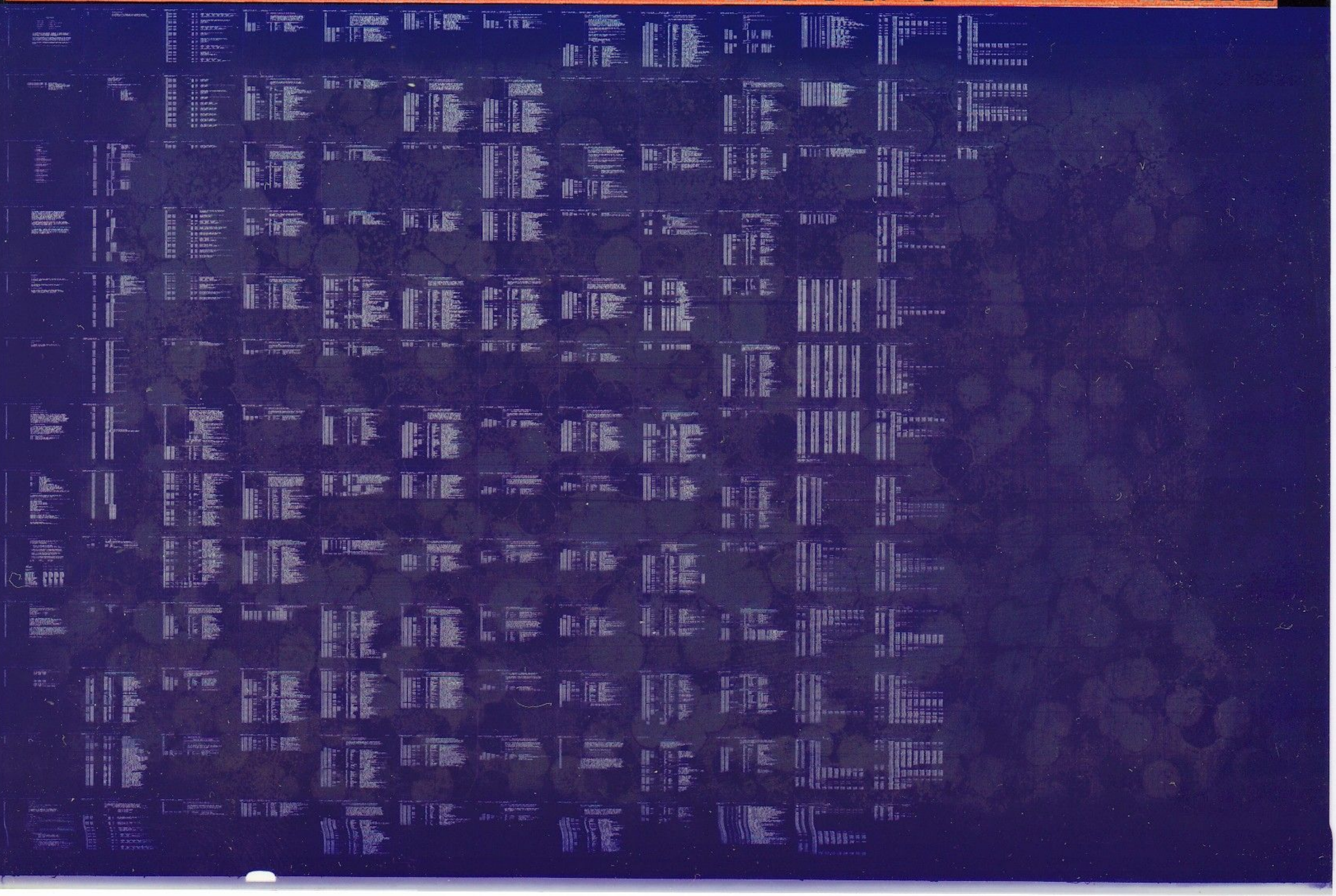
KD11-Z 11/44 UBI MAP AH-F629D-MC
CKKUADO FICHE 1 OF 1



```
1                                       .REM    a
2
3
4
5
6
7                                                       IDENTIFICATION
8                                                       --------------
9
10
11                                              PRODUCT CODE:       AC-F627D-MC
12
13                                              PRODUCT NAME:       CKKUAD0 11/24/44 UBI MAP
14
15                                              DATE CREATED:       OCTOBER, 1981
16
17                                              MAINTAINER:         DIAGNOSTIC ENGINEERING
18
19                                              AUTHOR:             DAN MILLEVILLE
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, 1982 BY DIGITAL EQUIPMENT CORPORATION
```

```
39
40
41
42                                  HISTORY SECTION
43
44
45                          CKKUAA0 WAS RELEASED OCTOBER, 1979 > INITIAL RELEASE
46                          CKKUAB0 WAS RELEASED OCTOBER, 1980 > ADDITION OF UNIBUS MEMORY TEST
47                          CKKUAC0 WAS RELEASED APRIL,   1981 > TESTING OF 11/24 MAP, 11/24 WITH UNIBUS
48                                                               MEMORY ONLY, & POWER MONITOR BIT CHECK.
49                          CKKUAD0 WAS RELEASED OCTOBER, 1981 > SETUP OF SOFTWARE/HARDWARE SWR ADDRESS
50                                                               WAS ADDED TO STARTUP CODE.
51                                                             > WHEN SELECTED, THE OPTIONAL CACHE TESTS
52                                                               NOW RUN CORRECTLY.
```

TABLE OF CONTENTS

1. ABSTRACT

THIS PROGRAM IS DESIGNED TO BE RUN ON A PDP 11/24 OR 11/44 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
INTERACTION BETWEEN MODULES PROHIBITS CLOSE ISOLATION, BUT
THE FAILING FUNCTION WILL BE CALLED OUT SO THE FIELD SERVICE
ENGINEER CAN COMPLETE THE ISOLATION PROCESS.

IF THE PROGRAM CATCHES AN ERROR IN AN EARLY TEST AND IS ALLOWED
TO CONTINUE RUNNING THROUGH THE LATER TESTS THE ERROR INDICATIONS
FROM THOSE LATER TESTS MAY BE INVALID. THIS IS DUE TO THE STRUCTURE
OF THE PROGRAM, WHICH ASSUMES THAT ALL AREAS TESTED PRIOR TO THE
CURRENT TEST ARE FUNCTIONING PROPERLY.

THE ERROR TYPE OUTS WILL BE IN TABLE FORMAT, WITH A MESSAGE INDICATING

THE CLASS OF ERROR, A HEADER IDENTIFYING EACH COLUMN AND A REPORT OF
ALL PERTINENT DATA. WHEN THE TEST CAN PRODUCE MORE THAN ONE ERROR
CONDITION, A SUMMARY OF ERRORS WILL BE GIVEN AT THE END OF THAT TEST
CONSISTING OF: THE LOGICAL 'AND' AND 'OR' OF THE DATA PREVIOUSLY
REPORTED AND THE NUMBER OF ERRORS IN THIS TEST. (SEE SECTION 6.3 FOR
AN EXAMPLE OF THE ERROR TYPEOUTS.)

2. REQUIREMENTS

2.1 EQUIPMENT

THE BASIC PDP-11/24 OR 44 COMPUTER, INCLUDING THE CPU, CACHE (11/44 ONLY), 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.
```

```
                        4.      STARTING PROCEDURE

                        4.1     STARTING ADDRESS

                                PROGRAM STARTS AT ADDRESS 200

                        4.2     PROGRAM AND/OR OPERATOR ACTION

                                THE PROGRAM WILL IDENTIFY ITSELF.  IF CPU IS AN 11/44, PROGRAM
                                WILL PROCEED WITH TESTING.  IF CPU IS AN 11/24, AND LOCATION 176
                                CONTAINS ZERO (SUCH AS JUST AFTER LOADING), PROGRAM WILL PROMPT
                                FOR A SWITCH REGISTER INPUT.  IF NONE IS REQUIRED, ENTER <CR>.
                                IF YOU DO SO, PROGRAM WILL DEPOSIT A "1" IN LOCATION 176 (MEANINGLESS
                                UNLESS BIT 8 IS ALSO SET) SO THAT SUBSEQUENT RUNNING OF THE PROGRAM
                                WILL NOT RESULT IN THE SAME REQUEST.  IF MANUFACTURING IS RUNNING
                                THIS PROGRAM ON AN 11/24 WITH UNIBUS MEMORY ONLY (NO MAIN MEMORY),
                                ENTER A NUMBER WITH AT LEAST BIT 11 SET (SWR=4000).  THIS WILL KEY
                                THE PROGRAM TO TEST THE 11/24 IN THAT CONFIGURATION.

                        4.3     SPECIAL STARTING PROCEDURE

                                IF IT APPEARS THAT THE CACHE IS CAUSING SOME TROUBLE AND
                                YOU STILL WANT TO RUN THIS PROGRAM, IT IS POSSIBLE TO RUN
                                WITH THE CACHE DISABLED.  SIMPLY LOAD THE CACHE CONTROL
                                REGISTER (17777746) WITH THE DESIRED NUMBER.  THEN LOAD
                                THE PC (17777707) WITH THE STARTING ADDRESS (200) AND
                                PRESS "CONTINUE".  THE PROGRAM WILL NOW RUN NORMALLY EXCEPT
                                THAT CERTAIN TESTS WILL BE SKIPPED SINCE THE CACHE IS DISABLED.
                                THIS FACT IS INDICATED IN THE ABSTRACT OF EACH TEST THAT
                                CHECKS THE CACHE CONTROL REGISTER.

                                DEFINITION OF THE BITS IN THE CACHE CONTROL REGISTER:
                                BIT00   -DISABLE TRAPS
                                BIT02   -FORCE MISS ON READ,WHERE ADDRESS BIT 12 IS 0
                                BIT03   -FORCE MISS ON READ,WHERE ADDRESS BIT 12 IS 1
                                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=      SET WHEN RUNNING AN 11/24 WITH NO MAIN MEMORY
                                                        AND ONLY UNIBUS MEMORY
                                        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.
                                        SW05    1=      SELECT UNIBUS MEMORY TESTS
                                                        SET WHEN A WINDOW EXISTS (SOME UBMAP JUMPERS CUT), AND
                                                        UNIBUS MEMORY OCCUPIES THE ENTIRE WINDOW ADDRESS AREA.

                                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) (SOME IN THIS SOURCE)

                                        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

                                ›.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
                                        SUBROUTINE TO DETERMINE 11/24 WITH UNIBUS MEM ONLY & CHECK LMA'S IF SO

                                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).

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.

4. ETABLE 'D' IS USED BY MANUFACTURING TO RUN AN 11/24 UNDER
   APT WITH UNIBUS MEMORY AND NO MAIN MEMORY, AND BECAUSE OF
   NO CACHE IN AN 11/24, DESELECTS THE OPTIONAL CACHE TESTS.

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

.....	E TABLES			.....
	A	B	C	D
E-MODE/S-MODE ($ENVM/$ENV)	000/000	240/001	240/001	240/001
SWITCH REGISTER 1 ($SWREG)	000100	000100	000000	004000
SWITCH REGISTER 2	000000	000000	000000	000000
CPU TYPE/OPTIONS	00/0000	00/0000	00/0000	00/0000
MEMORY MAP CODE 1	000/0000	000/0000	000/0000	000/0000
MEMORY MAP CODE 2	000/0000	000/0000	000/0000	000/0000
MEMORY MAP CODE 3	000/0000	000/0000	000/0000	000/0000
MEMORY MAP CODE 4	000/0000	000/0000	000/0000	000/0000
BUS PRIORITY/INTERRUPT 1	0000	0000	0000	0000
BUS PRIORITY/INTERRUPT 2	0000	0000	0000	0000
BASE ADDRESS CODE	000000	000000	000000	000000
DEVICE MAP CODE	000000	000000	000000	000000
CTLR. SPECIFIC WORD 1	000000	000000	000000	000000
CTLR. SPECIFIC WORD 2	000000	000000	000000	000000
DEVICE DESCRIPTOR WORD 0	177777	177777	177777	177777
DEVICE DESCRIPTOR WORD 1	177777	177777	177777	177777
DEVICE DESCRIPTOR WORD 2 THROUGH DEVICE DESCRIPTOR WORD 15	000000	000000	000000	000000

```
                                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.

                                        IF THE POWER MONITOR BIT ERROR IS CALLED, YOU MUST CORRECT THE POWER
                                        SUPPLY OR CPU ERROR REGISTER PROBLEM BEFORE YOU CAN RELY ON THE
                                        RESULTS OF THIS DIAGNOSTIC.  THE ERROR CAN BE CALLED AT 1 OF 2
                                        PLACES - 1) IN THE SCOPE ROUTINE EXECUTED AT THE BEGINNING OF EACH
                                        TEST, AND 2) IN AN ERROR CALL IN CASE BIT BECOMES SET AFTER THE SCOPE.
                                        IF THE BIT IS CAUGHT IN THE ERROR ROUTINE, *TWO* ERRORS WILL CALL -
                                        1) POWER MONITOR BIT ERROR WILL CALL FIRST TO ALERT YOU TO THE
                                        POSSIBILITY THE PROBLEM COULD BE CAUSED BY THE OUT-OF-SPEC POWER
                                        SUPPLY, AND THEN THE ERROR THAT WAS TO CALL.

                                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
```

```
351                          7.      RESTRICTIONS
352
353                          7.1     STARTING RESTRICTIONS
354
355                                  NONE
356
357                          7.2     OPERATING RESTRICTIONS
358
359                                  NONE
```

8. MISCELLANEOUS

8.1 EXECUTION TIME

THE RUN TIME FOR ANY PASS IS APPROXIMATELY 3 SECONDS.

8.2 ADDRESS GENERATION IN THE PDP-11/44

THE FOLLOWING IS AN EXAMPLE OF HOW A MEMORY ADDRESS IS GENERATED BY THE UNIBUS MAP. THIS ASSUMES THAT THE ADDRESS ORIGINATES IN THE CPU BUT THE PROCESS CAN APPLY TO ANY UNIBUS ADDRESS, STARTING AT LINE C2.

		15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
A.	VIRTUAL ADDRESS																
A1.	P.A.R PAGE NUMBER (0-7)																
A2.	OFFSET (FROM VIRTUAL ADDRESS)																
B.	P.A.R.[PAGE NO.] +																
C.	PHYS ADDRS (A2+B)																
C1.	17XXXXXX=> U.B.ADR.																
C2.	MAPPING REG.NO.(0-36)																
C3.	OFFSET																
D.	MAP REG.[NO.] +																
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
B.	P.A.R.[PAGE NO.] +																						
C.	PHYS ADDRS (A2+B)																						
C1.	17XXXXXX=> U.B.ADR.																						
C2.	MAPPING REG.NO.(0-36)																						
C3.	OFFSET																						
D.	MAP REG.[NO.] +																						
E.	PHYS ADDR (C3+D)																						

DESCRIPTION OF LINES:

A: VIRTUAL ADDRESS (16 BITS)
- A1: UPPER 3 BITS OF VIRTUAL ADDRESS, USED TO SELECT A PAGE ADDRESS REGISTER (PAR)
- A2: LOWER 13 BITS OF VIRTUAL ADDRESS, ADDED TO SELECTED PAR

B: PAGE ADDRESS REGISTER (16 BITS), IN ADDITION PROCESS THIS GETS LEFT SHIFTED 6 BITS BEFORE ADDITION TO A2

C: PHYSICAL ADDRESS CREATED BY MEMORY MANAGEMENT, (22 BITS)
- C1: IF UPPER 4 BITS ARE ALL ONES THEN BITS <17:00> GO OUT ON UNIBUS
- C2: IF MAP RELOCATION IS ENABLED THEN BITS <17:13> SELECT ONE OF THE 36 (OCTAL) MAP REGISTERS.
- C3: LOWER 13 BITS OF UNIBUS ADDRESS, ADDED TO SELECTED MAP REGISTER

D: MAP REGISTER (22 BITS), ADDED TO BITS <12:00> OF UNIBUS ADDRESS

E: PHYSICAL ADDRESS GENERATED BY UNIBUS MAP AND SENT TO THE CACHE.

```
                                9.      PROGRAM DESCRIPTION

                                        THE ASSEMBLED LISTING, CKKUAD.SEQ, HAS A PARAGRAPH DESCRIBING
                                        EACH OF THE TESTS.  THE PARAGRAPH WILL INDICATE IF THE TEST IS
                                        RUN CONDITIONALLY ON THE STATUS ON THE CACHE CONTROL REGISTER.@
```

```
1482                    .TITLE  CKKUADO 11/24/44 UBI MAP
                        ;*COPYRIGHT (C) APRIL 1981
                        ;*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-C5), JAN, 1981.
                        ;*
1483                    .SBTTL  OPERATIONAL SWITCH SETTINGS
                        ;*
                        ;*      SWITCH                  USE
                        ;*      ------          --------------------
                        ;*        15            HALT ON ERROR
                        ;*        14            LOOP ON TEST
                        ;*        13            INHIBIT ERROR TYPEOUTS
                        ;*        12            INHIBIT TRACE TRAP
                        ;*        11            INHIBIT TRACE TRAP
                        ;*        10            BELL ON ERROR
                        ;*         9            LOOP ON ERROR
                        ;*         8            LOOP ON TEST IN SWR<4:0>
1484                    ;*         7            INHIBIT MULTIPLE ERROR TYPEOUTS
                        ;*         6            SELECT CACHE-CIS TESTS
1485                    ;*         5            SELECT MEMORY ON UNIBUS TEST
```

```
1486

                        .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
177570                  DSWR=   177570          ;;HARDWARE SWITCH REGISTER
177570                  DDISP=  177570          ;;HARDWARE DISPLAY REGISTER
177546                  LKS=    177546          ;;LINE CLOCK (KW11-L) STATUS REGISTER
                        ;*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=     %6              ;;STACK POINTER
000006                          KSP=SP          ;
000006                          SSP=SP          ;
000006                          USP=SP          ;
000007                  PC=     %7              ;;PROGRAM COUNTER
                        ;*PRIORITY LEVEL DEFINITIONS
000000                  PR0=    0               ;;PRIORITY LEVEL 0
000040                  PR1=    40              ;;PRIORITY LEVEL 1
000100                  PR2=    100             ;;PRIORITY LEVEL 2
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
```

```
                                                                    E 2
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
000100                  LKVEC=  100             ;;LINE CLOCK (KW11-L) VECTER
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
```

```
                        KDPDR2= 172324
                        KDPDR3= 172326
                        KDPDR4= 172330
                        KDPDR5= 172332
                        KDPDR6= 172334
                        KDPDR7= 172336
                        ;*KERNEL "I" PAGE ADDRESS REGISTERS
                        KIPAR0= 172340
                        KIPAR1= 172342
                        KIPAR2= 172344
                        KIPAR3= 172346
                        KIPAR4= 172350
                        KIPAR5= 172352
                        KIPAR6= 172354
                        KIPAR7= 172356
                        ;*KERNEL "D" PAGE ADDRESS REGISTERS
                        KDPAR0= 172360
                        KDPAR1= 172362
                        KDPAR2= 172364
                        KDPAR3= 172366
                        KDPAR4= 172370
                        KDPAR5= 172372
                        KDPAR6= 172374
                        KDPAR7= 172376
                        .SBTTL  UNIBUS MAP REGISTER DEFINITIONS
                        ;*THE LOWER 16 BITS OF THE MAP REGISTERS ARE LABELED 'MAPLXX'
                        ;*THE UPPER 6 BITS OF THE MAP REGISTERS ARE LABELED 'MAPHXX'
                        MAPL00 = 170200
                        MAPH00 = 170202
                        MAPL01 = 170204
                        MAPH01 = 170206
                        MAPL02 = 170210
                        MAPH02 = 170212
                        MAPL03 = 170214
                        MAPH03 = 170216
                        MAPL04 = 170220
                        MAPH04 = 170222
                        MAPL05 = 170224
                        MAPH05 = 170226
                        MAPL06 = 170230
                        MAPH06 = 170232
                        MAPL07 = 170234
                        MAPH07 = 170236
                        MAPL10 = 170240
                        MAPH10 = 170242
                        MAPL11 = 170244
                        MAPH11 = 170246
                        MAPL12 = 170250
                        MAPH12 = 170252
                        MAPL13 = 170254
                        MAPH13 = 170256
                        MAPL14 = 170260
                        MAPH14 = 170262
                        MAPL15 = 170264
                        MAPH15 = 170266
                        MAPL16 = 170270
                        MAPH16 = 170272
```

UNIBUS MAP REGISTER DEFINITIONS

```
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 = 17 354
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            ;
```

TRAP CATCHER

```
1489
                000000                  .SBTTL  TRAP CATCHER
                                                .=0
                                        ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ''.+2,HALT''
                                        ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
                                        ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
                000174                          .=174
        000174  000000                  DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
        000176  000000                  SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER
                                        .SBTTL  STARTING ADDRESS(ES)
        000200  000137  010000                  JMP     @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
```

