE IT TO $TMP6 ALSO
 1793 012176  005037  001314          2$:   CLR     PCPUER          ;CLEAR ERROR CONDITION LOCATION
 1794 012202  013700  140000                MOV     140000,R0       ;TRY TO READ ADDRESS POINTED TO BY PAR6
 1795 012206  005737  001314                TST     PCPUER          ;SEE IF READ OF ADDRESS TIMED OUT
 1796 012212  001411                        BEQ     4$              ;BRANCH IF REFERENCE WAS GOOD
 1797 012214  062737  000200  172354  3$:   ADD     #200,KIPAR6     ;TRY NEXT 4K BLOCK OF MEMORY
 1798 012222  062737  000200  001206        ADD     #200,$TMP6      ;ADD 200 TO $TMP6 ALSO
 1799 012230  077416                        SOB     R4,2$           ;SUBTRACT 1 FROM R4 AND BRANCH IF NOT DONE
 1800 012232  104010                        ERROR   +10             ;NO UNIBUS ADDRESSES RESPOND
 1801 012234  000571                        BR      SIZEJ           ;BRANCH TO SIZE JUMPER TEST
 1802 012236  013737  172354  172352  4$:   MOV     KIPAR6,KIPAR5   ;PUT PAR6 INTO PAR5
 1803 012244  013737  001206  001204        MOV     $TMP6,$TMP5     ;DO SAME TRANSFER
 1804 012252  042737  170000  172352        BIC     #170000,KIPAR5  ;MAKE PAR5 A NON UNIBUS ADDRESS
 1805 012260  042737  170000  001204        BIC     #170000,$TMP5   ;CLEAR SAME BITS
 1806 012266  012737  173214  120000        MOV     #173214,120000  ;PUT RANDOM NUMBER INTO TEST LOCATION BY FAST BUS
 1807 012274  013701  140000                MOV     140000,R1       ;READ TEST LOCATION BY UNIBUS
 1808 012300  022701  173214                CMP     #173214,R1      ;SEE IF DATA WAS READ PROPERLY
 1809 012304  001406                        BEQ     5$              ;DATA OKAY NOW VERIFY DATA PATH
 1810 012306  020001                        CMP     R0,R1           ;SEE IF DATA CHANGED FROM FIRST READ
 1811 012310  001004                        BNE     5$              ;BRANCH AROUND NEXT TRY IF SO
 1812 012312  062737  000200  172354        ADD     #200,KIPAR6     ;TRY NEXT 4K BLOCK OF MEMORY
 1813 012320  077452                        SOB     R4,2$           ;BRANCH BACK TO TRY NEXT 4K BLOCK
 1814 012322  012701  021712          5$:   MOV     #PATRNS,R1      ;LOAD ADDRESS OF BIT PATTERNS IN R1
 1815 012326  012702  140000                MOV     #140000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
 1816 012332  012703  000007                MOV     #7,R3           ;DO 7 PATTERNS
 1817 012336  011112          6$:   MOV     (R1),(R2)       ;LOAD COUNT INTO TEST LOCATION VIA U.B.
 1818 012340  021112                        CMP     (R1),(R2)       ;COMPARE COUNT WITH DATA READ
 1819 012342  001426                        BEQ     7$              ;BRANCH IF DATA MATCHES
 1820 012344  011137  001172                MOV     (R1),$TMP0      ;MOVE EXPECTED PATTERN TO $TMP0
```

```
1821 012350  011257  001174                MOV     (R2),$TMP1      ;MOVE RECEIVED PATTERN TO $TMP1
1822 012354  011246                        MOV     (R2),-(SP)      ;PUT DATA ON STACK FOR DATEXT SUBROUTINE USE       SEQ 0091
1823 012356  004737  005200                JSR     PC,DATEXT       ;SUBROUTINE TO LOAD DATAOR AND DATAND
1824 012362  011146                        MOV     (R1),-(SP)      ;PUT PATTERN ON STACK FOR PATEXT SUBROUTINE USE
1825 012364  004737  005224                JSR     PC,PATEXT       ;GO SET DATA INTO PATTOR AND PATAND
1826 012370  012737  012400  001106        MOV     #61$,$LPERR     ;SET LOOP ON ERROR POINTER TO 61$
```

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

```
                                .SBTTL  TEST 6 - MAP DOESN'T RELOCATE IF NOT ENABLED
                                ;;*****************************************************************
                                ;*TEST 6        MAP DOESN'T RELOCATE IF NOT ENABLED
                                ;*
                                ;*      THIS TEST VERIFIES THAT THE UNIBUS MAP DOES NOT RELOCATE IF BIT5
                                ;*      OF MMR3 IS NOT SET.  THE TEST ASSUMES THAT THE PREVIOUS TEST HAS
                                ;*      RUN SUCCESSFULLY AND LEFT KIPAR6 POINTING TO THE FIRST UNIBUS
                                ;*      MAPPING REGISTER THAT THE UNIBUS MAP WILL RESPOND TO GREATER
                                ;*      THAN OR EQUAL TO MAPREG #2.  KIPAR5 IS ALSO POINTING TO THE
                                ;*      SAME MEMORY BASE ADDRESS EXCEPT IT POINTS OVER THE FASTBUS.
                                ;*      THE TEST THEN SETS ONE BIT IN EACH A.L.U. OF THE UNIBUS MAP
                                ;*      AND TRIES TO REFERENCE MAIN MEMORY OVER THE UNIBUS.  SINCE THE
                                ;*      MAP IS NOT ENABLED THE LOAD WILL GO TO MAIN MEMORY UNRELOCATED.
                                ;*
                                ;;*****************************************************************
     012470                     TST6:
     012470  000004                     SCOPE
     012472  004737  017662             JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     012476  012604  012556  000006     .WORD   TST7,20$,6      ;DATA USED BY PRETST
1860 012504  012737  000020  001312     MOV     #TIMOUT,CPUEXP  ;MIGHT TIME OUT OVER UNIBUS
1861 012512  052737  000001  177572     BIS     #BIT0,MMR0      ;TURN MEMORY MANAGEMENT BACK ON
1862 012520  013700  172354             MOV     KIPAR6,R0       ;PUT UNIBUS ADDRESS OF MAP REGISTER IN R0
1863 012524  072027  177773             ASH     #-5,R0          ;RIGHT SHIFT R0 5 PLACES
1864 012530  042700  17740C             BIC     #177400,R0      ;CLEAR UPPER BYTE
1865 012534  042737  17700C  172352     BIC     #177000,KIPAR5  ;MAKE KIPAR5 ACCESS THE FAST BUS
1866 012542  012720  021042             MOV     #021042,(R0)+   ;SET BOTTOM BIT IN EACH ALU
1867 012546  012710  000042             MOV     #42,(R0)        ;SET BOTTOM BIT IN EACH ALU
1868 012552  005037  120000             CLR     120000          ;CLEAR TEST LOCATION VIA FAST BUS
1869 012556  012737  043207  140000 20$: MOV     #43207,140000   ;LOAD TEST LOCATION VIA UNIBUS
1870                                                            ;THIS LOAD SHOULD NOT BE RELOCATED
1871                                                            ;BY THE UNIBUS MAP, SINCE BIT05 OF MMR3 IS CLEAR.
1872 012564  013703  120000             MOV     120000,R3       ;READ TEST LOCATION VIA FAST BUS
1873 012570  022703  043207             CMP     #43207,R3       ;SEE IF DATA MATCHES
1874 012574  001401                     BEQ     1$              ;BRANCH IF DATA GOOD
1875 012576  104012                     ERROR   +12             ;MAP RELOCATED WHEN NOT ENABLED
1876 012600  005037  001312       1$:   CLR     CPUEXP          ;ZERO EXPECTED CPU TRAP CONDITION
```

```
                        .SBTTL  TEST 7 - SIZE JUMPER LOCATION TEST
                        ;;*************************************************************
                        ;*TEST 7        SIZE JUMPER LOCATION TEST
                        ;*
                        ;*      THIS TEST DETERMINES THE SETTING OF THE JUMPERS ON THE UNIBUS
                        ;*      MAP WHICH ALLOW THE MAP TO RESPOND TO THOSE ADDRESSES BETWEEN
                        ;*      THE JUMPER RANGE.  THE DEFAULT SETTING ALLOWS THE MAP TO RESPOND
                        ;*      TO ADDRESSES 000000 - 757776 ON THE UNIBUS.  IF THE JUMPERS ARE
                        ;*      NOT SET IN THEIR DEFAULT POSITION AN INFORMATIONAL MESSAGE IS GIVEN.
                        ;*                      >>>>>>>>>NOTE<<<<<<<<<
                        ;*      THIS IS THE FIRST TEST IN WHICH THE UNIBUS MAP IS TURNED ON.
                        ;*
                        ;;*********  *************************************************
     012604             TST7:
     012604  000004             SCOPE
     012606  004737  017662     JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     012612  013440  012620  000007     .WORD   TST10,20$,7     ;DATA USED BY PRETST
1889 012620             20$:
1890 012620  012737  000020  001312 SIZEJ:  MOV     #TIMOUT,CPUEXP  ;EXPECTING CPU TIME OUT ON UNIBUS
1891 012626  012705  020100         MOV     #$DDW0,R5       ;PUT ADDRESS OF $DDW0 IN R5
1892 012632  012700  170200         MOV     #MAPL0,R0       ;LOAD ADDRESS OF FIRST MAP REGISTER IN R0
1893 012636  012701  000040         MOV     #40,R1          ;DO ALL 40 REGISTERS
1894 012642  012702  006162         MOV     #SPECST,R2      ;SET R2 TO BEGINING OF SPECIAL STACK
1895 012646  012720  020000   1$:   MOV     #20000,(R0)+    ;LOAD 4K INTO LOWER 16 BITS AND
1896 012652  005020         CLR     (R0)+           ;CLEAR THE UPPER 6 BITS
1897 012654  005022         CLR     (R2)+           ;CLEAR THE SPECIAL STACK LOCATION
1898 012656  077105         SOB     R1,1$           ;BRANCH IF THERE ARE MORE TO LOAD
1899 012660  012703  006162         MOV     #SPECST,R3      ;RESET SPECIAL STACK POINTER
1900 012664  052737  000040  172516 BIS     #BIT5,MMR3      ;TURN ON MAP RELOCATION
1901 012672  052737  000001  177572 BIS     #BIT0,MMR0      ;MAKE SURE MEMORY MANAGEMENT IS ON
1902 012700  012700  117776         MOV     #117776,R0      ;THIS WILL BE USED TO SELECT PAR 4; ADDRESS 17776
1903 012704  012737  167600  172350 MOV     #167600,KIPAR4  ;LOAD MAP REGISTER 0 -200 IN KIPAR4
1904 012712  012702  125252         MOV     #125252,R2      ;CONSTANT TO LOAD INTO LOCATION 17776
1905 012716  012737  177777  001172 MOV     #-1,$TMP0       ;MOVE -1 TO MAP REGISTER POINTER
1906 012724  012737  177777  005120 MOV     #-1,FTTHRU      ;INITIALIZE ONE TIME ENTRANCE FLAG IN SUBROUTINE
1907 012732  004737  005046   2$:   JSR     PC,TSTLOC       ;GO TO SUBROUTINE TO SEE IF LOCATION RESPONDS
1908 012736  000412         BR      3$              ;RETURN IS HERE IF LOWEST FOUND
1909 012740  004737  021616         JSR     PC,DSABLD       ;GO SEE IF REGISTER SHOULD BE DISABLED
1910 012744  104214         ERROR   +214            ;MAP REGISTER DISABLED WHEN SHOULD BE ENABLED
1911 012746  022737  177400  172350 CMP     #177400,KIPAR4  ;SEE IF WE ARE POINTING JUST BELOW THE I/O PAGE
1912 012754  001366         BNE     2$              ;GO TEST NEXT MAP REGISTER IF NOT
1913 012756  104013         ERROR   +13             ;FATAL ERROR, RESTARTING PROGRAM
1914 012760  000137  010000         JMP     START           ;JUMP TO RESTART PROGRAM
1915 012764  004737  021654   3$:   JSR     PC,ENABLD       ;GO CHECK TO SEE IF REGISTER SHOULD BE ENABLED
1916 012770  104213         ERROR   +213            ;MAP REGISTER ENABLED WHEN SHOULD BE DISABLED
1917 012772  013737  001172  001264 MOV     $TMP0,MMRLOW    ;MOVE REGISTER NUMBER FOUND USEABLE TO MMRLOW
1918 013000  013737  172350  001254 MOV     KIPAR4,LOWEST   ;MOVE LOWEST USEABLE REGISTER TO LOWEST
1919 013006  022737  170000  001254 CMP     #170000,LOWEST  ;SEE IF LOWEST REGISTER FOUND WAS THE LOWEST
1920 013014  001436         BEQ     5$              ;BRANCH AROUND SETUP IF IT WAS
1921 013016  005037  001270         CLR     UBRLOW          ;MAP REGISTER 0 IS LOWEST FOR UNIBUS MEMORY
1922 013022  012737  170000  001260 MOV     #170000,UBMLOW  ;MOVE PAR VALUE OF UB MEMORY TO UBMLOW
```

```
1923 013030  013737  001172  001272          MOV     $TMP0,UBRHI     ;MOVE REGISTER NUMBER FOUND USEABLE TO UBRHI
1924 013036  005337  001272                  DEC     UBRHI           ;POINT IT AT HIGHEST UB MEMORY MAP REGISTER        SEQ 0095
1925 013042  013737  172350  001262          MOV     KIPAR4,UBMHI    ;MOVE KIPAR4 TO UNIBUS MAP HIGHEST AND
1926 013050  162737  000200  001262          SUB     #200,UBMHI      ;SUBTRACT 200 FROM IT TO POINT TO LAST USEABLE UBMEM PAGE
1927 013056  012737  177400  001256          MOV     #177400,HIGEST  ;MOVE HIGHEST REGISTER TO HIGEST AND
1928 013064  012737  000031  001266          MOV     #31,MMRHI       ;POINT TO LAST USEABLE MAP REGISTER
```

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

```
1980 013356  001402                        BEQ     15$             ;BRANCH AROUND SETUP IF NONE                    SEQ 0097
1981 013360  011346                        MOV     (R3),-(SP)      ;PUSH THIS NUMBER ON THE STACK
1982 013362  000771                        BR      14$             ;BRANCH BACK TO PRINT IT
1983 013364  104400  001221          15$:  TYPE    ,$CRLF          ;TYPE ONE MORE <CRLF>
1984                                ;:
1985                                ;*      SETUP POINTERS TO THE LOWEST AND THE HIGHEST USABLE
```

```
1986                                    ;*      MAPPING REGISTERS TO CONTINUE WITH TEST.
1987                                    ;:
1988 013370  005037  001312            16$:    CLR     CPUFXP          ;NO CPU TRAPS EXPECTED IN THIS SECTION
1989 013374  010546                            MOV     R5,-(SP)        ;SAVE R5
1990 013376  013705  001254                    MOV     LOWEST,R5       ;MOVE PAR DATA TO R5 FOR CONVERSION
1991 013402  042705  170000                    BIC     #170000,R5      ;CLEAR BITS 15 TO 12
1992 013406  072527  177773                    ASH     #-5,R5          ;SHIFT INDICATOR BITS TO THE RIGHT 5 PLACES
1993 013412  052705  170200                    BIS     #170200,R5      ;FORM ADDRESS
1994 013416  010537  001276                    MOV     R5,LREGL        ;SAVE RESULTS
1995 013422  012605                            MOV     (SP)+,R5        ;RESTORE R5
1996 013424  013737  001276  001300            MOV     LREGL,LREGU     ;MOVE ADDRESS TO LREGU
1997 013432  062737  000002  001300            ADD     #2,LREGU        ;POINT TO UPPER 6 BITS OF MAP REG
```