ACT11 HOOKS

```
1491                                    .SBTTL  ACT11 HOOKS
                                        ;;******************************************************************
                                        ;HOOKS REQUIRED BY ACT11
                000204                          $SVPC=.                 ;SAVE PC
                000046                          .=46
        000046  021676                          $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
                000052                          .=52
        000052  000000                          .WORD   0               ;;2)SET LOC.52 TO ZERO
                000204                          .=$SVPC                 ;; RESTORE PC
```

```
1493                                    .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           ;DPM001
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 ADDRESS
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 ADDRESS OF 'GOOD' DATA
001120   000000                         $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
001122   000000                         $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
001124   000000                         $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
001126   000000                                 .WORD   0               ;;RESERVED--NOT TO BE USED
001130   000000                                 .WORD   0
001132      000                         $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
001133      000                         $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
001134   000000                                 .WORD   0
001136   177570                         SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
001140   177570                         DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
001142   177560                         $TKS:   177560                  ;;TTY KBD STATUS
001144   177562                         $TKB:   177562                  ;;TTY KBD BUFFER
001146   177564                         $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
001150   177566                         $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
001152      000                         $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
001153      002                         $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001154      012                         $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A 'LINE FEED''
001155      000                         $TPFLG: .BYTE   0               ;;''TERMINAL AVAILABLE'' FLAG (BIT<07>=0=YES)
001156   000000                         $REGAD: .WORD   0               ;;CONTAINS THE ADDRESS FROM
                                                                        ;;WHICH  ($REG0) WAS OBTAINED
        000006                                  .REPT   $CM3
001160   000000                         $REG0:  .WORD   0               ;;CONTAINS (($REGAD)+0)
001162   000000                         $REG1:  .WORD   0               ;;CONTAINS (($REGAD)+2)
001164   000000                         $REG2:  .WORD   0               ;;CONTAINS (($REGAD)+4)
001166   000000                         $REG3:  .WORD   0               ;;CONTAINS (($REGAD)+6)
001170   000000                         $REG4:  .WORD   0               ;;CONTAINS (($REGAD)+10)
001172   000000                         $REG5:  .WORD   0               ;;CONTAINS (($REGAD)+12)
        000007                          .REPT   7
001174   000000                         $TMP0:  .WORD   0               ;;USER DEFINED
001176   000000                         $TMP1:  .WORD   0               ;;USER DEFINED
001200   000000                         $TMP2:  .WORD   0               ;;USER DEFINED
001202   000000                         $TMP3:  .WORD   0               ;;USER DEFINED
001204   000000                         $TMP4:  .WORD   0               ;;USER DEFINED
001206   000000                         $TMP5:  .WORD   0               ;;USER DEFINED
001210   000000                         $TMP6:  .WORD   0               ;;USER DEFINED
001212   000000                         $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
001214   000000                         $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
001216      207     377     377         $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
```

```
001222     077                  $QUES:  .ASCII  /?/             ;;QUESTION MARK
001223     015                  $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001224     012     000          $LF:    .ASCIZ  <12>            ;;LINE FEED
                                ;;*****************************************************************
001226  000000                  PADRSL: .WORD   0               ;HOLDS THE LOWER 16 BITS OF A 22 BIT ADDRESS
                                                                ;GENERATED FOR TYPE OUT.
001230  000000                  PADRSH: .WORD   0               ;HOLDS THE UPPER 6 BITS OF A 22 BIT ADDRESS
                                                                ;GENERATED FOR TYPE OUT.
001232  000000  000077          ADRAND: .WORD   0,77            ;LOGICAL AND OF FAILING ADDRESSES
001236  000000  000077          ADDROR: .WORD   0,77            ;LOGICAL OR OF FAILING ADDRESSES
001242  000000  000077          DATAND: .WORD   0,77            ;LOGICAL AND OF BAD DATA
001246  000000  000077          DATAOR: .WORD   0,77            ;LOGICAL OR OF BAD DATA
001252  000000                  PATAND: .WORD   0               ;LOGICAL AND OF PATTERN LOADED
001254  000000                  PATTOR: .WORD   0               ;LOGICAL OR OF PATTERN LOADED
001256  000000                  LOWEST: .WORD   0               ;HOLDS NUMBER TO PUT IN PAR TO CAUSE THE
                                                                ;LOWEST USABLE MAP REGISTER TO RESPOND
001260  000000                  HIGEST: .WORD   0               ;HOLDS NUMBER TO PUT IN PAR TO CAUSE THE
                                                                ;HIGHEST USABLE MAP REGISTER TO RESPOND
001262  000000                  UBMLOW: .WORD   0               ;HOLDS NUMBER TO PUT IN PAR TO SIGNAL 1ST
                                                                ;ADDRESS OF UNIBUS MEMORY
001264  000000                  UBMHI:  .WORD   0               ;HOLDS NUMBER TO PUT IN PAR TO SIGNAL LAST
                                                                ;BLOCK OF 4K OF UNIBUS MEMORY
001266  000000                  MMRLOW: .WORD   0               ;HOLDS LOWEST MAP REGISTER NUMBER FROM 'LOWEST:'
001270  000000                  MMRHI:  .WORD   0               ;HOLDS HIGHEST MAP REGISTER NUMBER FROM 'HIGEST:'
001272  000000                  UBRLOW: .WORD   0               ;HOLDS LOWEST MAP REGISTER NUMBER FROM 'UBMLOW:'
001274  000000                  UBRHI:  .WORD   0               ;HOLDS HIGHEST MAP REGISTER NUMBER FROM 'UBMHI:'
001276  000000                  BUPWIN: .WORD   0               ;HOLDS LOWEST USEABLE PAR OF UPPER WINDOW
001300  000000                  LREGL:  .WORD   0               ;HOLDS I/O PAGE ADDR OF LOW 16 BITS OF
                                                                ;THE LOWEST USABLE MAP REGISTER
001302  000000                  LREGU:  .WORD   0               ;HOLDS I/O PAGE ADDR OF HIGH 6 BITS OF
                                                                ;OF THE LOWEST USABLE MAP REGISTER
001304  000000                  LMAH:   .WORD   0               ;LOCATION TO HOLD LMA HIGH REGISTER CONTENTS
001306  000000                  LMAL:   .WORD   0               ;LOCATION TO HOLD LMA LOW REGISTER CONTENTS
001310  000000                  UBM24L: .WORD   0               ;LOCATION USED TO HOLD LMA LOW EXPECTED VALUE
001312  000000                  UBM24U: .WORD   0               ;LOCATION USED TO HOLD LMA HIGH EXPECTED VALUE
001314  000000                  UBM24P: .WORD   0               ;LOCATION USED TO HOLD LMALOW PRELOAD VALUE
001316  000000                  NUMOFK: .WORD   0               ;LOCATION TO HOLD NUMBER OF K OF UB MEMORY
001320  000000                  ERRCNT: .WORD   0               ;MULTIPLE ERROR ERROR COUNTER
001322  000000                  CNTR:   .WORD   0               ;AUXILIARY COUNTER
001324  000000                  FLAG:   .WORD   0               ;FLAG TO INDICATE TO LAST PROGRAM PASS N
001326  000000                  CPUEXP: .WORD   0               ;HOLDS THE EXPECTED CPU ERROR CODE
001330  000000                  PCPUER: .WORD   0               ;HOLDS RECEIVED CPU ERROR CONDITION
001332  000000                  PPARER: .WORD   0               ;HOLDS RECEIVED PARITY ERROR CONDITION
001334  000000                  PCONTR: .WORD   0               ;HOLDS CONTENTS OF CONTROL REGISTER
001336  000000                  PMAINT: .WORD   0               ;HOLDS CONTENTS OF MAINTENENCE REGISTER
001340  000000                  BADPC:  .WORD   0               ;HOLDS PC OF INST THAT CAUSED TRAP
001342  000000                  OLDPC:  .WORD   0               ;HOLDS THE RETURN ADDRESS AFTER A TRAP
001344  000000                  OLDPS:  .WORD   0               ;HOLDS THE OLD PROCESSOR STATUS
001346  000000                  OLDPSW: .WORD   0               ;HOLDS OLD PSW FOR TBITRESTORE
001350  000000                  PMMR0:  .WORD   0               ;HOLDS CONTENTS OF MMR0 AFTER TRAP
001352  000000                  PMMR1:  .WORD   0               ;HOLDS CONTENTS OF MMR1 AFTER TRAP
001354  000000                  PMMR2:  .WORD   0               ;HOLDS CONTENTS OF MMR2 AFTER TRAP
001356  000000                  RSIZE:  .WORD   0               ;WILL HOLD P.A.R. DATA FOR TOP OF MEMORY
001360  000000                  RETRY:  .WORD   0               ;RETRY FLAG IN CASE OF PARITY ABORTS
001362  000000                  NXTTST: .WORD   0               ;LOCATION TO HOLD ESCAPE ADDRESS ON
                                                                ;PARITY ERRORS.
001364  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


     001366                     $ERRTB:
1494
1495 001366  024032             ;ITEM 1
                                        .WORD   EM1     ;NOT THE CORRECT CPU TRAP CONDITION THROUGH ERRVEC (#004)
1496 001370  027773                     .WORD   DH1     ;RECEIVD EXPECTD TESTNO  PC AT ABORT
1497 001372  032212                     .WORD   DT1     ;PCPUER,CPUEXP,$TESTN,BADPC,0
1498 001374  033513                     .WORD   DF1     ; 0, 0, 0, 0
1499
1500                            ;ITEM 2
1501 001376  024117                     .WORD   EM2     ;UNEXPECTED CPU TRAP THROUGH ERRVEC (#004)
1502 001400  030032                     .WORD   DH2     ;RECEIVD TESTNO  PC AT ABORT
1503 001402  032224                     .WORD   DT2     ;PCPUER,$TESTN,BADPC,0
1504 001404  033513                     .WORD   DF1     ; 0, 0, 0
1505
1506                            ;ITEM 3
1507 001406  024171                     .WORD   EM3     ;MEMORY MANAGEMENT TRAP, MEMORY MANAGEMENT STATUS REGISTERS
1508 001410  030061                     .WORD   DH3     ;STATUS  AUTOI/D VIRTADR
1509                                                    ;REGISTR REGISTR REGISTR TESTNO  PC AT ABORT
1510 001412  032234                     .WORD   DT3     ;PMMR0,PMMR1,PMMR2,$TESTN,BADPC,0
1511 001414  033513                     .WORD   DF1     ; 0, 0, 0, 0, 0
1512
1513                            ;ITEM 4
1514 001416  024277                     .WORD   EM4     ;SUMMARY OF MAP REGISTERS THAT TIMED OUT ON READ
1515 001420  030160                     .WORD   DH4     ;REGADRS   REGADRS
1516                                                    ; "OR"      "AND"     #ERRORS TESTNO  ERR PC
1517 001422  032250                     .WORD   DT4     ;ADDROR,ADRAND,ERRCNT,$TESTN,$ERRPC,0
1518 001424  033520                     .WORD   DF4     ; 2, 2, 1, 0, 0
1519
1520                            ;ITEM 5
1521 001426  024357                     .WORD   EM5     ;SUMMARY OF DUAL ADDRESSING ERRORS ON LOADING MAP REGISTERS
1522 001430  030255                     .WORD   DH5     ;REGLOAD   REGLOAD   REGDUAL   REGDUAL
1523                                                    ; "OR"      "AND"     "OR"      "AND"     #ERRORS TESTNO
1524 001432  032264                     .WORD   DT5     ;ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,$TESTN,0
1525 001434  033525                     .WORD   DF5     ; 2, 2, 2, 2, 1, 0
1526
1527                            ;ITEM 6
1528 001436  024452                     .WORD   EM6     ;SUMMARY OF BIT PATTERN FAILURES IN LOWER 16 BITS OF MAP REGISTERS
1529 001440  030412                     .WORD   DH6     ;MAPREG    MAPREG    EXPECTD EXPECTD RECEIVD RECEIVD
1530                                                    ; "OR"      "AND"     "OR"    "AND"   "OR"    "AND"   #ERRORS TESTNO
1531 001442  032302                     .WORD   DT6     ;ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,ERRCNT,$TESTN,0
1532 001444  033533                     .WORD   DF6     ; 2, 2, 0, 0, 0, 0, 1, 0
```

ERROR POINTER TABLE

```
                                        ;ITEM 7
001446  024554                                .WORD   EM7     ;SUMMARY OF BIT PATTERN FAILURES IN UPPER 6 BITS OF MAP REGISTERS
001450  030412                                .WORD   DH6     ;MAPREG    MAPREG    EXPECTD EXPECTD RECEIVD RECEIVD
                                                               ; "OR"       "AND"     "OR"    "AND"   "OR"    "AND"   #ERRORS TESTNO
001452  032302                                .WORD   DT6     ;ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,ERRCNT,$TESTN,0
001454  033533                                .WORD   DF6     ; 2, 2, 0, 0, 0, 0, 1, 0

                                        ;ITEM 10
001456  024655                                .WORD   EM10    ;CAN'T GET TO MAIN MEMORY FROM UNIBUS WITH THE MAP OFF
                                                               ;SO JUMPING TO THE SIZE JUMPER TEST FOR VERIFICATION
001460  030601                                .WORD   DH10    ;TESTNO  ERR PC
001462  032324                                .WORD   DT10    ;$TESTN,$ERRPC,0
001464  033513                                .WORD   DF1     ;0, 0

                                        ;ITEM 11
001466  025027                                .WORD   EM11    ;SUMMARY OF COUNT PATTERN FAILURES ON THE UNIBUS DATA PATH
001470  030626                                .WORD   DH11    ;EXPECTD EXPECTD RECEIVD RECEIVD
                                                               ; "OR"     "AND"   "OR"    "AND"  #ERRORS TESTNO
001472  032334                                .WORD   DT11    ;PATTOR,PATAND,DATAOR,DATAND,ERRCTN,$TESTN,0
001474  033543                                .WORD   DF11    ; 0, 0, 0, 0, 1, 0

                                        ;ITEM 12
001476  025121                                .WORD   EM12    ;UNIBUS MAP IS RELOCATING WHEN NOT ENABLED
001500  030601                                .WORD   DH10    ;TESTNO  ERR PC
001502  032324                                .WORD   DT10    ;$TESTN,$ERRPC,0
001504  033513                                .WORD   DF1     ; 0, 0

                                        ;ITEM 13
001506  025173                                .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.
001510  030601                                .WORD   DH10    ;TESTNO  ERR PC
001512  032324                                .WORD   DT10    ;$TESTN,$ERRPC,0
001514  033513                                .WORD   DF1     ; 0, 0

                                        ;ITEM 14
001516  025377                                .WORD   EM14    ;SUMMARY OF UNIBUS ADDRESS ERRORS, WITH MAP RELOCATION DISABLED
001520  030626                                .WORD   DH11    ;EXPECTD   EXPECTD   RECEIVD RECEIVD
                                                               ; "OR"       "AND"     "OR"    "AND"  #ERRORS TESTNO
001522  032352                                .WORD   DT14    ;ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,$TESTN,0
001524  033551                                .WORD   DF14    ; 2, 2, 0, 0, 1, 0

                                        ;ITEM 15
001526  025502                                .WORD   EM15    ;MAIN MEMORY TIME OUT OVER THE UNIBUS DID NOT OCCUR PROPERLY.
001530  030763                                .WORD   DH15    ;CONDITN CONDITN
                                                               ;EXPECTD RECEIVD TESTNO  ERR PC
001532  032370                                .WORD   DT15    ;CPUEXP,PCPUER,$TESTN,$ERRPC,0
001534  033513                                .WORD   DF1     ; 0, 0, 0, 0

                                        ;ITEM 16
001536  025575                                .WORD   EM16    ;SUMMARY OF DUAL MAPPING ERRORS
001540  030626                                .WORD   DH11    ;EXPECTD   EXPECTD   RECEIVD   RECEIVD
                                                               ; "OR"       "AND"     "OR"      "AND"    #ERRORS TESTNO
001542  032352                                .WORD   DT14    ;ADDROR,ADRAND,DATAOR,DATAND,$ERRPC,$TESTN,0
001544  033525                                .WORD   DF5     ; 2, 2, 2, 2, 1, 0
```

```
1588
1589 001546  025673                    ;ITEM 17
1590 001550  030601                            .WORD   EM17    ;NO UNIBUS MEMORY EXISTS
1591 001552  032324                            .WORD   DH10    ;TESTNO  ERR PC
1592 001554  033513                            .WORD   DT10    ;$TESTN,$ERRPC,0
1593                                           .WORD   DF1     ; 0, 0
1594
1595 001556  025723                    ;ITEM 20
1596 001560  030601                            .WORD   EM20    ;INTERRUPT/ABORT LOGIC TESTS TRAP TO LOCATION 114 DID NOT OCCUR
1597 001562  032324                            .WORD   DH10    ;TESTNO  ERR PC
1598 001564  033513                            .WORD   DT10    ;$TESTN,$ERRPC,0
1599                                           .WORD   DF1     ; 0, 0
1600
1601 001566  026022                    ;ITEM 21
1602                                           .WORD   EM21    ;INTERRUPT/ABORT TESTS R4 WAS OVERWRITTEN WITH
1603 001570  030601                                            ;DATA INDICATING THAT INSTRUCTION WAS NOT ABORTED
1604 001572  032324                            .WORD   DH10    ;TESTNO  ERR PC
1605 001574  033513                            .WORD   DT10    ;$TESTN,$ERRPC,0
1606                                           .WORD   DF1     ; 0, 0
1607
1608 001576  026100                    ;ITEM 22
1609 001600  030601                            .WORD   EM22    ;INTERRUPT/ABORT TESTS TRAP DID NOT OCCUR DUE TO ABORT
1610 001602  032324                            .WORD   DH10    ;TESTNO  ERR PC
1611 001604  033513                            .WORD   DT10    ;$TESTN,$ERRPC,0
1612                                           .WORD   DF1     ; 0, 0
1613
1614 001606  026166                    ;ITEM 23
1615 001610  031042                            .WORD   EM23    ;LMA NOT LOADED PROPERLY
1616 001612  032402                            .WORD   DH23    ;TESTNO  ERR PC  LMAEXP    LMARCV
1617 001614  033557                            .WORD   DT23    ;$TESTN,$ERRPC,EADRES,EADRS2,0
1618                                           .WORD   DF23    ; 0, 0, 2, 2
1619
1620 001616  026216                    ;ITEM 24
1621 001620  031103                            .WORD   EM24    ;LMA FORCE JUMPER BIT NOT ZERO
1622 001622  032414                            .WORD   DH24    ;TESTNO  ERR PC  LMAEXP  LMARCV
1623 001624  033513                            .WORD   DT24    ;$TESTN,$ERRPC,$REG1,LMAHI,0
1624                                           .WORD   DF1     ; 0, 0, 0, 0
1625
1626 001626  026254                    ;ITEM 25
1627 001630  031103                            .WORD   EM25    ;LMA FORCE JUMPER BIT NOT SET
1628 001632  032414                            .WORD   DH24    ;TESTNO  ERR PC  LMAEXP  LMARCV
1629 001634  033513                            .WORD   DT24    ;$TESTN,$ERRPC,$REG1,LMAHI,0
1630                                           .WORD   DF1     ; 0, 0, 0, 0
1631
1632 001636  026311                    ;ITEM 26
1633 001640  031103                            .WORD   EM26    ;LMA CONTROL BITS INCORRECT
1634 001642  032426                            .WORD   DH24    ;TESTNO  ERR PC  LMAEXP  LMARCV
1635 001644  033513                            .WORD   DT26    ;$TESTN,$ERRPC,$TMP0,$REG2,0
1636                                           .WORD   DF1     ; 0, 0, 0, 0
1637
1638 001646  026344                    ;ITEM 27
1639 001650  031142                            .WORD   EM27    ;FORCE JUMPER BIT FAILS TO REVERT MAP REGISTER STATUS TO DEFAULT
1640 001652  032440                            .WORD   DH27    ;TESTNO  ERR PC  LMARCV  KIPAR4
1641 001654  033513                            .WORD   DT27    ;$TESTN,$ERRPC,$TMP0,KIPAR4,0
                                               .WORD   DF1     ; 0, 0, 0
```

ERROR POINTER TABLE

```
1642                                    ;ITEM 30
1643 001656  026444                            .WORD   EM30    ;KIPAR5 NOT LOADED PROPERLY
1644 001660  031201                            .WORD   DH30    ;TESTNO  ERR PC  PR5EXP  PR5RCV
1645 001662  032452                            .WORD   DT30    ;$TESTN,$ERRPC,$TMP5,KIPAR5,0
1646 001664  033513                            .WORD   DF1     ; 0, 0, 0, 0
```

ERROR POINTER TABLE

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

                                ;ITEM 201
001666  026477                          .WORD   EM201   ;THE FOLLOWING REGISTERS TIMED OUT WHEN READ
001670  031240                          .WORD   DH201   ;REGADRS TESTNO  ERR PC
001672  032464                          .WORD   DT201   ;EADRES,$TESTN,$ERRPC,0
001674  033563                          .WORD   DF201   ; 2, 0, 0

                                ;ITEM 202
001676  026553                          .WORD   EM202   ;THE FOLLOWING ARE DUAL ADDRESSING ERRORS IN THE UNIBUS MAP
001700  031267                          .WORD   DH202   ;MAPREG    MAPREG    NON-ZER
                                                        ;TESTING   DUALED    CONTNTS TESTNO  ERR PC
001702  032474                          .WORD   DT202   ;EADRES,EADRS2,$TMP3,$TESTN,$ERRPC,0
001704  033533                          .WORD   DF6     ; 2, 2, 0, 0, 0

                                ;ITEM 203
001706  026646                          .WORD   EM203   ;THE BIT PATTERN THROUGH THE MAP REGISTERS FAILED
001710  031376                          .WORD   DH203   ;REGADRS   PATTRN  EXPCTD  RECEVD  TESTNO  ERR PC
001712  032510                          .WORD   DT203   ;EADRS2,$TMP0,$REG4,$REG3,$TESTN,$ERRPC,0
001714  033563                          .WORD   DF201   ; 2, 0, 0, 0, 0, 0

                                ;ITEM 204
001716  026727                          .WORD   EM204   ;UNIBUS DATA PATH COUNT PATTERN FAILURE
001720  031457                          .WORD   DH204   ;EXPECTD RECEIVD ADDRSLOAD TESTNO  ERR PC
001722  032526                          .WORD   DT204   ;$TMP0,$TMP1,$REG2,$TESTN,$ERRPC,0
001724  033571                          .WORD   DF204   ;0, 0, 3, 0, 0

                                ;ITEM 205
001726  026776                          .WORD   EM205   ;UNIBUS ADDRESSING ERRORS, MAP RELOCATION DISABLED
001730  031530                          .WORD   DH205   ;ADDRESS   ADDRESS
                                                        ;EXPECTD   RECEIVD   TESTNO ERR PC
001732  032542                          .WORD   DT205   ;EADRES,EADRS2,$TESTN,$ERRPC,0
001734  033551                          .WORD   DF14    ; 2, 2, 0, 0

                                ;ITEM 206
001736  027060                          .WORD   EM206   ;DATA PATTERN NOT CORRECT
001740  031615                          .WORD   DH206   ;ADDRESS  EXPCTD  RECVED  TESTNO  ERR PC
001742  032554                          .WORD   DT206   ;EADRES,$TMP4,$TMP5,$TESTN,$ERRPC,2
001744  033563                          .WORD   DF201   ; 2, 0, 0, 0, 0

                                ;ITEM 207
001746  027111                          .WORD   EM207   ;REFERENCED MAP REGISTER 0 WITH ADDRESS ONE BIT DIFFERENT THAN 17770
001750  031665                          .WORD   DH207   ;ADDRUSED  BITDIFF TESTNO  ERR PC
001752  032570                          .WORD   DT207   ;EADRES,$REG0,$TESTN,$ERRPC,0
001754  033563                          .WORD   DF201   ;2, 0, 0, 0

                                ;ITEM 210
001756  027220                          .WORD   EM210   ;MAP REGISTER UNDER TEST DID NOT RESPOND IN DUAL MAPPING TEST
001760  031726                          .WORD   DH210   ;TESTNO  ERR PC MAPREGADR
001762  032602                          .WORD   DT210   ;$TESTN,$ERRPC,EADRES,0
001764  033576                          .WORD   DF210   ; 0, 0, 2
```

ERROR POINTER TABLE

```
1700
1701 001766  027320                    ;ITEM 211
1702                                            .WORD   EM211   ;RELOCATION THROUGH THE MAP WAS NOT CORRECT, CARRY PROPAGATION
1703                                            ;TEST BEING RUN OVER UNIBUS
1704 001770  031760                             .WORD   DH211   ;CORRECT   EXPECTD RECEIVD
1705                                            ;ADDRESS   DATA    FROM UB TESTNO  ERR PC
1706 001772  032612                             .WORD   DT211   ;EADRES,$REG3,$REG2,$TESTN,$ERRPC,0
1707 001774  033563                             .WORD   DF201   ; 2, 0, 0, 0, 0
1708
1709 001776  027456                    ;ITEM 212
1710                                            .WORD   EM212   ;MAIN MEMORY TIME OUT OVER THE UNIBUS DID NOT OCCUR PROPERLY.
1711                                            ;TEST BEING RUN OVER UNIBUS
1712 002000  031760                             .WORD   DH211   ;CONDITN CONDITN
1713                                            ;EXPECTD RECEIVD TESTNO  ERR PC
1714 002002  032612                             .WORD   DT211   ;CPUEXP,PCPUER,$TESTN,$ERRPC,0
1715 002004  033563                             .WORD   DF201   ; 0, 0, 0, 0
1716
1717 002006  027611                    ;ITEM 213
1718                                            .WORD   EM213   ;MAP REGISTER ENABLED WHEN DDW SAYS IT SHOULD BE DISABLED
1719 002010  032142                             .WORD   DH213   ;TESTNO  ERR PC  REG NO  DDWDAT  DDWADR
1720 002012  032640                             .WORD   DT213   ;$TESTN,$ERRPC,$TMP0,$TMP1,$REG5,0
1721 002014  033513                             .WORD   DF1     ; 0, 0, 0, 0, 0
1722
1723 002016  027702                    ;ITEM 214
1724                                            .WORD   EM214   ;MAP REGISTER DISABLED WHEN DDW SAYS IT SHOULD BE ENABLED
1725 002020  032142                             .WORD   DH213   ;TESTNO  ERR PC  REG NO  DDWDAT  DDWADR
1726 002022  032640                             .WORD   DT213   ;$TESTN,$ERRPC,$TMP0,$TMP1,$REG5,0
     002024  033513                             .WORD   DF1     ; 0, 0, 0, 0, 0
```

SOFTWARE SWITCH REGISTER LOCATION

```
                                .SBTTL  SOFTWARE SWITCH REGISTER LOCATION
002026                          .$Y=.           ;SAVE ADDRESS LOCATION
000176                          .=176           ;ADDRESS TO SOFTWARE SWITCH REGISTER LOCATION
000176  000000          $SSWR:  .WORD   0       ;LOCATION FOR SOFTWARE SWITCH REGISTER
002026                          .=.$Y           ;RETURN TO PREVIOUS ADDRESS LOCATION
```

ERROR MESSAGE TYPE OUT ROUTINE

```
                                .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 ADFRESS 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
                                ;*
                                ;;*****************************************************************
002026  010046                  ERTYPE: MOV     R0,-(KSP)       ;SAVE R0 ON STACK
002030  005000                          CLR     R0              ;CLEAR R0
002032  113700  001112                  MOVB    $ITEMB,R0       ;PUT ITEM NUMBER IN R0
002036  001004                          BNE     1$              ;BRANCH IF IT IS NON-ZERO
002040  013746  001114                  MOV     $ERRPC,-(KSP)   ;PUT ERROR PC ON STACK FOR TYPING
002044  104402                          TYPOC                   ;TYPE FAILING PC
002046  000565                          BR      13$             ;GO TO RETURN
002050  122700  000177          1$:     CMPB    #177,R0         ;SEE IF THIS IS THE PWR MON BIT ERROR   ;DPM001
002054  001003                          BNE     200$            ;BRANCH IF NOT TO CALL ERROR
002056  012700  002432                  MOV     #PMBECW,R0      ;MOVE ADDRESS OF SPECIAL DATA HEADER TO R0
002062  000451                          BR      210$            ;BRANCH TO CALL ERROR
002064  005300                  200$:   DEC     R0              ;ADJUST ITEM NUMBER TO BE A POINTER
002066  072027  000003                  ASH     #3,R0           ;LEFT SHIFT ITEM NO. 3 PLACES
002072  100041                          BPL     22$             ;BRANCH IF ITEM NUMBER IS LESS THAN 200
002074  023727  001320  000020          CMP     ERRCNT,#20      ;;* SEE IF 20 (OCTAL) ERRORS HAVE PRINTED
```

```
1789 002102  002410                        BLT     40$             ;;* BRANCH TO PRINT THE ERROR IF LESS
1790 002104  001404                        BEQ     41$             ;;* BRANCH TO TYPE NO MORE DATA LINES IF EQUAL
1791 002106  062766  000004  000002        ADD     #4,2(KSP)       ;;* CORRECT PC RETURN TO RETURN AFTER <CRLF> PRINT
1792 002114  000542                        BR      13$             ;;* GO TO RETURN
1793 002116  104401  006417        41$:    TYPE    ,NOMORE         ;;* TYPE MESSAGE TO ANNOUNCE NO MORE PRINTING OF ERRORS
1794 002122  000537                        BR      13$             ;;* GO TO RETURN
1795 002124  022737  000001  001320  40$:   CMP     #1,ERRCNT       ;;* SEE IF THIS IS THE FIRST ERROR
1796 002132  001415                        BEQ     21$             ;;* BRANCH IF IT WAS AND GO TYPE ERROR MESSAGE
1797 002134  032777  000200  176774        BIT     #SW7,@SWR       ;SEE IF SWITCH 7 IS UP
1798 002142  001404                        BEQ     20$             ;BRANCH IF SWITCH NOT UP AND TYPE DATA
1799 002144  062766  000004  000002        ADD     #4,2(KSP)       ;SKIP 'TYPE  ,$CRLF' IF SW 7 IS UP
1800                                                                ;INHIBIT MULTIPLE ERROR TYPEOUTS
1801 002152  000523                        BR      13$             ;BRANCH TO EXIT
1802 002154  042700  177400        20$:    BIC     #177400,R0      ;CLEAR UPPER BYTE OF R0
1803 002160  062700  001672                ADD     #ER200+4,R0     ;POINT TO DATA TABLE ENTRY
1804 002164  000426                        BR      5$              ;GO TYPE DATA TABLE
1805 002166  042700  177000        21$:    BIC     #177000,R0      ;CLEAR UPPER BYTE OF R0
1806 002172  062700  000300                ADD     #<ER200-$ERRTB>,R0 ;ADD DIFFERENCE BETWEEN
1807                                                                ;ITEM 1 AND ITEM 201
1808                                                        ;:GET POINTER TO ERROR MESSAGE AND TYPE IT
1809                                                        ;:IF THE POINTER IS NOT ZERO
1810 002176  104401  001223        22$:    TYPE    ,$CRLF          ;TYPE A <CRLF>
1811 002202  062700  001366                ADD     #$ERRTB,R0      ;ADD BASE OF ERROR TABLE
1812 002206  012037  002216        210$:   MOV     (R0)+,2$        ;P M MESSAGE POINTER IN TYPE STATEMENT
1813 002212  001404                        BEQ     3$              ;BRANCH IF NO ERROR MESSAGE
1814 002214  104401                        TYPE                    ;TYPE ERROR MESSAGE
1815 002216  000000        2$:             .WORD   0               ;POINTER TO ERROR MESSAGE
1816 002220  104401  001223                TYPE    ,$CRLF          ;TYPE CRLF
1817                                                        ;:GET THE POINTER TO THE DATA HEADER AND
1818                                                        ;:TYPE IT IF THE POINTER IS NOT ZERO
1819 002224  012037  002234        3$:     MOV     (R0)+,4$        ;PUT HEADER POINTER IN TYPE STATEMENT
1820 002230  001404                        BEQ     5$              ;BRANCH IF NO DATA HEADER
1821 002232  104401                        TYPE                    ;TYPE THE DATA HEADER
1822 002234  000000        4$:             .WORD   0               ;POINTER TO DATA HEADER
1823 002236  104401  001223                TYPE    ,$CRLF          ;TYPE CRLF
1824                                                        ;:THIS IS THE START OF THE DATA OUTPUT IF THE
1825                                                        ;:DATA POINTER IS NOT ZERO.  R0 POINTS TO THE
1826                                                        ;:DATA FORMAT, R1 POINTS TO THE ADDRESS OF
1827                                                        ;:THE DATA WORDS.
1828 002242  010146        5$:     MOV     R1,-(KSP)       ;SAVE R1 ON THE STACK
1829         002244        TYPDAT=.
1830 002244  012001                MOV     (R0)+,R1        ;PUT DATA TABLE POINTER IN R1
1831 002246  001464                BEQ     12$             ;BRANCH IF NO DATA TABLE
1832 002250  012000                MOV     (R0)+,R0        ;PICK UP DATA FORMAT POINTER
1833 002252  105710        6$:     TSTB    (R0)            ;IS THIS WORD OCTAL
1834 002254  001003                BNE     7$              ;BRANCH IF NOT 16-BIT OCTAL
1835                                                        ;:WORD IS 16 BIT OCTAL FORMAT (DF = 0)
1836 002256  013146                MOV     @(R1)+,-(KSP)   ;PUSH NEXT 16-BIT WORD ON STACK
1837 002260  104402                TYPOC                   ;TYPE THE WORD ON STACK AS 16 BIT OCTAL
1838 002262  000451                BR      11$             ;GET READY FOR NEXT WORD
1839 002264  122710  000001  7$:   CMPB    #1,(R0)         ;IS THE WORD DECIMAL
1840 002270  001003                BNE     8$              ;BRANCH IF NOT DECIMAL
1841                                                        ;:WORD IS DECIMAL FORMAT (DF = 1)
1842 002272  013146                MOV     @(R1)+,-(KSP)   ;PUSH NEXT 16-BIT WORD ON STACK
1843 002274  104405                TYPDS                   ;TYPE THE WORD ON STACK AS DECIMAL
1844 002276  000443                BR      11$             ;GET READY FOR NEXT WORD
1845 002300  122710  000002  8$:   CMPB    #2,(R0)         ;IS WORD 22-BIT PHYSICAL ADDRESS
```

```
1846 002304  001012                          BNE     9$              ;BRANCH IF NOT 22-BIT PHYSICAL ADDR
1847                                                 ;:WORD IS 22-BIT PHYSICAL FORMAT (DF = 2)
1848 002306  012146                          MOV     (R1)+,-(KSP)    ;PUSH NEXT 16-BIT WORD ON STACK
1849 002310  004737  023712                  JSR     PC,$DB2O        ;CONVERT NUMBER TO OCTAL ASCIZ
1850 002314  062716  000003                  ADD     #3,(KSP)        ;ONLY WANT 8 DIGITS
1851 002320  012637  002326                  MOV     (KSP)+,30$      ;PUT POINTER AFTER 'TYPE' CALL
1852 002324  104401                          TYPE                    ;TYPE ASCIZ STRING
1853 002326  000000                  30$:    .WORD   0               ;WORD HOLDS POINTER TO ASCIZ STRING
1854 002330  000426                          BR      11$             ;GET READY FOR NEXT WORD
1855 002332  122710  000003          9$:     CMPB    #3,(R0)         ;IS THIS A 16-BIT VIRTUAL ADDRESS
1856 002336  001004                          BNE     10$             ;BRANCH IF NOT 16-BIT VIRT. ADDR.
1857                                                 ;:WORD IS 22-BIT VIRTUAL ADDRESS FORMAT
1858                                                 ;:KERNEL I-SPACE ASSUMED.  (DF = 3)
1859 002340  013146                          MOV     @(R1)+,-(KSP)   ;PUSH NEXT 16-BIT WORD ON STACK
1860 002342  004737  002542                  JSR     PC,TYPVAD       ;GO TYPE 22-BIT ADDRESS FROM 16-BIT V.A.
1861 002346  000417                          BR      11$             ;GET READY FOR NEXT WORD
1862 002350  122710  000004          10$:    CMPB    #4,(R0)         ;IS THIS A 16 BIT NUMBER TO BE CONVERTED TO
1863                                                                 ;AN 18 BIT UNIBUS ADDRESS LEFT SHIFTED 6?
1864 002354  001003                          BNE     100$            ;SKIP OVER FORMAT 4 ROUTINE IF NOT
1865                                                 ;:WORD IS FORMAT 4.  DATA WORD IS A UNIBUS
1866                                                 ;:ADDRESS OUTPUT WILL BE 18-BITS WORD LEFT SHIFTED 6.
1867
1868 002356  004737  002650                  JSR     PC,UBADDR       ;CONVERT TO 18-BIT UNIBUS ADDR AND TYPE
1869 002362  000411                          BR      11$             ;GET READY FOR NEXT WORD
1870 002364  122710  000005          100$:   CMPB    #5,(R0)         ;IS THIS A 16 BIT NUMBER TO BE PRINTED AS
1871                                                                 ;OCTAL WITH LEADING ZEROS SUPPRESSED?
1872 002370  001004                          BNE     110$            ;BRANCH TO DECIMAL LEADING SPACES SUPPRESS ROUTINE
1873                                                 ;:WORD IS FORMAT 5.  DATA WORD IS TO BE
1874                                                 ;:PRINTED IN OCTAL, LEADING ZEROS SUPPRESSED.
1875 002372  013146                          MOV     @(R1)+,-(KSP)   ;PUSH NEXT 16-BIT WORD ON STACK
1876 002374  104403                          TYPOS                   ;GO TYPE OCTAL SUPPRESS LEADING ZEROS
1877 002376     006                          .BYTE   6               ;TYPE 6 DIGITS AND
1878 002377     000                          .BYTE   0               ;SUPPRESS LEADING ZEROS
1879 002400  000402                          BR      11$             ;GET READY FOR NEXT WORD
1880 002402                          110$:                           ;:WORD IS FORMAT 6. DATA WORD IS TO BE
1881                                                                 ;:PRINTED IN DECIMAL, LEADING SPACES SUPPRESSED.
1882 002402  013146                          MOV     @(R1)+,-(KSP)   ;PUSH NEXT 16-BIT WORD ON STACK
1883 002404  104405                          TYPDS
1884 002406  005200                  11$:    INC     R0              ;POINT TO NEXT FORMAT BYTE
1885 002410  104401  002426                  TYPE    ,32$            ;TYPE TWO SPACES
1886 002414  005711                          TST     (R1)            ;IS THERE ANOTHER WORD?
1887 002416  001315                          BNE     6$              ;BRANCH IF NOT ALL DONE
1888 002420  012601                  12$:    MOV     (KSP)+,R1       ;RESTORE R1
1889 002422  012600                  13$:    MOV     (KSP)+,R0       ;RESTORE R0
1890 002424  000207                          RTS     PC              ;RETURN TO ERROR ROUTINE
1891 002426     040     040     000  32$:    .ASCIZ  ?  ?            ;TWO SPACES
1892                                          .EVEN
1893 002432  002442  002476  002526  PMBECW: .WORD   PMBECM,PMBECH,PMBECD,PMBECF     ;4 WORDS POINTING TO BELOW
1894 002442     120     117     127  PMBECM: .ASCIZ  ?POWER MONITOR BIT FOUND SET?
1895 002476     124     105     123  PMBECH: .ASCIZ  ?TESTNO  ERR PC  CPUERR?
1896                                          .EVEN
1897 002526  020020  001114  020552  PMBECD: .WORD   $TESTN,$ERRPC,CPSAVE,0
1898 002536     000     000     000  PMBECF: .BYTE   0,0,0,0
```

```
                                        .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.
                                        ;*
                                        ;;*****************************************************************
002542  104411                  TYPVAD: SAVREG                  ;SAVE ALL REGISTERS
002544  016601  000002                  MOV     2(KSP),R1       ;PUT VIRTUAL ADDR IN R1
002550  005000                          CLR     R0              ;CLEAR R0 FOR CALCULATIONS
002552  073027  000003                  ASHC    #3,R0           ;LEFT SHIFT R0,R1 3 PLACES
002556  006300                          ASL     R0              ;LEFT SHIFT R0 ONE MORE PLACE
002560  006001                          ROR     R1              ;RIGHT SHIFT R1 SO OFFSET IS CORRECT
002562  006001                          ROR     R1              ;RIGHT SHIFT R1
002564  006001                          ROR     R1              ;RIGHT SHIFT R1
002566  062700  172340                  ADD     #KIPAR0,R0      ;FORM DESIRED PAR ADDR IN R0
002572  011003                          MOV     (R0),R3         ;PUT CONTENTS OF PAR IN R3
002574  005002                          CLR     R2              ;CLEAR R2 FOR PHYSICAL ADDR CALCULATIONS
002576  073227  000006                  ASHC    #6,R2           ;LEFT SHIFT <R2,R3> 6 PLACES
002602  060103                          ADD     R1,R3           ;ADD OFFSET IN R1 TO BASE IN R3
002604  005502                          ADC     R2              ;ADD ANY POSSIBLE CARRY TO UPPER 6 BITS
002606  010237  001230                  MOV     R2,PADRSH       ;PUT UPPER 6 BITS OF ADDR IN CORE
002612  010337  001226                  MOV     R3,PADRSL       ;PUT LOWER 16 BITS OF ADDR IN CORE
002616  012746  001226                  MOV     #PADRSL,-(KSP)  ;PUT POINTER TO LOWER 16 BITS ON STACK
002622  004737  023712                  JSR     PC,$DB2O        ;CONVERT NUMBER TO OCTAL ASCIZ
002626  062716  000003                  ADD     #3,(KSP)        ;ONLY TYPE 8 DIGITS
002632  012637  002640                  MOV     (KSP)+,3$       ;PUT POINTER AFTER TYPE INST
002636  104401                          TYPE                    ;TYPE THE 22-BIT VIRTUAL ADDRESS
002640  000000                  3$:     .WORD   0               ;THIS WORD HOLDS THE POINTER TO
                                                                ;THE ASCIZ STRING
002642  104412                          RESREG                  ;RESTORE ALL THE REGISTERS
002644  012616                          MOV     (KSP)+,(KSP)    ;LEAVE ONLY RETURN ADDR ON STACK
002646  000207                          RTS     PC              ;RETURN TO ERROR HANDLER
```

```
                                        .SBTTL  SUBROUTINE TO CONVERT WORD TO A UNIBUS ADDRESS AND TYPE
                                        ;;*****************************************************************
                                        ;*THIS SUBROUTINE IS USED TO CONVERT THE A WORD PUSHED
                                        ;*ON THE STACK INTO A UNIBUS ADDRESS AND TYPE IT AS A
                                        ;*6 DIGIT NUMBER.  IT USES R1 & R0 AND LEAVES
                                        ;*ALL OTHER REGISTERS UNCHANGED.
                                        ;;*****************************************************************
002650  104411                  UBADDR: SAVREG
002652  016601  000002                  MOV     2(KSP),R1       ;LOAD 16 BIT ADDRESS INTO R1
002656  005000                          CLR     R0              ;CLEAR R0 FOR CALCULATIONS
002660  073027  000006                  ASHC    #6,R0           ;LEFT SHIFT <R0:R1> 6 PLACES
002664  010137  001226                  MOV     R1,PADRSL       ;PUT LOWER 16 BITS IN PADRSL
002670  010037  001230                  MOV     R0,PADRSH       ;PUT UPPER 6 BITS IN PADRSH
002674  012746  001226                  MOV     #PADRSL,-(KSP)  ;PUSH POINTER TO WORDS ON STACK
002700  004737  023712                  JSR     PC,$DB2O        ;JUMP TO CONVERT ROUTINE
002704  062716  000005                  ADD     #5,(KSP)        ;ONLY USE LOWER 6 CHARS.
002710  012637  002716                  MOV     (KSP)+,3$       ;PUT POINTER AFTER TYPE CALL.
002714  104401                          TYPE
002716  000000                  3$:     .WORD   0               ;HOLDS POINTER TO FIRST CHAR.
002720  104412                          RESREG
002722  012616                          MOV     (KSP)+,(KSP)    ;LEAVE ONLY RETURN ADDRESS ON STACK
002724  000207                          RTS     PC              ;RETURN TO ERROR TYPE ROUTINE.
```

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

                                        ;;**********************************************************************
                                        ;*
                                        ;*      THIS TRAP ROUTINE 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
002726  032766  000020  000002  TBITOF: BIT     #TBIT,2(KSP)    ;IS THE T-BIT ON?
002734  001406                        BEQ     1$              ;BRANCH TO EXIT IF IT IS NOT ON
002736  016637  000002  001346        MOV     2(KSP),OLDPSW   ;SAVE OLD PSW FOR RESTORING T BIT
002744  042766  000020  000002        BIC     #TBIT,2(KSP)    ;CLEAR T BIT
002752  000006                  1$:   RTT                     ;RETURN TO PROGRAM
```

```
                                        .SBTTL  RESTORE T-BIT TO ITS PREVIOUS CONDITION
                                        ;;*****************************************************************
                                        ;*
                                        ;*      THIS TRAP ROUTINE 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'.
                                        ;*
                                        ;;*****************************************************************
002754  013766  001346  000002  TBITRE: MOV     OLDPSW,2(KSP)   ;PUT OLD PSW ON STACK
002762  042737  000020  001346          BIC     #TBIT,OLDPSW    ;CLEAR T-BIT IN "OLDPSW"
                                                                ;SO THAT IT WON'T BE TURNED ON BY ACCIDENT
002770  000006                          RTT     ;RETURN TO PROGRAM AND INHIBIT T-BIT TRAP AFTER THIS INSTRUCTION
```

```
 1994                                   .SBTTL  SUBROUTINE TO CLEAR ALL OF THE MAP REGISTERS
 1995                                   ;;******************************************************************
 1996                                   ;*
 1997                                   ;*      THIS SUBROUTINE CLEARS ALL OF THE MAP REGISTERS IF MAPPING IS
 1998                                   ;*      DISABLED BY LOADING THE ADDRESS OF MAPL00 INTO R3 AND THEN
 1999                                   ;*      CLEARING THE REGISTER POINTED TO BY R3 UNTIL R3 POINTS ABOVE
 2000                                   ;*      MAPH37.  IF MAPPING IS ENABLED, ALL REGISTERS EXCEPT MAPL1
 2001                                   ;*      IS CLEARED.  THE LOWER WORD OF MAPL1 RECEIVES 20000.  THIS IS
 2002                                   ;*      SO APT CAN PROPERLY MONITOR THE PROGRESS OF THE DIAGNOSTIC.
 2003                                   ;*
 2004                                   ;;******************************************************************
 2005 002772  012703  170200           CLRMAP: MOV     #MAPL0,R3       ;PUT FIRST MAP ADDR IN R3
 2006 002776  005023                           CLR     (R3)+           ;CLEAR MAPL0
 2007 003000  005023                           CLR     (R3)+           ;CLEAR MAPL0+2
 2008 003002  032737  000040  172516           BIT     #BIT5,MMR3      ;SEE IF MAPPING IS ENABLED
 2009 003010  001402                           BEQ     1$              ;BRANCH TO CLEAR ALL IF NOT ENABLED
 2010 003012  012723  020000                   MOV     #20000,(R3)+    ;LOAD 20000 INTO MAPL1 FOR POSSIBLE APT USE
 2011 003016  005023                   1$:     CLR     (R3)+           ;CLEAR MAP REGISTERS
 2012 003020  022703  170400                   CMP     #MAPH37+2,R3    ;SEE IF LAST ADDR+2 IS IN R3
 2013 003024  001374                           BNE     1$              ;BRANCH IF NOT DONE YET
 2014 003026  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.
                                        ;*
                                        ;;*****************************************************************
003030  005227                  TIMEOUT:INC     (PC)+           ;INCREMENT ONE TIME GATE
003032  177777                  TOFLAG: .WORD   -1              ;ONE TIME ENTANCE FLAG
003034  001403                          BEQ     10$             ;BRANCH IF FLAG IS NOW ZERO
003036  005237  020014                  INC     $MSGTY          ;INDICATE TO APT A FATAL ERROR OCCURED
003042  000000                          HALT    ;I HAVE ENTERED THIS ROUTINE BEFORE I FINISHED REPORTING THE FIRST ERROR.
                                                                ;THE SECOND ENTRY ADDRESS IS ON THE STACK AND THE
                                                                ;FIRST ERROR CONDITION IS PROBABLY STILL LOCKED UP.
003044  012637  001342          10$:    MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS
003050  012637  001344                  MOV     (KSP)+,OLDPS    ;SAVE OLD PSW
003054  105737  007136                  TSTB    CPUTYP          ;SEE IF THIS IS AN 11/44
003060  001406                          BEQ     1$              ;BRANCH TO CONTINUE IF IT IS
003062  005237  001330                  INC     PCPUER          ;INCREMENT PCPUER TO SHOW A TIMEOUT OCCURED
003066  005737  001326                  TST     CPUEXP          ;SEE IF THERE WAS AN EXPECTED ERROR
003072  001435                          BEQ     3$              ;GO REPORT ERROR IF NONE EXPECTED
003074  000442                          BR      4$              ;BRANCH TO EXIT IF TIMEOUT WAS EXPECTED
003076  013737  177766  001330  1$:     MOV     CPUERR,PCPUER   ;SAVE CPU ERROR REGISTER
003104  013737  001330  177766          MOV     PCPUER,CPUERR   ;CLEAR CPU ERROR REGISTER
003112  023737  001330  001326          CMP     PCPUER,CPUEXP   ;SEE IF EXPECTED CONDITION CAME UP.
003120  001405                          BEQ     2$              ;BRANCH IF IT WAS A TIMEOUT
003122  012737  177777  003032          MOV     #-1,TOFLAG      ;RESET ONE TIME GATE
003130  104001                          ERROR   +1              ;NOT THE CORRECT CPU TRAP THROUGH 4
003132  000423                          BR      4$              ;BRANCH TO EXIT
003134  105737  007136          2$:     TSTB    CPUTYP          ;IS THIS AN 11/24?
003140  001403                          BEQ     25$             ;BRANCH IF NOT
003142  005237  001320                  INC     ERRCNT          ;COUNT THIS AS A TIMEOUT
003146  000415                          BR      4$              ;GO TO EXIT
003150  022737  000020  001326  25$:    CMP     #TIMOUT,CPUEXP  ;SEE IF A TIMEOUT WAS EXPECTED
003156  001411                          BEQ     4$              ;BRANCH TO EXIT THIS ROUTINE IF IT WAS
003160  010046                          MOV     R0,-(SP)        ;PUT VIRTUAL ADDRESS ON STACK FOR ADREXT SUBROUTINE USE
003162  013746  172356                  MOV     KIPAR7,-(SP)    ;PUT PAR ON STACK FOR ADREXT SUBROUTINE USE
003166  004737  003674          3$:     JSR     PC,ADREXT       ;GO SET DATA IN THE 4 WORDS OF ADDROR AND ADRAND
003172  012737  177777  003032          MOV     #-1,TOFLAG      ;RESET ONE TIME GATE
003200  104201                          ERROR   +201            ;THE FOLLOWING REGISTERS TIMED OUT WHEN READ
003202  012737  177777  003032  4$:     MOV     #-1,TOFLAG      ;RESET ONE TIME GATE
003210  013746  001344                  MOV     OLDPS,-(KSP)    ;RESTORE OLD PSW
003214  013746  001342                  MOV     OLDPC,-(KSP)    ;PUSH RETURN ADDRESS BACK ON THE STACK
003220  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
                                        ;*
                                        ;;*****************************************************************
003222  005227                  CPUER:  INC     (PC)+           ;MAKE FLAG ZERO IF FIRST TIME
003224  177777                  CPFLAG: .WORD   -1              ;NEGATIVE ONE FOR A FLAG
003226  001403                          BEQ     10$             ;BRANCH IF FIRST TIME IN
003230  005237  020014                  INC     $MSGTY          ;INDICATE TO APT A FATAL ERROR OCCURED
003234  000000                          HALT                    ;I HAVE ENTERED THIS ROUTINE BEFORE
                                        ;I FINISHED REPORTING THE FIRST ERROR.  THE SECOND ENTRY ADDRESS IS ON
                                        ;THE STACK, AND THE FIRST ERROR CONDITION IS PROBABLY STILL LOCKED UP.
003236  012637  001342          10$:    MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS IN CASE OF LOOP
003242  012637  001344                  MOV     (KSP)+,OLDPS    ;SAVE OLD PSW IN CASE OF LOOP
003246  013737  177766  001330          MOV     CPUERR,PCPUER   ;SAVE CPU ERROR REGISTER
003254  013737  001342  001340          MOV     OLDPC,BADPC     ;SAVE PC+2 AT TIME OF ABORT
003262  005737  001326                  TST     CPUEXP          ;SEE IF ANY CONDITION WAS EXPECTED
003266  001414                          BEQ     1$              ;BRANCH IF NO TRAP WAS EXPECTED
003270  105737  007136                  TSTB    CPUTYP          ;SEE IF THIS WAS AN 11/44
003274  001016                          BNE     2$              ;BRANCH TO CONTINUE IF AN 11/24
003276  023737  001330  001326          CMP     PCPUER,CPUEXP   ;SEE IF EXPECTED ERROR OCCURED
003304  001417                          BEQ     3$              ;BRANCH IF ERROR CODES MATCH
003306  012737  177777  003224          MOV     #-1,CPFLAG      ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
003314  104001                          ERROR   +1              ;NOT THE CORRECT CPU TRAP THROUGH 4
003316  000412                          BR      3$              ;SKIP NEXT INSTRUCTION
003320  012737  177777  003224  1$:     MOV     #-1,CPFLAG      ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
003326  104002                          ERROR   +2              ;UNEXPECTED CPU TRAP THROUGH 4
003330  000405                          BR      3$              ;SKIP NEXT INSTRUCTION
003332  005237  001320          2$:     INC     ERRCNT          ;INCREMENT ERRCNT TO SHOW AN ERROR FOR 11/24
003336  013737  001326  001330          MOV     CPUEXP,PCPUER   ;PUT EXPECTED CONTENTS IN PCPUER
003344  012737  177777  003224  3$:     MOV     #-1,CPFLAG      ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
003352  013737  001330  177766          MOV     PCPUER,CPUERR   ;CLEAR CPU ERROR REGISTER
003360  013746  001344                  MOV     OLDPS,-(KSP)    ;PUSH OLD PSW BACK ON STACK
003364  013746  001342                  MOV     OLDPC,-(KSP)    ;PUSH RETURN ADDRESS BACK ON STACK
003370  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.
                                        ;*
                                        ;;*****************************************************************
003372  005227                          MMTRAP: INC     (PC)+           ;MAKE FLAG ZERO IF FIRST TIME
003374  177777                          MMFLAG: .WORD   -1              ;FLAG SHOULD BE NEG ONE
003376  001403                                  BEQ     10$             ;BRANCH IF FIRST TIME INTO ROUTINE
003400  005237  020014                          INC     $MSGTY          ;INDICATE TO APT A FATAL ERROR OCCURED
003404  000000                                  HALT    ;I HAVE ENTERED THIS ROUTINE BEFORE I FINISHED REPORTING THE
                                        ;FIRST ERROR.  THE SECOND ENTRY ADDRESS IS ON THE STACK AND THE FIRST ERROR
                                        ;CONDITION IS PROBABLY STILL LOCKED UP .
003406  011637  001340                  10$:    MOV     (KSP),BADPC     ;SAVE PC AT TIME OF ABORT OR TRAP
003412  012637  001342                          MOV     (KSP)+,OLDPC    ;SAVE RETURN ADDRESS IN CASE OF LOOP
003416  012637  001344                          MOV     (KSP)+,OLDPS    ;SAVE OLD PSW IN CASE OF LOOP
003422  013737  177572  001350                  MOV     MMR0,PMMR0      ;SAVE STATUS REGISTER
003430  013737  177574  001352                  MOV     MMR1,PMMR1      ;SAVE AUTO INC/DEC REGISTER
003436  013737  177576  001354                  MOV     MMR2,PMMR2      ;SAVE VIRTUAL ADDRESS REGISTER
003444  104003                                  ERROR   +3              ;UNEXPECTED M.M. ABORT OR TRAP
003446  042737  177776  177572  1$:     BIC     #177776,MMR0    ;CLEAR ALL BITS EXCEPT 0
003454  012737  177777  003374          MOV     #-1,MMFLAG      ;RESTORE A NEGATIVE ONE TO FLAG
003462  013746  001344                  MOV     OLDPS,-(KSP)    ;PUSH OLD PSW ONTO STACK
003466  013746  001342                  MOV     OLDPC,-(KSP)    ;PUSH RETURN ADDRESS ON STACK
003472  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.
                                        ;*
                                        ;;*****************************************************************
003474  005237  001174        TSTLOC: INC     $TMP0           ;INCREMENT REGISTER COUNTER
003500  005737  005772                TST     FLOATR          ;SEE IF BIT 15 OF FLOATR IS SET
003504  100011                        BPL     1$              ;BRANCH IF STILL PLUS
003506  022705  020102                CMP     #$DDW1,R5       ;SEE IF R5 IS POINTING TO UPPER DDW
003512  001417                        BEQ     NEXT            ;BRANCH IF SO - ALL DONE
003514  012705  020102                MOV     #$DDW1,R5       ;MOVE ADDRESS OF DDW1 TO R5 AND
003520  012737  000001  005772        MOV     #BIT0,FLOATR    ;RESET BIT 0 IN FLOATR
003526  000411                        BR      NEXT            ;BRANCH OVER ASL
003530  005227                1$:     INC     (PC)+           ;INCREMENT NEXT LOCATION FOR FIRST TIME THROUGH CHECK
003532  177777                FTTHRU: .WORD   -1              ;FIRST TIME ENTRANCE FLAG
003534  001004                        BNE     1$              ;BRANCH IF NOT FIRST TIME
003536  012737  000001  005772        MOV     #BIT0,FLOATR    ;MOVE BIT 0 TO LOCATION FLOATR
003544  000402                        BR      NEXT            ;BRANCH OVER THE ASL
003546  006337  005772        1$:     ASL     FLOATR          ;ROTATE THE TEST BIT TO THE LEFT
003552  005037  037776        NEXT:   CLR     37776           ;CLEAR TEST LOCATION
003556  005037  001330                CLR     PCPUER          ;CLEAR ERROR LOCATION
003562  010210                        MOV     R2,(R0)         ;TRY TO LOAD TEST CELL THROUGH MAP
003564  023702  037776                CMP     37776,R2        ;SEE IF TEST LOCATION WAS LOADED
003570  001414                        BEQ     2$              ;BRANCH IF IT WAS LOADED
003572  004737  005076                JSR     PC,CHKLMA       ;GO SEE IF USER SAYS THIS IS AN 11/24 WITH UB MEMORY
003576  177736  177734                .WORD   LMAHI,LMALOW    ;ADDRESSES OF LMA REGISTERS
003602  000407                        BR      2$              ;RETURN IS HERE IF OK
003604  005737  001330                TST     PCPUER          ;SEE IF A TIMEOUT OCCURED
003610  001402                        BEQ     1$              ;BRANCH OVER SPECIAL STACK PUSH IF NOT
003612  013743  001174                MOV     $TMP0,-(R3)     ;PUSH REGISTER NUMBER THAT TIMED OUT ON SPECIAL ST^CK
003616  062716  000002        1$:     ADD     #2,(SP)         ;CORRECT PC RETURN FOR LOAD FAILURE INDICATION
003622  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.
                                        ;*
                                        ;*****************************************************************
003624  056637  000002  001246  DATEXT: BIS     2(SP),DATAOR    ;SET THE 'OR' PATTERN TO DATAOR
003632  005166  000002                  COM     2(SP)           ;COMPLIMENT THE DATA
003636  046637  000002  001242          BIC     2(SP),DATAND    ;CLEAR THE 'AND' PATTERN TO DATAND
003644  012616                          MOV     (SP)+,(SP)      ;CLEAN THE STACK FOR THE RETURN
003646  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.
                                        ;*
                                        ;;*****************************************************************
003650  056637  000002  001254  PATEXT: BIS     2(SP),PATTOR    ;SET THE 'OR' PATTERN TO PATTOR
003656  005166  000002                  COM     2(SP)           ;COMPLIMENT THE PATTERN
003662  046637  000002  001252          BIC     2(SP),PATAND    ;CLEAR THE 'AND' PATTERN TO PATAND
003670  012616                          MOV     (SP)+,(SP)      ;CLEAN UP STACK FOR RETURN
003672  000207                          RTS     PC              ;RETURN
```