```
   2016                                 .SBTTL  TEST 10 - ENSURE THAT THERE IS NO DUAL MAPPING
                                        ;;*****************************************************************
                                        ;*TEST 10       ENSURE THAT THERE IS NO DUAL MAPPING
                                        ;*
                                        ;*      THIS TEST VERIFIES THAT THERE IS NO DUAL MAPPING. IT CLEARS
                                        ;*      ALL THE MAP REGISTERS EXCEPT THE ONE UNDER TEST, AND LOADS
                                        ;*      THAT ONE WITH 00040000.  IF MAP RELOCATION IS ENABLED (AND
                                        ;*      IN THIS TEST IT IS), SUBROUTINE CLRMAP CLEARS ALL BUT THE
                                        ;*      LOWER 16 BITS OF MAPL1, AND LOADS 20000 THERE.  THIS IS SO
                                        ;*      THAT APT, IF CONTROLLING THIS DIAGNOSTIC, CAN STILL EXAMINE
                                        ;*      THE PROPER LOCATIONS.  THE TEST THEN USES A VIRTUAL ADDRESS
                                        ;*      TO SELECT THAT MAP REGISTER AND ADD 17776, SO THAT IT SHOULD
                                        ;*      REFERENCE ADDRESS 00057776 (00037776 IF MAPL1 CONTAINS 20000
                                        ;*      AS PER CONDITIONS DESCRIBED ABOVE).  A REFERENCE IS MADE THROUGH
                                        ;*      EACH OF THE REGISTERS AND ANY THAT FETCH THE CORRECT DATA ARE
                                        ;*      CHECKED TO SEE THAT IT WAS THE MAP REGISTER UNDER TEST.  IF
                                        ;*      NOT, BOTH THE MAP REGISTER UNDER TEST AND THE DUALED REGISTER
                                        ;*      ARE REPORTED.
                                        ;*
                                        ;;*****************************************************************
   013440                               TST10:
   013440  000004                               SCOPE
   013442  004737  017662                       JSR     PC,PRETST       ;GO SET UP PRETEST DATA
   013446  014130  013606  000010               .WORD   TST11,20$,10    ;DATA USED BY PRETST
2017 013454  005037  172516                     CLR     MMR3            ;CLEAR MMR3
2018 013460  052737  000060  172516           BIS     #60,MMR3        ;ENABLE 22-BIT ADDRESSING AND MAP RELOCATION
2019 013466  004737  004352                     JSR     PC,CLRMAP       ;CLEAR ALL MAP REGISTERS
2020 013472  042737  000001  177572           BIC     #1,MMR0         ;TURN OFF MEMORY MANAGEMENT
2021 013500  005037  001172                     CLR     $TMP0           ;$TMP0 IS USED AS A FLAG IN THIS TEST
2022 013504  012703  117776                     MOV     #117776,R3      ;SELECT P.A.R. 4 OFFSET OF 17776
2023 013510  013702  001276                     MOV     LREGL,R2        ;PUT ADDRESS OF LOWEST USABLE MAP REGISTER IN R2
2024 013514  013700  001254                     MOV     LOWEST,R0       ;LOAD PAR POINTING TO MAP REGISTER UNDER TEST IN R0
2025 013520  052737  000001  177572           BIS     #1,MMR0         ;MAKE SURE MEMORY MANAGEMENT IS ON
2026 013526  005037  001302                     CLR     ERRCNT          ;CLEAR THE ERROR COUNT FOR ERROR 202 BELOW
2027 013532  013737  001254  172350  1$:      MOV     LOWEST,KIPAR4   ;PAR OF LOWEST USEABLE MAP REGISTER IS LOADED IN KIPAR4
2028 013540  022702  170204                     CMP     #MAPL01,R2      ;SEE IF WE ARE POINTING AT MAPL1
2029 013544  001003                             BNE     13$             ;BRANCH IF NOT
2030 013546  022712  020000                     CMP     #20000,(R2)     ;SEE IF IT CONTAINS 20000 (FOR APT USE)
2031 013552  001405                             BEQ     14$             ;BRANCH IF SO
2032 013554  012712  040000          13$:      MOV     #40000,(R2)     ;LOAD MAP REGISTER UNDER TEST WITH 8K BASE
2033 013560  010237  057776                     MOV     R2,57776        ;LOAD TEST LOCATION WITH THE ADDRESS
2034                                                                    ;OF THE MAP REGISTER UNDER TEST
2035 013564  000402                             BR      15$             ;BRANCH OVER LOCATION SETUP
2036 013566  010237  037776          14$.      MOV     R2,37776        ;LOAD TEST LOCATION WITH THE ADDRESS
2037                                                                    ;OF THE MAP REGISTER UNDER TEST
2038 013572  162737  000200  172350  15$:      SUB     #200,KIPAR4     ;PREPARE KIPAR4 FOR FIRST 'ADD 200'
2039 013600  062737  000200  172350  21$:      ADD     #200,KIPAR4     ;TRY NEXT MAP REGISTER
2040 013606  005037  001314          20$:      CLR     PCPUER          ;CLEAR THE ERROR RECEIVER
2041 013612  011304                             MOV     (R3),R4         ;READ THROUGH THE MAP REGISTER
2042 013614  005737  001314                     TST     PCPUER          ;SEE IF THERE WAS AN ERROR
2043 013620  001023                             BNE     4$              ;BRANCH AROUND DATA TEST IF SO
```

```
2044 013622  020402                CMP     R4,R2           ;SEE IF CORRECT DATA WAS FETCHED                    SEQ 0100
2045 013624  001021                BNE     4$              ;BRANCH IF NO MATCH
2046 013626  020037  172350        CMP     R0,KIPAR4       ;SEE IF MAP REGISTERS ARE THE SAME
2047 013632  001414                BEQ     3$              ;BRANCH IF CORRECT MAP REGISTER WAS USED
2048 013634  011337  001200        MOV     (R3),$TMP3      ;SAVE CONTENTS FOR ERROR PRINTING
2049 013640  013746  172356        MOV     KIPAR7,-(SP)    ;PUT PAR ON STACK FOR SUBROUTINE ADREXT USE
```

```
2050 013644  010246                        MOV     R2,-(SP)        ;PUT ADDRESS OF REGISTER ON STACK FOR ADREXT USE
2051 013646  004737  005250                JSR     PC,ADREXT       ;GO SET DATA IN THE 4 WORDS OF ADDROR AND ADRAND
2052 013652  010246                        MOV     R2,-(SP)        ;PUT ADDRESS OF REGISTER ON STACK FOR DATEXT USE
2053 013654  004737  005200                JSR     PC,DATEXT       ;GO SET DATA IN DATAOR AND DATAND
2054 013660  104210                        ERROR   +210            ;DUAL MAPPING ERROR IN THE UNIBUS MAP
2055 013662  000402                        BR      4$              ;BRANCH AROUND ADDRESS MATCH SETTING
2056 013664  005237  001172          3$:   INC     $TMP0           ;SET FLAG WHEN ADDRESSES MATCH
2057 013670  022737  177400  172350  4$:   CMP     #177400,KIPAR4  ;SEE IF LAST REGISTER HAS BEEN TRIED
2058 013676  001423                        BEQ     6$              ;BRANCH TO CONTINUE IF SO
2059 013700  023737  001256  172350        CMP     HIGEST,KIPAR4   ;SEE IF ALL HAVE BEEN TRIED
2060 013706  001334                        BNE     21$             ;BRANCH IF STILL MORE TO TRY
2061 013710  062737  000200  172350  5$:   ADD     #200,KIPAR4     ;MAP TO NEXT MAP REGISTER
2062 013716  022737  177400  172350        CMP     #177400,KIPAR4  ;SEE IF WE ARE AT THE TOP
2063 013724  001410                        BEQ     6$              ;BRANCH TO CONTINUE IF SO
2064 013726  023737  001274  172350        CMP     BUPWIN,KIPAR4   ;SEE IF WE ARE POINTING TO THE UPPER WINDOW START
2065 013734  001365                        BNE     5$              ;BRANCH BACK FOR ANOTHER INCREMENTING SET IF NOT
2066 013736  162737  000200  172350        SUB     #200,KIPAR4     ;PREPARE FOR THE 'ADD 200'
2067 013744  000720                        BR      20$             ;GO BACK FOR A NEW TRY
2068 013746  005737  001172          6$:   TST     $TMP0           ;SEE THAT THERE WAS A SUCCESSFUL MATCH
2069 013752  001017                        BNE     7$              ;BRANCH IF THERE WAS
2070 013754  010005                        MOV     R0,R5           ;MOVE R0 TO R5 FOR PREPARATION AND SHIFTING
2071 013756  042705  170000                BIC     #170000,R5      ;CLEAR THE UPPER 4 BITS
2072 013762  072527  177773                ASH     #-5,R5          ;SHIFT RIGHT 5 PLACES
2073 013766  062705  170200                ADD     #170200,R5      ;FORM LOWER 16 BITS OF PHYSICAL ADDRESS OF MAP REGISTER
2074 013772  010537  006144                MOV     R5,EADRES       ;MOVE THE OBTAINED ADDRESS TO THE ERROR LOCATION
2075 013776  013746  172356                MOV     KIPAR7,-(SP)    ;PUT PAR ON STACK FOR SUBROUTINE ADREXT USE
2076 014002  010246                        MOV     R2,-(SP)        ;PUT ADDRESS OF REGISTER ON STACK FOR ADREXT USE
2077 014004  004737  005250                JSR     PC,ADREXT       ;GO SET DATA IN THE 4 WORDS OF ADDROR AND ADRAND
2078 014010  104210                        ERROR   +210            ;DUAL MAPPING ERROR IN THE UNIBUS MAP
2079 014012  005037  001172          7$:   CLR     $TMP0           ;CLEAR FLAG FOR NEXT REGISTER
2080 014016  022702  170204                CMP     #MAPL01,R2      ;SEE IF R2 IS POINTING TO MAPL1
2081 014022  001006                        BNE     75$             ;BRANCH IF NOT
2082 014024  022712  040000                CMP     #40000,(R2)     ;DOES MAPL1 CONTAIN 40000?
2083 014030  001403                        BEQ     75$             ;BRANCH IF IT DOES
2084 014032  005037  037776                CLR     37776           ;CLEAR LOCATION 37776 ONLY - MAPL1 IS TO BE LEFT ALONE
2085 014036  000401                        BR      76$             ;SKIP OVER REGISTER CLEAR STEP
2086 014040  005012                75$:   CLR     (R2)            ;CLEAR MAP REGISTER JUST TESTED
2087 014042  062700  000200          76$:  ADD     #200,R0         ;POINT TO NEXT MAP REGISTER UNDER TEST
2088 014046  062702  000004                ADD     #4,R2           ;POINT TO NEXT MAP REGISTER TO LOAD
2089 014052  022700  177400                CMP     #177400,R0      ;SEE IF LAST REGISTER HAS BEEN TRIED
2090 014056  001416                        BEQ     9$              ;BRANCH TO NEXT SECTION IF SO
2091 014060  023700  001256                CMP     HIGEST,R0       ;SEE IF ALL MAP REGS HAVE BEEN TESTED
2092 014064  001222                        BNE     1$              ;BRANCH IF STILL MORE TO TEST
2093 014066  062700  000200          8$:   ADD     #200,R0         ;POINT TO NEXT MAP REGISTER UNDER TEST
2094 014072  062702  000004                ADD     #4,R2           ;POINT TO NEXT MAP REGISTER TO LOAD
2095 014076  022700  177400                CMP     #177400,R0      ;SEE IF LAST REGISTER HAS BEEN TRIED
2096 014102  001404                        BEQ     9$              ;BRANCH TO NEXT SECTION IF SO
2097 014104  020037  001274                CMP     R0,BUPWIN       ;SEE IF WE ARE POINTING TO UPPER WINDOW START
2098 014110  001366                        BNE     8$              ;BRANCH FOR ANOTHER INCREMENT SET IF NOT
2099 014112  000607                        BR      1$              ;BRANCH BACK FOR ANOTHER RUN
2100 014114  005737  001302          9$:   TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
2101 014120  001403                        BEQ     TST11           ;;BRANCH TO NEXT TEST IF NO ERRORS
2102 014122  005337  001110                DEC     $ERTTL          ;DON'T COUNT ERROR +16 AS ANOTHER ERROR
2103 014126  104016                        ERROR   +16             ;SUMMARY OF DUAL MAPPING ERRORS
```

```
                                        .SBTTL  TEST 11 - LOAD LOC'S 40000-77776 WITH THEIR ADRES'S
                                        ;;*****************************************************************
                                        ;*TEST 11       LOAD LOC'S 40000-77776 WITH THEIR ADRES'S
                                        ;*
                                        ;*      THIS TEST IS USED TO LOAD MAIN MEMORY FROM ADDRESS 00040000 TO
                                        ;*      ADDRESS 000077776 WITH ITS OWN ADDRESS.  IT THEN CHECKS THAT
                                        ;*      MEMORY OVER THE UNIBUS AND LOGS ANG REPORTS ANY ERRORS THAT
                                        ;*      IT FINDS.
                                        ;*
                                        ;;*****************************************************************
     014130                             TST11:
     014130  000004                             SCOPE
     014132  004737  017662                     JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     014136  014362  014252  000011             .WORD   TST12,20$,11    ;DATA USED BY PRETST
2113 014144  042737  000040  172516             BIC     #BIT5,MMR3      ;TURN OFF MAP RELOCATION
2114 014152  012737  000400  172350             MOV     #400,KIPAR4     ;MAP PAGE 4 TO 8K
2115 014160  012700  040000                     MOV     #40000,R0       ;STARTING ADDRESS FOR DATA PATTERN
2116 014164  012701  100000             1$:     MOV     #100000,R1      ;VIRTUAL ADDRESS
2117 014170  012702  010000                     MOV     #4096.,R2       ;LOAD 4096 LOCATIONS AT A TIME
2118 014174  010021                     2$:     MOV     R0,(R1)+        ;LOAD PHY. ADDR. INTO EACH MEMORY LOC.
2119 014176  062700  000002                     ADD     #2,R0           ;POINT TO NEXT PHYSICAL ADDRESS
2120 014202  077204                             SOB     R2,2$           ;BRANCH IF 4K OF MEMORY NOT LOADED
2121 014204  062737  000200  172350             ADD     #200,KIPAR4     ;POINT TO NEXT 4K BANK OF MEMORY
2122 014212  022737  001000  172350             CMP     #1000,KIPAR4    ;SEE IF 16K IS LOADED
2123 014220  101361                             BHI     1$              ;BRANCH IF MORE MEMORY TO LOAD
2124                                    ;*
2125                                    ;*      MEMORY FROM 8K - 16K IS NOW LOADED WITH ITS OWN ADDRESS
2126                                    ;*
2127 014222  022737  171000  172354             CMP     #171000,KIPAR6  ;DID I USE ANY MAP REGISTER
2128                                                                     ;BELOW REGISTER 6 (UB. ADDR 100000)
2129 014230  101454                             BLOS    TST12           ;;BRANCH TO NEXT TEST IF NOT
2130 014232  013700  172354                     MOV     KIPAR6,R0       ;LOAD PAR6 INTO R0 TO GET
2131                                                                     ;THE STARTING DATA PATTERN
2132 014236  072027  000006                     ASH     #6,R0           ;R0 NOW HOLDS THE STARTING DATA PATTERN
2133 014242  012701  140000             3$:     MOV     #140000,R1      ;STARTING VIRTUAL ADDRESS
2134 014246  012702  010000                     MOV     #4096.,R2       ;PREPARE TO READ 4K AT A TIME
2135 014252  011103                     20$:    MOV     (R1),R3         ;READ MAIN MEMORY THROUGH UNIBUS
2136 014254  020003                             CMP     R0,R3           ;SEE IF THE ADDRESSES MATCH
2137 014256  001015                             BNE     6$              ;BRANCH IF ERROR
2138 014260  062701  000002             5$:     ADD     #2,R1           ;CHANGE VIRTUAL ADDRESS
2139 014264  062700  000002                     ADD     #2,R0           ;CHANGE PHYSICAL ADDRESS
2140 014270  077210                             SOB     R2,20$          ;BRANCH IF 4K OF MEMORY NOT READ
2141 014272  062737  000200  172354             ADD     #200,KIPAR6     ;POINT TO NEXT BANK OF 4K THROUGH UNIBUS
2142 014300  022737  171000  172354             CMP     #171000,KIPAR6  ;SEE IF THIS POINTS TO 16K PLUS 2
2143 014306  101355                             BHI     3$              ;BRANCH IF 16K OF MEMORY NOT CHECKED
2144 014310  000416                             BR      10$             ;TEST FINISHED, BRANCH TO EXIT
2145 014312  013746  172344             6$:     MOV     KIPAR2,-(SP)    ;PUT PAR ON STACK FOR ADREXT SUBROUTINE USE
2146 014316  010046                             MOV     R0,-(SP)        ;PUT VIRTUAL ADDRESS ON STACK FOR ADREXT SUBROUTINE USE
2147 014320  004737  005250                     JSR     PC,ADREXT       ;GO SET DATA IN THE 4 WORDS OF ADDROR AND ADRAND
2148 014324  010346                             MOV     R3,-(SP)        ;PUT DATA ON STACK FOR DATEXT SUBROUTINE
2149 014326  004737  005200                     JSR     PC,DATEXT       ;SUBROUTINE TO LOAD DATAOR AND DATAND
```

```
2150 014332  010357  006150              MOV     R3,EADRS2       ;MOVE DATA IN R3 TO EADRS2 FOR ERROR CALL
2151 014336  005037  006152              CLR     EADRS2+2        ;CLEAR UPPER 6 BITS TO PRINT                    SEQ 0103
2152 014342  104205                      ERROR   +205            ;DIDN'T READ ADDRESSES CORRECTLY FROM UNIBUS
2153 014344  000745                      BR      5$              ;CONTINUE TESTING
2154 014346  005737  001302      10$:    TST     ERRCNT          ;WERE THERE ANY ERRORS ON THIS TEST?
2155 014352  001403                      BEQ     TST12           ;;BRANCH IF NO ERRORS ON THIS TEST
```

```
 2156 014354  005337  001110                DEC     $ERTTL          ;DON'T COUNT ERROR +14 AS ANOTHER ERROR
 2157 014360  104014                        ERROR   +14             ;SUMMARY OF UNIBUS ADDRESS FAILURES
```

```
                                .SBTTL  TEST 12 - MAIN MEMORY TIMEOUT THROUGH MAP
                                ;;*****************************************************************
                                ;*TEST 12       MAIN MEMORY TIMEOUT THROUGH MAP
                                ;*
                                ;*      THIS TEST GENERATES A TIME OUT THROUGH THE UNIBUS MAP BY TRYING
                                ;*      TO REFERENCE ADDRESS 17000000 IN MAIN MEMORY.  IT USES THE LOWEST
                                ;*      USABLE MAP REGISTER, WHICH IN THE DEFAULT CASE IS MAP REGISTER
                                ;*      ZERO.
                                ;*
                                ;;*****************************************************************
     014362                     TST12:
     014362  000004                     SCOPE
     014364  004737  017662             JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     014370  014544  014410  000012     .WORD   TST13,20$,12    ;DATA USED BY PRETST
2167 014376  052737  000040  172516     BIS     #BIT5,MMR3      ;TURN MAP RELOCATION BACK ON
2168 014404  005037  001202             CLR     $TMP4           ;CLEAR $TMP4 FOR ERROR LOOP INDICATOR
2169 014410  004737  014420     20$:    JSR     PC,MMTOTM       ;GO DO TEST BELOW
2170 014414  104015                     ERROR   +15             ;RETURN HERE IF ERROR - UNIBUS DID NOT TIME OUT
2171 014416  000452                     BR      TST13           ;;SKIP OVER THIS PORTION OF THE TEST
2172
2173 014420  005737  001202     MMTOTM: TST     $TMP4           ;SEE IF THIS ENTRY IS FROM ERROR LOOPING
2174 014424  001402                     BEQ     1$              ;BRANCH IF NOT
2175 014426  062706  000002             ADD     #2,SP           ;CLEAN EXTRA RETURN OFF STACK
2176 014432  005037  001302     1$:     CLR     ERRCNT          ;CLEAR ERRCNT FOR THIS TEST
2177 014436  012737  000020  001312     MOV     #TIMOUT,CPUEXP  ;EXPECTING TIMEOUT IN THIS TEST
2178 014444  013737  001254  172350     MOV     LOWEST,KIPAR4   ;LOAD PAR 4 WITH LOWEST USABLE MAP REG
2179 014452  012777  000074  164620     MOV     #74,@LREGU      ;LOAD UPPER 6 BITS OF LOWEST MAP REG
2180 014460  005077  164612             CLR     @LREGL          ;LOAD LOWER 16 BITS OF LOWEST MAP REG
2181 014464  005037  001314     20$:    CLR     PCPUER          ;CPU ERROR REGISTER LOCATION
2182 014470  013703  100000             MOV     100000,R3       ;TRY TO READ THROUGH PAGE 4 THIS REFERENCE WILL GO OUT
2183                            ;ON THE UNIBUS TO SELECT THE LOWEST USABLE MAP REGISTER (DEFAULT MAP REG. 0).  PHYSICAL
2184                            ;ADDRESS 17700000 IS THEN GENERATED, WHICH SHOULD TIME OUT SINCE IT IS THE FIRST
2185                            ;NON-EXISTENT LOCATION.
2186 014474  022737  000020  001314     CMP     #TIMOUT,PCPUER  ;THE UNIBUS SHOULD HAVE TIMED OUT
2187 014502  001413                     BEQ     3$              ;BRANCH IF CONDITION WAS CORRECT
2188 014504  105737  007774             TSTB    CPUTYP          ;IS THIS AN 11/44?
2189 014510  001403                     BEQ     2$              ;BRANCH IF IT IS
2190 014512  005737  001302             TST     ERRCNT          ;SEE IF A TIMEOUT WAS REPORTED
2191 014516  001005                     BNE     3$              ;BRANCH IF SO
2192 014520  011646             2$:     MOV     (SP),-(SP)      ;PUSH ANOTHER RETURN ONTO THE STACK FOR POSSIBLE ERROR LOOP
2193 014522  012737  000001  001202     MOV     #1,$TMP4        ;SET ERROR LOOP FLAG
2194 014530  000207                     RTS     PC              ;EXIT
2195 014532  062716  000002     3$:     ADD     #2,(SP)         ;CORRECT RETURN PC OVER ERROR CALL
2196 014536  005037  001312             CLR     CPUEXP          ;NO CPU TRAPS EXPECTED FOR AWHILE
2197 014542  000207                     RTS     PC              ;EXIT
```