SUBROUTINE TO TAKE PAR AND LOAD 2 WORDS EACH OF ADDROR & ADRAN

```
                                        .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.
                                        ;*
                                        ;;*****************************************************************
003674  010546                  ADREXT: MOV     R5,-(SP)        ;SAVE R5
003676  016605  000004                  MOV     4(SP),R5        ;MOVE PAR CONTENTS TO R5 FOR SHIFTING
003702  072527  177766                  ASH     #-10.,R5        ;SHIFT R5 TO THE RIGHT 10 PLACES
003706  042705  177700                  BIC     #177700,R5      ;CLEAR BITS 15 TO 6
003712  022705  000074                  CMP     #74,R5          ;SEE IF I/O PAGE
003716  001002                          BNE     1$              ;BRANCH IF NOT
003720  012705  000077                  MOV     #77,R5          ;RESET R5 TO 77
003724  010537  001206          1$:     MOV     R5,$TMP5        ;MOVE OBTAINED UPPER 6 BITS TO $TMP5 FOR FUTURE TRANSFER
003730  050537  001240                  BIS     R5,ADDROR+2     ;SET THE 'OR' PATTERN OF UPPER 6 BITS TO ADDROR+2
003734  005105                          COM     R5              ;COMPLIMENT R5
003736  040537  001234                  BIC     R5,ADRAND+2     ;CLEAR THE 'AND' PATTERN OF UPPER 6 BITS TO ADRAND+2
003742  016605  000004                  MOV     4(SP),R5        ;PUT PAR CONTENTS BACK IN R5
003746  013766  001206  000004          MOV     $TMP5,4(SP)     ;MOVE UPPER 6 BITS OF PHYSICAL ADDRESS ON STACK
003754  022737  000074  001206          CMP     #74,$TMP5       ;SEE IF I/O PAGE
003762  001007                          BNE     2$              ;BRANCH IF NOT
003764  012737  000077  005764          MOV     #77,EADRES+2    ;SET 77 IN UPPER ERROR LOCATION
003772  016637  000006  005762          MOV     6(SP),EADRES    ;SET LOWER ADDRESS IN LOWER ERROR LOCATION
004000  000411                          BR      3$              ;BRANCH OVER PREP
004002  042705  176000          2$:     BIC     #176000,R5      ;STRIP OFF UPPER 6 BITS OF PAR CONTENTS
004006  072527  000006                  ASH     #6,R5           ;SHIFT REMAINING BITS 6 PLACES TO THE LEFT
004012  042766  160000  000006          BIC     #160000,6(SP)   ;STRIP OFF PAR PAGE BITS FROM ADDRESS TO FORM OFFSET
004020  060566  000006                  ADD     R5,6(SP)        ;FORM LOWER 16 BITS OF ADDRESS
004024  012605                  3$:     MOV     (SP)+,R5        ;RESTORE R5
004026  056637  000004  001236          BIS     4(SP),ADDROR    ;SET THE 'OR' PATTERN TO ADDROR
004034  005166  000004                  COM     4(SP)           ;COMPLIMENT THE ADDRESS
004040  046637  000004  001232          BIC     4(SP),ADRAND    ;CLEAR THE 'AND' PATTERN TO ADRAND
004046  005166  000004                  COM     4(SP)           ;RETURN ADDRESS TO ITS ORIGINAL STATE
004052  022737  000077  005764          CMP     #77,EADRES+2    ;SEE IF I/O PAGE IN UPPER LOCATION
004060  001406                          BEQ     4$              ;BRANCH TO EXIT IF SO
004062  016637  000004  005762          MOV     4(SP),EADRES    ;PUT LOWER 16 BITS IN ERROR STATUS LOCATION
004070  016637  000002  005764          MOV     2(SP),EADRES+2  ;PUT UPPER 6 BITS IN ERROR STATUS LOCATION
004076  012666  000002          4$:     MOV     (SP)+,2(SP)     ;PUT RETURN ADDRESS WHERE IT BELONGS
004102  005726                          TST     (SP)+           ;CLEAN STACK
004104  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
                                        ;;**************************************************************
004106  005737  004312          MAPADD: TST     LOEFLG          ;SEE IF THIS ENTRY WAS AN ERROR
004112  001070                          BNE     3$              ;GO CONTINUE TEST
004114  011646                          MOV     (SP),-(SP)      ;MOVE RETURN UP ONE NOTCH
004116  062766  000004  000002          ADD     #4,2(SP)        ;CREATE ADDRESS ON STACK FOR FINAL RETURN
004124  012703  004314                  MOV     #R3STAK,R3      ;SET THE SPECIAL STACK POINTER
004130  012704  000013                  MOV     #13,R4          ;SET THE LOOP COUNTER
004134  012737  004162  001106          MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
004142  005077  175134                  CLR     @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
004146  012377  175126          1$:     MOV     (R3)+,@LREGL    ;LOAD LOWER BITS OF MAPPING REG
004152  013737  001256  172354          MOV     LOWEST,KIPAR6   ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
004160  012300                          MOV     (R3)+,R0        ;SELECT PAR6, OFFSET IS AUGEND
004162  011001                  2$:     MOV     (R0),R1         ;READ LOCATION DEFINED BY 4TH WORD IN TABLE
004164  012337  001310                  MOV     (R3)+,UBM24L    ;MOVE ANTICIPATED PHYSICAL ADDRESS TO UBM24L
004170  020137  001310                  CMP     R1,UBM24L       ;SEE IF THE MAP'S FETCH WAS CORRECT
004174  001443                          BEQ     4$              ;BRANCH IF FETCHED DATA MATCHES ADDRESS
004176  004737  005076                  JSR     PC,CHKLMA       ;GO SEE IF USER SAYS THIS IS AN 11/24 WITH UB MEMORY
004202  177736  177734                  .WORD   LMAHI,LMALOW    ;ADDRESSES OF LMA REGISTERS
004206  000436                          BR      4$              ;RETURN IS HERE IF OK
004210  032777  004000  174720          BIT     #BIT11,@SWR     ;CHECK TO SEE IF 11/24 WITH UB MEMORY ONLY
004216  001406                          BEQ     25$             ;BRANCH IF NOT
004220  017737  175064  001176          MOV     @UBM24L,$TMP1   ;GET EXPECTED DATA FOR ERROR CALL
004226  004737  005000                  JSR     PC,PTMP2        ;GO PREPARE $TMP2 FOR ERROR CALL
004232  000404                          BR      26$             ;GO CALL ERROR
004234  011037  001176          25$:    MOV     (R0),$TMP1      ;GET EXPECTED DATA FOR ERROR CALL
004240  010137  001200                  MOV     R1,$TMP2        ;GET DATA FROM R1 FOR ERROR CALL
004244  162703  000002          26$:    SUB     #2,R3           ;ANTICIPATE ERROR LOOPING BY UNDOING AUTOINC
004250  013737  001310  005762          MOV     UBM24L,EADRES   ;PUT ADDRESS IN EADRES FOR ERROR CALL
004256  005037  005764                  CLR     EADRES+2        ;CLEAR UPPER LOCATION
004262  011646                          MOV     (SP),-(SP)      ;PUT AN EXTRA RETURN ON FOR POSSIBLE ERROR LOOPING
004264  012737  000001  004312          MOV     #1,LOEFLG       ;SET FLAG SHOWING ERROR CALL WAS CALLED
004272  000207                          RTS     PC              ;EXIT TO ERROR CALL AT TEST
004274  062703  000002          3$:     ADD     #2,R3           ;RESTORE AUTOINC, ERROR LOOPING NOT DONE
004300  062706  000002                  ADD     #2,SP           ;CLEAN EXTRA RETURN OFF STACK - LOOPING NOT DONE
004304  077460                  4$:     SOB     R4,1$           ;SUBTRACT 1 FROM R4 AND BRANCH IF NOT 0
004306  005726                          TST     (SP)+           ;EXPOSE NEXT TEST RETURN ADDRESS
004310  000207                          RTS     PC              ;EXIT TO NEXT TEST
004312  000000          LOEFLG: .WORD   0               ;LOOP ON ERROR FLAG LOCATION
```

SUBROUTINE TESTING RELOCATION ADDER

```
2300                                                ;DATA IN THE R3STAK DATA BANK BELOW IS ARRANGED IN THE FOLLOWING ORDER:
2301                                                ;>>>NOTE<<<: THE 'OFFSET' COLUMN IS NOT IN THE STACK DUE TO THE DELIMITER
2302                                                ;          (;) BETWEEN THE EXPECTED ADDRESS AND THE OFFSET VALUES
2303                                                ;               !VRTUAL!PHYSCL!
2304                                                ;        BASE   !ADDRES!ADDRES!OFFSET     R4 VALUE
2305                                                ;                      !EXPCTD!
2306 004314  060000  140000  060000  R3STAK:  .WORD   060000,140000,060000;000000     R4=13
2307 004322  052524  145252  057776           .WORD   052524,145252,057776;005252     R4=12
2308 004330  045252  152524  057776           .WORD   045252,152524,057776;012524     R4=11
2309 004336  050420  150420  061040           .WORD   050420,150420,061040;010420     R4=10
2310 004344  054630  144210  061040           .WORD   054630,144210,061040;004210     R4=7
2311 004352  044210  154630  061040           .WORD   044210,154630,061040;014630     R4=6
2312 004360  056734  142104  061040           .WORD   056734,142104,061040;002104     R4=5
2313 004366  042104  156734  061040           .WORD   042104,156734,061040;016734     R4=4
2314 004374  057776  141042  061040           .WORD   057776,141042,061040;001042     R4=3
2315 004402  041042  157776  061040           .WORD   041042,157776,061040;017776     R4=2
2316 004410  057776  140002  060000           .WORD   057776,140002,060000;000002     R4=1
```

```
                                        .SBTTL  SUBROUTINE TO TEST CARRY PROP OF MAP'S RELOC ADDER
                                ;;*****************************************************************
                                ;*      THIS SUBROUTINE IS USED BY TESTS 14 AND 17.  CODE CALLING THIS SUB-
                                ;*      ROUTINE IS AS FOLLOWS:
                                ;*      CLR     $TMP4           ;CLEAR $TMP4 - USED IN ERROR RETURN
                                ;*LAB:  JSR     PC,TCPMRA       ;GO DO THE TEST
                                ;*      ERROR   +211            ;RETURN IS HERE IF AN ERROR
                                ;*      BR      LAB             ;BRANCH BACK TO CONTINUE TEST
                                ;;*****************************************************************
004416  005737  004312        TCPMRA: TST     LOEFLG          ;TEST ERROR FLAG TO SEE IF THIS ENTRY IS FROM ERROR
004422  001132                        BNE     4$              ;BRANCH TO CONTINUE TEST IF SO
004424  011646                        MOV     (SP),-(SP)      ;MOVE RETURN ADDRESS UP ONE NOTCH
004426  062766  000004  000002        ADD     #4,2(SP)        ;CREATE CORRECT FINAL RETURN ADDRESS
004434  012737  177777  004674        MOV     #-1,35$         ;INITIALIZE FLAG AS NEGATIVE ONE
004442  005077  174634                CLR     @LREGU          ;CLEAR UPPER 6 BITS OF MAP REG
004446  012777  020000  174624        MOV     #20000,@LREGL   ;LOAD 4K BASE INTO MAP REGISTER
004454  012701  100100                MOV     #100100,R1      ;LOAD BITS TO SELECT PAR 4, OFFSET 100
004460  012700  150000                MOV     #150000,R0      ;LOAD BITS TO SELECT PAR 6, OFFSET 2K
004464  012737  000277  172350        MOV     #277,KIPAR4     ;START WITH PHYSICAL 6K
004472  013737  001256  172354        MOV     LOWEST,KIPAR6   ;LOAD PAR 6 WITH MAP REG'S ADDR
004500  012737  004552  001106        MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
004506  005037  001330        1$:     CLR     PCPUER          ;CLEAR TIME OUT FLAG
004512  062737  010000  001310        ADD     #10000,UBM24L   ;FORM EXPECTED LMA FOR POSSIBLE USE
004520  001002                        BNE     15$             ;BRANCH AROUND UBM24U INCREMENT IF NOT ZERO
004522  005237  001312                INC     UBM24U          ;INCREMENT UPPER LOCATION
004526  012737  000020  001326  15$:  MOV     #20,CPUEXP      ;EXPECTING A UNIBUS TIME OUT DURING TEST
004534  013710  001364                MOV     DATA,(R0)       ;THIS LOAD WILL TIME OUT WHEN YOU HAVE REACHED THE TOP
                                ;OF MEMORY IT SELECTS PAR 6 WHICH WILL PUT ADDR <XXX1>0000 ON THE UNIBUS. THE X'S WILL
                                ;SELECT THE LOWEST USABLE MAPPING REGISTER.  THE DEFAULT CASE IS 00010000, SELECTING
                                ;MAP REGISTER 0.
004540  005037  001326                CLR     CPUEXP          ;CLEAR CPUEXP - DON'T EXPECT TRAPS FOR A WHILE
004544  005737  001330                TST     PCPUER          ;SEE IF THERE WAS MAIN MEMORY
004550  001050                        BNE     3$              ;BRANCH IF NO MAIN MEMORY FROM UNIBUS
004552  012737  000040  001326  2$:   MOV     #NEXMEM,CPUEXP  ;POSSIBLE CACHE NON-EXISTENT MEMORY
004560  011103                        MOV     (R1),R3         ;READ TEST LOCATION VIA FASTBUS
004562  005037  001326                CLR     CPUEXP          ;CLEAR CPUEXP - DON'T EXPECT TRAPS FOR A WHILE
004566  022737  000040  001330        CMP     #NEXMEM,PCPUER  ;WAS THIS CACHE NON-EXISTENT MEMORY
004574  001436                        BEQ     3$              ;BRANCH IF NON-EXISTENT MEMORY
004576  011002                        MOV     (R0),R2         ;READ TEST LOCATION VIA UNIBUS MAP
004600  020203                        CMP     R2,R3           ;COMPARE TEST DATA R2=MAP DATA, R3=FASTBUS DATA
004602  001433                        BEQ     3$              ;BRANCH IF IT WAS THE SAME
004604  004737  005076                JSR     PC,CHKLMA       ;GO CHECK FOR 11/24 WITH UB MEMORY
004610  177736  177734                .WORD   LMAHI,LMALOW    ;ADDRESSES OF LMA REGISTERS
004614  000426                        BR      3$              ;RETURN IS HERE IF OK
004616  032777  004000  174312        BIT     #BIT11,@SWR     ;SEE IF THIS IS AN 11/24 WITH UB MEMORY ONLY
004624  001403                        BEQ     25$             ;BRANCH IF NOT
004626  004737  005000                JSR     PC,PTMP2        ;GO PREPARE $TMP2 FOR ERROR CALL
004632  000402                        BR      26$             ;BRANCH TO CALL ERROR
004634  011037  001200        25$:    MOV     (R0),$TMP2      ;GET RECEIVED DATA FOR ERROR CALL
004640  010337  001176        26$:    MOV     R3,$TMP1        ;GET EXPECTED DATA FOR ERROR CALL
004644  011646                        MOV     (SP),-(SP)      ;PUT AN EXTRA RETURN ADDRESS ON STACK FOR POSSIBLE LOOP
004646  013737  001310  005762        MOV     UBM24L,EADRES   ;LOAD LOWER 16 BITS OF ADDRESS FOR ERROR CALL
004654  013737  001312  005764        MOV     UBM24U,EADRES+2 ;LOAD UPPER 6 BITS OF ADDRESS FOR ERROR CALL
004662  012737  000001  004312        MOV     #1,LOEFLG       ;SET FLAG SHOWING AN ERROR RETURN
004670  000207                        RTS     PC              ;RETURN TO THE ERROR
004672  005227                3$:     INC     (PC)+           ;INCREMENT ONE TIME ENTRANCE FLAG
004674  177777                35$:    .WORD   -1              ;FLAG
```

```
2374 004676  001006                        BNE     5$              ;BRANCH IF I'VE BEEN HERE BEFORE
2375 004700  013737  172350  001356        MOV     KIPAR4,RSIZE    ;SAVE UPPER LIMIT OF MEMORY
2376 004706  000402                        BR      5$              ;BRANCH OVER ERROR LOOP CORRECTION
2377 004710  062706  000002          4$:   ADD     #2,SP           ;POP EXCESS RETURN ADDRESS OFF STACK
2378 004714  062737  000100  001364  5$:   ADD     #100,DATA       ;CHANGE PATTERN FOR NEXT LOAD
2379 004722  062737  000100  172350        ADD     #100,KIPAR4     ;ADD 2K TO PAR4
2380 004730  062777  010000  174342        ADD     #10000,@LREGL   ;ADD 2K TO MAP REGISTER
2381 004736  001263                        BNE     1$              ;BRANCH IF MAP REGISTER NOT ZERO
2382 004740  005277  174336                INC     @LREGU          ;ADD ONE TO UPPER 6 BITS OF MAP REG
2383 004744  022777  000073  174330        CMP     #73,@LREGU      ;SEE IF TOP 128K BLOCK HAS BEEN PASSED
2384 004752  103255                        BHIS    1$              ;BRANCH IF NOT PAST IT
2385 004754  005237  001364                INC     DATA            ;CHANGE DATA PATTERN FOR NEXT PASS
2386 004760  042737  177700  001364        BIC     #177700,DATA    ;CLEAR UPPER 10 BITS OF DATA PATTERN
2387 004766  052737  000300  001364        BIS     #300,DATA       ;START WITH 3XX IN DATA PATTERN
2388 004774  005726                        TST     (SP)+           ;POP STACK EXPOSING FINAL RETURN ADDRESS
2389 004776  000207                        RTS     PC              ;EXIT
```

SUBROUTINE TO OBTAIN CONTENTS OF LMA CONTENTS LOCATION

```
                                                .SBTTL  SUBROUTINE TO OBTAIN CONTENTS OF LMA CONTENTS LOCATION
2391 005000  013705  177734          PTMP2:     MOV     LMALOW,R5       ;MOVE LMA LOW REGISTER CONTENTS TO R5 FOR SHIFTING
2392 005004  072527  177772                     ASH     #-6,R5          ;SHIFT PHYSICAL UPPER 3 BITS TO THE RIGHT 6 PLACES
2393 005010  042705  176177                     BIC     #176177,R5      ;CLEAR ALL BUT THE THREE SHIFTED BITS
2394 005014  010546                             MOV     R5,-(SP)        ;SAVE THIS NUMBER ON THE STACK
2395 005016  013705  177736                     MOV     LMAHI,R5        ;MOVE LMA HIGH REGISTER CONTENTS TO R5 FOR SHIFTING
2396 005022  072527  000012                     ASH     #10.,R5         ;SHIFT LOWER 6 BITS TO THE LEFT 10 PLACES
2397 005026  062605                             ADD     (SP)+,R5        ;ADD PREVIOUSLY SHIFTED CONTENTS TO R5
2398 005030  013746  172346                     MOV     KIPAR3,-(SP)    ;SAVE PAR3 ON STACK
2399 005034  010537  172346                     MOV     R5,KIPAR3       ;MOVE OBTAINED PAR VALUE TO PAR3
2400 005040  013737  177734  001200            MOV     LMALOW,$TMP2    ;GET ADDRESS MAP CREATED
2401 005046  042737  100000  001200            BIC     #BIT15,$TMP2    ;CLEAR BIT 15 AND
2402 005054  052737  060000  001200            BIS     #60000,$TMP2    ;SET BITS 13 & 14 TO PUT PAR PAGE 3 IN $TMP2
2403 005062  017737  174112  001200            MOV     @$TMP2,$TMP2    ;MOVE CONTENTS OF LOCATION TO $TMP2
2404 005070  012637  172346                     MOV     (SP)+,KIPAR3    ;RESTORE PAR3
2405 005074  000207                             RTS     PC              ;EXIT
```

```
                                        .SBTTL  SUBROUTINE TO CHECK FOR 11/24 WITH UBMEMORY
                                ;;**************************************************************
                                ;*
                                ;*      THIS SUBROUTINE CHECKS THE SWITCH REGISTER TO SEE IF USER STATES THAT
                                ;*      THIS IS AN 11/24 CPU WITH UNIBUS MEMORY ONLY.  IF NOT, AN EXIT IS
                                ;*      EXECUTED WITH THE RETURN BEING UPDATED TO THE 2ND LOCATION AFTER THE
                                ;*      JSR CALL.  IF IT IS, THE LMA HIGH REGISTER IS CHECKED FOR BEING EQUAL
                                ;*      TO THE CONTENTS OF UBM24U, (ASSUMED TO BE PRELOADED. THESE 6 BITS ARE
                                ;*      THE UPPER 6 BITS OF THE MAPPED ADDRESS FORMED BY THE MAP REGISTER
                                ;*      LOGIC), IF NOT, EXIT TO FUDGE RETURN.  IF SO, THE LMA LOW REGISTER IS
                                ;*      CHECKED.  IT IS ASSUMED THAT LOCATION UBM24L CONTAINS THE EXPECTED
                                ;*      PHYSICAL ADDRESS.  IT COMPARES THE LMA LOW REGISTER WITH UBM24L, AND IF
                                ;*      EQUAL, EXECUTES A RETURN WITHOUT THE FUDGING OF THE RETURN ADDRESS,
                                ;*      OTHERWISE IT IS FUDGED.
                                ;*
                                ;;**************************************************************
005076  017637  000000  005134  CHKLMA: MOV     @0(SP),10$      ;MOVE LMAHI ADDRESS TO ACCESS TO 10$
005104  062 16  000002                  ADD     #2,(SP)         ;ADVANCE TO NEXT PARAMETER
005110  017637  000000  005154          MOV     @0(SP),11$      ;MOVE LMALOW ADDRESS TO ACCESS TO 11$
005116  062716  000002                  ADD     #2,(SP)         ;CORRECT RETURN OVER PARAMETER
005122  032777  004000  174006          BIT     #BIT11,@SWR     ;SEE IF USER SAYS THIS IS AN 11/24 WITH UB MEMORY
005130  001413                          BEQ     1$              ;BRANCH OUT IF NOT
005132  013746                          MOV     @(PC)+,-(SP)    ;MOVE LMA HIGH REGISTER CONTENTS TO STACK
005134  000000                  10$:    .WORD   0               ;LOCATION FOR ADDRESS TO ACCESS
005136  042716  177700                  BIC     #177700,(SP)    ;CLEAR ALL BUT LOWER 6 BITS (UPPER 6 BITS OF ADDRESS)
005142  023726  001312                  CMP     UBM24U,(SP)+    ;SEE IF UPPER 6 BITS ARE AS EXPECTED
005146  001004                          BNE     1$              ;BRANCH TO PREPARE FOR 2ND RETURN LOCATION IF NOT
005150  023737  001310                  CMP     UBM24L,@(PC)+   ;SEE IF EXPECTED DATA WAS CLOCKED PROPERLY
005154  000000                  11$:    .WORD   0               ;LOCATION FOR ADDRESS TO ACCESS
005156  001402                          BEQ     2$              ;BRANCH AROUND STACK RETURN FUDGE IF OK
005160  062716  000002          1$:     ADD     #2,(SP)         ;FUDGE RETURN OVER NON-ERROR BRANCH
005164  000207                  2$:     RTS     PC              ;EXIT
```

```
                                        .SBTTL  PRETEST DATA SETUP SUBROUTINE
                                ;;*****************************************************************
005166  011600                  PRETST: MOV     (SP),R0         ;MOVE RETURN ADDRESS TO R0
005170  012037  001362                  MOV     (R0)+,NXTTST    ;SAVE STARTING ADDRESS OF NEXT TEST FOR ESCAPE ON PAR ERRORS
005174  012037  001106                  MOV     (R0)+,$LPERR    ;SET LOOP ON ERROR POINTER TO 20$ IN TEST
005200  012037  001100                  MOV     (R0)+,$TSTNM    ;SETUP TEST NUMBER AND CLEAR THE ERROR FLAG
005204  013777  001100  173726          MOV     $TSTNM,@DISPLAY ;DISPLAY TEST NUMBER FOR ALL TO SEE
005212  010016                          MOV     R0,(SP)         ;FUDGE RETURN OVER PARAMETERS
005214  010037  001104                  MOV     R0,$LPADR       ;SET LOOP ON TEST POINTER TO START OF TEST
005220  012737  000001  001212          MOV     #1,$TIMES       ;RESET ITERATIONS COUNTER TO 1
005226  000207                          RTS     PC              ;RETURN TO BEGIN TEST
```

```
                                        .SBTTL  DISABLE CHECK SUBROUTINE
                                ;;**************************************************************
                                ;;**************************************************************
                                ;*
                                ;*      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.
005230  105737  020034          DSABLD: TSTB    $ENV            ;TEST APT STATUS
005234  001411                          BEQ     1$              ;BRANCH IF NOT APT
005236  105737  020035                  TSTB    $ENVM           ;DOES APT SAY TO SIZE
005242  100006                          BPL     1$              ;BRANCH TO EXIT IF NOT
005244  033715  005772                  BIT     FLOATR,(R5)     ;TEST DISABLE STATUS
005250  001403                          BEQ     1$              ;BRANCH IF IT SHOULD BE DISABLED
005252  011537  001176                  MOV     (R5),$TMP1      ;MOVE CONTENTS OF DEVICE DESCRIPTOR WORD TO $TMP1
005256  000402                          BR      2$              ;BRANCH OVER RETURN CORRECTION
005260  062716  000002          1$:     ADD     #2,(SP)         ;CHOCOLATE FUDGE RETURN OVER ERROR CALL
005264  000207                  2$:     RTS     PC              ;RETURN WITHOUT CORRECTING STACK SO ERROR WILL CALL
```

```
                                        .SBTTL  ENABLE CHECK SUBROUTINE
                                ;;**************************************************************
                                ;*      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 ENABLED.
005266  105737  020034          ENABLD: TSTB    $ENV            ;TEST APT STATUS
005272  001411                          BEQ     1$              ;BRANCH IF NOT APT
005274  105737  020035                  TSTB    $ENVM           ;DOES APT SAY TO SIZE
005300  100006                          BPL     1$              ;BRANCH TO EXIT IF NOT
005302  033715  005772                  BIT     FLOATR,(R5)     ;TEST DISABLE STATUS
005306  001003                          BNE     1$              ;BRANCH IF IT SHOULD BE ENABLED
005310  011537  001176                  MOV     (R5),$TMP1      ;MOVE CONTENTS OF DEVICE DESCRIPTOR WORD TO $TMP1
005314  000402                          BR      2$              ;BRANCH OVER RETURN CORRECTION
005316  062716  000002          1$:     ADD     #2,(SP)         ;VANILLA FUDGE RETURN OVER ERROR CALL
005322  000207                  2$:     RTS     PC              ;RETURN WITHOUT CORRECTING STACK SO ERROR WILL CALL
```

```
                                        .SBTTL  CACHE TEST IN SUBROUTINE FORM
                                ;;**************************************************************
                                        .ENABL  LSB
005324  042737  000001  177572  CASHSR: BIC     #BIT00,MMR0     ;TURN OFF RELOCATION
005332  052737  000400  177746          BIS     #BIT08,CACHE    ;FLUSH CACHE TO INVALIDATE ALL CACHE LOCATIONS
005340  032737  010000  177746  1$:     BIT     #BIT12,CACHE    ;WAIT TILL DONE
005346  001374                          BNE     1$
005350  013702  000000                  MOV     0,R2            ;SAVE ADDR. 0 CONTENTS
005354  005037  000000                  CLR     0               ;0'S TO MAIN MEMORY LOCATION 0.
005360  005003                          CLR     R3              ;CLEAR ERROR FLAG
005362  012704  177777                  MOV     #-1,R4          ;ALL 1'S TO R4
005366  013700  000114                  MOV     CTRAPV,R0       ;SAVE VECTORS
005372  013701  000116                  MOV     CTRAPS,R1
005376  012737  005522  000114          MOV     #3$,CTRAPV      ;SETUP FOR CACHE TRAP
005404  012737  000340  000116          MOV     #340,CTRAPS
005412  112737  000002  177750          MOVB    #2,MAINT        ;HODO ALLOWS CACHE UPDATES AND CLOCKING OF
                                                                ;PARITY INFO TO INTERRUPT LOGIC ONLY DURING
                                                                ;THE DESTINATION ACCESS OF AN INSTRUCTION.
005420  012737  000015  177746          MOV     #15,CACHE       ;NO UCB SO AS TO WRITE CACHE STORES
005426  005737  040000                  TST     40000           ;UPDATE CACHE LOCATION 0000 WITH CORRECT
                                                                ;PARITY STORAGE
005432  052737  000100  177746          BIS     #BIT06,CACHE    ;ALLOW WRITE WRONG PARITY DATA TO LO & HI BYTE
                                                                ;PARITY STORE.
005440  005737  000000                  TST     0               ;READ UPDATE TO CACHE LOCATION 0000;
                                                                ;WRITE WRONG PARITY TO HI/LO BYTE PARITY STORES
005444  042737  000100  177746          BIC     #BIT06,CACHE    ;DISABLE WWPD
005452  005037  177744                  CLR     CMPE            ;CLEAR CMPE AND PARITY DETECT LOGIC
005456  142737                          BICB    (PC)+,@(PC)+    ;ALLOW INT & ENABLE LOW CACHE (1ST       ;DPM002
005460  000000                  CASH1:  .WORD   0               ;LOCATION LOADED BY THE TEST            ;DPM002
005462  177746                          .WORD   CACHE           ;ADDRESS OF CACHE LOCATION              ;DPM002
005464  122727                          CMPB    (PC)+,(PC)+     ;WHICH TEST                             ;DPM002
005466     002                          .BYTE   2               ;TEST 2 IS TESTED FOR                   ;DPM002
005467     000                  CASH2:  .BYTE   0               ;THIS LOCATION LOADED BY THE TEST       ;DPM002
005470  001405                          BEQ     2$              ;GO TO 2ND TEST SECTION IF 2ND TEST EXECUTING
005472  005737  000000                  TST     0               ;READ HIT LO & HI BYTE PARITY CHECK GENERATORS
                                                                ;WILL DETECT  WRONG PARITY AND THE PARITY
                                                                ;ERROR WILL BE CLOCKED TO INTERRUPT LOGIC
005476  000240                          NOP                     ;NEEDED FOR 11/44
005500  005203                          INC     R3              ;INDICATE THAT TRAP DID NOT OCCUR
005502  000410                          BR      4$              ;BRANCH OVER STACK CORRECTION AND 2ND TEST SECTION
005504  052737  000200  177746  2$:     BIS     #BIT07,CACHE    ;ALLOW FOR ABORT
005512  011304                          MOV     (R3),R4         ;READ HIT LO & HI BYTE PARITY CHECK GENERATORS WILL
                                ;DETECT WRONG PARITY USING HODO AND SOURCE MODE FOR READING LOCATION 0 WILL
                                ;INHIBIT PARITY ERROR FROM BEING CLOCKED TO INTERRUPT LOGIC.  HOWEVER, THE PARITY
                                ;ERROR SIGNAL WILL CAUSE THE ABORT SIGNAL TO BE ASSERTED.  THE ABORT SIGNAL WILL
                                ;BECAUSE CMPE<15> TO BE SET.  THIS INSTRUCTION SHOULD BE ABORTED
005514  000240                          NOP                     ;NEEDED IN AN 11/44 TO ALLOW 1 INSTRUCTION BEFORE ABORT
005516  005203                          INC     R3              ;INDICATE NO TRAP OCCURED
005520  000401                          BR      4$              ;BRANCH OVER STACK CORRECTION
005522  022626                  3$:     CMP     (R6)+,(R6)+     ;READJUST STACK DUE TO INTERRUPT
005524  005037  177744          4$:     CLR     CMPE            ;CLEAR CMPE
005530  012737  001015  177746          MOV     #1015,CACHE     ;DISABLE CACHE
005536  105037  177750                  CLRB    MAINT           ;DISABLE MAINT. MODE
005542  010237  000000                  MOV     R2,0            ;RESTORE LOCATION 0
005546  010037  000114                  MOV     R0,CTRAPV       ;RESTORE CACHE INTERRUPT VECTORS
005552  010137  000116                  MOV     R1,CTRAPS
005556  052737  000400  177746          BIS     #BIT08,CACHE    ;BEFORE LEAVING TEST FLUSH CACHE TO ELIMINATE EFFECTS OF WWP
```

```
2538 005564  032737  010000  177746  5$:     BIT     #BIT12,CACHE    ;WAIT TILL DONE
2539 005572  001374                          BNE     5$
2540 005574  000207                          RTS     PC              ;EXIT
2541                                         .DSABL  LSB
2542                                         .SBTTL  SUBROUTINE TO PREPARE AND CHECK DATA PATTERN
2543                                 ;;*************************************************************
2544 005576  011504                 CHKPAT: MOV     (R5),R4         ;MOVE NEXT COUNT PATTERN TO R2
2545 005600  043704  005774                  BIC     MASK1,R4        ;USE MASK1 PRE-LOADED FOR PROPER LOADING
2546 005604  010410                          MOV     R4,(R0)         ;LOAD MAP REGISTER WITH COUNT PATTERN
2547 005606  011504                          MOV     (R5),R4         ;RELOAD PATTERN
2548 005610  043704  005776                  BIC     MASK2,R4        ;USE THE 2ND MASK TO CONSTRUCT EXPECTED VALUE
2549 005614  020410                          CMP     R4,(R0)         ;COMPARE EXPECTED WITH RECEIVED
2550 005616  001402                          BEQ     1$              ;BRANCH IF DATA IS OK
2551 005620  062716  000002                  ADD     #2,(SP)         ;FUDGE RETURN TO SHOW ERROR
2552 005624  000207                 1$:     RTS     PC              ;EXIT
```

```
                                        .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
                                ;;**********************************************************************
                                ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY, CHANGE IT TO
                                ;*BINARY AND PUT THE NUMBER ON THE STACK.
                                ;*CALL:
                                ;*      RDOCT                           ;READ AN OCTAL NUMBER
                                ;*      MOV     (SP)+,LOCATION          ;POP THE INPUTED NUMBER OFF STACK
                                ;*                                      ;HIGH ORDER BITS ARE IN $HIOCT
                                
005626  011646                  $RDOCT: MOV     (SP),-(SP)              ;MAKE ROOM FOR THE
005630  016666  000004  000002          MOV     4(SP),2(SP)             ;OCTAL NUMBER
005636  010046                          MOV     R0,-(SP)                ;SAVE R0
005640  010146                          MOV     R1,-(SP)                ;SAVE R1
005642  010246                          MOV     R2,-(SP)                ;SAVE R2
005644  104407                  1$:     RDLIN                           ;READ THE OCTAL NUMBER
005646  012600                          MOV     (SP)+,R0                ;GET ADDRESS OF 1ST CHARACTER
005650  005001                          CLR     R1                      ;CLEAR R1
005652  005002                          CLR     R2                      ;CLEAR R2
005654  112046                  2$:     MOVB    (R0)+,-(SP)             ;MOVE CHARACTER TO STACK
005656  001415                          BEQ     3$                      ;IF ZERO, EXIT
005660  000241                          CLC                             ;CLEAR THE CARRY BIT
005662  006101                          ROL     R1                      ;SHIFT MSB TO THE CARRY BIT
005664  006102                          ROL     R2                      ;SHIFT IT TO UPPER RECEIVER
005666  000241                          CLC                             ;CLEAR THE CARRY BIT
005670  006101                          ROL     R1                      ;SHIFT MSB TO THE CARRY BIT
005672  006102                          ROL     R2                      ;SHIFT IT TO UPPER RECEIVER
005674  000241                          CLC                             ;CLEAR THE CARRY BIT
005676  006101                          ROL     R1                      ;SHIFT MSB TO THE CARRY BIT
005700  006102                          ROL     R2                      ;SHIFT IT TO UPPER RECEIVER
005702  042716  177770                  BIC     #177770,(SP)            ;STRIP ALL BUT BINARY EQUIVALENT
005706  052601                          BIS     (SP)+,R1                ;SET THE BINARY EQUIVALENT TO LOWER RECEIVER
005710  000761                          BR      2$                      ;BRANCH BACK
005712  005726                  3$:     TST     (SP)+                   ;CLEAN TERMINATOR FROM STACK
005714  010166  000012                  MOV     R1,12(SP)               ;SAVE THE RESULT
005720  010237  005734                  MOV     R2,$HIOCT
005724  012602                          MOV     (SP)+,R2                ;RESTORE R2
005726  012601                          MOV     (SP)+,R1                ;RESTORE R1
005730  012600                          MOV     (SP)+,R0                ;RESTORE R0
005732  000002                          RTI                             ;RETURN
005734  000000                  $HIOCT: .WORD   0                       ;HIGH ORDER BITS GO HERE
```

DATA TABLES AND ASCII STRINGS USED IN THIS DIAGNOSTIC

```
                                                .SBTTL  DATA TABLES AND ASCII STRINGS USED IN THIS DIAGNOSTIC
005736  005742  006116                DTMS:     .WORD   DTMSG,DFMSG     ;POINTER TO DATA VARIABLE ADDRESSES
005742  001266  001270  001272        DTMSG:    .WORD   MMRLOW,MMRHI,UBRLOW,UBRHI,$TESTN,0 ;DATA VARIABLES TO BE PRINTED
005756  000000                        CHARCT:   .WORD   0       ;THIS LOCATION HOLDS CHARACTERS INPUTED DURING THE TYPE ROUTINE
005760  000000                        EXTOUT:   .WORD   0       ;THIS LOCATION STORES THE OUTPUT OF THE EXTRACTION ROUTINE
005762  000000  000077                EADRES:   .WORD   0,77    ;LOCATIONS FOR STORING 22 BITS OF THE UBMAP REGISTER ADDRESS
005766  000000  000077                EADRS2:   .WORD   0,77    ;LOCATIONS FOR STORING ANOTHER UBMAP REGISTER ADDRESS
005772  000000                        FLOATR:   .WORD   0       ;LOCATION TO HOLD BIT TO FLOAT TO TEST DDW'S STATUS
005774  000000                        MASK1:    .WORD   0       ;PRE-LOADED BY THE TEST WITH THE 1ST BIT-MASK
005776  000000                        MASK2:    .WORD   0       ;PRE-LOADED BY THE TEST WITH THE 2ND BIT-MASK
006000                                SPECST:   .BLKW   40
        000002                        .RADIX  2               ;THIS ENABLES YOU TO SEE THE BIT PATTERNS BELOW
006100  177777                        PATRNS:   .WORD   1111111111111111        ;ALL BITS SET
006102  000000                                  .WORD   0000000000000000        ;ALL BITS CLEAR
006104  125252                                  .WORD   1010101010101010        ;ODD BITS SET, EVEN BITS CLEAR
006106  052525                                  .WORD   0101010101010101        ;EVEN BITS SET, ODD BITS CLEAR
006110  031463                                  .WORD   0011001100110011        ;ALTERNATING PAIRS OF BITS SET
006112  007417                                  .WORD   0000111100001111        ;ALTERNATING GROUPS OF 4 BITS SET
006114  000377                                  .WORD   0000000011111111        ;LOWER BYTE SET, UPPER BYTE CLEAR
        000010                        .RADIX  8               ;THIS RETURNS MODE BACK TO OCTAL
006116     000     000     000        DFMSG:    .BYTE   0,0,0,0,0       ;ALL NUMBERS ARE TO BE PRINTED IN OCTAL
006123     200     125     116        UBMAVA:   .ASCIZ  <CRLF>?UNIBUS MEMORY AVAILABLE = ?
006157     123     127     122        NEWSWR:   .ASCIZ  ?SWR INPUT ?
006172     040     113     200        UBMEND:   .ASCIZ  ? K?<CRLF>
```

DATA TABLES AND ASCII STRINGS USED IN THIS DIAGNOSTIC

```
2617 006176  200  104  111  CPUMSG: .ASCIZ  <CRLF>?DIAGNOSTIC HAS DETERMINED THAT CPU IS AN 11/ 4?<CRLF>
2618 006257  101  103  103  TOMSG:  .ASCII  ?ACCESSING MEMORY THROUGH THE FOLLOWING REGISTER(S)?<CRLF>
2619 006342  103  101  125          .ASCIZ  ?CAUSED TIME OUTS (THEY MIGHT BE UNEXPECTED)?<CRLF>
2620 006417  124  110  105  NOMORE: .ASCII  ?THERE ARE STILL MORE ERRORS BUT WILL NOT BE TYPED.  EACH?<CRLF>
2621 006510  105  122  122          .ASCIZ  ?ERROR WILL BE COUNTED AND NUMBER PRINTED AT THE END-OF-PASS?<CRLF>
2622 006605  124  110  111  BADCPU: .ASCII  ?THIS DIAGNOSTIC IS DESIGNED FOR AN 11/24 OR 11/44.?<CRLF>
2623 006670  104  111  101          .ASCIZ  ?DIAGNOSTIC DOES NOT INTERPRET CPU AS EITHER ONE.?<CRLF>
2624 006752  111  123  040  MRQUES: .ASCIZ  ?IS THERE A MAP REGISTER MODULE IN THIS CPU (Y OR <CR>)?
2625 007041  040  200  000  GMRMD1: .ASCIZ  ? ?<CRLF> ;PLACE FOR USER INPUT CHARACTER
2626 007044  104  111  101  GMRMOD: .ASCIZ  ?DIAGNOSTIC CHECKS THIS MODULE - INSERT BEFORE RE-RUNNING?<CRLF>
2627 007136  000            CPUTYP: .BYTE   0               ;LOCATION TO STORE THE CPU TYPE NUMBER
2628                                 .EVEN
2629                         ;RESERVE REMAINING AREA FOR PERMANENT PATCHES
2630                         ;THIS AREA MAY GO BEYOND 10000.  USE ONLY THAT AREA UP TO 7776.
2631 007140  000000  000000  000000  PATCH:  .WORD   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2632 007240  000000  000000  000000          .WORD   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

PRE-TESTING SETUP

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

                 010000                         .=10000                 ;START TEST CODE AT ADDRESS 10000 (2K)
010000  012737  000340  177776  START:  MOV     #340,PS                 ;LOCK OUT ALL INTERRUPTS
010006  012700  001100                  MOV     #$CMTAG,R0              ;FIRST LOCATION TO BE CLEARED
010012  012701  000021                  MOV     #<$TKS-$CMTAG>/2,R1;MOVE LOOP COUNTER TO R1
010016  005020                  11$:    CLR     (R0)+                   ;CLEAR MEMORY LOCATION
010020  077102                          SOB     R1,11$                  ;SUBTRACT 1 AND BRANCH IF NOT DONE YET
010022  005037  020022                  CLR     $PASS                   ;INITIALIZE PASS COUNT
010026  012706  001100                  MOV     #STACK,SP               ;INITILAIZE STACK
010032  012700  010054                  MOV     #13$,R0                 ;MOVE ADDRESS OF VECTORS TO R0
010036  012701  000005                  MOV     #5,R1                   ;DO 5 LOADS
010042  012030                  12$:    MOV     (R0)+,@(R0)+            ;MOVE VECTOR TO LOCATION
010044  012730  000340                  MOV     #340,@(R0)+             ;MOVE PRIORITY 7 TO NEXT LOCATION
010050  077104                          SOB     R1,12$                  ;BRANCH BACK IF NOT DONE YET
010052  000417                          BR      14$                     ;BRANCH OVER DATA WORDS
010054  020140  000020  000022  13$:    .WORD   $SCOPE,IOTVEC,IOTVEC+2,$ERROR,EMTVEC,EMTVEC+2,$TRAP,TRAPVE
010074  000036  023462  000024          .WORD   TRAPVE+2,$PWRDN,PWRVEC,PWRVEC+2,$RTRN,TBITVE,TBITVE+2
010112  013737  021516  021510  14$:    MOV     $ENDCT,$EOPCT           ;SETUP END-OF-PROGRAM COUNTER
010120  005037  001214                  CLR     $ESCAPE                 ;CLEAR THE ESCAPE ON ERROR ADDRESS
010124  112737  000001  001113          MOVB    #1,$ERMAX               ;ALLOW ONE ERROR PER TEST
010132  012737  000002  021742          MOV     #RTI,$RTRN              ;SET $RTRN TO AN RTI
010140  012737  010166  000010          MOV     #16$,RESVEC             ;MOVE 16$ (FAILURE) TO RESVEC
010146  012746  000340                  MOV     #340,-(SP)              ;MOVE PRIORITY 7 TO STACK
010152  012746  010160                  MOV     #15$,-(SP)              ;MOVE 15$ (SUCCESS) TO THE STACK
010156  000006                          RTT                             ;TRY TO DO AN RTT
010160  012737  000006  021742  15$:    MOV     #RTT,$RTRN              ;RTT IS LEGAL--SET $RTRN TO AN RTT
010166  012706  001100          16$:    MOV     #STACK,SP               ;RESET STACK IF NECESSARY
010172  012737  000012  000010          MOV     #RESVEC+2,RESVEC;RESET TRAP CATCHER
010200  005037  021750                  CLR     $TBIT                   ;CLEAR "T" BIT SWITCH
010204  012737  010700  001104          MOV     #TST1+2,$LPADR          ;SETUP $LPADR
010212  012737  010700  001106          MOV     #TST1+2,$LPERR          ;SETUP $LPERR
010220  005227  177777                  INC     #-1                     ;FIRST TIME?
010224  001017                          BNE     18$                     ;BRANCH IF NOT
010226  104401  010234                  TYPE    ,17$                    ;TYPE THE TEST TITLE
010232  000414                          BR      18$                     ;BRANCH OVER ASCIZ
010234     103     113     113  17$:    .ASCIZ  'CKKUAD 11/24/44 UBI MAP'
                                        .EVEN
010264  005737  020022          18$:    TST     $PASS                   ;IS THIS THE FIRST PASS?
010270  001140                          BNE     8$                      ;BRANCH IF NOT
010272  013746  000004                  MOV     4,-(SP)                 ;SAVE TIMEOUT VECTOR
010276  012737  010470  000004          MOV     #2$,4                   ;TIMEOUTS TO 104$
010304  013746  000006                  MOV     6,-(SP)                 ;SAVE PS VECTOR
010310  012737  000340  000006          MOV     #340,6                  ;PRIORITY 7
010316  000007                          MFPT                            ;DETERMINE PROCESSOR TYPE
010320  110037  007136                  MOVB    R0,CPUTYP               ;MOVE THE CPU NUMBER TO CPUTYP
010324  105337  007136                  DECB    CPUTYP                  ;MAKE THE 11/44 VALUE ZERO
010330  001021                          BNE     1$                      ;BRANCH TO NEXT TEST IF NOT AN 11/44
010332  022737  177777  177570          CMP     #-1,177570              ;SEE IF SSWR IS TO BE USED              ;DPM002
010340  001004                          BNE     1000$                   ;BRANCH TO SETUP FOR HARDWARE SWR       ;DPM002
010342  012737  000176  001136          MOV     #176,SWR                ;SET UP FOR SSWR                        ;DPM002
010350  000403                          BR      1010$                   ;BRANCH                                 ;DPM002
010352  012737  177570  001136  1000$:  MOV     #177570,SWR             ;SET UP FOR HARDWARE SWITCH REGISTER    ;DPM002
010360  112737  000064  006253  1010$:  MOVB    #'4,CPUMSG+55           ;MOVE ASCII 4 TO LOCATION IN MESSAGE TO PRINT
```

```
2690 010366  104401  006176                    TYPE    ,CPUMSG          ;TYPE THE CPU TYPE HEADER
2691 010372  000446                            BR      6$               ;CONTINUE
2692 010374  122737  000002  007136  1$:       CMPB    #2,CPUTYP        ;SEE IF THIS IS AN 11/24
2693 010402  001034                            BNE     3$               ;BRANCH TO FATAL ERROR MESSAGE PRINTING IF NOT
2694 010404  012737  000176  001136            MOV     #$SSWR,SWR       ;SETUP SOFTWARE SWITCH REGISTER          ;DPM002
2695 010412  112737  000062  006253            MOVB    #'2,CPUMSG+55    ;MOVE ASCII 2 TO LOCATION IN MESSAGE TO PRINT
2696 010420  104401  006176                    TYPE    ,CPUMSG          ;TYPE THE CPU TYPE HEADER
2697 010424  105737  020035                    TSTB    $ENVM            ;IS APT SIZING
2698 010430  100427                            BMI     6$               ;BRANCH IF NO
2699 010432  012737  000176  001136            MOV     #$SSWR,SWR       ;SETUP SOFTWARE SWITCH REGISTER
2700 010440  005737  000176                    TST     $SSWR            ;SEE IF SWR IS NON-ZERO
2701 010444  001021                            BNE     6$               ;BRANCH IF SO
2702 010446  104401  006157                    TYPE    ,NEWSWR          ;TYPE:  'SWR INPUT '
2703 010452  104410                            RDOCT                    ;GO READ USER OCTAL INPUT
2704 010454  012637  000176                    MOV     (SP)+,$SSWR      ;MOVE NEW CONTENTS TO THE SOFTWARE SWR LOCATION
2705 010460  001013                            BNE     6$               ;BRANCH IF NON-ZERO
2706 010462  005237  000176                    INC     $SSWR            ;MAKE SWR NON-ZERO FOR POSSIBLE NEXT RUN
2707 010466  000410                            BR      6$               ;CONTINUE
2708 010470  062706  000004          2$:       ADD     #4,SP            ;CLEAN STACK AFTER TIMEOUT
2709 010474  005237  020014          3$:       INC     $MSGTY           ;TELL APT THIS IS A FATAL ERROR
2710 010500  104401  006605          4$:       TYPE    ,BADCPU          ;TYPE THE BAD CPU MESSAGE
2711 010504  000000                            HALT                     ;FATAL ERROR - THIS DIAGNOSTIC IS WRITTEN
2712                                                                    ;FOR 11/24 AND 11/44 PROCESSORS ONLY
2713 010506  000774                            BR      4$               ;DON'T ALLOW CONTINUE
2714 010510  012637  000006          6$:       MOV     (SP)+,6          ;RESTORE TIMEOUT PS
2715 010514  012637  000004                    MOV     (SP)+,4          ;RESTORE TIMEOUT VECTOR
2716 010520  013746  172346                    MOV     KIPAR3,-(SP)     ;SAVE PAR3
2717 010524  012737  000400  172346            MOV     #400,KIPAR3      ;START WITH ADDRESS 40000
2718 010532  012704  060000                    MOV     #60000,R4        ;ADDRESS PAR3, OFFSET=0
2719 010536  012705  040222                    MOV     #40222,R5        ;DATA TO LOAD TO 40000
2720 010542  010514                  7$:       MOV     R5,(R4)          ;MOVE DATA TO LOCATION
2721 010544  062705  010000                    ADD     #10000,R5        ;MOVE DATA UP 10000
2722 010550  062737  000100  172346            ADD     #100,KIPAR3      ;MOVE ADDRESS UP 10000
2723 010556  022737  007600  172346            CMP     #7600,KIPAR3     ;SEE IF ALL DONE YET
2724 010564  001366                            BNE     7$               ;BRANCH BACK IF NOT
2725 010566  012637  172346                    MOV     (SP)+,KIPAR3     ;RESTORE PAR3
2726 010572  105737  020035          8$:       TSTB    $ENVM            ;IS APT SIZING
2727 010576  100003                            BPL     LOOP             ;BRANCH IF NOT
2728 010600  012737  020036  001136            MOV     #$SWREG,SWR      ;USE APT SWITCH REGISTER
2729 010606  012737  003372  000250    OOP:    MOV     #MMTRAP,MMVEC    ;LOAD MEMORY MANAGEMENT TRAP SERVICE ROUTINE ADDRESS
2730 010614  012737  000340  000252            MOV     #340,MMVEC+2     ;SET PRIORITY SEVEN
2731 010622  012737  003222  000004            MOV     #CPUER,ERRVEC    ;LOAD CPU TRAP SERVICE ROUTINE ADDR
2732 010630  012737  000340  000006            MOV     #340,ERRVEC+2    ;SET PRIORITY SEVEN
2733 010636  005037  001326                    CLR     CPUEXP           ;NOT EXPECTING ANY CPU ERRORS
2734 010642  005037  177572                    CLR     MMR0             ;START IN 16 BIT MAPPING
2735 010646  005037  172516                    CLR     MMR3             ;DISABLE MAP AND 22-BIT MAPPING
2736 010652  012700  177777                    MOV     #-1,R0           ;NEGATIVE ONE USED TO INITIALIZE FLAGS
2737 010656  010037  003224                    MOV     R0,CPFLAG        ;INITIALIZE FLAGS
2738 010662  010037  003032                    MOV     R0,TOFLAG        ;INITIALIZE FLAGS
2739 010666  010037  003374                    MOV     R0,MMFLAG        ;INITIALIZE FLAGS
2740 010672  010037  177766                    MOV     R0,CPUERR        ;CLEAR CPU ERROR REGISTER
```

```
2750                                    .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.
                                        ;*
                                        ;;*****************************************************************
     010676                             TST1:
     010676  000004                             SCOPE
     010700  004737  005166                     JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     010704  011122  011042  000001             .WORD   TST2,20$,1      ;DATA USED BY PRETST
2751 010712  012737  003030  000004             MOV     #TIMEOUT,ERRVEC ;LOAD ERRVEC WITH ROUTINE ADDRESS
2752 010720  105737  007136                     TSTB    CPUTYP          ;TEST TO SEE WHICH CPU IS RUNNING THIS DIAGNOSTIC
2753 010724  001435                             BEQ     3$              ;BRANCH AROUND MAP REGISTER EXISTENCE CHECK IF 11/44
2754 010726  005037  001320                     CLR     ERRCNT          ;CLEAR THE ERROR COUNTER
2755 010732  013702  170200             1$:     MOV     MAPL:  2        ;READ FIRST MAP REGISTER TO R2
2756 010736  005737  001320                     TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
2757 010742  001426                             BEQ     3$              ;BRANCH TO CHECK THEM ALL IF NONE, THEY ARE IN THIS 11/24
2758 010744  105737  020034                     TSTB    $ENV            ;ARE WE RUNNING UNDER APT
2759 010750  100004                             BPL     2$              ;BRANCH IF NOT
2760 010752  005237  020014                     INC     $MSGTY          ;TELL APT THIS IS A FATAL ERROR
2761 010756  000000                             HALT                    ;HALT - FATAL ERROR
2762 010760  000764                             BR      1$              ;TRY AGAIN - RESTART REQUESTED
2763 010762  104401  006752             2$:     TYPE    ,MRQUES         ;ASK USER IF THERE ARE MAP REGISTERS IN THIS 11/24
2764 010766  104406                             RDCHR                   ;GO READ USER INPUT
2765 010770  112637  007041                     MOVB    (SP)+,GMRMD1    ;MOVE THE CHARACTER TO THE PRINTING LOCATION
2766 010774  104401  007041                     TYPE    ,GMRMD1         ;TYPE THE CHARACTER
2767 011000  122737  000131  007041             CMPB    #'Y,GMRMD1      ;SEE IF THIS WAS A 'Y'
2768 011006  001404                             BEQ     3$              ;BRANCH AROUND FATAL MESSAGE IF EQUAL TO A 'Y'
2769 011010  104401  007044                     TYPE    ,GMRMOD         ;TYPE:  'DIAGNOSTIC CHECKS THIS MODULE - INSERT
2770                                            ;       'BEFORE RE-RUNNING'
2771 011014  000000                             HALT                    ;FATAL ERROR - WAIT FOR USER ACTION
2772 011016  000745                             BR      1$              ;TRY AGAIN - RESTART REQUESTED
2773 011020  012700  170200             3$:     MOV     #MAPL0,R0       ;PUT FIRST MAP REGISTER ADDR IN R0
2774 011024  012703  000100                     MOV     #100,R3         ;TEST ALL MAP REGISTERS
2775 011030  005037  001320                     CLR     ERRCNT          ;CLEAR THE ERROR COUNTER
2776 011034  012737  011042  001106             MOV     #20$,$LPERR     ;MAKE SURE $LPERR IS POINTING AT 20$
2777 011042  011002                     20$:    MOV     (R0),R2         ;READ MAP REGISTERS TO R2
2778 011044  032777  001000  170064             BIT     #BIT09,@SWR     ;SEE IF LOOP ON ERROR IS SET
2779 011052  001403                             BEQ     4$              ;BRANCH AROUND SETUP IF NOT
2780 011054  005737  001320                     TST     ERRCNT          ;SEE IF AN ERROR OCCURED
2781 011060  001370                             BNE     20$             ;BRANCH BACK IF SO
2782 011062  005720                     4$:     TST     (R0)+           ;INCREMENT R0 TO NEXT REGISTER LOCATION
2783 011064  077312                             SOB     R3,20$          ;DO ALL OF THEM
2784 011066  012737  003222  000004             MOV     #CPUER,ERRVEC   ;RESTORE CPU TRAP SERVICE ROUTINE ADDRESS TO ERRVEC
2785 011074  012737  011020  001106             MOV     #3$, $LPERR     ;MOVE 3$ TO LOOP ON ERROR FOR ERROR +4 BELOW
2786 011102  005737  001320                     TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
2787 011106  001405                             BEQ     TST2            ;;GO TO NEXT TEST IF NO ERRORS
2788 011110  005337  001110                     DEC     $ERTTL          ;DON'T COUNT SUMMARY AS AN ADDITIONAL ERROR
2789 011114  005337  001320                     DEC     ERRCNT          ;DECREMENT ERRCNT FOR SAME REASON
2790 011120  104004                             ERROR   +4              ;SUMMARY OF MAP REGISTERS THAT TIMED OUT ON READ
```

```
                                .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.
                                ;*
                                ;;*****************************************************************
                                TST2:
      011122                            SCOPE
      011122  000004
      011124  004737  005166            JSR     PC,PRETST       ;GO SET UP PRETEST DATA
      011130  011444  011254  000002    .WORD   TST3,20$,2      ;DATA USED BY PRETST
 2806 011136  005037  001320            CLR     ERRCNT          ;CLEAR THE ERROR COUNT LOCATION
 2807 011142  012737  011254  001106  1$:     MOV     #20$,$LPERR     ;SETUP 20$ AS LOOP ON ERROR INDICATOR (IF NOT ALREADY)
 2808 011150  012700  170200            MOV     #MAPL00,R0      ;MOVE STARTING ADDRESS OF LOWER 16 BITS REGISTER TO R0
 2809 011154  005037  005774            CLR     MASK1           ;MOVE '0' TO MASK1
 2810 011160  012737  000001  005776    MOV     #1,MASK2        ;MOVE '1' TO MASK2
 2811 011166  012701  000040          2$:     MOV     #40,R1          ;DO 40 REGISTERS LOOP COUNTER
 2812 011172  012702  000007          3$:     MOV     #7,R2           ;MOVE PATTERN LOOP COUNTER TO R2
 2813 011176  012705  006100            MOV     #PATRNS,R5      ;MOVE PATTERN START TO R5
 2814 011202  004737  005576          4$:     JSR     PC,CHKPAT       ;GO CHECK PATTERN
 2815 011206  000435                    BR      7$              ;RETURN IS HERE IF DATA IS OK
 2816 011210  011003                    MOV     (R0),R3         ;READ BAD MAP REGISTER DATA INTO R3
 2817 011212  011537  001174            MOV     (R5),$TMP0      ;MOVE PATTERN TO $TMP0
 2818 011216  011546                    MOV     (R5),-(SP)      ;MOVE PATTERN TO STACK FOR SUBROUTINE USE
 2819 011220  004737  003650            JSR     PC,PATEXT       ;GO SET DATA INTO PATTOR AND PATAND
 2820 011224  010346                    MOV     R3,-(SP)        ;MOVE BAD DATA TO STACK FOR SUBROUTINE USE
 2821 011226  004737  003624            JSR     PC,DATEXT       ;SUBROUTINE TO LOAD DATAOR AND DATAND
 2822 011232  043737  005774  001174    BIC     MASK1,$TMP0     ;FORM INPUT PATTERN FOR ERROR PRINTING
 2823 011240  010046                    MOV     R0,-(SP)        ;PUT VIRTUAL ADDRESS ON STACK FOR ADREXT SUBROUTINE
 2824 011242  013746  172356            MOV     KIPAR7,-(SP)    ;PUT PAR CONTENTS ON STACK FOR ADREXT SUBROUTINE
 2825 011246  004737  003674            JSR     PC,ADREXT       ;GO SET DATA IN THE 4 WORDS OF ADDROR AND ADRAND
 2826                                                           ;AND FORM A PHYSICAL 22-BIT ADDRESS
 2827 011252  000403                    BR      5$              ;BRANCH OVER LOOP ON ERROR SECTION
 2828 011254  004737  005576        20$:    JSR     PC,CHKPAT       ;GO TO SUBROUTINE TO CHECK PATTERN
 2829 011260  000401                    BR      6$              ;RETURN IS HERE IF DATA IS OK
 2830 011262  104203                  5$:     ERROR   +203            ;THE BIT PATTERN THROUGH THE MAP REGISTERS FAILED
 2831 011264  032777  001000  167644  6$:     BIT     #BIT9,@SWR      ;SEE IF LOOP ON ERROR IS SET
 2832 011272  001370                    BNE     20$             ;LOOP BACK IF SO
 2833 011274  012737  011172  001106    MOV     #3$,$LPERR      ;MOVE 3$ TO LOOP ON ERROR INDICATOR
 2834 011302  005725                  7$:     TST     (R5)+           ;POINT R5 TO NEXT PATTERN
 2835 011304  077242                    SOB     R2,4$           ;DECREMENT LOOP COUNTER
 2836 011306  062700  000004            ADD     #4,R0           ;POINT TO NEXT MAP REGISTER UNDER TEST
 2837 011312  077151                    SOB     R1,3$           ;DECREMENT LOOP COUNTER AND
 2838 011314  005737  001320            TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
 2839 011320  001416                    BEQ     9$              ;BRANCH IF NO ERRORS
 2840 011322  012737  011142  001106    MOV     #1$,$LPERR      ;MOVE 1$ TO LOOP ON ERROR FOR ERROR +6 OR +7 BELOW
 2841 011330  005337  001110            DEC     $ERTTL          ;SUMMATION ERROR NOT COUNTED AS ANOTHER ERROR
 2842 011334  005337  001320            DEC     ERRCNT          ;SAME AS ABOVE
```