```
                                        .SBTTL  TEST 13 - RELOC USING LOWEST USABLE MAP REG THRU UNIBUS
                                        ;;*****************************************************************
                                        ;*TEST 13       RELOC USING LOWEST USABLE MAP REG THRU UNIBUS
                                        ;*
                                        ;*      THIS TEST CHECKS OUT THE FULL ADDITION PROPERTIES OF THE UNIBUS
                                        ;*      MAP A.L.U..  IN THE DEFAULT CASE IT USES MAP REGISTER ZERO BUT
                                        ;*      IF THE MAP JUMPERS HAVE BEEN ALTERED TO DE-SELECT SOME MAP REGISTERS
                                        ;*      THIS TEST WILL USE THE LOWEST USABLE MAP REGISTER.
                                        ;*      IF AN ERROR OCCURS THE TEST WILL REPORT THE PHYSICAL ADDRESS
                                        ;*      THAT WAS DESIRED, AND THE DATA AT THE ADDRESS THAT WAS REFERENCED.
                                        ;*
                                        ;;*****************************************************************
     014544                             TST13:
     014544  000004                             SCOPE
     014546  004737  017662                     JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     014552  014574  014560  000013             .WORD   TST14,20$,13    ;DATA USED BY PRETST
2209 014560  005037  001202             20$:    CLR     $TMP4           ;CLEAR $TMP4 - FLAG USED FOR ERROR RETURN
2210 014564  004737  005404             1$:     JSR     PC,MAPADD       ;GO DO THE TEST
2211 014570  104211                             ERROR   +211            ;RETURN IS HERE FOR ERROR
2212 014572  000774                             BR      1$              ;GO BACK TO THE SUBROUTINE
```

```
2221                                    .SBTTL  TEST 14 - CARRY PROP OF MAP'S RELOC ADDER THRU UNIBUS
                                        ;;*****************************************************************
                                        ;*TEST 14       CARRY PROP OF MAP'S RELOC ADDER THRU UNIBUS
                                        ;*
                                        ;*      EVERY ADDRESS OF THE FORM XXXX0000 IS GENERATED HERE STARTING
                                        ;*      WITH 00030000 UP TO 17000000.  THAT IS, THE FIRST OF EVERY 2K
                                        ;*      WORDS IS ADDRESSED, TO INSURE THAT THE ADDER IN THE MAP IS
                                        ;*      WORKING PROPERLY .
                                        ;*
                                        ;;*****************************************************************
     014574  000004                     TST14:  SCOPE
2222 014576  004737  017662                     JSR     PC,PRETST       ;GO SET UP PRETEST DATA
2223                                    ;;*****************************************************************
2224                                    ;**IMPORTANT**: IF THE POSITION OF THIS TEST IS CHANGED, CHANGE THE THIRD
2225                                    ;               WORD BELOW TO THE NEW TEST NUMBER THIS TEST WILL OCCUPY.
2226                                    ;;*****************************************************************
2227 014602  014624  014610  000014             .WORD   2$,20$,14       ;DATA USED BY PRETST
2228         000040                     NEXMEM=BIT5             ;BIT05 IS NON-EXISTENT MEMORY BIT IN THE CPU ERROR REGISTER
2229 014610  005037  001202             20$:    CLR     $TMP4           ;CLEAR $TMP4 - USED AS AN ERROR RETURN FLAG
2230 014614  004737  005632             1$:     JSR     PC,TCPMRA       ;GO DO THE TEST
2231 014620  104211                             ERROR   +211            ;RETURN IS HERE IF AN ERROR
2232 014622  000774                             BR      1$              ;RETURN TO THE TEST
2233 014624                             2$:
```

```
                                .SBTTL  UNIBUS MAP SETTUP
                        ;;******************************************************************
                        ;       THE NEXT 3 TESTS ARE RUN THROUGH THE UNIBUS MAP
                        ;;******************************************************************

014624  000004                  SCOPE                   ;LOOP ON PREVIOUS TEST
014626  104420                  TBITR                   ;RESTORE T-BIT IF IT WAS ON

                        ;;******************************************************************
                        ;*
                        ;*      THIS CODE SETS UP THE TWO MAP REGISTERS ABOVE THE LOWEST
                        ;*      USABLE ONE TO POINT TO PHYSICAL MEMORY FROM 0 - 8K.  IN THE
                        ;*      DEFAULT CASE, IN MANUFACTURING AND IF THERE IS NO UNIBUS MEMORY,
                        ;*      THIS WILL BE MAP REGISTERS 1 AND 2.  MAP REGISTER 1 WILL POINT
                        ;*      TO PHYSICAL 4K - 8K SO "ACT-11" WILL WORK PROPERLY AND MAP
                        ;*      REGISTER 2 WILL POINT TO PHYSICAL 0 - 4K.  THIS MEANS THAT
                        ;*      KIPAR0 SHOULD GET 170400 SO IT PUTS ADDRESSES 040000 TO 057776
                        ;*      ON THE UNIBUS AND KIPAR1 SHOULD GET 170200 SO IT PUTS ADDRESSES
                        ;*      020000 TO 037776 ON THE UNIBUS.
                        ;*
                        ;;******************************************************************

014630  013701  001300          MOV     LREGU,R1        ;PUT POINTER TO LOWEST MAP REGISTER IN R1
014634  005721                  TST     (R1)+           ;POINT TO LOWER 16 BITS OF LOWEST + 1
014636  012721  020000          MOV     #20000,(R1)+    ;LOAD LOWER 16 BITS OF (LOWEST + 1), POINTING TO 4-8K
014642  005021                  CLR     (R1)+           ;CLEAR UPPER 6 BITS OF (LOWEST + 1)
014644  005021                  CLR     (R1)+           ;CLEAR LOWER 16 BITS OF (LOWEST + 2)
014646  005021                  CLR     (R1)+           ;CLEAR UPPER 6 BITS OF (LOWEST + 2)
014650  013701  001254          MOV     LOWEST,R1       ;LOAD R1
014654  062701  000200          ADD     #200,R1         ;POINT R1 TO (LOWEST + 1)
014660  010137  172342          MOV     R1,KIPAR1       ;LOAD PAR1
014664  062701  000200          ADD     #200,R1         ;POINT R1 TO (LOWEST + 2)
014670  010137  172340          MOV     R1,KIPAR0       ;LOAD PAR0
```

```
                                .SBTTL  TEST 15 - MAIN MEM. T.O. THROUGH MAP, CODE RUN OVER U.B.
                                ;;*****************************************************************
                                ;*TEST 15       MAIN MEM. T.O. THROUGH MAP, CODE RUN OVER U.B.
                                ;*
                                ;*      THIS TEST GENERATES A TIME OUT THROUGH THE UNIBUS MAP BY TRYING
                                ;*      TO REFERENCE ADDRESS 17000000 IN MAIN MEMORY.  IT USES THE LOWEST
                                ;*      USABLE MAP REGISTER, WHICH IN THE DEFAULT CASE IS MAP REGISTER
                                ;*      ZERO.
                                ;*
                                ;*      THIS TEST IS BEING RUN WITH ALL MEMORY REFERENCES GOING THROUGH
                                ;*      THE UNIBUS MAP.
                                ;*
                                ;;*****************************************************************
     014674                     TST15:
     014674  000004                     SCOPE
     014676  004737  017662             JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     014702  014722  014714  000015     .WORD   TST16,20$,15    ;DATA USED BY PRETST
2279 014710  005037  001202             CLR     $TMP4           ;CLEAR LOOP ON ERROR FLAG
2280 014714  004737  014420     20$:    JSR     PC,MMTOTM       ;GO DO TEST ON PAGE 60 OF THIS LISTING
2281 014720  104015                     ERROR   +15             ;RETURN HERE IF ERROR - UNIBUS DID NOT TIME OUT
```

```
                        .SBTTL  TEST 16 - RELOC USING LOWEST USABLE MAP REG USING MAP REG
                        ;;*****************************************************************
                        ;*TEST 16       RELOC USING LOWEST USABLE MAP REG USING MAP REG
                        ;*
                        ;*      THIS TEST CHECKS OUT THE FULL ADDITION PROPERTIES OF THE UNIBUS
                        ;*      MAP A.L.U..  IN THE DEFAULT CASE IT USES MAP REGISTER ZERO BUT
                        ;*      IF THE MAP JUMPERS HAVE BEEN ALTERED TO DE-SELECT SOME MAP REGISTERS
                        ;*      THIS TEST WILL USE THE LOWEST USABLE MAP REGISTER.
                        ;*      IF AN ERROR OCCURS THE TEST WILL REPORT THE PHYSICAL ADDRESS
                        ;*      THAT WAS DESIRED, AND THE DATA AT THE ADDRESS THAT WAS REFERENCED.
                        ;*
                        ;*      THIS TEST IS BEING RUN WITH ALL MEMORY REFERENCES GOING THROUGH
                        ;*      THE UNIBUS MAP.
                        ;*
                        ;;*****************************************************************
     014722             TST16:
     014722  000004             SCOPE
     014724  004737  017662     JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     014730  014760  014744  000016     .WORD   TST17,20$,16    ;DATA USED BY PRETST
2296 014736  012737  060000  060000     MOV     #060000,060000  ;MAKE SURE THAT ADDRESS 060000
2297                                    ;CONTAINS ITS OWN ADDRESS AS DATA
2298 014744  005037  001202     20$:    CLR     $TMP4           ;CLEAR $TMP4 - USED IN ERROR RETURN
2299 014750  004737  005404     1$:     JSR     PC,MAPADD       ;GO DO THE TEST
2300 014754  104211             ERROR   +211            ;RETURN IS HERE FOR ERROR
2301 014756  000774             BR      1$              ;RETURN TO THE SUBROUTINE
```

```
2310                                    .SBTTL  TEST 17 - CARRY PROP OF MAP'S RELOC ADDER USING MAP REG
                                        ;;*****************************************************************
                                        ;*TEST 17       CARRY PROP OF MAP'S RELOC ADDER USING MAP REG
                                        ;*
                                        ;*      EVERY ADDRESS OF THE FORM XXXX0000 IS GENERATED HERE STARTING
                                        ;*      WITH 00030000 UP TO 17000000.  THAT IS THE FIRST OF EVERY 2K
                                        ;*      WORDS IS ADDRESSED, TO INSURE THAT THE ADDER IN THE MAP IS
                                        ;*      WORKING PROPERLY .
                                        ;*
                                        ;;*****************************************************************
     014760                             TST17:
     014760  000004                             SCOPE
     014762  004737  017662                     JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     014766  015010  014774  000017             .WORD   TST20,20$,17    ;DATA USED BY PRETST
2311 014774  005037  001202             20$:    CLR     $TMP4           ;CLEAR $TMP4 - USED AS ERROR RETURN FLAG
2312 015000  004737  005632             1$:     JSR     PC,TCPMRA       ;GO DO THE TEST
2313 015004  104211                             ERROR   +211            ;RETURN IS HERE IF AN ERROR
2314 015006  000774                             BR      1$              ;RETURN TO THE TEST
```

```
                        .SBTTL  TEST 20 - VERIFY TRAP DUE TO CACHE PARITY INTERRUPT
                        ;;*****************************************************************
                        ;*TEST 20       VERIFY TRAP DUE TO CACHE PARITY INTERRUPT
                        ;;*****************************************************************
                        ;*              ***********NOTE***********
                        ;*      THE MAP WILL BE SHUT OFF FOR THE REMAINDER OF THE DIAGNOSTIC
                        ;*      BEFORE ANY TEST CODE IS EXECUTED.  IT IS NOT NEEDED.
                        ;;*****************************************************************
                        ;;*****************************************************************
                        ;*      THIS TEST IS OPTIONAL AND IS SELECTED BY SETTING MFM HARDWARE
                        ;*      SWITCH REGISTER BIT 08 TO A 1 IN THE CASE OF STANDALONE OPERATION
                        ;*      OF THE DIAGNOSTIC. IN THE CASE OF MANUFACTURING APT RUNTIME MODE
                        ;*      THEN BIT 08 OF $SWREG IS SET TO 1 THROUGH APT SCRIPTING.

                        ;*       THE TEST VERIFIES THE SIGNAL GENERATED FROM THE CACHE
                        ;*      TO THE UBI MODULE WHICH INDICATES TO THE UBI THAT A CACHE INTERRUPT
                        ;*      IS BEING CALLED FOR(CACHE PE INTR L).
                        ;*
                        ;*      THIS TEST ASSUMES THAT ALL MODULES EXCEPT THE UBI MODULE ARE KNOWN
                        ;*      GOOD MODULES.

                        ;*      THIS TEST TOGETHER WITH OTHER CACHE TESTS,ALLOW MFG. TO ELIMINATE
                        ;*      HAVING TO RUN THE  CACHE  DIAGNOSTIC DURING QUICK
                        ;*      VERIFY TESTING  OF THE UBI MODULE.

                        ;*      TEST DESCRIPTION:
                        ;*      VERIFY INTERRUPT LOGIC BY ASSURING THAT A TRAP OCCURS TO LOCATION
                        ;*      114 WHEN A LOCATION PREVIOUSLY WRITTEN
                        ;*      WITH WRONG HI/LO BYTE PARITY IS ACCESSED.
                        ;*              CONDITIONS:     PEA=0
                        ;*                              DCPI=0
                        ;;*****************************************************************
 015010                 TST20:
 015010  000004                SCOPE
 015012  004737  017662        JSR     PC,PRETST       ;GO SET UP PRETEST DATA
 015016  015352  015116  000020 .WORD   TST21,20$,20    ;DATA USED BY PRETST
2346 015024  005037  177572    CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2347 015030  005037  172516    CLR     MMR3            ;TURN OFF MAP RELOCATION
2348 015034  005037  172340    CLR     KIPAR0          ;PUT PAR0 BACK WHERE IT SHOULD AND
2349 015040  012737  000200  172342 MOV     #200,KIPAR1     ;PUT PAR1 BACK WHERc IT SHOULD
2350          177746          CACHE   =177746         ;CACHE LOCATION =177746
2351          177744          CMPE    =177744         ;CMPE LOCATION =177744
2352          177750          MAINT   =177750         ;MAINTENANCE LOCATION =177750
2353          000114          CTRAPV  =114            ;CACHE TRAP VECTOR =114
2354          000116          CTRAPS  =116            ;CACHE TRAP STATUS =116
2355 015046  105737  007774        TSTB    CPUTYP          ;IS THIS AN 11/24?
2356 015052  001404            BEQ     21$             ;BRANCH OVER JUMP IF 11/44
2357 015054  105237  001100        INCB    $TSTNM          ;INCREMENT TEST NUMBER TO SIMULATE SKIPPING TEST
2358 015060  000137  015706        JMP     TST22           ;JUMP OVER THIS AND NEXT TESTS
2359 015064  132737  000200  020035 21$:   BITB    #200,$ENVM      ;IS APT SIZING?
2360 015072  001405            BEQ     12$             ;NO;TRY HARDWARE SWITCH REGISTER
```

```
2361 015074  032737  000100  020036          BIT     #100,$SWREG     ;YES APT IS SIZING;DOES APT SAY TO DO THIS TEST
2362 015102  001523                          BEQ     TST21           ;;NO - SKIP TEST                                    SEQ 0113
2363 015104  000404                          BR      20$             ;YES,DO TEST
2364 015106  032777  000100  164034  12$:    BIT     #100,@SWR       ;DOES HARDWARE SWITCH REGISTER SAY TO DO TEST?
2365 015114  001516                          BEQ     TST21           ;;NO - SKIP TEST
2366 015116  042737  000001  177572  20$:    BIC     #1,MMR0         ;TURN OFF RELOCATION
```

```
2367 015124  052737  000400  177746          BIS     #400,CACHE      ;FLUSH CACHE TO INVALIDATE ALL CACHE LOCATIONS
2368 015132  032737  010000  177746  10$:    BIT     #10000,CACHE    ;WAIT TILL DONE
2369 015140  001374                          BNE     10$
2370 015142  013702  000000                  MOV     0,R2            ;SAVE ADDR. 0 CONTENTS
2371 015146  005037  000000                  CLR     0               ;0'S TO MAIN MEMORY LOCATION 0.
2372 015152  013700  000114                  MOV     CTRAPV,R0       ;SAVE VECTORS
2373 015156  013701  000116                  MOV     CTRAPS,R1
2374 015162  012737  015264  000114          MOV     #4$,CTRAPV      ;SETUP FOR CACHE TRAP
2375 015170  012737  000340  000116          MOV     #340,CTRAPS
2376 015176  112737  000002  177750          MOVB    #2,MAINT        ;HODO ALLOWS CACHE UPDATES AND CLOCKING OF
2377                                                                  ;PARITY INFO TO INTERRUPT LOGIC ONLY DURING
2378                                                                  ;THE DESTINATION ACCESS OF AN INSTRUCTION.
2379 015204  005003                          CLR     R3              ;CLEAR ERROR FLAG
2380 015206  012737  000015  177746          MOV     #15,CACHE       ;NO UCB SO AS TO WRITE CACHE STORES
2381 015214  005737  040000                  TST     40000           ;UPDATE CACHE LOCATION 0000 WITH CORRECT
2382                                                                  ;PARITY STORAGE
2383 015220  052737  000100  177746          BIS     #100,CACHE      ;ALLOW WRITE WRONG PARITY DATA TO LO & HI BYTE
2384                                                                  ;PARITY STORE.
2385 015226  005737  000000                  TST     0               ;READ UPDATE TO CACHE LOCATION 0000;
2386                                                                  ;WRITE WRONG PARITY TO HI/LO BYTE PARITY STORES
2387 015232  042737  000100  177746          BIC     #100,CACHE      ;DISABLE WWPD
2388 015240  005037  177744                  CLR     CMPE            ;CLEAR CMPE AND PARITY DETECT LOGIC
2389 015244  042737  000005  177746          BIC     #5,CACHE        ;ALLOW FOR INTERRUPT TO OCCUR AND ENABLE LOW
2390                                                                  ;CACHE
2391 015252  005737  000000                  TST     0               ;READ HIT LO & HI BYTE PARITY CHECK GENERATORS
2392                                                                  ;WILL DETECT  WRONG PARITY AND THE PARITY
2393                                                                  ;ERROR WILL BE CLOCKED TO INTERRUPT LOGIC
2394 015256  000240                          NOP                     ;NEEDED FOR 11/44
2395 015260  005203                          INC     R3              ;INDICATE THAT TRAP DID NOT OCCUR
2396 015262  000401                          BR      25$             ;BRANCH OVER STACK CORRECTION
2397 015264  022626                  4$:     CMP     (R6)+,(R6)+     ;READJUST STACK DUE TO INTERRUPT
2398 015266  005037  177744          25$:    CLR     CMPE            ;CLEAR CMPE
2399 015272  012737  001015  177746          MOV     #1015,CACHE     ;DISABLE CACHE
2400 015300  105037  177750                  CLRB    MAINT           ;DISABLE MAINT. MODE
2401 015304  010237  000000                  MOV     R2,0            ;RESTORE LOCATION 0
2402 015310  010037  000114                  MOV     R0,CTRAPV       ;RESTORE CACHE INTERRUPT VECTORS
2403 015314  010137  000116                  MOV     R1,CTRAPS
2404 015320  052737  000400  177746          BIS     #400,CACHE      ;BEFORE LEAVING TEST FLUSH CACHE TO
2405                                                                  ;ELIMINATE ANY EFFECTS OF WWPD
2406 015326  032737  010000  177746  2$:     BIT     #10000,CACHE    ;WAIT TILL DONE
2407 015334  001374                          BNE     2$
2408 015336  012737  000000  177746          MOV     #0,CACHE        ;TURN CACHE ON
2409 015344  005703                          TST     R3              ;DID TRAP OCCUR?
2410 015346  001401                          BEQ     TST21           ;;BRANCH TO NEXT TEST IF YES
2411 015350  104020                          ERROR   +20             ;INTERRUPT/ABORT LOGIC TESTS TRAP TO LOCATION
2412                                                                  ;114 DID NOT OCCUR
```

```
2446                    .SBTTL  TEST 21 - VERIFY TRAP DUE TO CACHE PARITY ABORT
                        ;*****************************************************************
                        ;*TEST 21       VERIFY TRAP DUE TO CACHE PARITY ABORT
                        ;*      THIS TEST IS OPTIONAL AND IS SELECTED BY SETTING MFM HARDWARE
                        ;*      SWITCH REGISTER BIT 08 TO A 1 IN THE CASE OF STANDALONE OPERATION
                        ;*      OF THE DIAGNOSTIC. IN THE CASE OF MANUFACTURING APT RUNTIME MODE
                        ;*      THEN BIT 08 OF $SWREG IS SET TO 1 THROUGH APT SCRIPTING.
                        ;*
                        ;*       THE TEST VERIFIES THE SIGNAL GENERATED FROM THE CACHE(BUS PBL)
                        ;*      TO THE UBI MODULE WHICH INDICATES TO THE UBI THAT A CACHE ABORT
                        ;*      IS BEING CALLED FOR.
                        ;*
                        ;*      THIS TEST ASSUMES THAT ALL MODULES EXCEPT UBI ARE KNOWN GOOD  MODULES
                        ;*
                        ;*      THIS TEST TOGETHER WITH OTHER CACHE TESTS,ALLOW MFG. TO ELIMINATE
                        ;*      HAVING TO RUN THE  CACHE  DIAGNOSTIC DURING QUICK
                        ;*      VERIFY TESTING  OF THE UBI MODULE.
                        ;*
                        ;*      TEST DESCRIPTION:
                        ;*
                        ;*      VERIFY ABORT LOGIC BY THE FOLLOWING RESULTS WHEN A LOCATION
                        ;*      PREVIOUSLY WRITTEN WITH WRONG HI/LO BYTE PARITY IS ACCESSED.
                        ;*              1. INSTRUCTION CYCLE WILL BE ABORTED
                        ;*              2. THE ABORT CAUSES TRAP TO 114
                        ;*
                        ;*              PROCEDURE:      INHIBIT CLOCKING OF PARITY ERROR SIGNAL TO
                        ;*                              INTERRUPT LOGIC. ALLOW CMPE<15> TO BE SET
                        ;*                              BY ABORT SIGNAL WHICH IS ASSERTED BY PARITY
                        ;*                              ERROR SIGNAL TO ABORT LOGIC.
                        ;*
                        ;*
                        ;*              CONDITIONS:     PEA=1
                        ;*                              DCPI=1
                        ;*
                        ;*****************************************************************
                        TST21:
     015352                             SCOPE
     015352  000004
     015354  004737  017662             JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     015360  015706  015420  000021      .WORD   TST22,20$,21    ;DATA USED BY PRETST
2447 015366  132737  000200  020035      BITB    #200,$ENVM      ;IS APT SIZING?
2448 015374  001405                     BEQ     12$             ;NO;TRY HARDWARE SWITCH REGISTER
2449 015376  032737  000100  020036      BIT     #100,$SWREG     ;YES APT IS SIZING;DOES APT SAY TO DO THIS TEST
2450 015404  001533                     BEQ     21$             ;NO - SKIP TEST
2451 015406  000404                     BR      20$             ;YES,DO TEST
2452 015410  032737  000100  177570  12$:BIT     #100,177570     ;DOES HARDWARE SWITCH REGISTER SAY TO DO TEST?
2453 015416  001526                     BEQ     21$             ;NO - SKIP TEST
2454 015420  042737  000001  177572  20$:BIC     #1,177572       ;TURN OFF RELOCATION
2455 015426  052737  000400  177746          BIS     #400,CACHE      ;FLUSH CACHE TO INVALIDATE ALL CACHE LOCATIONS
2456 015434  032737  010000  177746  2$: BIT     #10000,CACHE    ;WAIT TILL DONE
2457 015442  001374                     BNE     2$
2458 015444  013702  000000             MOV     0,R2            ;SAVE ADDRESS 0 CONTENTS
```

```
2459 015450  005057  000000                CLR     0               ;ALL 0'S TO LOCATION 0
2460 015454  005003                        CLR     R3              ;ADDRESS 0 TO R3                    SEQ 0116
2461 015456  013700  000114                MOV     CTRAPV,R0       ;SAVE VECTORS
2462 015462  013701  000116                MOV     CTRAPS,R1
2463 015466  012737  015576  000114        MOV     #4$,CTRAPV      ;SETUP FOR TRAP
2464 015474  012737  000340  000116        MOV     #340,CTRAPS
```

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

```
                                        .SBTTL  TEST 22 - LMA REGISTER PHYSICAL ADDRESS CHECK
                                        ;;*****************************************************************
                                        ;*TEST 22       LMA REGISTER PHYSICAL ADDRESS CHECK
                                        ;*
                                        ;*      THE NEXT 7 TESTS ARE EXECUTED ON THE 11/24 ONLY.
                                        ;*
                                        ;*      THIS TEST IS TO CHECK OUT THE LMA (LAST MAPPED ADDRESS) REGISTER FOR
                                        ;*      PROPER CONTENTS.  FIRST, THE PAR AND MAP REGISTERS ARE SET, THEN A
                                        ;*      PHYSICAL ADDRESS IS LOADED INTO AN EXPECTED DATA LOCATION.  THEN THE
                                        ;*      MAP IS INSURED TO BE ON AND A MEMORY ACCESS IS DONE, USING THE MAP
                                        ;*      REGISTER SO THE LMA IS LOADED.  THE LMA IS THEN CHECKED FOR CONTAINING
                                        ;*      THE PROPER CONTENTS, CALLING AN ERROR IF EXPECTED DATA DID NOT APPEAR.
                                        ;*
                                        ;;*****************************************************************
 015706                                 TST22:
 015706  000004                                 SCOPE
 015710  004737  017662                         JSR     PC,PRETST       ;GO SET UP PRETEST DATA
 015714  016112  015722  000022                 .WORD   TST23,20$,22    ;DATA USED BY PRETST
2523 015722  042737  000001  177572  20$:     BIC     #BIT0,MMR0      ;TURN OFF MEMORY MANAGEMENT
2524 015730  042737  000060  172516           BIC     #60,MMR3        ;TURN OFF 22-BIT AND MAP RELOCATION
2525 015736  012737  015722  006144           MOV     #20$,EADRES     ;LOAD EADRES WITH LOWER 16 BITS OF THE PHYSICAL ADDRESS
2526 015744  005037  006146                   CLR     EADRES+2        ;LOAD UPPER 6 BITS WITH PHYSICAL BITS EXPECTED
2527 015750  012700  035722                   MOV     #20$+BIT13,R0   ;MOVE ADDRESS +20000 (TO REFERENCE PAR1) TO R0
2528 015754  005037  170200                   CLR     MAPL0           ;CLEAR MAP 0 LOWER 16
2529 015760  005037  170202                   CLR     MAPH0           ;CLEAR MAP 0 UPPER 6
2530 015764  013746  172340                   MOV     KIPAR0,-(SP)    ;SAVE KIPAR0
2531 015770  013746  172342                   MOV     KIPAR1,-(SP)    ;SAVE KIPAR1
2532 015774  005037  172340                   CLR     KIPAR0          ;CLEAR PAR0 FOR NO MAP USE FOR THIS AREA
2533 016000  012737  170000  172342           MOV     #170000,KIPAR1  ;PUT 170000 IN PAR1 FOR PROPER MEMORY REFERENCE
2534 016006  052737  000060  172516           BIS     #60,MMR3        ;TURN ON 22-BIT AND MAP RELOCATION
2535 016014  052737  000001  177572           BIS     #BIT0,MMR0      ;TURN ON  MEMORY MANAGEMENT
2536 016022  011001                           MOV     (R0),R1         ;DO THE MAP REGISTER READ THROUGH THE MAP
2537 016024  023737  006144  177734           CMP     EADRES,LMALOW   ;SEE IF LMA LOWER 16 WERE LOADED PROPERLY
2538 016032  001007                           BNE     1$              ;BRANCH TO CALL ERROR IF NOT
2539 016034  013737  177736  001172           MOV     LMAHI,$TMP0     ;MOVE HI 6 BITS TO $TMP0 FOR PREPARATION
2540 016042  042737  177700  001172           BIC     #177700,$TMP0   ;CLEAR ALL BUT LOWER 6 BITS
2541 016050  001414                           BEQ     2$              ;BRANCH AROUND ERROR IF OK
2542 016052  013737  177734  006150  1$:      MOV     LMALOW,EADRS2   ;MOVE LOWER 16 BITS OF RECEIVED DATA TO EADRS2 FOR ERROR
2543 016060  013737  177736  006152           MOV     LMAHI,EADRS2+2  ;MOVE UPPER 6 BITS OF RECEIVED DATA TO EADRS2+2 FOR ERROR
2544 016066  012637  172342                   MOV     (SP)+,KIPAR1    ;RESTORE KIPAR1
2545 016072  012637  172340                   MOV     (SP)+,KIPAR0    ;RESTORE KIPAR0
2546 016076  104023                           ERROR   +23             ;LMA NOT LOADED PROPERLY
2547 016100  000404                           BR      TST23           ;;BRANCH OVER PAR RESTORATION - NOT NEEDED
2548 016102  012637  172342          2$:      MOV     (SP)+,KIPAR1    ;RESTORE KIPAR1
2549 016106  012637  172340                   MOV     (SP)+,KIPAR0    ;RESTORE KIPAR0
```

```
                                        .SBTTL  TEST 23 - LMA FORCE JUMPER BIT TEST
                                        ;;*****************************************************************
                                        ;*TEST 23       LMA FORCE JUMPER BIT TEST
                                        ;*
                                        ;*      THIS TEST DETERMINES THAT THE FORCE JUMPER BIT OF THE LMA IS ZERO AFTER
                                        ;*      A SYSTEM RESET.
                                        ;*
                                        ;;*****************************************************************
     016112                             TST23:
     016112  000004                             SCOPE
     016114  004737  017662                     JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     016120  016152  016126  000023             .WORD   TST24,20$,23    ;DATA USED BY PRETST
2557         000100                     FJBIT   =100                    ;FORCE JUMPER BIT IS BIT 6
2558 016126  000005                     20$:    RESET                   ;RESET THE WORLD, CLEARING THE FJBIT
2559 016130  032737  000100  177736             BIT     #FJBIT,LMAHI    ;CHECK THE BIT FOR BEING ZERO
2560 016136  001405                             BEQ     TST24           ;;BRANCH TO NEXT TEST IF OK
2561 016140  013701  177736                     MOV     LMAHI,R1        ;MOVE LMAHI TO R1 FOR ERROR CALL
2562 016144  042701  000100                     BIC     #FJBIT,R1       ;CLEAR THE BIT THAT SHOULD HAVE BEEN CLEAR
2563 016150  104024                             ERROR   +24             ;LMA FORCE JUMPER BIT NOT ZERO
```

```
                                        .SBTTL  TEST 24 - SETTING LMA FORCE JUMPER BIT TEST
                                        ;;*****************************************************************
                                        ;*TEST 24       SETTING LMA FORCE JUMPER BIT TEST
                                        ;*
                                        ;*      THIS TEST SETS THE FORCE JUMPER BIT AND TESTS ITS FUNCTIONALITY IF
                                        ;*      THE JUMPERS ARE NOT IN THEIR DEFAULT STATE.  IF NOT ('LOWEST' OR
                                        ;*      'HIGEST' DO NOT CONTAIN THE DEFAULT VALUES OF 170000 OR 177400
                                        ;*      RESPECTIVELY), THIS TEST INSURES THAT THE PREVIOUSLY DISABLED MAP
                                        ;*      REGISTERS ARE ENABLED WITH THE FJ BIT SET.
                                        ;*
                                        ;;*****************************************************************
     016152                             TST24:
     016152  000004                             SCOPE
     016154  004737  017662                     JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     016160  016352  016166  000024             .WORD   TST25,20$,24    ;DATA USED BY PRETST
2574 016166  052737  000100  177736  20$:      BIS     #FJBIT,LMAHI    ;SET THE BIT
2575 016174  032737  000100  177736             BIT     #FJBIT,LMAHI    ;SEE IF IT WAS SET
2576 016202  001005                             BNE     1$              ;BRANCH IF SET
2577 016204  013701  177736                     MOV     LMAHI,R1        ;MOVE LMAHI TO R1 FOR ERROR CALL
2578 016210  052701  000100                     BIS     #FJBIT,R1       ;SET THE BIT THAT SHOULD HAVE BEEN SET
2579 016214  104025                             ERROR   +25             ;LMA FORCE JUMPER BIT NOT SET
2580 016216  022737  170000  001254  1$:       CMP     #170000,LOWEST  ;SEE IF MAP REGISTER 0 IS LOWEST
2581 016224  001004                             BNE     2$              ;BRANCH AROUND UPPER LIMIT CHECK IF NOT
2582 016226  022737  177400  001256             CMP     #177400,HIGEST  ;SEE IF MAP REGISTER 31 IS HIGHEST
2583 016234  001446                             BEQ     TST25           ;;BRANCH TO NEXT TEST IF SO
2584 016236  012737  016276  001106  2$:       MOV     #3$,$LPERR      ;RESET LOOP ON ERROR TO 3$
2585 016244  013746  172350                     MOV     KIPAR4,-(SP)    ;SAVE PAR4
2586 016250  013737  001260  172350             MOV     UBMLOW,KIPAR4   ;MOVE LOWEST PAGE OF MEMORY WINDOW TO PAR4
2587 016256  162737  000200  172350             SUB     #200,KIPAR4     ;SUBTRACT 200 FROM KIPAR4 TO PREPARE FOR SUBROUTINE
2588 016264  012700  117776                     MOV     #117776,R0      ;THIS WILL BE USED TO SELECT PAR4, ADDRESS 17776
2589 016270  012702  125252                     MOV     #125252,R2      ;LOAD CONSTANT TO R2
2590 016274  000417                             BR      5$              ;BRANCH OVER LOOP SETUP
2591 016276  032777  001000  162644  3$:       BIT     #BIT9,@SWR      ;SEE IF LOOP ON ERROR IS STILL SET
2592 016304  001407                             BEQ     4$              ;BRANCH OUT IF NOT
2593 016306  162737  000200  172350             SUB     #200,KIPAR4     ;UNDO NEXT PAGE MAP FOR LOOPING
2594 016314  004737  005046                     JSR     PC,TSTLOC       ;GO TEST LOCATION FOR WRITEABILITY
2595 016320  000766                             BR      3$              ;BRANCH BACK FOR LOOPBACK
2596 016322  104027                             ERROR   +27             ;FORCE JUMPER BIT FAILS TO REVERT MAP REGISTER STATUS TO DEF
2597 016324  023737  001262  172350  4$:       CMP     UBMHI,KIPAR4    ;SEE IF HIGHEST HAS BEEN REACHED
2598 016332  001405                             BEQ     6$              ;BRANCH TO RESTORE PAR4 AND LEAVE TEST IF SO
2599 016334  004737  005046          5$:       JSR     PC,TSTLOC       ;GO TEST LOCATION FOR WRITEABILITY
2600 016340  000771                             BR      4$              ;BRANCH BACK FOR ANOTHER TEST IF LOCATION LOADED
2601 016342  104027                             ERROR   +27             ;FORCE JUMPER BIT FAILS TO REVERT MAP REGISTER STATUS TO DEF
2602 016344  000767                             BR      4$              ;BRANCH BACK FOR ANOTHER TEST
2603 016346  012637  172350          6$:       MOV     (SP)+,KIPAR4    ;RESTORE KIPAR4
```