TEST # 2 - BIT PATTERN AND CLEAR TEST OF 40 MAP REGISTERS

```
 2843 011340  022700  170402                CMP     #MAPH37+4,R0    ;SEE IF THIS PASS WAS UPPER 6 BITS
 2844 011344  001402                        BEQ     8$              ;GO SERVICE UPPER 6 BITS ERROR IF SO
 2845 011346  104006                        ERROR   +6              ;SUMMARY OF BIT PATTERN FAILURES, LOWER 16 BITS
 2846 011350  000405                        BR      10$             ;GO SET UP DATA FOR TESTING UPPER 6 BITS
 2847 011352  104007                 8$:    ERROR   +7              ;SUMMARY OF BIT PATTERN FAILURES, UPPER 6 BITS
 2848 011354  000433                        BR      TST3            ;;GO TO NEXT TEST - LOWER 16 BITS ALREADY TESTED
 2849 011356  022700  170402           9$:    CMP     #MAPH37+4,R0    ;SEE IF THIS PASS WAS FOR THE UPPER 6 BITS
 2850 011362  001430                        BEQ     TST3            ;;GO TO NEXT TEST IF IT WAS
 2851 011364  012737  011254  001106  10$:    MOV     #20$,$LPERR     ;RESET LOOP ON ERROR TO 20$
 2852 011372  012700  170202                MOV     #MAPH00,R0      ;MOVE STARTING ADDRESS OF UPPER 6 BITS REGISTER TO R0
 2853 011376  012737  177700  005774        MOV     #177700,MASK1   ;MOVE 1ST MASK TO MASK1
 2854 011404  012737  177700  005776        MOV     #177700,MASK2   ;MOVE 2ND MASK TO MASK2
 2855 011412  005037  001254                CLR     PATTOR          ;CLEAR PATTOR FOR NEXT PASS
 2856 011416  005037  001246                CLR     DATAOR          ;CLEAR DATAOR FOR NEXT PASS
 2857 011422  012737  177777  001252        MOV     #-1,PATAND      ;MOVE -1 TO PATAND FOR NEXT PASS
 2858 011430  012737  177777  001242        MOV     #-1,DATAND      ;MOVE -1 TO DATAND FOR NEXT PASS
 2859 011436  005037  001320                CLR     ERRCNT          ;CLEAR ERROR COUNT FOR NEXT PASS
 2860 011442  000651                        BR      2$              ;GO REACCOMPLISH TEST FOR UPPER 6 BITS
```

```
                                        .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.
                                        ;*
                                        ;;*****************************************************************
     011444                             TST3:
     011444  000004                             SCOPE
     011446  004737  005166                     JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     011452  011762  011616  000003             .WORD   TST4,20$,3      ;DATA USED BY PRETST
2874 011460  042737  000001  177572             BIC     #1,MMR0         ;TURN OFF MEMORY MANAGEMENT
2875 011466  012700  170200                     MOV     #MAPL00,R0      ;LOAD ADDRESS OF MAPL00 IN R0
2876 011472  012702  177700                     MOV     #177700,R2      ;SET UP XOR DATA
2877 011476  012701  170400             1$:     MOV     #MAPH37+2,R1    ;PUT ADDRESS OF MAPH37+2 IN R1
2878 011502  012737  000077  001174             MOV     #77,$TMP0       ;SET UP TEST PATTERN IN $TMP0
2879 011510  004737  002772                     JSR     PC,CLRMAP       ;CLEAR ALL MAP REGISTERS FOR TEST
2880 011514  012703  000100                     MOV     #100,R3         ;SET UP LOOP COUNTER FOR TESTING 40 REGISTERS, 2 WORDS EACH
2881 011520  005037  172516                     CLR     MMR3            ;CLEAR MMR3
2882 011524  012704  000100             2$:     MOV     #100,R4         ;SET UP LOOP COUNTER FOR CHECKING 40 REGISTERS, 2 WORDS EACH
2883 011530  074237  001174                     XOR     R2,$TMP0        ;REVERSE BIT STATES OF $TMP0 IN BITS 7 TO 15
2884 011534  013710  001174                     MOV     $TMP0,(R0)      ;LOAD MAP REGISTER UNDER TEST
2885 011540  012701  170400                     MOV     #MAPH37+2,R1    ;PUT ADDRESS OF MAPH37+2 IN R1
2886 011544  005741                     3$:     TST     -(R1)           ;SEE IF MAP REGISTER IS ZERO
2887 011546  001435                             BEQ     7$              ;GO SEE IF MORE REGISTERS TO TEST IF = 0
2888 011550  005011                             CLR     (R1)            ;CLEAR THE FAILED LOCATION
2889 011552  020100                             CMP     R1,R0           ;SEE IF NON-ZERO REGISTER ADDRESS MATCHES ADDRESS LOADED
2890 011554  001432                             BEQ     7$              ;GO SEE IF MORE REGISTERS TO TEST IF SO
2891 011556  012137  001202                     MOV     (R1)+,$TMP3     ;MOVE FAULTY DATA TO $TMP3 & PREPARE R1 FOR POSSIBLE LOOP
2892 011562  013746  001202                     MOV     $TMP3,-(SP)     ;MOVE FAULTY DATA TO STACK FOR SUBROUTINE USE
2893 011566  013746  172350                     MOV     KIPAR4,-(SP)    ;MOVE PAR4 TO STACK FOR SUBROUTINE USE
2894 011572  004737  003674                     JSR     PC,ADREXT       ;GO SET DATA IN THE 4 WORDS OF ADDROR AND ADRAND
2895                                                                     ;AND FORM A PHYSICAL 22-BIT ADDRESS
2896 011576  013746  172356                     MOV     KIPAR7,-(SP)    ;PUT PAR ON STACK FOR DATEXT SUBROUTINE USE
2897 011602  010146                             MOV     R1,-(SP)        ;PUT VIRTUAL ADDRESS ON STACK FOR DATEXT SUBROUTINE USE
2898 011604  004737  003624                     JSR     PC,DATEXT       ;SUBROUTINE TO LOAD DATAOR AND DATAND
2899 011610  010037  005766                     MOV     R0,EADRS2       ;MOVE ADDRESS IN R0 TO EADRS2 FOR ERROR CALL
2900 011614  000405                             BR      5$              ;BRANCH OVER LOOP ON ERROR SECTION
2901 011616  013710  001174             20$:    MOV     $TMP0,(R0)      ;LOAD MAP REGISTER UNDER TEST
2902 011622  005711                             TST     (R1)            ;SEE IF MAP REGISTER IS ZERO
2903 011624  001402                             BEQ     6$              ;GO SEE IF LOOP ON ERROR IS STILL SET IF ZERO
2904 011626  005011                             CLR     (R1)            ;CLEAR THE FAILED LOCATION
2905 011630  104202                     5$:     ERROR   +202            ;DUAL ADDRESSING ERROR IN THE UNIBUS MAP
2906 011632  032777  001000  167276     6$:     BIT     #BIT09,@SWR     ;SEE IF LOOP ON ERROR IS SET
2907 011640  001366                             BNE     20$             ;BRANCH BACK IF SO
2908 011642  077440                     7$:     SOB     R4,3$           ;BRANCH IF MORE REGISTERS TO CHECK
2909 011644  005020                             CLR     (R0)+           ;CLEAR THE REGISTER JUST TESTED AND POINT TO NEXT REGISTER
2910 011646  005303                             DEC     R3              ;DECREMENT LOOP COUNTER AND
2911 011650  001325                             BNE     2$              ;BRANCH IF MORE REGISTERS TO TEST
2912 011652  005737  001320                     TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
```

TEST # 3 - DUAL ADDRESS LOADS & READS MAP REG'S

```
2913 011656  001405                    BEQ     RELC22          ;GO TO NEXT SECTION IF NO ERRORS
2914 011660  005337  001110              DEC     $ERTTL          ;DON'T COUNT ERROR +5 AS ANOTHER ERROR
2915 011664  005337  001320              DEC     ERRCNT          ;SAME AS ABOVE
2916 011670  104005                    ERROR   +5              ;SUMMARY OF DUAL ADDRESSING ERRORS ON LOADING MAP REGISTERS
```

```
2917 011672  104414                  RELC22: TBITR                   ;RESTORE THE T BIT TO ITS CONDITION
2918                                                                 ;BEFORE THE LAST TEST
2919 011674  012700  077406                  MOV     #77406,R0       ;MAKE THE KERNEL I-SPACE PAGES ALL
2920                                                                 ;4K, UPWARD EXPANDABLE, READ/WRITE
2921 011700  012701  172300                  MOV     #KIPDR0,R1      ;MOVE ADDRESS OF KIPDR0 TO R1
2922 011704  012702  000010                  MOV     #8.,R2          ;LOAD 8 PDR'S
2923 011710  010021                  1$:     MOV     R0,(R1)+        ;KERNEL I-SPACE PAGE X KIPDRX
2924 011712  077202                          SOB     R2,1$           ;SUBTRACT 1 AND BRANCH BACK IF NOT DONE
2925 011714  005000                          CLR     R0              ;CLEAR R0
2926 011716  012701  172340                  MOV     #KIPAR0,R1      ;MOVE ADDRESS OF KIPAR0 TO R1
2927 011722  012702  000006                  MOV     #6,R2           ;LOAD FIRST 6 PAR'S
2928 011726  010021                  2$:     MOV     R0,(R1)+        ;MAP KIPARX TO NEXT PHYSICAL PAGE
2929 011730  062700  000200                  ADD     #200,R0         ;MAP R0 TO NEXT PAGE
2930 011734  077204                          SOB     R2,2$           ;SUBTRACT 1 AND BRANCH BACK IF NOT DONE
2931 011736  012721  170000                  MOV     #170000,(R1)+   ;MAP KIPAR6 TO UNIBUS
2932 011742  012721  177600                  MOV     #177600,(R1)+   ;MAP KIPAR7 TO I/O PAGE
2933 011746  012737  000001  177572          MOV     #BIT0,MMR0      ;ENABLE FULL 18-BIT MAPPING
2934 011754  012737  000020  172516          MOV     #BIT4,MMR3      ;ENABLE 22-BIT MAPPING
2935                                 ;*      AT THIS POINT 22-BIT RELOCATION FROM MEMORY MANAGEMENT
2936                                 ;*      IS ENABLED, WITH THE KIPAR'S MAPPED TO PHYSICAL 0-24K.
2937                                 ;*      KIPAR6 IS MAPPED TO THE UNIBUS (170000) AND
2938                                 ;*      KIPAR7 IS MAPPED TO THE I/O PAGE (177600).
```

```
2949                                    .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.
                                        ;*
                                        ;;*****************************************************************
     011762                             TST4:
     011762  000004                             SCOPE
     011764  004737  005166                     JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     011770  012242  012126  000004             .WORD   TST5,20$,4      ;DATA USED BY PRETST
2950 011776  013737  172354  001200             MOV     KIPAR6,$TMP2    ;SAVE KIPAR6 FOR RESTORATION LATER
2951 012004  012737  177702  172354             MOV     #177702,KIPAR6  ;PUT MAP REGISTER 0 ADDR IN PAR6
2952 012012  012702  175254                     MOV     #175254,R2      ;PATTERN FOR TESTING.
2953 012016  010237  170200                     MOV     R2,MAPL0        ;LOAD MAP REGISTER 0
2954 012022  012700  004000                     MOV     #BIT11,R0       ;SET BIT 11 TO FLOAT THROUGH PAR6
2955 012026  012704  140000                     MOV     #140000,R4      ;VIRT.ADDR. TO SELECT PAR6
2956 012032  074037  172354             1$:     XOR     R0,KIPAR6       ;CHANGE A BIT OF MAP REGISTER 0'S ADDP
2957 012036  010046                             MOV     R0,-(SP)        ;SAVE R0 ON STACK
2958 012040  013746  172354                     MOV     KIPAR6,-(SP)    ;SAVE KIPAR6 ON STACK
2959 012044  012737  000020  001326             MOV     #TIMOUT,CPUEXP  ;EXPECTING CPU TIME OUT ON UNIBUS
2960 012052  011401                             MOV     (R4),R1         ;READ LOCATION POINTED TO BY PAR6
2961 012054  005037  001326                     CLR     CPUEXP          ;CLEAR CPIUEXP - DON'T EXPECT TRAPS FOR A WHILE
2962 012060  020201                             CMP     R2,R1           ;SEE IF DATA FETCHED MATCHES PATTERN
2963 012062  001047                             BNE     4$              ;BRANCH IF NOT SAME
2964 012064  012737  000020  001326             MOV     #TIMOUT,CPUEXP  ;EXPECTING CPU TIME OUT ON UNIBUS
2965 012072  005014                             CLR     (R4)            ;TRY TO CLEAR THIS LOCATION
2966 012074  005037  001326                     CLR     CPUEXP          ;CLEAR CPIUEXP - DON'T EXPECT TRAPS FOR A WHILE
2967 012100  005737  170200                     TST     MAPL0           ;SEE IF MAP REGISTER 0 GOT CLEARED
2968 012104  001036                             BNE     4$              ;BRANCH IF MAP REGISTER NOT ZERO
2969 012106  010237  170200                     MOV     R2,MAPL0        ;RESTORE MAPL0
2970 012112  010446                             MOV     R4,-(SP)        ;PUT VIRTUAL ADDRESS ON STACK FOR ADREXT SUBROUTINE USE
2971 012114  013746  172354                     MOV     KIPAR6,-(SP)    ;PUT PAR ON STACK FOR ADREXT SUBROUTINE USE
2972 012120  004737  003674                     JSR     PC,ADREXT       ;GO SET DATA IN THE 4 WORDS OF ADDROR AND ADRAND
2973 012124  000416                             BR      2$              ;GO CALL ERROR
2974 012126  012737  000020  001326     20$:    MOV     #TIMOUT,CPUEXP  ;EXPECTING CPU TIME OUT ON UNIBUS
2975 012134  011401                             MOV     (R4),R1         ;READ LOCATION POINTED TO BY PAR6
2976 012136  005037  001326                     CLR     CPUEXP          ;CLEAR CPIUEXP - DON'T EXPECT TRAPS FOR A WHILE
2977 012142  020201                             CMP     R2,R1           ;SEE IF DATA FETCHED MATCHES PATTERN
2978 012144  001012                             BNE     3$              ;BRANCH IF NOT SAME
2979 012146  005014                             CLR     (R4)            ;TRY TO CLEAR THIS LOCATION
2980 012150  005737  170200                     TST     MAPL0           ;SEE IF MAP REGISTER 0 GOT CLEARED
2981 012154  001006                             BNE     3$              ;BRANCH IF MAP REGISTER NOT ZERO
2982 012156  010237  170200                     MOV     R2,MAPL0        ;RESTORE MAPL0
2983 012162  104207                     2$:     ERROR   +207            ;GOT TO MAPL0 WITH ONE BIT DIFFERENT IN ADDRESS FROM 1777020
2984 012164  012737  000077  005764             MOV     #77,EADRES+2    ;RESTORE 77 TO EADRES+2
2985 012172  032777  001000  166736     3$:     BIT     #BIT09,@SWR     ;SEE IF LOOP ON ERROR IS SET
2986 012200  001352                             BNE     20$             ;BRANCH BACK TO LOOP IF SO
2987 012202  012737  000020  001326     4$:     MOV     #TIMOUT,CPUEXP  ;EXPECTING CPU TIME OUT ON UNIBUS
2988 012210  010114                             MOV     R1,(R4)         ;RELOAD THE LOCATION
2989 012212  005037  001326                     CLR     CPUEXP          ;CLEAR CPIUEXP - DON'T EXPECT TRAPS FOR A WHILE
2990 012216  012637  172354             5$:     MOV     (SP)+,KIPAR6    ;RESTORE KIPAR6
```

```
 2991 012222  012600                          MOV     (SP)+,R0        ;RESTORE R0
 2992 012224  074037  172354                  XOR     R0,KIPAR6       ;RESTORE BIT TO ORIGINAL STATUS
 2993 012230  006200                          ASR     R0              ;RIGHT SHIFT ONE PLACE
 2994 012232  001277                          BNE     1$              ;GO CONTINUE TEST IF BIT NOT SHIFTED OUT YET
 2995 012234  013737  001200  172354  6$:     MOV     $TMP2,KIPAR6    ;RESTORE KIPAR6
```