```
                                        .SBTTL  TEST 25 - CLEARING THE FORCE JUMPER BIT
                                        ;;*****************************************************************
                                        ;*TEST 25       CLEARING THE FORCE JUMPER BIT
                                        ;*
                                        ;*      THIS TEST CLEARS THE FJ BIT AND INSURES THAT I' IS SUCCESSFULLY CLEARED.
                                        ;*
                                        ;;*****************************************************************
     016352                             TST25:
     016352  000004                             SCOPE
     016354  004737  017662                      JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     016360  016416  016366  000025              .WORD   TST26,20$,25    ;DATA USED BY PRETST
2610 016366  042737  000100  177736  20$:       BIC     #FJBIT,LMAHI    ;CLEAR THE BIT
2611 016374  032737  000100  177736             BIT     #FJBIT,LMAHI    ;CHECK TO SEE THAT IT WAS CLEARED
2612 016402  001405                             BEQ     TST26           ;;BRANCH IF CLEARED
2613 016404  013701  177736                      MOV     LMAHI,R1        ;MOVE LMAHI TO R1 FOR ERROR CALL
2614 016410  042701  000100                      BIC     #FJBIT,R1       ;CLEAR THE BIT THAT SHOULD HAVE BEEN CLEAR
2615 016414  104024                             ERROR   +24             ;LMA FORCE JUMPER BIT NOT ZERO
```

```
                                        .SBTTL  TEST 26 - LMA CONTROL BITS TEST - DATI
                                        ;;*****************************************************************
                                        ;*TEST 26       LMA CONTROL BITS TEST - DATI
                                        ;*
                                        ;*      THIS TEST INSURES THE CONTROL BITS 14 AND 15 LOAD PROPERLY DOING A
                                        ;*      DATI.
                                        ;*
                                        ;;*****************************************************************
016416                                  TST26:
016416  000004                                  SCOPE
016420  004737  017662                          JSR     PC,PRETST       ;GO SET UP PRETEST DATA
016424  016616  016516  000026                  .WORD   TST27,20$,26    ;DATA USED BY PRETST
2623 016432  013746  172340                     MOV     KIPAR0,-(SP)    ;SAVE KIPAR0
016436  013746  172342                          MOV     KIPAR1,-(SP)    ;SAVE KIPAR1
016442  012737  170000  172342                  MOV     #170000.KIPAR1  ;LOAD KIPAR1
016450  005037  172340                          CLR     KIPAR0          ;CLEAR KIPAR0
2624 016454  012737  000001  177572             MOV     #1,MMR0         ;TURN ON MEMORY MANAGEMENT
2625 016462  012737  000020  172516             MOV     #20,MMR3        ;TURN ON 22-BIT ADDRESSING
2626 016470  012700  036540                     MOV     #1$+BIT13,R0    ;MOVE LOCATION ADDRESS +20000 (TO REF PAR1) TO R0
2627 016474  011001                             MOV     (R0),R1         ;DO A DATI
2628 016476  013702  177736                     MOV     LMAHI,R2        ;MOVE LMAHI TO R2 FOR CONTROL
2629 016502  042702  037777                     BIC     #37777,R2       ;CLEAR ALL BUT THE CONTROL BITS
2630 016506  001437                             BEQ     3$              ;BRANCH TO FINISH TEST IF ALL CLEAR
2631 016510  005037  001172                     CLR     $TMP0           ;CLEAR THE EXPECTED LOCATION
2632 016514  000426                             BR      2$              ;GO CALL ERROR
2633 016516                             20$:
016516  013746  172340                          MOV     KIPAR0,-(SP)    ;SAVE KIPAR0
016522  013746  172342                          MOV     KIPAR1,-(SP)    ;SAVE KIPAR1
016526  012737  170000  172342                  MOV     #170000,KIPAR1  ;LOAD KIPAR1
016534  005037  172340                          CLR     KIPAR0          ;CLEAR KIPAR0
2634 016540  011001                     1$:     MOV     (R0),R1         ;DO A DATI
2635 016542  013702  177736                     MOV     LMAHI,R2        ;MOVE LMAHI TO R2 FOR CONTROL
2636 016546  042702  037777                     BIC     #37777,R2       ;CLEAR ALL BUT THE CONTROL BITS
2637 016552  020237  001172                     CMP     R2,$TMP0        ;SEE IF EXPECTED CAME UP
2638 016556  001005                             BNE     2$              ;BRANCH TO CALL ERROR IF IT DIDN'T
2639 016560  032777  001000  162362             BIT     #BIT9,@SWR      ;SEE IF LOOP ON ERROR IS SET
2640 016566  001364                             BNE     1$              ;BRANCH BACK FOR ANOTHER TRY IF SET
2641 016570  000406                             BR      3$              ;GO EXIT TEST - ALL DONE
2642 016572  012637  172342             2$:     MOV     (SP)+,KIPAR1    ;RESTORE KIPAR1
2643 016576  012637  172340                     MOV     (SP)+,KIPAR0    ;RESTORE KIPAR0
2644 016602  104026                             ERROR   +26             ;LMA CONTROL BITS INCORRECT
2645 016604  000404                             BR      TST27           ;;BRANCH OVER PAR RESTORATIONS
2646 016606  012637  172342             3$:     MOV     (SP)+,KIPAR1    ;RESTORE KIPAR1
2647 016612  012637  172340                     MOV     (SP)+,KIPAR0    ;RESTORE KIPAR0
```

```
                                        .SBTTL  TEST 27 - LMA CONTROL BITS TEST - DATO
                                        ;;*****************************************************************
                                        ;*TEST 27       LMA CONTROL BITS TEST - DATO
                                        ;*
                                        ;*      THIS TEST INSURES THE CONTROL BITS 14 AND 15 LOAD PROPERLY DOING A
                                        ;*      DATO.
                                        ;*
                                        ;;*****************************************************************
016616                                  TST27:
016616  000004                                SCOPE
016620  004737  017662                        JSR     PC,PRETST       ;GO SET UP PRETEST DATA
016624  017010  016710  000027                .WORD   TST30,20$,27    ;DATA USED BY PRETST
        100000                  DATO    =100000
016632  013746  172340                        MOV     KIPAR0,-(SP)    ;SAVE KIPAR0
016636  013746  172342                        MOV     KIPAR1,-(SP)    ;SAVE KIPAR1
016642  012737  170000  172342                MOV     #170000,KIPAR1  ;LOAD KIPAR1
016650  005037  172340                        CLR     KIPAR0          ;CLEAR KIPAR0
016654  012700  036732                        MOV     #1$+BIT13,R0    ;MOVE LOCATION ADDRESS +20000 (TO REF PAR1) TO R0
016660  010110                              MOV     R1,(R0)         ;DO A DATO
016662  013702  177736                        MOV     LMAHI,R2        ;MOVE LMAHI TO R2 FOR CONTROL BIT ANALYSIS
016666  042702  037777                        BIC     #37777,R2       ;CLEAR ALL BUT THE CONTROL BITS
016672  022702  100000                        CMP     #DATO,R2        ;SEE IF BIT 15 IS SET AND 14 IS CLEAR
016676  001440                              BEQ     3$              ;BRANCH IF OK
016700  012737  100000  001172                MOV     #DATO,$TMP0     ;MOVE EXPECTED DATA TO $TMP0
016706  000426                              BR      2$              ;GO CALL ERROR
016710                                  20$:
016710  013746  172340                        MOV     KIPAR0,-(SP)    ;SAVE KIPAR0
016714  013746  172342                        MOV     KIPAR1,-(SP)    ;SAVE KIPAR1
016720  012737  170000  172342                MOV     #170000,KIPAR1  ;LOAD KIPAR1
016726  005037  172340                        CLR     KIPAR0          ;CLEAR KIPAR0
016732  010110                          1$:     MOV     R1,(R0)         ;DO A DATO
016734  013702  177736                        MOV     LMAHI,R2        ;MOVE LMAHI TO R2 FOR CONTROL
016740  042702  037777                        BIC     #37777,R2       ;CLEAR ALL BUT THE CONTROL BITS
016744  020237  001172                        CMP     R2,$TMP0        ;SEE IF EXPECTED CAME UP
016750  001005                              BNE     2$              ;BRANCH TO CALL ERROR IF IT DIDN'T
016752  032777  001000  162170                BIT     #BIT9,@SWR      ;SEE IF LOOP ON ERROR IS SET
016760  001364                              BNE     1$              ;BRANCH BACK FOR ANOTHER TRY IF SET
016762  000406                              BR      3$              ;GO EXIT - ALL DONE
016764  012637  172342                  2$:     MOV     (SP)+,KIPAR1    ;RESTORE KIPAR1
016770  012637  172340                        MOV     (SP)+,KIPAR0    ;RESTORE KIPAR0
016774  104026                              ERROR   +26             ;LMA CONTROL BITS INCORRECT
016776  000404                              BR      TST30           ;;BRANCH OVER PAR RESTORATION - NOT NEEDED
017000  012637  172342                  3$:     MOV     (SP)+,KIPAR1    ;RESTORE KIPAR1
017004  012637  172340                        MOV     (SP)+,KIPAR0    ;RESTORE KIPAR0
```

```
2686                                    .SBTTL  TEST 30 - LMA CONTROL BITS TEST - DATOB
                                        ;;*****************************************************************
                                        ;*TEST 30       LMA CONTROL BITS TEST - DATOB
                                        ;*
                                        ;*      THIS TEST INSURES THE CONTROL BITS 14 AND 15 LOAD PROPERLY DOING A
                                        ;*      DATOB.
                                        ;*
                                        ;;*****************************************************************
     017010                             TST30:
     017010  000004                             SCOPE
     017012  004737  017662                     JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     017016  017202  017102  000030             .WORD   TST31,20$,30    ;DATA USED BY PRETST
2687         140000                     DATOB   =140000                 ;DATOB CONTROL BITS STATUS=140000
2688 017024  013746  172340                     MOV     KIPAR0,-(SP)    ;SAVE KIPAR0
     017030  013746  172342                     MOV     KIPAR1,-(SP)    ;SAVE KIPAR1
     017034  012737  170000  172342             MOV     #170000,KIPAR1  ;LOAD KIPAR1
     017042  005037  172340                     CLR     KIPAR0          ;CLEAR KIPAR0
2689 017046  012700  037124                     MOV     #1$+BIT13,R0    ;MOVE LOCATION ADDRESS +20000 (TO REF PAR1) TO R0
2690 017052  110110                             MOVB    R1,(R0)         ;DO A DATOB
2691 017054  013702  177736                     MOV     LMAHI,R2        ;MOVE LMAHI TO R2 FOR CONTROL
2692 017060  042702  037777                     BIC     #37777,R2       ;CLEAR ALL BUT THE CONTROL BITS
2693 017064  022702  140000                     CMP     #DATOB,R2       ;SEE IF BIT 15 AND 14 ARE SET
2694 017070  001440                             BEQ     3$              ;BRANCH IF OK
2695 017072  012737  140000  001172             MOV     #DATOB,$TMP0    ;MOVE EXPECTED DATA TO $TMP0
2696 017100  000426                             BR      2$              ;BRANCH TO CALL ERROR
2697 017102                             20$:
     017102  013746  172340                     MOV     KIPAR0,-(SP)    ;SAVE KIPAR0
     017106  013746  172342                     MOV     KIPAR1,-(SP)    ;SAVE KIPAR1
     017112  012737  170000  172342             MOV     #170000,KIPAR1  ;LOAD KIPAR1
     017120  005037  172340                     CLR     KIPAR0          ;CLEAR KIPAR0
2698 017124  110110                     1$:     MOVB    R1,(R0)         ;DO A DATOB
2699 017126  013702  177736                     MOV     LMAHI,R2        ;MOVE LMAHI TO R2 FOR CONTROL
2700 017132  042702  037777                     BIC     #37777,R2       ;CLEAR ALL BUT THE CONTROL BITS
2701 017136  020237  001172                     CMP     R2,$TMP0        ;SEE IF EXPECTED CAME UP
2702 017142  001005                             BNE     2$              ;BRANCH IF IT DIDN'T
2703 017144  032777  001000  161776             BIT     #BIT9,@SWR      ;SEE IF LOOP ON ERROR IS SET
2704 017152  001364                             BNE     1$              ;BRANCH BACK FOR ANOTHER TRY IF SET
2705 017154  000406                             BR      3$              ;BRANCH TO COMPLETE TEST - ALL DONE
2706 017156  012637  172342             2$:     MOV     (SP)+,KIPAR1    ;RESTORE KIPAR1
2707 017162  012637  172340                     MOV     (SP)+,KIPAR0    ;RESTORE KIPAR0
2708 017166  104026                             ERROR   +26             ;LMA CONTROL BITS INCORRECT
2709 017170  000404                             BR      TST31           ;;BRANCH OVER PAR RESTORATION - NOT NEEDED
2710 017172  012637  172342             3$:     MOV     (SP)+,KIPAR1    ;RESTORE KIPAR1
2711 017176  012637  172340                     MOV     (SP)+,KIPAR0    ;RESTORE KIPAR0
2712                                    ;;*****************************************************************
2713                                    ;*
2714                                    ;*      >>NOTE<<: 'DATIP' CANNOT BE CHECKED IN THE 11/24 BECAUSE THE LMA IS
2715                                    ;*                WRITTEN TWICE WHEN A 'DATIP' IS EXECUTED, DESTROYING THE
2716                                    ;*                'DATIP' STATE THAT WAS WRITTEN FIRST.
2717                                    ;*
2718                                    ;;*****************************************************************
```

```
                        .SBTTL  TEST 31 - MEMORY ON UNIBUS TEST
                        ;*****************************************************************
                        ;*TEST 31       MEMORY ON UNIBUS TEST
                        ;*
                        ;*      THIS TEST DETERMINES IF IT HAS BEEN SELECTED BY A '1' IN BIT 5 OF
                        ;*      THE SWITCH REGISTER.  IF IT HAS, IT THEN DETERMINES IF THERE IS ANY
                        ;*      UNIBUS MEMORY - AN ERROR RESULTS IF THERE IS NONE.  IF THERE IS MEMORY,
                        ;*      IT THEN  SIZES THE AMOUNT OF MEMORY ON THE UNIBUS, INFORMS THE USER
                        ;*      HOW MUCH MEMORY IT FOUND ON THE FIRST PASS, THEN SETS AND CLEARS ALL
                        ;*      BITS OF ALL LOCATIONS IN THE UNIBUS MEMORY FOUND USING THE 'MARCH'
                        ;*      ALGORITHM.
                        ;*
                        ;*              M7098 FOR THE 11/44, M7134 FOR THE 11/24
                        ;*               JUMPER SETTINGS FOR THE MAP REGISTERS
                        ;*
                        ;*         V<<<<<LOWER LIMIT>>>>>V    V<<<<UPPER LIMIT>>>>V
                        ;*         0     0     0    0    0    0    0    0    0    0
                        ;*         '     '     '    '    '    '    '    '    '    '
                        ;*      W12'  W11'  W10'  W9'  W8.  W7   W6   W5   W4.  W3.
                        ;*         .     .     .    .    .    .    .    .    .    .
                        ;*         0     0     0    0    0    0    0    0    0    0
                        ;*
                        ;*W3-W7 AND W8-W12 ARE THE BINARY-CODED PAGE NUMBER LIMIT (UPPER OR LOWER).
                        ;*A JUMPER IN CORRESPONDS TO A LOGIC "0"; A JUMPER OUT TO A LOGIC "1".  TO
                        ;*SET THE JUMPERS, DETERMINE WHICH UNIBUS PAGE THE MEMORY RESIDES IN (0 TO 31)
                        ;*BY CHECKING WHICH OF THE 5 ADDRESS BITS BA17-BA13 ARE ASSERTED.  ALL ZEROS
                        ;*IS PAGE 0, 10000 IS PAGE 1, 01000 IS PAGE 2, 11000 IS PAGE 3, ETC.  UP TO
                        ;*PAGE 31 (11111).  FOR THE MEMORY TO BE DETECTED, IT MUST LIE AT OR ABOVE
                        ;*THE LOWER LIMIT JUMPER SETTING AND BELOW THE UPPER LIMIT JUMPER SETTING.
                        ;*THUS TO HAVE UNIBUS MEMORY IN PAGES 5-9, ONE WOULD SET THE LOWER LIMIT TO
                        ;*PAGE 5 (10100) OR W10 AND W12 OUT, AND W8, W9 AND W11 IN, AND THE UPPER LIMIT
                        ;*TO PAGE 10 (01010) OR W4 AND W6 OUT, AND W3, W5 AND W7 IN.  UNIBUS MEMORY
                        ;*MUST BE CONTIGUOUS SINCE NO GAPS ARE PERMITTED.
                        ;;*****************************************************************
     017202  000004     TST31:  SCOPE
2752 017204  042737  000001  177572          BIC     #1,MMR0         ;TURN OFF MEMORY MANAGEMENT
2753 017212  032777  000040  161730          BIT     #BIT5,@SWR      ;SEE IF THIS TEST HAS BEEN SELECTED
2754 017220  001417                          BEQ     2$              ;BRANCH TO JUMP IF TEST NOT SELECTED
2755 017222  012737  021306  001346          MOV     #$EOP,NXTTST    ;POINT TO ESCAPE VECTOR
2756 017230  022737  170000  001254  1$:     CMP     #170000,LOWEST  ;SEE IF LOWEST IS LOWEST
2757 017236  001012                          BNE     3$              ;GO DO TEST IF IT ISN'T - UNIBUS MEMORY EXISTS
2758 017240  022737  177400  001256          CMP     #177400,HIGEST  ;SEE IF HIGEST IS HIGHEST
2759 017246  001006                          BNE     3$              ;GO DO TEST IF IT ISN'T - UNIBUS MEMORY EXISTS
2760 017250  012737  012620  001106          MOV     #SIZEJ,$LPERR   ;MOVE SIZE JUMPER ROUTINE TO LOOP ON ERROR
2761 017256  104017                          ERROR   +17             ;NO UNIBUS MEMORY EXISTS
2762 017260  000137  021306          2$:     JMP     $EOP            ;JUMP TO END OF PASS
2763 017264  005037  001302          3$:     CLR     ERRCNT          ;CLEAR THE ERROR COUNT INDICATOR
2764 017270  005037  001314                  CLR     PCFUER          ;CLEAR THE ERROR REGISTER RECEIVER
2765 017274  005037  172516                  CLR     MMR3            ;CLEAR MEMORY MANAGEMENT REGISTER MMR3
2766 017300  052737  000020  172516          BIS     #20,MMR3        ;TURN ON 22-BIT MAPPING
2767 017306  005037  172340                  CLR     KIPAR0          ;MAP PAR0 TO 0-4K
```

```
2768 017312  012737  000200  172342          MOV     #200,KIPAR1     ;MAP PAR1 TO 4-8K
2769 017320  013700  001272                  MOV     UBRHI R0        ;MOVE UBRHI TO R0                                  SEQ 0126
2770 017324  163700  001270                  SUB     UBRLOW,R0       ;SUBTRACT UBRLOW FROM IT, AND
2771 017330  005200                          INC     R0              ;ADD LAST BLOCK OF 4K TO LOOP COUNTER
2772 017332  010001                          MOV     R0,R1           ;SAVE R0 IN R1
2773 017334  005046                  4$:     CLR     -(SP)           ;CLEAR THE MAJOR LOOP INDICATOR ON STACK
```

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

```
2825 017612  077050                        SOB     R0,9$           ;BRANCH BACK IF MORE BLOCKS TO CHECK
2826 017614  005766  000006                TST     6(SP)           ;TEST TO SEE IF THIS IS SECOND PASS                 SEQ 0128
2827 017620  001404                        BEQ     17$             ;BRANCH TO 2ND PASS SETUP IF NOT
2828 017622  062706  000010                ADD     #10,SP          ;CLEAN UP STACK
2829 017626  000137  021306                JMP     $EOP            ;JUMP TO END OF PASS
2830 017632  012766  017776  000006  17$:  MOV     #17776,6(SP)    ;MOVE 2K WORDS -2 (ALSO 2ND PASS INDICATOR) TO STACK
```

SEQ 0129

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

SEQ 0130

```
                                        .SBTTL  PRETEST DATA SETUP SUBROUTINE
                                ;;*****************************************************************
017662  013777  001100  161262  PRETST: MOV     $TSTNM,@DISPLAY ;DISPLAY TEST NUMBER FOR ALL TO SEE
017670  010046                  MOV     R0,-(SP)        ;SAVE R0
017672  016600  000002          MOV     2(SP),R0        ;MOVE ORIGINAL RETURN ADDRESS TO R0
017676  012037  001346          MOV     (R0)+,NXTTST    ;SAVE STARTING ADDRESS OF NEXT TEST FOR ESCAPE ON PAR ERRORS
017702  012037  001106          MOV     (R0)+,$LPERR    ;SET LOOP ON ERROR POINTER TO 20$
017706  012037  001100          MOV     (R0)+,$TSTNM    ;SETUP TEST NUMBER AND CLEAR THE ERROR FLAG
017712  010066  000002          MOV     R0,2(SP)        ;FUDGE RETURN OVER DATA
017716  012600                  MOV     (SP)+,R0        ;RESTORE R0
017720  011637  001104          MOV     (SP),$LPADR     ;SET LOOP ON TEST POINTER TO START OF TEST
017724  000207                  RTS     PC              ;RETURN TO TEST
```

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

```
2851                                    .SBTTL  APT PARAMETER BLOCK
                                        ;;*****************************************************************
                                        ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                        ;;*****************************************************************
                 020000                         .$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 020000                         $APTHDR  ;;POINT TO APT HEADER BLOCK
                 020000                         .=.$X    ;;RESET LOCATION COUNTER
                                        ;;*****************************************************************
                                        ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                                        ;INTERFACE SPEC.
          020000                        $APTHD:
          020000 000000                 $HIBTS: .WORD   0        ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
          020002 020014                 $MBADR: .WORD   $MAIL    ;;ADDRESS OF APT MAILBOX (BITS 0-15)
          020004 000005                 $TSTM:  .WORD   5        ;;RUN TIM OF LONGEST TEST
          020006 000010                 $PASTM: .WORD   10       ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
          020010 000000                 $UNITM: .WORD   0        ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
          020012 000052                         .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

```
2853                                    .SBTTL  APT MAILBOX-ETABLE
                                        ;;*****************************************************************
                                        .EVEN
020014                                  $MAIL:                          ;;APT MAILBOX
020014  000000                          $MSGTY: .WORD   AMSGTY   ;;MESSAGE TYPE CODE
020016  000000                          $FATAL: .WORD   AFATAL   ;;FATAL ERROR NUMBER
020020  000000                          $TESTN: .WORD   ATESTN   ;;TEST NUMBER
020022  000000                          $PASS:  .WORD   APASS    ;;PASS COUNT
020024  000000                          $DEVCT: .WORD   ADEVCT   ;;DEVICE COUNT
020026  000000                          $UNIT:  .WORD   AUNIT    ;;I/O UNIT NUMBER
020030  000000                          $MSGAD: .WORD   AMSGAD   ;;MESSAGE ADDRESS
020032  000000                          $MSGLG: .WORD   AMSGLG   ;;MESSAGE LENGTH
020034                                  $ETABLE:                        ;;APT ENVIRONMENT TABLE
020034     000                          $ENV:   .BYTE   AENV     ;;ENVIRONMENT BYTE
020035     000                          $ENVM:  .BYTE   AENVM    ;;ENVIRONMENT MODE BITS
020036  000000                          $SWREG: .WORD   ASWREG   ;;APT SWITCH REGISTER
020040  000000                          $USWR:  .WORD   AUSWR    ;;USER SWITCHES
020042  000000                          $CPUOP: .WORD   ACPUOP   ;;CPU TYPE,OPTIONS
                                        ;*                              BITS 15-11=CPU TYPE
                                        ;*                                      11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
                                        ;*                                      11/70=06,PDQ=07,Q=10
                                        ;*                              BIT 10=REAL TIME CLOCK
                                        ;*                              BIT  9=FLOATING POINT PROCESSOR
                                        ;*                              BIT  8=MEMORY MANAGEMENT
020044     000                          $MAMS1: .BYTE   AMAMS1   ;;HIGH ADDRESS,M.S. BYTE
020045     000                          $MTYP1: .BYTE   AMTYP1   ;;MEM. TYPE,BLK#1
                                        ;*                              MEM.TYPE BYTE   --  (HIGH BYTE)
                                        ;*                                      900 NSEC CORE=001
                                        ;*                                      300 NSEC BIPOLAR=002
                                        ;*                                      500 NSEC MOS=003
020046  000000                          $MADR1: .WORD   AMADR1   ;;HIGH ADDRESS,BLK#1
                                        ;*                                MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF ''TYPE'' ABOVE
020050     000                          $MAMS2: .BYTE   AMAMS2   ;;HIGH ADDRESS,M.S. BYTE
020051     000                          $MTYP2: .BYTE   AMTYP2   ;;MEM.TYPE,BLK#2
020052  000000                          $MADR2: .WORD   AMADR2   ;;MEM.LAST ADDRESS,BLK#2
020054     000                          $MAMS3: .BYTE   AMAMS3   ;;HIGH ADDRESS,M.S.BYTE
020055     000                          $MTYP3: .BYTE   AMTYP3   ;;MEM.TYPE,BLK#3
020056  000000                          $MADR3: .WORD   AMADR3   ;;MEM.LAST ADDRESS,BLK#3
020060     000                          $MAMS4: .BYTE   AMAMS4   ;;HIGH ADDRESS,M.S.BYTE
020061     000                          $MTYP4: .BYTE   AMTYP4   ;;MEM.TYPE,BLK#4
020062  000000                          $MADR4: .WORD   AMADR4   ;;MEM.LAST ADDRESS,BLK#4
020064  000000                          $VECT1: .WORD   AVECT1   ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
020066  000000                          $VECT2: .WORD   AVECT2   ;;INTERRUPT VECTOR#2BUS PRIORITY#2
020070  000000                          $BASE:  .WORD   ABASE    ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
020072  000000                          $DEVM:  .WORD   ADEVM    ;;DEVICE MAP
020074  000000                          $CDW1:  .WORD   ACDW1    ;;CONTROLLER DESCRIPTION WORD#1
020076  000000                          $CDW2:  .WORD   ACDW2    ;;CONTROLLER DESCRIPTION WORD#2
020100  177777                          $DDW0:  .WORD   ADDW0    ;;DEVICE DESCRIPTOR WORD#0
020102  177777                          $DDW1:  .WORD   ADDW1    ;;DEVICE DESCRIPTOR WORD#1
020104  000000                          $DDW2:  .WORD   ADDW2    ;;DEVICE DESCRIPTOR WORD#2
020106  000000                          $DDW3:  .WORD   ADDW3    ;;DEVICE DESCRIPTOR WORD#3
```

```
020110  000000                  $DDW4:  .WORD   ADDW4    ;;DEVICE DESCRIPTOR WORD#4
020112  000000                  $DDW5:  .WORD   ADDW5    ;;DEVICE DESCRIPTOR WORD#5
020114  000000                  $DDW6:  .WORD   ADDW6    ;;DEVICE DESCRIPTOR WORD#6
020116  000000                  $DDW7:  .WORD   ADDW7    ;;DEVICE DESCRIPTOR WORD#7
020120  000000                  $DDW8:  .WORD   ADDW8    ;;DEVICE DESCRIPTOR WORD#8
020122  000000                  $DDW9:  .WORD   ADDW9    ;;DEVICE DESCRIPTOR WORD#9
```

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

```
2855                                    .SBTTL  SCOPE HANDLER ROUTINE
                                        ;;*****************************************************************
                                        ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
                                        ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
                                        ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
                                        ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                        ;*SW14=1         LOOP ON TEST
                                        ;*SW11=1         INHIBIT ITERATIONS
                                        ;*SW09=1         LOOP ON ERROR
                                        ;*SW08=1         LOOP ON TEST IN SWR<4:0>
                                        ;*CALL
                                        ;*      SCOPE           ;;SCOPE=IOT
020140                                  $SCOPE:
020140  005037  001344                          CLR     RETRY           ;CLEAR RETRY FLAG AN THE START OF EACH TEST
020144  005037  001302                          CLR     ERRCNT          ;CLEAR THE MULTIPLE ERROR COUNTER
020150  005037  001244                          CLR     DATAOR          ;LOCATION FOR LOGICAL OR OF BAD DATA
020154  005037  001234                          CLR     ADDROR          ;LOCATION FOR LOGICAL OR OF ADDRESS
020160  005037  001236                          CLR     ADDROR+2        ;LOCATION FOR UPPER 6 BITS OF LOGICAL OR OF ADDRESS
020164  005037  001252                          CLR     PATTOR          ;LOCATION FOR LOGICAL OR OF PATTERN LOADED
020170  012737  177777  001240                MOV     #-1,DATAND      ;LOCATION FOR LOGICAL AND OF BAD DATA
020176  012737  177777  001230                MOV     #-1,ADRAND      ;LOCATION FOR LOGICAL AND OF ADDRESS
020204  012737  000077  001232                MOV     #77,ADRAND+2    ;LOCATION FOR UPPER 6 BITS OF LOGICAL AND OF ADDRESS
020212  012737  177777  001250                MOV     #-1,PATAND      ;LOCATION FOR LOGICAL AND OF PATTERN LOADED
020220  012737  000077  006146                MOV     #77,EADRES+2    ;RESTORE UPPER 6 BIT LOCATION OF EADRES+2
020226  012737  000077  006152                MOV     #77,EADRS2+2    ;RESTORE UPPER 6 BIT LOCATION OF EADRS2+2
020234  032777  040000  160706                BIT     #40000,@SWR     ;&& LOOP ON PRESENT TEST?
020242  001117                                BNE     $OVER           ;&& YES IF SW14=1
                                        ;#####START OF CODE FOR THE XOR TESTER#####
020244  000416                          $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                                                        ;;THIS INSTRUCTION TO A "NOP" (NOP-240)
020246  013746  000004                          MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
020252  012737  020272  000004                MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
020260  005737  177060                        TST     @#177060        ;;TIME OUT ON XOR?
020264  012637  000004                        MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
020270  000466                                BR      $SVLAD          ;;GO TO THE NEXT TEST
020272  022626                          5$:   CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
020274  012637  000004                        MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
020300  000426                                BR      7$              ;;LOOP ON THE PRESENT TEST
020302                                  6$:;#####END OF CODE FOR THE XOR TESTER#####
020302  032777  000400  160640                BIT     #BIT08,@SWR     ;&& LOOP ON SPEC. TEST?
020310  001407                                BEQ     2$              ;;BR IF NO
020312  017746  160632                        MOV     @SWR,-(SP)      ;&& SET DESIRED TEST NUM. FROM SWR
020316  042716  000340                        BIC     #$SWRMK,(SP)    ;;STRIP AWAY UNDESIRED BITS
020322  122637  001100                        CMPB    (SP)+,$TSTNM    ;;ON THE RIGHT TEST?
020326  001465                                BEQ     $OVER           ;;BR IF YES
020330  105737  001101                  2$:   TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
020334  001421                                BEQ     3$              ;;BR IF NO
020336  123737  001113  001101                CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
020344  101015                                BHI     3$              ;;BR IF NO
020346  032777  001000  160574                BIT     #BIT09,@SWR     ;&& LOOP ON ERROR?
020354  001404                                BEQ     4$              ;;BR IF NO
```

```
020356  013757  001106  001104  7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
020364  000446                          BR      $OVER                                                   SEQ 0137
020366  105037  001101          4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
020372  005037  001210                  CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
020376  000415                          BR      1$              ;;ESCAPE TO THE NEXT TEST
020400  032777  004000  160542  3$:     BIT     #BIT11,@SWR     ;&& INHIBIT ITERATIONS?
```

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

```
2857                                    .SBTTL  ERROR HANDLER ROUTINE
                                        ;;*****************************************************************
                                        ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
                                        ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
                                        ;*AND GO TO ERTYPE ON ERROR
                                        ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                        ;*SW15=1         HALT ON ERROR
                                        ;*               HALT CAN OCCUR BEFORE AND AFTER THE ERROR TYPEOUT
                                        ;*SW13=1         INHIBIT ERROR TYPEOUTS
                                        ;*SW10=1         BELL ON ERROR
                                        ;*SW09=1         LOOP ON ERROR
                                        ;*CALL
                                        ;*       ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
020520                                  $ERROR:
020520  113737  001100  001310                MOVB    $TSTNM,TESTNO   ;SAVE TEST NUMBER FOR ERROR TYPE OUT
020526  005237  001302                        INC     ERRCNT          ;COUNT ALL MULTIPLE ERRORS
020532  010037  001156                        MOV     R0,$REG0        ;SAVE R0 FOR POSSIBLE TYPE OUT
020536  010137  001160                        MOV     R1,$REG1        ;SAVE R1 FOR POSSIBLE TYPE OUT
020542  010237  001162                        MOV     R2,$REG2        ;SAVE R2 FOR POSSIBLE TYPE OUT
020546  010337  001164                        MOV     R3,$REG3        ;SAVE R3 FOR POSSIBLE TYPE OUT
020552  010437  001166                        MOV     R4,$REG4        ;SAVE R4 FOR POSSIBLE TYPE OUT
020556  010537  001170                        MOV     R5,$REG5        ;SAVE R5 FOR POSSIBLE TYPE OUT
020562  105237  001101          7$:           INCB    $ERFLG          ;;SET THE ERROR FLAG
020566  001775                                BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
020570  013777  001100  160354                MOV     $TSTNM,@DISPLAY ;&& DISPLAY TEST NUMBER AND ERROR FLAG
                                              ;TST    @SWR            ;&& HALT ON ERROR = 1?\
                                              ;BPL    8$              ;;.BRANCH IF NO         >(REMOVED FOR CKKUAB)
                                              ;HALT                   ;;YES--HALT            /
020576  032777  002000  160344    8$:         BIT     #BIT10,@SWR     ;&& BELL ON ERROR?
020604  001402                                BEQ     1$              ;;NO - SKIP
020606  104400  001214                        TYPE    ,$BELL          ;;RING BELL
020612  005237  001110          1$:           INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
020616  011637  001114                        MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
020622  162737  000002  001114                SUB     #2,$ERRPC
020630  117737  160260  001112                MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
020636  032777  020000  160304                BIT     #BIT13,@SWR     ;&& SKIP TYPEOUT IF SET
020644  001017                                BNE     2$              ;;SKIP TYPEOUTS
020646  004737  002012                        JSR     PC,ERTYPE       ;;GO TO USER ERROR ROUTINE
020652  104400  001221                        TYPE    ,$CRLF
020656  122737  000001  020034                CMPB    #1,$ENV         ;&& IS THIS APT?
020664  001007                                BNE     2$              ;&& NO
020666  113737  001112  020700                MOVB    $ITEMB,21$      ;&&
020674  004737  021056                        JSR     PC,$ATY4        ;&&
020700     000          21$:                 .BYTE   0               ;&&
020701     000                               .BYTE   0               ;&&
020702  000777          22$:                 BR      22$             ;&&
020704  005777  160240          2$:           TST     @SWR            ;&& HALT ON ERROR
020710  100007                                BPL     9$              ;;SKIP IF CONTINUE
020712  032777  000200  160174                BIT     #BIT7,@$ERRPC   ;&& LOOK AT ERROR AND SEE IF A +2XX ERROR
020720  001402                                BEQ     25$             ;&& BRANCH IF NOT TO HALT
020722  005037  001302                        CLR     ERRCNT          ;&& CLEAR ERROR COUNT SO RESTART PRINTS ANOTHER HEADER
```