TEST # 5 - DATA PATH, UNIBUS TO MAIN MEMORY

```
                                        .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.
                                        ;*
                                        ;*********************************************************************
     012242                             TST5:
     012242  000004                             SCOPE
     012244  004737  005166                     JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     012250  012630  012262  000005             .WORD   TST6,20$,5      ;DATA USED BY PRETST
3009 012256  004737  002772                     JSR     PC,CLRMAP       ;CLEAR ALL MAP REGISTERS
3010 012262  012704  000035             20$:    MOV     #35,R4          ;DO 35 ACCESSES
3011 012266  012737  170400  172354             MOV     #170400,KIPAR6  ;START WITH ADDRESS 8K FROM UNIBUS
3012 012274  012737  170400  001210             MOV     #170400,$TMP6   ;MOVE IT TO $TMP6 ALSO
3013 012302  005037  001330             2$:     CLR     PCPUER          ;CLEAR ERROR CONDITION LOCATION
3014 012306  012737  000020  001326             MOV     #TIMOUT,CPUEXP  ;TIMEOUTS MIGHT OCCUR IN THIS TEST.
3015 012314  013700  140000                     MOV     140000,R0       ;TRY TO READ ADDRESS POINTED TO BY PAR6
3016 012320  005037  001326                     CLR     CPUEXP          ;CLEAR CPUEXP - NO TIMEOUTS FOR A WHILE
3017 012324  005737  001330                     TST     PCPUER          ;SEE IF READ OF ADDRESS TIMED OUT
3018 012330  001411                             BEQ     4$              ;BRANCH IF REFERENCE WAS GOOD
3019 012332  062737  000200  172354     3$:     ADD     #200,KIPAR6     ;TRY NEXT 4K BLOCK OF MEMORY
3020 012340  062737  000200  001210             ADD     #200,$TMP6      ;ADD 200 TO $TMP6 ALSO
3021 012346  077423                             SOB     R4,2$           ;SUBTRACT 1 FROM R4 AND BRANCH IF NOT DONE
3022 012350  104010                             ERROR   +10             ;NO UNIBUS ADDRESSES RESPOND
3023 012352  000574                             BR      SIZEJ0          ;BRANCH TO SIZE JUMPER SECTION
3024 012354  013737  172354  172352     4$:     MOV     KIPAR6,KIPAR5   ;PUT PAR6 INTO PAR5
3025 012362  013737  001210  001206             MOV     $TMP6,$TMP5     ;DO SAME TRANSFER
3026 012370  042737  170000  172352             BIC     #170000,KIPAR5  ;MAKE PAR5 A NON UNIBUS ADDRESS
3027 012376  042737  170000  001206             BIC     #170000,$TMP5   ;CLEAR SAME BITS
3028 012404  012737  173214  120000             MOV     #173214,120000  ;PUT RANDOM NUMBER INTO TEST LOCATION BY FAST BUS
3029 012412  012737  000020  001326             MOV     #TIMOUT,CPUEXP  ;TIMEOUTS MIGHT OCCUR IN THIS TEST.
3030 012420  013701  140000                     MOV     140000,R1       ;READ TEST LOCATION BY UNIBUS
3031 012424  005037  001326                     CLR     CPUEXP          ;CLEAR CPUEXP - NO TIMEOUTS FOR A WHILE
3032 012430  022701  173214                     CMP     #173214,R1      ;SEE IF DATA WAS READ PROPERLY
3033 012434  001406                             BEQ     5$              ;DATA OKAY NOW VERIFY DATA PATH
3034 012436  020001                             CMP     R0,R1           ;SEE IF DATA CHANGED FROM FIRST READ
3035 012440  001004                             BNE     5$              ;BRANCH AROUND NEXT TRY IF SO
3036 012442  062737  000200  172354             ADD     #200,KIPAR6     ;TRY NEXT 4K BLOCK OF MEMORY
3037 012450  077464                             SOB     R4,2$           ;BRANCH BACK TO TRY NEXT 4K BLOCK
3038 012452  012701  006100             5$:     MOV     #PATRNS,R1      ;LOAD ADDRESS OF BIT PATTERNS IN R1
3039 012456  012702  140000                     MOV     #140000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3040 012462  012703  000007                     MOV     #7,R3           ;DO 7 PATTERNS
3041 012466  012737  000020  001326             MOV     #TIMOUT,CPUEXP  ;TIMEOUTS MIGHT OCCUR IN THIS TEST.
3042 012474  011112                     6$:     MOV     (R1),(R2)       ;LOAD COUNT INTO TEST LOCATION VIA U.B.
3043 012476  021112                             CMP     (R1),(R2)       ;COMPARE COUNT WITH DATA READ
3044 012500  001426                             BEQ     10$             ;BRANCH IF DATA MATCHES
```

TEST # 5 - DATA PATH, UNIBUS TO MAIN MEMORY

```
3045 012502  011137  001174                    MOV     (R1),$TMP0      ;MOVE EXPECTED PATTERN TO $TMP0
3046 012506  011237  001176                    MOV     (R2),$TMP1      ;MOVE RECEIVED PATTERN TO $TMP1
3047 012512  011246                            MOV     (R2),-(SP)      ;PUT DATA ON STACK FOR DATEXT SUBROUTINE USE
3048 012514  004737  003624                    JSR     PC,DATEXT       ;SUBROUTINE TO LOAD DATAOR AND DATAND
3049 012520  011146                            MOV     (R1),-(SP)      ;PUT PATTERN ON STACK FOR PATEXT SUBROUTINE USE
3050 012522  004737  003650                    JSR     PC,PATEXT       ;GO SET DATA INTO PATTOR AND PATAND
3051 012526  012737  012536  001106            MOV     #7$,$LPERR      ;SET LOOP ON ERROR POINTER TO 61$
3052 012534  000403                            BR      8$              ;BRANCH OVER LOOP ON ERROR SECTION
3053 012536  011112                    7$:     MOV     (R1),(R2)       ;LOAD COUNT INTO TEST LOCATION VIA U.B.
3054 012540  021112                            CMP     (R1),(R2)       ;COMPARE COUNT WITH DATA READ
3055 012542  001401                            BEQ     9$              ;BRANCH IF OK NOW
3056 012544  104204                    8$:     ERROR   +204            ;REPORT ERROR(3) ON UNIBUS DATA PATH
3057 012546  032777  001000  166362    9$:     BIT     #BIT9,@SWR      ;SEE IF LOOP ON ERROR IS SET
3058 012554  001370                            BNE     7$              ;BRANCH BACK IF SO
3059 012556  062701  000002            10$:    ADD     #2,R1           ;MOVE TO NEXT PATTERN
3060 012562  077334                            SOB     R3,6$           ;DECREMENT LOOP COUNTER AND BRANCH IF NOT DONE
3061 012564  005037  001326                    CLR     CPUEXP          ;CLEAR CPUEXP - NO TIMEOUTS FOR A WHILE
3062 012570  005737  001320                    TST     ERRCNT          ;WERE THERE ANY ERRORS ON THIS TEST
3063 012574  001405                            BEQ     11$             ;BRANCH IF NO ERRORS ON THIS TEST
3064 012576  005337  001110                    DEC     $ERTTL          ;DON'T COUNT ERROR +11 AS ANOTHER ERROR
3065 012602  005337  001320                    DEC     ERRCNT          ;SAME AS ABOVE
3066 012606  104011                            ERROR   +11             ;SUMMARY OF ERRORS ON THE UNIBUS DATA PATH
3067 012610  023737  001206  172352    11$:    CMP     $TMP5,KIPAR5    ;MAKE SURE PAR5 CONTAINS EXPECTED CONTENTS
3068 012616  001404                            BEQ     TST6            ;;BRANCH IF OK
3069 012620  104030                            ERROR   +30             ;KIPAR5 NOT LOADED PROPERLY - SKIPPING NEXT TEST
3070 012622  013737  001206  172352            MOV     $TMP5,KIPAR5    ;RESET KIPAR5 FOR EXECUTION OF NEXT TEST
```

```
3084                                    .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.
                                        ;*
                                        ;;*****************************************************************
     012630  000004                     TST6:   SCOPE
3085 012632  004737  005166                     JSR     PC,PRETST       ;GO SET UP PRETEST DATA
3086 012636  012744  012710  000006             .WORD   SIZEJ0,20$,6    ;DATA USED BY PRETST
3087 012644  052737  000001  177572             BIS     #BIT0,MMR0      ;TURN MEMORY MANAGEMENT BACK ON
3088 012652  013700  172354                     MOV     KIPAR6,R0       ;PUT UNIBUS ADDRESS OF MAP REGISTER IN R0
3089 012656  072027  177773                     ASH     #-5,R0          ;RIGHT SHIFT R0 5 PLACES
3090 012662  042700  177400                     BIC     #177400,R0      ;CLEAR UPPER BYTE
3091 012666  042737  177000  172352             BIC     #177000,KIPAR5  ;MAKE KIPAR5 ACCESS THE FAST BUS
3092 012674  012720  021042                     MOV     #021042,(R0)+   ;SET BOTTOM BIT IN EACH ALU
3093 012700  012710  000042                     MOV     #42,(R0)        ;SET BOTTOM BIT IN EACH ALU
3094 012704  005037  120000                     CLR     120000          ;CLEAR TEST LOCATION VIA FAST BUS
3095 012710  012737  000020  001326  20$:      MOV     #TIMOUT,CPUEXP  ;TIMEOUTS MIGHT OCCUR IN THIS TEST.
3096 012716  012737  043207  140000             MOV     #43207,140000   ;LOAD TEST LOCATION VIA UNIBUS THIS LOAD SHOULD NOT BE
3097                                                                    ;RELOCATED BY THE UNIBUS MAP, SINCE BIT05 OF MMR3 IS CLEAR.
3098 012724  005037  001326                     CLR     CPUEXP          ;CLEAR CPUEXP - NO TIMEOUTS FOR A WHILE
3099 012730  013703  120000                     MOV     120000,R3       ;READ TEST LOCATION VIA FAST BUS
3100 012734  022703  043207                     CMP     #43207,R3       ;SEE IF DATA MATCHES
3101 012740  001401                             BEQ     SIZEJ0          ;BRANCH IF DATA GOOD
3102 012742  104012                             ERROR   +12             ;MAP RELOCATED WHEN NOT ENABLED
```

```
                                        .SBTTL  DETERMINATION OF FIRST USEABLE MAP REGISTER
                                ;;**************************************************************
                                ;*
                                ;*      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.
                                ;*
                                ;;**************************************************************
012744  012705  020100          SIZEJ0: MOV     #$DDW0,R5       ;PUT ADDRESS OF $DDW0 IN R5
012750  012700  170200                  MOV     #MAPL0,R0       ;LOAD ADDRESS OF FIRST MAP REGISTER IN R0
012754  012701  000040                  MOV     #40,R1          ;DO ALL 40 REGISTERS
012760  012702  006000                  MOV     #SPECST,R2      ;SET R2 TO BEGINING OF SPECIAL STACK
012764  012720  020000          1$:     MOV     #20000,(R0)+    ;LOAD 4K INTO LOWER 16 BITS AND
012770  005020                  CLR     (R0)+           ;CLEAR THE UPPER 6 BITS
012772  005022                  CLR     (R2)+           ;CLEAR THE SPECIAL STACK LOCATION
012774  077105                  SOB     R1,1$           ;BRANCH IF THERE ARE MORE TO LOAD
012776  012703  006000                  MOV     #SPECST,R3      ;RESET SPECIAL STACK POINTER
013002  052737  000040  172516          BIS     #BIT5,MMR3      ;TURN ON MAP RELOCATION
013010  052737  000001  177572          BIS     #BIT0,MMR0      ;MAKE SURE MEMORY MANAGEMENT IS ON
013016  012700  117776                  MOV     #117776,R0      ;THIS WILL BE USED TO SELECT PAR 4, ADDRESS 17776
013022  012737  170000  172350          MOV     #170000,KIPAR4  ;LOAD MAP REGISTER 0 -200 IN KIPAR4
013030  012702  125252                  MOV     #125252,R2      ;CONSTANT TO LOAD INTO LOCATION 17776
013034  012737  177777  001174          MOV     #-1,$TMP0       ;MOVE -1 TO MAP REGISTER POINTER
013042  012737  177777  003532          MOV     #-1,FTTHRU      ;INITIALIZE ONE TIME ENTRANCE FLAG IN SUBROUTINE
013050  012737  037776  001310          MOV     #37776,UBM24L   ;MOVE TEST CELL ADDRESS TO LOCATION UBM24L
013056  005037  001312                  CLR     UBM24U          ;CLEAR UPPER LOCATION UBM24U
013062  012737  013070  001106          MOV     #2$,$LPERR      ;MOVE LOOP ON ERRROR ADDRESS TO $LPERR
013070  012737  000020  001326  2$:     MOV     #TIMOUT,CPUEXP  ;EXPECTING CPU TIME OUT ON UNIBUS
013076  004737  003474                  JSR     PC,TSTLOC       ;GO TO SUBROUTINE TO SEE IF LOCATION RESPONDS
013102  000417                  BR      3$              ;RETURN IS HERE IF LOWEST FOUND
013104  005037  001326                  CLR     CPUEXP          ;CLEAR CPUEXP - NO TIMEOUTS FOR A WHILE
013110  004737  005230                  JSR     PC,DSABLD       ;GO SEE IF REGISTER SHOULD BE DISABLED
013114  104214                  ERROR   +214            ;MAP REGISTER DISABLED WHEN SHOULD BE ENABLED
013116  062737  000200  172350          ADD     #200,KIPAR4     ;MAP TO NEXT REGISTER
013124  022737  177400  172350          CMP     #177400,KIPAR4  ;SEE IF WE ARE POINTING JUST BELOW THE I/O PAGE
013132  001356                  BNE     2$              ;GO TEST NEXT MAP REGISTER IF NOT
013134  104013                  ERROR   +13             ;FATAL ERROR, RESTARTING PROGRAM
013136  000137  010000                  JMP     START           ;JUMP TO RESTART PROGRAM
013142  005037  001326          3$:     CLR     CPUEXP          ;CLEAR CPUEXP - NO TIMEOUTS FOR A WHILE
013146  004737  005266                  JSR     PC,ENABLD       ;GO CHECK TO SEE IF REGISTER SHOULD BE ENABLED
013152  104213                  ERROR   +213            ;MAP REGISTER ENABLED WHEN SHOULD BE DISABLED
```

```
                                        .SBTTL  SECTION TO DETERMINE NUMBER OF USEABLE MAP REGISTERS
                                ;;******************************************************************
                                ;*      THIS SECTION DETERMINES THE NUMBER OF USEABLE MAP REGISTERS BY ACCOM-
                                ;*      PLISHING THE SAME LOCATION ACCESS TEST AS IN THE PREVIOUS SECTION.
                                ;*
                                ;;******************************************************************
013154  013737  001174  001266  SIZEJ1: MOV     $TMP0,MMRLOW    ;MOVE REGISTER NUMBER FOUND USEABLE TO MMRLOW
013162  013737  172350  001256          MOV     KIPAR4,LOWEST   ;MOVE LOWEST USEABLE REGISTER TO LOWEST
013170  022737  170000  001256          CMP     #170000,LOWEST  ;SEE IF LOWEST REGISTER FOUND WAS THE LOWEST
013176  001427                          BEQ     1$              ;BRANCH AROUND SETUP IF IT WAS
013200  005037  001272                  CLR     UBRLOW          ;MAP REGISTER 0 IS LOWEST FOR UNIBUS MEMORY
013204  012737  170000  001262          MOV     #170000,UBMLOW  ;MOVE PAR VALUE OF UB MEMORY TO UBMLOW
013212  013737  001174  001274          MOV     $TMP0,UBRHI     ;MOVE REGISTER NUMBER FOUND USEABLE TO UBRHI
013220  005337  001274                  DEC     UBRHI           ;POINT IT AT HIGHEST UB MEMORY MAP REGISTER
013224  013737  172350  001264          MOV     KIPAR4,UBMHI    ;MOVE KIPAR4 TO UNIBUS MAP HIGHEST AND
013232  162737  000200  001264          SUB     #200,UBMHI      ;SUBTRACT 200 FROM IT TO POINT TO LAST USEABLE UBMEM PAGE
013240  012737  177400  001260          MOV     #177400,HIGEST  ;MOVE HIGHEST REGISTER TO HIGEST AND
013246  012737  000031  001270          MOV     #31,MMRHI       ;POINT TO LAST USEABLE MAP REGISTER
013254  000532                          BR      YESMSG          ;GO TYPE MESSAGE OF NON-DEFAULT INFORMATION
013256  012737  013324  001106  1$:     MOV     #3$,$LPERR      ;SET LOOP ON ERROR TO 3$
013264  062737  000200  172350          ADD     #200,KIPAR4     ;MAP TO NEXT REGISTER
013272  000414                          BR      3$              ;BRANCH OVER POST-TSTLOC CODE
013274  005037  001326          2$:     CLR     CPUEXP          ;CLEAR CPUEXP - DON'T EXPECT TRAPS FOR A WHILE
013300  062737  000200  172350          ADD     #200,KIPAR4     ;MAP TO NEXT REGISTER
013306  022737  177400  172350          CMP     #177400,KIPAR4  ;SEE IF ALL MAP REGISTERS HAVE BEEN TRIED
013314  001416                          BEQ     4$              ;BRANCH IF ALL ARE DONE
013316  004737  005266                  JSR     PC,ENABLD       ;GO SEE IF MAP REGISTER SHOULD BE ENABLED
013322  104213                          ERROR   +213            ;MAP REGISTER ENABLED WHEN SHOULD BE DISABLED
013324  012737  000020  001326  3$:     MOV     #TIMOUT,CPUEXP  ;TIMEOUTS MIGHT OCCUR IN THIS TEST.
013332  004737  003474                  JSR     PC,TSTLOC       ;GO TO SUBROUTINE TO SEE IF IT RESPONDS
013336  000756                          BR      2$              ;RETURN IS HERE IF IT WAS LOADED
013340  005037  001326                  CLR     CPUEXP          ;CLEAR CPUEXP - NO TIMEOUTS FOR A WHILE
013344  004737  005230                  JSR     PC,DSABLD       ;RETURN IS HERE IF NOT LOADED - GO SEE IF
                                                                ;REGISTER SHOULD BE DISABLED
013350  104214                          ERROR   +214            ;MAP REGISTER DISABLED WHEN SHOULD BE ENABLED
013352  013737  001174  001270  4$:     MOV     $TMP0,MMRHI     ;MOVE REGISTER FOUND TO MMRHI
013360  005337  001270                  DEC     MMRHI           ;DECREMENT THIS VALUE - IT IS ONE TOO MANY
013364  013737  172350  001260          MOV     KIPAR4,HIGEST   ;MOVE FIRST UNUSABLE REGISTER TO HIGEST
013372  023727  001260  177400          CMP     HIGEST,#177400  ;SEE IF UPPER JUMPER IS DEFAULT.
013400  001522                          BEQ     NOMSG           ;BRANCH AROUND MESSAGE TYPEOUT IF IT IS DEFAULT
```

```
                                        .SBTTL  DETERMINE UNUSEABLE MAP REGISTER WINDOW SIZE
                                ;;******************************************************************
                                ;*      THIS SECTION PERFORMS THE SAME LOCATION ACCESS TEST PREVIOUSLY DONE
                                ;*      LOOKING FOR THE FIRST ACCESSIBLE REGISTER, OR THE UPPER LIMIT.  ONCE
                                ;*      FOUND, THE SIZE OF THE WINDOW HAS BEEN DETERMINED AND THE MESSAGE
                                ;*      TELLING THE USER OF THE SIZE IS PRINTED.
                                ;*
                                ;;******************************************************************
013402  013737  001260  001262  SIZEJ2: MOV     HIGEST,UBMLOW   ;MOVE UPPER LIMIT TO UBMLOW LOCATION
013410  062737  000200  001262          ADD     #200,UBMLOW     ;POINT UBMLOW TO FIRST USABLE UNIBUS MEM PAGE
013416  013737  001174  001272          MOV     $TMP0,UBRLOW    ;MOVE REGISTER NUMBER TO UBRLOW
013424  012737  013460  001106          MOV     #2$,$LPERR      ;SET LOOP ON ERROR TO 2$
013432  000412                        BR      2$              ;BRANCH OVER CHECK FOR NON-EXISTENT LOCATION
013434  062737  000200  172350  1$:     ADD     #200,KIPAR4     ;MAP TO NEXT REGISTER
013442  022737  177400  172350          CMP     #177400,KIPAR4  ;SEE IF WE ARE POINTING JUST BELOW THE I/O PAGE
013450  001414                        BEQ     3$              ;GO INITIALIZE UBMHI IF WE ARE
013452  004737  005230                JSR     PC,DSABLD       ;GO SEE IF LOCATION SHOULD BE DISABLED
013456  104214                        ERROR   +214            ;MAP REGISTER DISABLED WHEN SHOULD BE ENABLED
013460  012737  000020  001326  2$:     MOV     #TIMOUT,CPUEXP  ;TIMEOUTS MIGHT OCCUR IN THIS TEST.
013466  004737  003474                JSR     PC,TSTLOC       ;GO TO SUBROUTINE TO SEE IF IT DOESN'T RESPOND
013472  000403                        BR      3$              ;RETURN IS HERE IF IT WAS LOADED - GO INITIALIZE UBMHI
013474  005037  001326                CLR     CPUEXP          ;CLEAR CPUEXP - NO TIMEOUTS FOR A WHILE
013500  000755                        BR      1$              ;RETURN IS HERE IF NOT - GO BACK FOR ANOTHER TRY
                                        ;AT THIS POINT, KIPAR4 POINTS JUST ABOVE HIGHEST ADDRESS OF UNIBUS MEMORY
013502  005037  001326          3$:     CLR     CPUEXP          ;CLEAR CPUEXP - NO TIMEOUTS FOR A WHILE
013506  004737  005266                JSR     PC,ENABLD       ;GO SEE IF MAP REGISTER SHOULD BE ENABLED
013512  104213                        ERROR   +213            ;MAP REGISTER ENABLED WHEN SHOULD BE DISABLED
013514  013737  172350  001276          MOV     KIPAR4,BUPWIN   ;MOVE THIS VALUE TO 'B'EGINING 'UP'PER 'WIN'DOW
013522  013737  172350  001264          MOV     KIPAR4,UBMHI    ;MOVE THIS TO UBMHI ALSO
013530  013737  001174  001274          MOV     $TMP0,UBRHI     ;MOVE MAP REGISTER POINTER TO UBRHI
013536  005337  001274                DEC     UBRHI           ;DECREMENT THIS VALUE - IT IS ONE TOO MANY
```

ROUTINE TO PRINT THE WINDOW SIZE MESSAGE

```
                                        .SBTTL  ROUTINE TO PRINT THE WINDOW SIZE MESSAGE
                                ;;*****************************************************************
3221 013542  005737  020022    YESMSG: TST     $PASS           ;SEE IF THIS IS FIRST PASS
3222 013546  001037                    BNE     NOMSG           ;BRANCH TO NEXT SECTION IF NOT FIRST PASS
3223 013550  104401  032654            TYPE    ,JMPMSG         ;TYPE SIZE JUMPERS NOT IN DEFAULT - FOR INFO ONLY
3224 013554  012700  005736            MOV     #DTMS,R0        ;SET UP MESSAGE POINTER
3225 013560  012746  013574            MOV     #1$,-(SP)       ;PUSH RETURN ON THE STACK
3226 013564  010046                    MOV     R0,-(SP)        ;PUSH R0 ON THE STACK
3227 013566  010146                    MOV     R1,-(SP)        ;PUSH R1 ON THE STACK
3228 013570  000137  002244            JMP     TYPDAT          ;GO TYPE THE DATA
3229 013574  104401  001223    1$:     TYPE    ,$CRLF          ;TYPE A <CRLF>
3230 013600  104401  001223            TYPE    ,$CRLF          ;TYPE ONE MORE <CRLF>
3231 013604  022703  006000            CMP     #SPECST,R3      ;SEE IF ANY TIMEOUTS OCCURED
3232 013610  001416                    BEQ     NOMSG           ;BRANCH TO NEXT SECTION IF NONE
3233 013612  104401  006257            TYPE    ,TOMSG          ;TYPE THE TIMEOUTS MESSAGE
3234 013616  012703  006000            MOV     #SPECST,R3      ;RESET R3
3235 013622  014346            2$:     MOV     -(R3),-(SP)     ;PUSH THE REGISTER # ONTO THE STACK IN ORDER IT WAS PUSHED
3236 013624  104405            3$:     TYPDS                   ;TYPE THE NUMBER IN DECIMAL, LEADING ZEROS SUPPRESSED
3237 013626  104401  001223            TYPE    ,$CRLF          ;TYPE A <CRLF>
3238 013632  005743                    TST     -(R3)           ;SEE IF THERE IS ANOTHER REGISTER NUMBER TO PRINT
3239 013634  001402                    BEQ     4$              ;BRANCH AROUND SETUP IF NONE
3240 013636  011346                    MOV     (R3),-(SP)      ;PUSH THIS NUMBER ON THE STACK
3241 013640  000771                    BR      3$              ;BRANCH BACK TO PRINT IT
3242 013642  104401  001223    4$:     TYPE    ,$CRLF          ;TYPE ONE MORE <CRLF>
```