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

```
020752  013716  001106                  MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
020756  005737  001212          4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
020762  001402                          BEQ     5$              ;;BR IF NONE
020764  013716  001212                  MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
020770                          5$:
020770  032777  001000  160152          BIT     #SW9,@SWR       ;ARE WE LOOPING ON THIS ERROR?
020776  001417                          BEQ     1000$           ;BRANCH IF NOT
021000  012737  177777  177766          MOV     #-1,CPUERR      ;CLEAR CPU ERROR REGISTER
021006  042737  177776  177572          BIC     #177776,MMR0    ;CLEAR MEMORY MANAGEMENT STATUS REGISTER
021014  012737  177777  004412          MOV     #-1,TOFLAG      ;INITIALIZE TRAP FLAG
021022  012737  177777  004604          MOV     #-1,CPFLAG      ;INITIALIZE CP TRAP FLAG
021030  012737  177777  004746          MOV     #-1,MMFLAG      ;INITIALIZE MEMORY MANAGEMENT TRAP FLAG
021036  000002                  1000$:  RTI                     ;RETURN TO TEST
```

```
2859                                    .SBTTL  APT COMMUNICATIONS ROUTINE
                                        ;;*****************************************************************
021040  112737  000001  021304  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
021046  112737  000001  021302  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
021054  000403                        BR      $ATYC
021056  112737  000001  021304  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
021064                          $ATYC:
021064  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
021066  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
021070  105737  021302                TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
021074  001450                        BEQ     5$              ;;IF NOT:  BR
021076  122737  000001  020034        CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
021104  001031                        BNE     3$              ;;IF NOT:  BR
021106  132737  000100  020035        BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
021114  001425                        BEQ     3$              ;;IF NOT:  BR
021116  017600  000004                MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
021122  062766  000002  000004        ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
021130  005737  020014          1$:   TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
021134  001375                        BNE     1$              ;;IF NOT:  WAIT
021136  010037  020030                MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
021142  105720                  2$:   TSTB    (R0)+           ;;FIND END OF MESSAGE
021144  001376                        BNE     2$
021146  163700  020030                SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
021152  006200                        ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
021154  010037  020032                MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
021160  012737  000004  020014        MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
021166  000413                        BR      5$
021170  017637  000004  021214  3$:   MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
021176  062766  000002  000004        ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
021204  013746  177776                MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
021210  004737  002674                JSR     PC,$TYPE        ;;CALL TYPE MACRO
021214  000000                  4$:   .WORD   0
021216                          5$:
021216  105737  021304          10$:  TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
021222  001416                        BEQ     12$             ;;IF NOT:  BR
021224  005737  020034                TST     $ENV            ;;RUNNING UNDER APT?
021230  001413                        BEQ     12$             ;;IF NOT:  BR
021232  005737  020014          11$:  TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
021236  001375                        BNE     11$             ;;IF NOT:  WAIT
021240  017637  000004  020016        MOV     @4(SP),$FATAL   ;;GET ERROR #
021246  062766  000002  000004        ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
021254  005237  020014                INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
021260  105037  021304          12$:  CLRB    $FFLG           ;;CLEAR FATAL FLAG
021264  105037  021303                CLRB    $LFLG           ;;CLEAR LOG FLAG
021270  105037  021302                CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
021274  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
021276  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
021300  000207                        RTS     PC              ;;RETURN
021302     000                $MFLG:  .BYTE   0               ;;MESSG. FLAG
021303     000                $LFLG:  .BYTE   0               ;;LOG FLAG
021304     000                $FFLG:  .BYTE   0               ;;FATAL FLAG
                                      .EVEN
            000200          APTSIZE=200
            000001          APTENV=001
            000100          APTSPOOL=100
            000040          APTCSUP=040
```

```
   2861                                 .SBTTL  END OF PASS ROUTINE
                                        ;;*****************************************************************
                                        ;*INCREMENT THE PASS NUMBER ($PASS)
                                        ;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
                                        ;*TYPE "END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYY"
                                        ;*WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS
                                        ;*IF SW12=1 INHIBIT TRACE TRAP
                                        ;*IF THERES A MONITOR GO TO IT
                                        ;*IF THERE ISN'T JUMP TO LOOP
   021306                               $EOP:
   021306  000004                               SCOPE                   ;LOOP ON LAST TEST
   021310  005037  177572                       CLR     MMR0            ;TURN OFF FULL RELOCATION
   021314  005037  172516                       CLR     MMR3            ;DISABLE THE UNIBUS MAP
   021320  104420                               TBITR                   ;RESTORE THE T BIT IF IT WAS ON
   021322  005037  001100                       CLR     $TSTNM          ;;ZERO THE TEST NUMBER
   021326  005037  001210                       CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
   021332  005237  020022                       INC     $PASS           ;;INCREMENT THE PASS NUMBER
   021336  042737  100000  020022               BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
   021344  005327                               DEC     (PC)+           ;;LOOP?
   021346  000001                      $EOPCT:  .WORD   1
   021350  003075                               BGT     $DOAGN          ;;YES
   021352  012737                               MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
   021354  000001                      $ENDCT:  .WORD   1
   021356  021346                               $EOPCT
   021360                               $NEXT:
   021360  104400  021366                       TYPE    ,65$            ;;TYPE ASCIZ STRING
   021364  000407                               BR      64$             ;;GET OVER THE ASCIZ
   021366     015     012      105      65$:    .ASCIZ  <15><12>/END PASS #/
                                        .EVEN
   021404                               64$:
   021404  013746  020022                       MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
                                                                        ;;TYPE PASS NUMBER
   021410  104410                               TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
   021412  005737  001110                       TST     $ERTTL          ;! SEE IF THERE WERE ANY ERRORS THIS PASS
   021416  001427                               BEQ     70$             ;! BRANCH OVER ERROR REPORT IF NONE
   021420  104400  021426                       TYPE    ,67$            ;;TYPE ASCIZ STRING
   021424  000421                               BR      66$             ;;GET OVER THE ASCIZ
   021426     040     040      124      67$:    .ASCIZ  /  TOTAL ERRORS SINCE LAST REPORT /
                                        .EVEN
   021470                               66$:
   021470  013746  001110                       MOV     $ERTTL,-(SP)    ;;SAVE $ERTTL FOR TYPEOUT
                                                                        ;;TOTAL NUMBER OF ERRORS
   021474  104410                               TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
   021476  104400  001221               70$:    TYPE    ,$CRLF          ;;TYPE CARRIAGE RETURN, LINE FEED
   021502  005037  001110                       CLR     $ERTTL          ;;CLEAR ERROR TOTAL
   021506  013700  000042               $GET42:  MOV     @#42,R0         ;;GET MONITOR ADDRESS
   021512  001414                               BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
   021514  005046                               CLR     -(SP)           ;;INSURE THE "T" BIT IS CLEAR
   021516  012746  021524                       MOV     #$CLR.T,-(SP)   ;;SETUP FOR AN RTI OR RTT
   021522  000427                               BR      $RTRN           ;;GO DO AN RTI OR RTT TO LOAD THE PSW
                                                                        ;;WITH A CLEARED "T" BIT
```

```
021524                  $CLR.T:
021524  013700  000042          MOV     @#42,R0         ;;INSURE R0 CONTAINS THE MONITORS                SEQ 0144
021530  001405                  BEQ     $DOACN          ;;RETURN ADDRESS
021532  000005                  RESET                   ;;CLEAR THE WORLD
021534  004710          $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
021536  000240                  NOP                     ;;SAVE ROOM
```

```
021540  000240                          NOP                     ;;FI
021542  000240                          NOP                     ;;ACT11
021544                          $DOAGN:
021544  013746  177776                  MOV     @#PS,-(SP)      ;;PUT THE PS ON THE STACK AND
021550  042716  000020                  BIC     #20,(SP)        ;;CLEAR THE "T" BIT
021554  032777  010000  157366          BIT     #BIT12,@SWR     ;;& RUN WITH TRACE TRAP?
021562  001005                          BNE     1$              ;;BR IF NO
021564  005137  021610                  COM     $TBIT           ;;IS IT TIME FOR TRACE TRAP
021570  100402                          BMI     1$              ;;BR IF NO
021572  052716  000020                  BIS     #20,(SP)        ;;SET TRACE TRAP
021576  012746  021604          1$:     MOV     #$LOOP,-(SP)    ;;JUMP TO START OF TEST
021602  000002                  $RTRN:  RTI                     ;;RETURN--THIS IS CHANGED TO
                                                                ;;AN "RTT" IF "RTT" IS A LEGAL
                                                                ;;INSTRUCTION
021604                          $LOOP:
021604  000137  010510                  JMP     @#LOOP          ;;RETURN
021610  000000                  $TBIT:  .WORD   0               ;;"T" BIT STATE INDICATOR
021612     377     377     000  $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
                                        .EVEN
```

```
                                        .SBTTL  SUBROUTINES UNIQUE TO TEST #7 ONLY
                                ;;**********************************************************************
                                ;;**********************************************************************
                                ;*
                                ;*      THIS SUBROUTINE CHECKS THE STATUS OF THE BIT POINTED TO IN FLOATR
                                ;*      IN THE LOCATION POINTED TO BY R5 (EITHER $DDW0 OR $DDW1), AND DETER-
                                ;*      MINES IF THE LOCATION SHOULD BE DISABLED.
2870 021616  105737  020034    DSABLD: TSTB    $ENV            ;TEST APT STATUS
2871 021622  001411                    BEQ     1$              ;BRANCH IF NOT APT
2872 021624  105737  020035            TSTB    $ENVM           ;DOES APT SAY TO SIZE
2873 021630  100006                    BPL     1$              ;BRANCH TO EXIT IF NOT
2874 021632  033715  006154            BIT     FLOATR,(R5)     ;TEST DISABLE STATUS
2875 021636  001403                    BEQ     1$              ,BRANCH IF IT SHOULD BE DISABLED
2876 021640  011537  001174            MOV     (R5),$TMP1      ;MOVE CONTENTS OF DEVICE DESCRIPTOR WORD TO $TMP1
2877 021644  000402                    BR      2$              ;BRANCH OVER RETURN CORRECTION
2878 021646  062716  000002    1$:     ADD     #2,(SP)         ;CHOCOLATE FUDGE RETURN OVER ERROR CALL
2879 021652  000207            2$:     RTS     PC              ;RETURN WITHOUT CORRECTING STACK SO ERROR WILL CALL
2880
2881                            ;;**********************************************************************
2882                            ;*      THIS SUBROUTINE CHECKS THE STATUS OF THE BIT POINTED TO IN FLOATR
2883                            ;*      IN THE LOCATION POINTED TO BY R5 (EITHER $DDW0 OR $DDW1), AND DETER-
2884                            ;*      MINES IF THE LOCATION SHOULD BE ENABLED.
2885 021654  105737  020034    ENABLD: TSTB    $ENV            ;TEST APT STATUS
2886 021660  001411                    BEQ     1$              ;BRANCH IF NOT APT
2887 021662  105737  020035            TSTB    $ENVM           ;DOES APT SAY TO SIZE
2888 021666  100006                    BPL     1$              ;BRANCH TO EXIT IF NOT
2889 021670  033715  006154            BIT     FLOATR,(R5)     ;TEST DISABLE STATUS
2890 021674  001003                    BNE     1$              ;BRANCH IF IT SHOULD BE ENABLED
2891 021676  011537  001174            MOV     (R5),$TMP1      ;MOVE CONTENTS OF DEVICE DESCRIPTOR WORD TO $TMP1
2892 021702  000402                    BR      2$              ;BRANCH OVER RETURN CORRECTION
2893 021704  062716  000002    1$:     ADD     #2,(SP)         ;VANILLA FUDGE RETURN OVER ERROR CALL
2894 021710  000207            2$:     RTS     PC              ;RETURN WITHOUT CORRECTING STACK SO ERROR WILL CALL
2895                            ;;**********************************************************************
2896                            ;;**********************************************************************
```

```
2898                                    .SBTTL  ASCII AND ERROR MESSAGES
2899            000002                  .RADIX  2               ;THIS ENABLES YOU TO SEE THE BIT PATTERNS BELOW
2900 021712    177777                   PATRNS: .WORD   1111111111111111        ;ALL BITS SET
2901 021714    000000                           .WORD   0000000000000000        ;ALL BITS CLEAR
2902 021716    125252                           .WORD   1010101010101010        ;ODD BITS SET, EVEN BITS CLEAR
2903 021720    052525                           .WORD   0101010101010101        ;EVEN BITS SET, ODD BITS CLEAR
2904 021722    031463                           .WORD   0011001100110011        ;ALTERNATING PAIRS OF BITS SET
2905 021724    007417                           .WORD   0000111100001111        ;ALTERNATING GROUPS OF 4 BITS SET
2906 021726    000377                           .WORD   0000000011111111        ;LOWER BYTE SET, UPPER BYTE CLEAR
2907            000010                  .PADIX  8               ;THIS RETURNS MODE BACK TO OCTAL
2908 021730       000      000      000 DFMSG:  .BYTE   0,0,0,0,0       ;ALL NUMBERS ARE TO BE PRINTED IN OCTAL
2909 021735       200      125      116 UBMAJA: .ASCIZ  <CRLF>?UNIBUS MEMORY AVAILABLE = ?
2910 021771       040      113      200 UBMEND: .ASCIZ  ? K?<CRLF>
```

```
2911 021775  104  111  10*  CPUMSG: .ASCIZ  ?DIAGNOSTIC HAS DETERMINED THAT CPU IS AN 11/?
2912 022052  062  064  200  ELEV24: .ASCIZ  ?24?<CRLF>
2913 022056  064  064  200  ELEV44: .ASCIZ  ?44?<CRLF>
2914 022062  116  117  124  EM1:    .ASCIZ  ?NOT THE CORRECT TRAP CONDITION THROUGH ERRVEC (#004)?
2915 022147  125  116  105  EM2:    .ASCIZ  ?UNEXPECTED CPU TRAP THROUGH ERRVEC (#004)?
2916 022221  125  116  105  EM3:    .ASCIZ  ?UNEXPECTED MEMORY MANAGEMENT TRAP, MEMORY MANAGEMENT STATUS REGISTERS?
2917 022327  123  125  115  EM4:    .ASCIZ  ?SUMMARY OF MAP REGISTERS THAT TIMED OUT ON READ?
2918 022407  123  125  115  EM5:    .ASCIZ  ?SUMMARY OF DUAL ADDRESSING ERRORS ON LOADING MAP REGISTERS?
2919 022502  123  125  115  EM6:    .ASCIZ  ?SUMMARY OF BIT PATTERN FAILURES IN LOWER 16 BITS OF MAP REGISTERS?
2920 022604  123  125  115  EM7:    .ASCIZ  ?SUMMARY OF BIT PATTERN FAILURES IN UPPER 6 BITS OF MAP REGISTERS?
2921 022705  103  101  116  EM10:   .ASCII  ?CAN'T GET TO MAIN MEMORY FROM UNIBUS WITH THE MAP OFF?<CRLF>
2922 022773  123  117  040          .ASCIZ  ?SO I'LL JUMP TO THE SIZE JUMPER TEST FOR VERIFICATION?
2923 023061  123  125  115  EM11:   .ASCIZ  ?SUMMARY OF COUNT PATTERN FAILURES ON THE UNIBUS DATA PATH?
2924 023153  125  116  111  EM12:   .ASCIZ  ?UNIBUS MAP IS RELOCATING WHEN NOT ENABLED?
2925 023225  103  101  116  EM13:   .ASCII  ?CANNOT USE ANY OF THE MAP REGISTERS OR PHYSICAL?<CRLF>
2926 023305  101  104  104          .ASCII  ?ADDRESS BIT14 IS STUCK LOW, MUST RESTART PROGRAM?<CRLF>
2927 023366  111  106  040          .ASCIZ  ?IF YOU DON'T LOOP ON THIS PROBLEM.?
2928 023431  123  125  115  EM14:   .ASCIZ  ?SUMMARY OF UNIBUS ADDRESS ERRORS, WITH THE MAP RELOCATION DISABLED?
2929 023534  115  101  111  EM15:   .ASCIZ  ?MAIN MEMORY TIMEOUT OVER THE UNIBUS DID NOT OCCUR PROPERLY?
2930 023627  122  105  114  EM16:   .ASCIZ  ?RELOCATION THROUGH THE MAP WAS NOT CORRECT, CARRY PROPAGATION?
2931 023725  116  117  040  EM17:   .ASCIZ  ?NO UNIBUS MEMORY EXISTS?
2932 023755  111  116  124  EM20:   .ASCIZ  ?INTERRUPT/ABORT LOGIC TESTS TRAP TO LOCATION 114 DID NOT OCCUR?
2933 024054  111  116  124  EM21:   .ASCIZ  ?INTERRUPT/ABORT TESTS R4 WAS OVERWRITTEN WITH?
2934 024132  111  116  124  EM22:   .ASCIZ  ?INTERRUPT/ABORT TESTS TRAP DID NOT OCCUR DUE TO ABORT?
2935 024220  114  115  101  FM23:   .ASCIZ  ?LMA NOT LOADED PROPERLY?
2936 024250  114  115  101  EM24:   .ASCIZ  ?LMA FORCE JUMPER BIT NOT ZERO?
2937 024306  114  115  101  EM25:   .ASCIZ  ?LMA FORCE JUMPER BIT NOT SET?
2938 024343  114  115  101  EM26:   .ASCIZ  ?LMA CONTROL BITS INCORRECT?
2939 024376  106  117  122  EM27:   .ASCIZ  ?FORCE JUMPER BIT FAILS TO REVERT MAP REGISTER STATUS TO DEFAULT?
2940 024476  113  111  120  EM30:   .ASCIZ  ?KIPAR5 NOT LOADED PROPERLY - SKIPPING NEXT TEST?
2941 024556  124  110  105  EM201:  .ASCIZ  ?THE FOLLOWING REGISTERS TIMED OUT WHEN READ?
2942 024632  124  110  105  EM202:  .ASCIZ  ?THE FOLLOWING ARE DUAL ADDRESSING ERRORS IN THE UNIBUS MAP?
2943 024725  124  110  105  EM203:  .ASCIZ  ?THE BIT PATTERN THROUGH THE MAP REGISTERS FAILED?
2944 025006  125  116  111  EM204:  .ASCIZ  ?UNIBUS DATA PATH COUNT PATTERN FAILURE?
2945 025055  125  116  111  EM205:  .ASCIZ  ?UNIBUS ADDRESSING ERRORS, MAP RELOCATION DISABLED?
2946 025137  104  101  124  EM206:  .ASCIZ  ?DATA PATTERN NOT CORRECT?
2947 025170  122  105  106  EM207:  .ASCIZ  ?REFERENCED MAP REGISTER 0 WITH ADDRESS ONE BIT DIFFERENT THAN 17770200?
2948 025277  115  101  120  EM210:  .ASCIZ  ?MAP REGISTER(S) UNDER TEST DID NOT RESPOND IN DUAL MAPPING TEST?
2949 025377  122  105  114  EM211:  .ASCII  ?RELOCATION THROUGH THE MAP WAS NOT CORRECT, CARRY PROPAGATION?<CRLF>
2950 025475  124  105  123          .ASCIZ  ?TEST CODE BEING RUN OVER UNIBUS?
2951 025535  115  101  111  EM212:  .ASCII  ?MAIN MEMORY TIMEOUT OVER THE UNIBUS DID NOT OCCUR PROPERLY?<CRLF>
2952 025630  124  105  123          .ASCIZ  ?TEST CODE BEING RUN OVER UNIBUS?
2953 025670  115  101  120  EM213:  .ASCIZ  ?MAP REGISTER ENABLED WHEN DDW SAYS IT SHOULD BE DISABLED?
2954 025761  115  101  120  EM214:  .ASCIZ  ?MAP REGISTER DISABLED WHEN DDW SAYS IT SHOULD BE ENABLED?
```

```
                                                .SBTTL  DATA HEADERS
026052  122  105  103  DH1:    .ASCIZ  ?RECEIVD EXPECTD TESTNO  ERR PC?
026111  122  105  103  DH2:    .ASCIZ  ?RECEIVD TESTNO  ERR PC?
026140  123  124  101  DH3:    .ASCII  ?STATUS  AUTOI/D VIRTUAL?<CRLF>
026170  122  105  107          .ASCIZ  ?REGISTR REGISTR ADDRESS TESTNO  ERR PC?
026237  122  105  107  DH4:    .ASCII  ?REGADRS   REGADRS?<CRLF>
026261  040  042  117          .ASCIZ  ? "OR"      "AND"    #ERRORS TESTNO  ERR PC?
026334  122  105  107  DH5:    .ASCII  ?REGLOAD   REGLOAD   REGDUAL   REGDUAL?<CRLF>
026402  040  042  117          .ASCIZ  ? "OR"      "AND"    "OR"      "AND"    #ERRORS TESTNO?
026471  115  101  120  DH6:    .ASCII  ?MAPREG    MAPREG    EXPECTD EXPECTD RECEIVD RECEIVD?<CRLF>
026555  040  042  117          .ASCIZ  ? "OR"      "AND"    "OR"    "AND"   "OR"    "AND"  #ERRORS TESTNO?
026660  124  105  123  DH10:   .ASCIZ  ?TESTNO  ERR PC?
026677  105  130  120  DH11:   .ASCII  ?EXPECTD   EXPECTD   RECEIVD   RECEIVD?<CRLF>
026745  040  042  117          .ASCIZ  ? "OR"      "AND"    "OR"      "AND"    #ERRORS TESTNO?
027034  103  117  116  DH15:   .ASCII  ?CONDITN CONDITN?<CRLF>
027054  105  130  120          .ASCIZ  ?EXPECTD RECEIVD TESTNO  ERR PC?
027113  124  105  123  DH23:   .ASCIZ  ?TESTNO  ERR PC  LMAEXP    LMARCV?
027154  124  105  123  DH24:   .ASCIZ  ?TESTNO  ERR PC  LMAEXP  LMARCV?
027213  124  105  123  DH27:   .ASCIZ  ?TESTNO  ERR PC  LMARCV  KIPAR4?
027252  124  105  123  DH30:   .ASCIZ  ?TESTNO  ERR PC  PR5EXP  PR5RCV?
027311  122  105  107  DH201:  .ASCIZ  ?REGADRS TESTNO  ERR PC?
027340  115  101  120  DH202:  .ASCII  ?MAPREG    MAPREG    NON-ZER?<CRLF>
027374  124  105  123          .ASCIZ  ?TESTING   DUALED    CONTNTS TESTNO  ERR PC?
027447  122  105  107  DH203:  .ASCIZ  ?REGADRS   PATTRN  EXPCTD  RECEVD  TESTNO  ERR PC?
027530  105  130  120  DH204:  .ASCIZ  ?EXPECTD RECEIVD ADDRSLOAD TESTNO  ERR PC?
027601  101  104  104  DH205:  .ASCII  ?ADDRESS   ADDRESS?<CRLF>
027623  105  130  120          .ASCIZ  ?EXPECTD   RECEIVD   TESTNO  ERR PC?
027666  101  104  104  DH206:  .ASCIZ  ?ADDRESS  EXPCTD  RECVED  TESTNO  ERR PC?
027736  101  104  104  DH207:  .ASCIZ  ?ADDRUSED  BITDIFF TESTNO  ERR PC?
027777  124  105  123  DH210:  .ASCIZ  ?TESTNO  ERR PC  MAPREGADR?
030031  103  117  122  DH211:  .ASCII  ?CORRECT   EXPECTD RECEIVD?<CRLF>
030063  101  104  104          .ASCIZ  ?ADDRESS   DATA    FROM UB TESTNO  ERR PC?
030134  103  117  116  DH212:  .ASCII  ?CONDITN CONDITN?<CRLF>
030154  105  130  120          .ASCIZ  ?EXPECTD RECEIVD TESTNO  ERR PC?
030213  124  105  123  DH213:  .ASCIZ  ?TESTNO  ERR PC  REG NO  DDWDAT  DDWADR?
                                .EVEN
```

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

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

```
ABASE = 000000          BIT0  = 000001          DF210   031005          DT5     030334          HITMIS= 177752
ACDW1 = 000000          BIT00 = 000001          DF23    030766          DT6     030352          HT    = 000011
ACDW2 = 000000          BIT01 = 000002          DF4     030727          EADRES  006144          IOTVEC= 000020
ACPUOP= 000000          BIT02 = 000004          DF5     030734          EADRS2  006150          JMPMSG  006422
ADDROR  001234          BIT03 = 000010          DF6     030742          ELEV24  022052          KDPAR0= 172360
ADDW0 = 177777          BIT04 = 000020          DH1     026052          ELEV44  022056          KDPAR1= 172362
ADDW1 = 177777          BIT05 = 000040          DH10    026660          EMTVEC= 000030          KDPAR2= 172364
ADDW10= 000000          BIT06 = 000100          DH11    026677          EM1     022062          KDPAR3= 172366
ADDW11= 000000          BIT07 = 000200          DH15    027034          EM10    022705          KDPAR4= 172370
ADDW12= 000000          BIT08 = 000400          DH2     026111          EM11    023061          KDPAR5= 172372
ADDW13= 000000          BIT09 = 001000          DH201   027311          EM12    023153          KDPAR6= 172374
ADDW14= 000000          BIT1  = 000002          DH202   027340          EM13    023225          KDPAR7= 172376
ADDW15= 000000          BIT10 = 002000          DH203   027447          EM14    023431          KDPDR0= 172320
ADDW2 = 000000          BIT11 = 004000          DH204   027530          EM15    023534          KDPDR1= 172322
ADDW3 = 000000          BIT12 = 010000          DH205   027601          EM16    023627          KDPDR2= 172324
ADDW4 = 000000          BIT13 = 020000          DH206   027666          EM17    023725          KDPDR3= 172326
ADDW5 = 000000          BIT14 = 040000          DH207   027736          EM2     022147          KDPDR4= 172330
ADDW6 = 000000          BIT15 = 100000          DH210   027777          EM20    023755          KDPDR5= 172332
ADDW7 = 000000          BIT2  = 000004          DH211   030031          EM201   024556          KDPDR6= 172334
ADDW8 = 000000          BIT3  = 000010          DH212   030134          EM202   024632          KDPDR7= 172336
ADDW9 = 000000          BIT4  = 000020          DH213   030213          EM203   024725          KERSTK= 001100
ADEVCT= 000000          BIT5  = 000040          DH23    027113          EM204   025006          KIPAR0= 172340
ADEVM = 000000          BIT6  = 000100          DH24    027154          EM205   025055          KIPAR1= 172342
ADRAND  001230          BIT7  = 000200          DH27    027213          EM206   025137          KIPAR2= 172344
ADREXT  005250          BIT8  = 000400          DH3     026140          EM207   025170          KIPAR3= 172346
AENV  = 000000          BIT9  = 001000          DH30    027252          EM21    024054          KIPAR4= 172350
AENVM = 000000          BPTVEC= 000014          DH4     026237          EM210   025277          KIPAR5= 172352
AFATAL= 000000          BUPWIN  001274          DH5     026334          EM211   025377          KIPAR6= 172354
AMADR1= 000000          CACHE = 177746          DH6     026471          EM212   025535          KIPAR7= 172356
AMADR2= 000000          CACHVE= 000114          DISPLA  001152          EM213   025670          KIPDR0= 172300
AMADR3= 000000          CHARCT  006140          DSABLD  021616          EM214   025761          KIPDR1= 172302
AMADR4= 000000          CLRMAP  004352          DTMS    006120          EM22    024132          KIPDR2= 172304
AMAMS1= 000000          CMPE  = 177744          DTMSG   006124          EM23    024220          KIPDR3= 172306
AMAMS2= 000000          CNTR    001304          DT1     030262          EM24    024250          KIPDR4= 172310
AMAMS3= 000000          CONTRL= 177746          DT10    030374          EM25    024306          KIPDR5= 172312
AMAMS4= 000000          CPFLAG  004604          DT11    030402          EM26    024343          KIPDR6= 172314
AMSGAD= 000000          CPUER   004602          DT14    030420          EM27    024376          KIPDR7= 172316
AMSGLG= 000000          CPUERR= 177766          DT15    030436          EM3     022221          KSP   =%000006
AMSGTY= 000000          CPUEXP  001312          DT2     030274          EM30    024476          LF    = 000012
AMTYP1= 000000          CPUMSG  021775          DT201   030532          EM4     022327          LMAHI = 177736
AMTYP2= 000000          CPUTYP  007774          DT202   030542          EM5     022407          LMALOW= 177734
AMTYP3= 000000          CR    = 000015          DT203   030556          EM6     022502          LOADRS= 177740
AMTYP4= 000000          CRLF  = 000200          DT204   030574          EM7     022604          LOOP    010510
APASS = 000000          CTRAPS= 000116          DT205   030610          ENABLD  021654          LOWEST  001254
APRIOR= 000000          CTRAPV= 000114          DT206   030622          ERRCNT  001302          LREGL   001276
APTCSU= 000040          DATA    001350          DT207   030636          ERROR = 104000          LREGU   001300
APTENV= 000001          DATAND  001240          DT210   030650          ERRVEC= 000004          MAINT = 177750
APTSIZ= 000200          DATAOR  001244          DT211   030660          ERTYPE  002012          MAPADD  005404
APTSPO= 000100          DATEXT  005200          DT212   030674          ER200   001652          MAPH0 = 170202
ASWREG= 000000          DATO  = 100000          DT213   030706          EXTOUT  006142          MAPH00- 170202
ATESTN= 000000          DATOB = 140000          DT23    030450          FJBIT = 000100          MAPH01- 170206
```

AUNIT	= 000000	DFMSG	021730	DT24	030462	FLAG	001306	MAPH02=	170212		
AUSWR	= 000000	DF1	030722	DT26	030474	FLOATR	006154	MAPH03=	170216		SEQ 0153
AVECT1	= 000000	DF11	030752	DT27	030506	FTTHRU	005120	MAPH04=	170222		
AVECT2	= 000000	DF14	030760	DT3	030304	GMRMOD	007702	MAPH05=	170226		
BADCPU	007446	DF201	030772	DT30	030520	HIADRS	= 177742	MAPH06=	170232		
BADPC	001324	DF204	031000	DT4	030320	HIGEST	001256	MAPH07=	170236		

SYMBOL TABLE

SEQ 0154

```
MAPH1 = 170206          MAPL27= 170334          PR5   = 000240          SIZELO= 177760          TST1    010600
MAPH10= 170242          MAPL3 = 170214          PR6   = 000300          SPECST  006162          TST10   013440
MAPH11= 170246          MAPL30= 170340          PR7   = 000340          SPSUPP  002413          TST11   014130
MAPH12= 170252          MAPL31= 170344          PS    = 177776          SR0   = 177572          TST12   014362
MAPH13= 170256          MAPL32= 170350          PSW   = 177776          SR1   = 177574          TST13   014544
MAPH14= 170262          MAPL33= 170354          PWRMSG  004122          SR2   = 177576          TST14   014574
MAPH15= 170266          MAPL34= 170360          PWRVEC= 000024          SR3   = 172516          TST15   014674
MAPH16= 170272          MAPL35= 170364          RELC22  011600          SSP   =%000006          TST16   014722
MAPH17= 170276          MAPL36= 170370          RESREG= 104414          STACK = 001100          TST17   014760
MAPH2 = 170212          MAPL37= 170374          RESVEC= 000010          START   010000          TST2    011022
MAPH20= 170302          MAPL4 = 170220          RETRY   001344          STKLMT= 177774          TST20   015010
MAPH21= 170306          MAPL5 = 170224          RSIZE   001342          SUPSTK= 000700          TST21   015352
MAPH22= 170312          MAPL6 = 170230          R10   =%000000          SWR     001150          TST22   015706
MAPH23= 170316          MAPL7 = 170234          R11   =%000001          SW0   = 000001          TST23   016112
MAPH24= 170320          MASK1   006156          R12   =%000002          SW00  = 000001          TST24   016152
MAPH25= 170326          MASK2   006160          R13   =%000003          SW01  = 000002          TST25   016352
MAPH26= 170332          MEMERR= 177744          R14   =%000004          SW02  = 000004          TST26   016416
MAPH27= 170336          MFPT  = 000007          R15   =%000005          SW03  = 000010          TST27   016616
MAPH3 = 170216          MMFLAG  004746          R3STAK  005530          SW04  = 000020          TST3    011400
MAPH30= 170342          MMRHI   001266          R6    =%000006          SW05  = 000040          TST30   017010
MAPH31= 170346          MMRLOW  001264          R7    =%000007          SW06  = 000100          TST31   017202
MAPH32= 170352          MMR0  = 177572          SAVREG= 104412          SW07  = 000200          TST4    011710
MAPH33= 170356          MMR1  = 177574          SCOPE = 000004          SW08  = 000400          TST5    012130
MAPH34= 170362          MMR2  = 177576          SDPAR0= 172260          SW09  = 001000          TST6    012470
MAPH35= 170366          MMR3  = 172516          SDPAR1= 172262          SW1   = 000002          TST7    012604
MAPH36= 170372          MMTOTM  014420          SDPAR2= 172264          SW10  = 002000          TYPDAT= 002214
MAPH37= 170376          MMTRAP  004744          SDPAR3= 172266          SW11  = 004000          TYPDS = 104410
MAPH4 = 170222          MMVEC = 000250          SDPAR4= 172270          SW12  = 010000          TYPE  = 104400
MAPH5 = 170226          MRQUES  007613          SDPAR5= 172272          SW13  = 020000          TYPOC = 104402
MAPH6 = 170232          NEXMEM= 000040          SDPAR6= 172274          SW14  = 040000          TYPON = 104406
MAPH7 = 170236          NEXT    005140          SDPAR7= 172276          SW15  = 100000          TYPOS = 104404
MAPL0 = 170200          NOMORE  007260          SDPDR0= 172220          SW2   = 000004          TYPVAD  002414
MAPLU0= 170200          NXTTST  001346          SDPDR1= 172222          SW3   = 000010          UBADDR  002522
MAPL01= 170204          OLDPC   001326          SDPDR2= 172224          SW4   = 000020          UBMAVA  021735
MAPL02= 170210          OLDPS   001330          SDPDR3= 172226          SW5   = 000040          UBMEND  021771
MAPL03= 170214          OLDPSW  001332          SDPDR4= 172230          SW6   = 000100          UBMHI   001262
MAPL04= 170220          PADRSH  001226          SDPDR5= 172232          SW7   = 000200          UBMLOW  001260
MAPL05= 170224          PADRSL  001224          SDPDR6= 172234          SW8   = 000400          UBRHI   001272
MAPL06= 170230          PATAND  001250          SDPDR7= 172236          SW9   = 001000          UBRLOW  001270
MAPL07= 170234          PATEXT  005224          SIPAR0= 172240          SYSTID= 177764          UDPAR0= 177660
MAPL1 = 170204          PATRNS  021712          SIPAR1= 172242          TBIT  = 000020          UDPAR1= 177662
MAPL10= 170240          PATTOR  001252          SIPAR2= 172244          TBITO = 104416          UDPAR2= 177664
MAPL11= 170244          PCONTR  001320          SIPAR3= 172246          TBITOF  004306          UDPAR3= 177666
MAPL12= 170250          PCPUER  001314          SIPAR4= 172250          TBITR = 104420          UDPAR4= 177670
MAPL13= 170254          PIRQ  = 177772          SIPAR5= 172252          TBITRE  004334          UDPAR5= 177672
MAPL14= 170260          PIRQVE= 000240          SIPAR6= 172254          TBITVE= 000014          UDPAR6= 177674
MAPL15= 170264          PMAINT  001322          SIPAR7= 172256          TCPMRA  005632          UDPAR7= 177676
MAPL16= 170270          PMMRU   001334          SIPDR0= 172200          TESTNO  001310          UDPDR0= 177620
MAPL17= 170274          PMMR1   001336          SIPDR1= 172202          TIMEOU  004410          UDPDR1= 177622
MAPL2 = 170210          PMMR2   001340          SIPDR2= 172204          TIMOUT= 000020          UDPDR2= 177624
MAPL20= 170300          PPARER  001316          SIPDR3= 172206          TKVEC = 000060          UDPDR3= 177626
```

```
MAPL21= 170304          PRETST  017662          SIPDR4= 172210          TOFLAG  004412          UDPDR4= 177630
MAPL22= 170310          PR0     = 000000          SIPDR5= 172212          TOMSG   006262          UDPDR5= 177632
MAPL23= 170314          PR1     = 000040          SIPDR6= 172214          TPVEC   = 000064          UDPDR6= 177634
MAPL24= 170320          PR2     = 000100          SIPDR7= 172216          TRAPVE= 000034          UDPDR7= 177636
MAPL25= 170324          PR3     = 000140          SIZEHI= 177762          TRTVEC= 000014          UIPAR0= 177640
MAPL26= 170330          PR4     = 000200          SIZEJ   012620          TSTLOC  005046          UIPAR1= 177642
```

SEQ 0155

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

. ABS.  031010      000
        000000      001
ERRORS DETECTED:  0


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