SETUP POINTERS FOR LOWEST AND HIGHEST MAP REGISTERS

```
                                        .SBTTL  SETUP POINTERS FOR LOWEST AND HIGHEST MAP REGISTERS
                                ;;**************************************************************
                                ;;
                                ;*      SETUP POINTERS TO THE LOWEST AND THE HIGHEST USABLE
                                ;*      MAPPING REGISTERS TO CONTINUE WITH TEST.
                                ;;
                                ;;**************************************************************
013646  010546                  NOMSG:  MOV     R5,-(SP)        ;SAVE R5
013650  013705  001256                  MOV     LOWEST,R5       ;MOVE PAR DATA TO R5 FOR CONVERSION
013654  042705  170000                  BIC     #170000,R5      ;CLEAR BITS 15 TO 12
013660  072527  177773                  ASH     #-5,R5          ;SHIFT INDICATOR BITS TO THE RIGHT 5 PLACES
013664  052705  170200                  BIS     #170200,R5      ;FORM ADDRESS
013670  010537  001300                  MOV     R5,LREGL        ;SAVE RESULTS
013674  012605                          MOV     (SP)+,R5        ;RESTORE R5
013676  013737  001300  001302          MOV     LREGL,LREGU     ;MOVE ADDRESS TO LREGU
013704  062737  000002  001302          ADD     #2,LREGU        ;POINT TO UPPER 6 BITS OF MAP REG
```

```
                                .SBTTL  TEST # 7 - ENSURE THAT THERE IS NO DUAL MAPPING
                                ;;*****************************************************************
                                ;*TEST 7        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.
                                ;*
                                ;;*****************************************************************
      013712                    TST7:
      013712  000004                    SCOPE
      013714  004737  005166            JSR     PC,PRETST       ;GO SET UP PRETEST DATA
      013720  014616  014260  000007    .WORD   TST10,20$,7     ;DATA USED BY PRETST
3278  013726  005037  172516            CLR     MMR3            ;CLEAR MMR3
3279  013732  005037  001312            CLR     UBM24U          ;CLEAR THE UPPER EXPECTED LMA LOCATION
3280  013736  052737  000060  172516    BIS     #60,MMR3        ;ENABLE 22-BIT ADDRESSING AND MAP RELOCATION
3281  013744  004737  002772            JSR     PC,CLRMAP       ;CLEAR ALL MAP REGISTERS
3282  013750  042737  000001  177572    BIC     #1,MMR0         ;TURN OFF MEMORY MANAGEMENT
3283  013756  005037  001174            CLR     $TMP0           ;$TMP0 IS USED AS A FLAG IN THIS TEST
3284  013762  012703  117776            MOV     #117776,R3      ;SELECT P.A.R. 4 OFFSET OF 17776
3285  013766  013702  001300            MOV     LREGL,R2        ;PUT ADDRESS OF LOWEST USABLE MAP REGISTER IN R2
3286  013772  013700  001256            MOV     LOWEST,R0       ;LOAD PAR POINTING TO MAP REGISTER UNDER TEST IN R0
3287  013776  052737  000001  177572    BIS     #1,MMR0         ;MAKE SURE MEMORY MANAGEMENT IS ON
3288  014004  005037  001320            CLR     ERRCNT          ;CLEAR THE ERROR COUNT FOR ERROR 202 BELOW
3289  014010  013737  001256  172350  100$: MOV    LOWEST,KIPAR4   ;PAR OF LOWEST USEABLE MAP REGISTER IS LOADED IN KIPAR4
3290  014016  022702  170204            CMP     #MAPL01,R2      ;SEE IF WE ARE POINTING AT MAPL1
3291  014022  001003                    BNE     1$              ;BRANCH IF NOT
3292  014024  022712  020000            CMP     #20000,(R2)     ;SEE IF IT CONTAINS 20000 (FOR APT USE)
3293  014030  001410                    BEQ     2$              ;BRANCH IF SO
3294  014032  012712  040000        1$: MOV     #40000,(R2)     ;LOAD MAP REGISTER UNDER TEST WITH 8K BASE
3295  014036  010237  057776            MOV     R2,57776        ;LOAD TEST LOCATION WITH THE ADDRESS
3296                                                            ;OF THE MAP REGISTER UNDER TEST
3297  014042  012737  057776  001314    MOV     #57776,UBM24P   ;MOVE ANTICIPATED ADDRESS TO LOCATION UBM24P
3298  014050  000405                    BR      3$              ;BRANCH OVER LOCATION SETUP
3299  014052  010237  037776        2$: MOV     R2,37776        ;LOAD TEST LOCATION WITH THE ADDRESS
3300                                                            ;OF THE MAP REGISTER UNDER TEST
3301  014056  012737  037776  001314    MOV     #37776,UBM24P   ;MOVE ANTICIPATED ADDRESS TO LOCATION
3302  014064  162737  000200  172350  3$: SUB     #200,KIPAR4     ;PREPARE KIPAR4 FOR FIRST 'ADD 200'
3303  014072  062737  000200  172350  4$: ADD     #200,KIPAR4     ;TRY NEXT MAP REGISTER
3304  014100  005037  001330      41$: CLR     PCPUER          ;CLEAR THE ERROR RECEIVER
3305  014104  012737  017776  001310    MOV     #17776,UBM24L   ;MOVE DEFAULT LMA ADDRESS TO UBM24L
3306  014112  020037  172350            CMP     R0,KIPAR4       ;SEE IF REGISTER UNDER TEST IS POINTED TO BY KIPAR4
3307  014116  001003                    BNE     45$             ;BRANCH AROUND SETUP IF NOT
3308  014120  013737  001314  001310    MOV     UBM24P,UBM24L   ;MOVE SPECIAL LMA ADDRESS TO LOCATION
3309  014126  012737  000020  001326  45$: MOV    #TIMOUT,CPUEXP  ;POSSIBLE NON-EXISTENT MEMORY
3310  014134  011304                    MOV     (R3),R4         ;READ THROUGH THE MAP REGISTER
```

TEST # 7 - ENSURE THAT THERE IS NO DUAL MAPPING

```
3311 014136  005037  001326                    CLR     CPUEXP          ;NO MORE TIMEOUTS FOR AWHILE
3312 014142  005037  001320                    CLR     ERRCNT          ;CLEAR ERROR COUNT
3313 014146  005737  001330                    TST     PCPUEP          ;SEE IF THERE WAS AN ERROR
3314 014152  001110                           BNE     8$              ;BRANCH AROUND DATA TEST IF SO
3315 014154  020402                           CMP     R4,R2           ;SEE IF CORRECT DATA WAS FETCHED
3316 014156  001075                           BNE     6$              ;BRANCH IF NO MATCH
3317 014160  032777  004000  164750           BIT     #BIT11,@SWR     ;SEE IF THIS IS AN 11/24 WITH UB MEM ONLY
3318 014166  001406                           BEQ     49$             ;BRANCH IF NOT
3319 014170  013737  177736  001304           MOV     LMAHI,LMAH      ;SAVE LMA HIGH REG IN LMAH
3320 014176  013737  177734  001306           MOV     LMALOW,LMAL     ;SAVE LMA LOW REG IN LMAL
3321 014204  020037  172350          49$:     CMP     R0,KIPAR4       ;SEE IF MAP REGISTERS ARE THE SAME
3322 014210  001455                           BEQ     5$              ;BRANCH IF CORRECT MAP REGISTER WAS USED
3323 014212  004737  005076                    JSR     PC,CHKLMA       ;GO CHECK FOR 11/24 WITH UB MEMORY ONLY
3324 014216  001304  001306                    .WORD   LMAH,LMAL       ;ADDRESSES OF LOCATIONS CONTAINING LMA CONTENTS
3325 014222  000450                           BR      5$              ;RETURN IS HERE IF MATCH ACHIEVED
3326 014224  011337  001202                    MOV     (R3),$TMP3      ;SAVE CONTENTS FOR ERROR PRINTING
3327 014230  013746  001202                    MOV     $TMP3,-(SP)     ;MOVE FAULTY DATA TO STACK FOR SUBROUTINE USE
3328 014234  013746  172350                    MOV     KIPAR4,-(SP)    ;MOVE PAR4 CONTENTS TO STACK FOR SUBROUTINE USE
3329 014240  004737  003674                    JSR     PC,ADREXT       ;GO SET DATA IN THE 4 WORDS OF ADDROR AND ADRAND
3330 014244  010246                           MOV     R2,-(SP)        ;PUT ADDRESS OF REGISTER ON STACK FOR DATEXT USE
3331 014246  013746  172356                    MOV     KIPAR7,-(SP)    ;MOVE PAR CONTENTS TO STACK FOR SUBROUTINE USE
3332 014252  004737  003624                    JSR     PC,DATEXT       ;GO SET DATA IN DATAOR AND DATAND
3333 014256  000424                           BR      46$             ;BRANCH OVER LOOP ON ERROR SETUP
3334 014260  020402                  20$:     CMP     R4,R2           ;SEE IF CORRECT DATA WAS FETCHED
3335 014262  001023                           BNE     47$             ;BRANCH IF NO MATCH
3336 014264  032777  004000  164644           BIT     #BIT11,@SWR     ;SEE IF THIS IS AN 11/24 WITH UB MEM ONLY
3337 014272  001406                           BEQ     59$             ;BRANCH IF NOT
3338 014274  013737  177736  001304           MOV     LMAHI,LMAH      ;SAVE LMA HIGH REG IN LMAH
3339 014302  013737  177734  001306           MOV     LMALOW,LMAL     ;SAVE LMA LOW REG IN LMAL
3340 014310  020037  172350          59$:     CMP     R0,KIPAR4       ;SEE IF MAP REGISTERS ARE THE SAME
3341 014314  001406                           BEQ     47$             ;BRANCH IF CORRECT MAP REGISTER WAS USED
3342 014316  004737  005076                    JSR     PC,CHKLMA       ;GO CHECK FOR 11/24 WITH UB MEMORY ONLY
3343 014322  001304  001306                    .WORD   LMAH,LMAL       ;ADDRESSES OF LOCATIONS CONTAINING LMA CONTENTS
3344 014326  000401                           BR      47$             ;RETURN IS HERE IF MATCH ACHIEVED
3345 014330  104210                  46$:     ERROR   +210            ;DUAL MAPPING ERROR IN THE UNIBUS MAP
3346 014332  032777  001000  164576  47$:     BIT     #BIT9,@SWR      ;SEE IF LOOP ON ERROR IS SET
3347 014336  001347                           BNE     20$             ;BRANCH BACK IF SO
3348 014342  000414                           BR      8$              ;BRANCH AROUND ADDRESS MATCH SETTING
3349 014344  005237  001174          5$:      INC     $TMP0           ;SET FLAG WHEN ADDRESSES MATCH
3350 014350  000411                           BR      8$              ;BRANCH AROUND SUBROUTINE CHECK - NOT NEEDED
3351 014352  004737  005076          6$:      JSR     PC,CHKLMA       ;GO CHECK FOR 11/24 WITH UB MEMORY ONLY
3352 014356  001304  001306                    .WORD   LMAH,LMAL       ;ADDRESSES OF LOCATIONS CONTAINING LMA CONTENTS
3353 014362  000401                           BR      7$              ;RETURN IS HERE IF MATCH ACHIEVED
3354 014364  000403                           BR      8$              ;BRANCH OVER LMA SPECIFIC COMPARE CODE
3355 014366  020037  172350          7$:      CMP     R0,KIPAR4       ;SEE IF MAP REGISTERS ARE THE SAME
3356 014372  001764                           BEQ     5$              ;BRANCH IF CORRECT MAP REGISTER WAS USED
3357 014374  022737  177400  172350  8$:      CMP     #177400,KIPAR4  ;SEE IF LAST REGISTER HAS BEEN TRIED
3358 014402  001420                           BEQ     10$             ;BRANCH TO CONTINUE IF SO
3359 014404  023737  001260  172350           CMP     HIGEST,KIPAR4   ;SEE IF ALL HAVE BEEN TRIED
3360 014412  001227                           BNE     4$              ;BRANCH IF STILL MORE TO TRY
3361 014414  062737  000200  172350  9$:      ADD     #200,KIPAR4     ;MAP TO NEXT MAP REGISTER
3362 014422  022737  177400  172350           CMP     #177400,KIPAR4  ;SEE IF WE ARE AT THE TOP
3363 014430  001405                           BEQ     10$             ;BRANCH TO CONTINUE IF SO
3364 014432  023737  001276  172350           CMP     BUPWIN,KIPAR4   ;SEE IF WE ARE POINTING TO THE UPPER WINDOW START
3365 014440  001365                           BNE     9$              ;BRANCH BACK FOR ANOTHER INCREMENTING SET IF NOT
3366 014442  000616                           BR      41$             ;GO BACK FOR A NEW TRY
3367 014444  005737  001174          10$:     TST     $TMP0           ;SEE THAT THERE WAS A SUCCESSFUL MATCH
```

TEST # 7 - ENSURE THAT THERE IS NO DUAL MAPPING

```
3368 014450  001006                    BNE     11$             ;BRANCH IF THERE WAS
3369 014452  010246                    MOV     R2,-(SP)        ;PUT ADDRESS OF REGISTER ON STACK FOR ADREXT USE
3370 014454  013746  172356            MOV     KIPAR7,-(SP)    ;PUT PAR ON STACK FOR SUBROUTINE ADREXT USE
3371 014460  004737  003674            JSR     PC,ADREXT       ;GO SET DATA IN THE 4 WORDS OF ADDROR AND ADRAND
3372 014464  104210                    ERROR   +210            ;DUAL MAPPING ERROR IN THE UNIBUS MAP
3373 014466  005037  001174    11$:    CLR     $TMP0           ;CLEAR FLAG FOR NEXT REGISTER
3374 014472  022702  170204            CMP     #MAPL01,R2      ;SEE IF R2 IS POINTING TO MAPL1
3375 014476  001006                    BNE     12$             ;BRANCH IF NOT
3376 014500  022712  040000            CMP     #40000,(R2)     ;DOES MAPL1 CONTAIN 40000?
3377 014504  001403                    BEQ     12$             ;BRANCH IF IT DOES
3378 014506  005037  037776            CLR     37776           ;CLEAR LOCATION 37776 ONLY - MAPL1 IS TO BE LEFT ALONE
3379 014512  000401                    BR      13$             ;SKIP OVER REGISTER CLEAR STEP
3380 014514  005012            12$:    CLR     (R2)            ;CLEAR MAP REGISTER JUST TESTED
3381 014516  062700  000200    13$:    ADD     #200,R0         ;POINT TO NEXT MAP REGISTER UNDER TEST
3382 014522  062702  000004            ADD     #4,R2           ;POINT TO NEXT MAP REGISTER TO LOAD
3383 014526  022700  177400            CMP     #177400,R0      ;SEE IF LAST REGISTER HAS BEEN TRIED
3384 014532  001421                    BEQ     15$             ;BRANCH TO NEXT SECTION IF SO
3385 014534  023700  001260            CMP     HIGEST,R0       ;SEE IF ALL MAP REGS HAVE BEEN TESTED
3386 014540  001402                    BEQ     14$             ;BRANCH IF NO MORE TO TEST
3387 014542  000137  014010            JMP     100$            ;JUMP TO BEGIN AGAIN
3388 014546  062700  000200    14$:    ADD     #200,R0         ;POINT TO NEXT MAP REGISTER UNDER TEST
3389 014552  062702  000004            ADD     #4,R2           ;POINT TO NEXT MAP REGISTER TO LOAD
3390 014556  022700  177400            CMP     #177400,R0      ;SEE IF LAST REGISTER HAS BEEN TRIED
3391 014562  001405                    BEQ     15$             ;BRANCH TO NEXT SECTION IF SO
3392 014564  020037  001276            CMP     R0,BUPWIN       ;SEE IF WE ARE POINTING TO UPPER WINDOW START
3393 014570  001336                    BNE     11$             ;BRANCH FOR ANOTHER INCREMENT SET IF NOT
3394 014572  000137  014072            JMP     4$              ;JUMP BACK FOR ANOTHER RUN
3395 014576  005737  001320    15$:    TST     ERRCNT          ;SEE IF THERE WERE ANY ERRORS
3396 014602  001405                    BEQ     TST10           ;;BRANCH TO NEXT TEST IF NO ERRORS
3397 014604  005337  001110            DEC     $ERTTL          ;DON'T COUNT ERROR +16 AS ANOTHER ERROR
3398 014610  005337  001320            DEC     ERRCNT          ;SAME AS ABOVE
3399 014614  104016                    ERROR   +16             ;SUMMARY OF DUAL MAPPING ERRORS
```

```
                                        .SBTTL  TEST # 10 - LOAD LOC'S 40000-77776 WITH THEIR ADRES'S
                                        ;;*****************************************************************
                                        ;*TEST 10       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.
                                        ;*
                                        ;;*****************************************************************
     014616                             TST10:
     014616  000004                             SCOPE
     014620  004737  005166                     JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     014624  015072  015030  000010             .WORD   TST11,20$,10    ;DATA USED BY PRETST
3409 014632  042737  000040  172516             BIC     #BIT5,MMR3      ;TURN OFF MAP RELOCATION
3410 014640  012737  000400  172350             MOV     #400,KIPAR4     ;MAP PAGE 4 TO 8K
3411 014646  012700  040000                     MOV     #40000,R0       ;STARTING ADDRESS FOR DATA PATTERN
3412 014652  012701  100000             1$:     MOV     #100000,R1      ;VIRTUAL ADDRESS
3413 014656  012702  010000                     MOV     #4096.,R2       ;LOAD 4096 LOCATIONS AT A TIME
3414 014662  010021                     2$:     MOV     R0,(R1)+        ;LOAD PHY. ADDR. INTO EACH MEMORY LOC.
3415 014664  062700  000002                     ADD     #2,R0           ;POINT TO NEXT PHYSICAL ADDRESS
3416 014670  077204                             SOB     R2,2$           ;BRANCH IF 4K OF MEMORY NOT LOADED
3417 014672  062737  000200  172350             ADD     #200,KIPAR4     ;POINT TO NEXT 4K BANK OF MEMORY
3418 014700  022737  001000  172350             CMP     #1000,KIPAR4    ;SEE IF 16K IS LOADED
3419 014706  101361                             BHI     1$              ;BRANCH IF MORE MEMORY TO LOAD
3420                                    ;*
3421                                    ;*      MEMORY FROM 8K - 16K IS NOW LOADED WITH ITS OWN ADDRESS
3422                                    ;*
3423 014710  022737  171000  172354             CMP     #171000,KIPAR6  ;DID I USE ANY MAP REGISTER
3424                                                                     ;BELOW REGISTER 6 (UB. ADDR 100000)
3425 014716  101465                             BLOS    TST11           ;;BRANCH TO NEXT TEST IF NOT
3426 014720  013700  172354                     MOV     KIPAR6,R0       ;LOAD PAR6 INTO R0 TO GET
3427                                                                     ;THE STARTING DATA PATTERN
3428 014724  072027  000006                     ASH     #6,R0           ;R0 NOW HOLDS THE STARTING DATA PATTERN
3429 014730  012701  140000             3$:     MOV     #140000,R1      ;STARTING VIRTUAL ADDRESS
3430 014734  012702  010000                     MOV     #4096.,R2       ;PREPARE TO READ 4K AT A TIME
3431 014740  011103                     4$:     MOV     (R1),R3         ;READ MAIN MEMORY THROUGH UNIBUS
3432 014742  020003                             CMP     R0,R3           ;SEE IF THE ADDRESSES MATCH
3433 014744  001012                             BNE     6$              ;BRANCH IF ERROR
3434 014746  022120                     5$:     CMP     (R1)+,(R0)+     ;CHANGE VIRTUAL & PHYSICAL ADDRESSES
3435 014750  077205                             SOB     R2,4$           ;BRANCH IF 4K OF MEMORY NOT READ
3436 014752  062737  000200  172354             ADD     #200,KIPAR6     ;POINT TO NEXT BANK OF 4K THROUGH UNIBUS
3437 014760  022737  171000  172354             CMP     #171000,KIPAR6  ;SEE IF THIS POINTS TO 16K PLUS 2
3438 014766  101360                             BHI     3$              ;BRANCH IF 16K OF MEMORY NOT CHECKED
3439 014770  000430                             BR      10$             ;TEST FINISHED, BRANCH TO EXIT
3440 014772  011146                     6$:     MOV     (R1),-(SP)      ;PUT VIRTUAL ADDRESS ON STACK FOR ADREXT SUBROUTINE USE
3441 014774  013746  172354                     MOV     KIPAR6,-(SP)    ;PUT PAR6 CONTENTS ON STACK FOR SUBROUTINE USE
3442 015000  004737  003674                     JSR     PC,ADREXT       ;GO SET DATA IN THE 4 WORDS OF ADDROR AND ADRAND
3443 015004  010146                             MOV     R1,-(SP)        ;PUT DATA ON STACK FOR DATEXT SUBROUTINE
3444 015006  013746  172344                     MOV     KIPAR2,-(SP)    ;MOVE PAR CONTENTS TO STACK FOR SUBROUTINE USE
3445 015012  004737  003624                     JSR     PC,DATEXT       ;SUBROUTINE TO LOAD DATAOR AND DATAND
3446 015016  010337  005766                     MOV     R3,EADRS2       ;MOVE DATA IN R3 TO EADRS2 FOR ERROR CALL
3447 015022  005037  005770                     CLR     EADRS2+2        ;CLEAR UPPER 6 BITS TO PRINT
3448 015026  000403                             BR      7$              ;BRANCH OVER LOOP ON ERROR SETUP
3449 015030  011103                     20$:    MOV     (R1),R3         ;READ MAIN MEMORY THROUGH UNIBUS
3450 015032  020003                             CMP     R0,R3           ;SEE IF THE ADDRESSES MATCH
3451 015034  001401                             BEQ     8$              ;BRANCH IF OK
```

TEST # 10 - LOAD LOC'S 40000-77776 WITH THEIR ADRES'S

```
3452 015036  104205                    7$:     ERROR   +205            ;DIDN'T READ ADDRESSES CORRECTLY FROM UNIBUS
3453 015040  032777  001000  164070    8$:     BIT     #BIT9,@SWR      ;SEE IF LOOP ON ERROR IS SET
3454 015046  001370                            BNE     20$             ;BRANCH BACK IF SO
3455 015050  000736                            BR      5$              ;CONTINUE TESTING
3456 015052  005737  001320            10$:    TST     ERRCNT          ;WERE THERE ANY ERRORS ON THIS TEST?
3457 015056  001405                            BEQ     TST11           ;;BRANCH IF NO ERRORS ON THIS TEST
3458 015060  005337  001110                    DEC     $ERTTL          ;DON'T COUNT ERROR +14 AS ANOTHER ERROR
3459 015064  005337  001320                    DEC     ERRCNT          ;SAME AS ABOVE
3460 015070  104014                            ERROR   +14             ;SUMMARY OF UNIBUS ADDRESS FAILURES
```

```
                                        .SBTTL  TEST # 11 - MAIN MEMORY TIMEOUT THROUGH MAP
                                        ;;*****************************************************************
                                        ;*TEST 11       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.
                                        ;*
                                        ;;*****************************************************************
     015072                             TST11:
     015072  000004                             SCOPE
     015074  004737  005166                     JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     015100  015140  015132  000011             .WORD   TST12,20$,11    ;DATA USED BY PRETST
3470 015106  052737  000040  172516             BIS     #BIT5,MMR3      ;TURN MAP RELOCATION BACK ON
3471 015114  005037  004312                     CLR     LOEFLG          ;CLEAR LOEFLG FOR ERROR LOOP INDICATOR
3472 015120  012737  000074  001312             MOV     #74,UBM24U      ;EXPECTING 74 IN UPPER POSITION OF 11/24 LMA
3473 015126  005037  001310                     CLR     UBM24L          ;EXPECTING ZERO IN LOWER POSITION
3474 015132  004737  017616             20$:    JSR     PC,MMTOTM       ;GO DO THE TEST
3475 015136  104015                             ERROR   +15             ;RETURN HERE IF ERROR - UNIBUS DID NOT TIME OUT
```

```
                                        .SBTTL  TEST # 12 - RELOC USING LOWEST USABLE MAP REG THRU UNIBUS
                                        ;;*****************************************************************
                                        ;*TEST 12       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.
                                        ;*
                                        ;;*****************************************************************
     015140                             TST12:
     015140  000004                             SCOPE
     015142  004737  005166                     JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     015146  015174  015154  000012             .WORD   TST13,20$,12    ;DATA USED BY PRETST
3487 015154  005037  004312             20$:    CLR     LOEFLG          ;CLEAR LOEFLG - FLAG USED FOR ERROR RETURN
3488 015160  005037  001312                     CLR     UBM24U          ;CLEAR UPPER LOCATION
3489 015164  004737  004106             1$:     JSR     PC,MAPADD       ;GO DO THE TEST
3490 015170  104211                             ERROR   +211            ;RETURN IS HERE FOR ERROR
3491 015172  000774                             BR      1$              ;GO BACK TO THE SUBROUTINE
```

```
3500                                    .SBTTL  TEST # 13 - CARRY PROP OF MAP'S RELOC ADDER THRU UNIBUS
                                        ;;*****************************************************************
                                        ;*TEST 13       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 .
                                        ;*
                                        ;;*****************************************************************
     015174  000004                     TST13:  SCOPE
3501 015176  004737  005166                     JSR     PC,PRETST       ;GO SET UP PRETEST DATA
3502                                    ;;*****************************************************************
3503                                    ;**IMPORTANT**: IF THE POSITION OF THIS TEST IS CHANGED, CHANGE THE THIRD
3504                                    ;               WORD BELOW TO THE NEW TEST NUMBER THIS TEST WILL OCCUPY.
3505                                    ;;*****************************************************************
3506 015202  015236  015210  000013             .WORD   2$,20$,13       ;DATA USED BY PRETST
3507         000040                             NEXMEM=BIT5             ;BIT05 IS NON-EXISTENT MEMORY BIT IN THE CPU ERROR REGISTER
3508 015210  005037  004312             20$:    CLR     LOEFLG          ;CLEAR LOEFLG - USED AS AN ERROR RETURN FLAG
3509 015214  005037  001312                     CLR     UBM24U          ;CLEAR UPPER LOCATION
3510 015220  012737  020000  001310             MOV     #20000,UBM24L   ;PUT 20000 IN LOWER LOCATION
3511 015226  004737  004416             1$:     JSR     PC,TCPMRA       ;GO DO THE TEST
3512 015232  104211                             ERROR   +211            ;RETURN IS HERE IF AN ERROR
3513 015234  000774                             BR      1$              ;RETURN TO THE TEST
3514 015236                             2$:
```

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

015236  000004                UBMSU:  SCOPE                   ;LOOP ON PREVIOUS TEST
015240  105337  001100                DECB    $TSTNM          ;DECREMENT $TSTNM - THIS IS NOT A TEST
015244  104414                        TBITR                   ;RESTORE T-BIT IF IT WAS ON
015246  032777  004000  163662        BIT     #BIT11,@SWR     ;SEE IF THIS IS AN 11/24 WITH UB MEMORY ONLY
015254  001405                        BEQ     1$              ;BRANCH TO SETUP IF NOT
015256  062737  000005  001100        ADD     #5,$TSTNM       ;FUDGE $TSTNM 5 TESTS AHEAD AND
015264  000137  015744                JMP     TST21           ;JUMP OVER NEXT 5 TESTS

                                ;;**************************************************************
                                ;*
                                ;*      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.
                                ;*
                                ;;**************************************************************

015270  013701  001302        1$:     MOV     LREGU,R1        ;PUT POINTER TO LOWEST MAP REGISTER IN R1
015274  005721                        TST     (R1)+           ;POINT TO LOWER 16 BITS OF LOWEST + 1
015276  012721  020000                MOV     #20000,(R1)+    ;LOAD LOWER 16 BITS OF (LOWEST + 1), POINTING TO 4-8K
015302  005021                        CLR     (R1)+           ;CLEAR UPPER 6 BITS OF (LOWEST + 1)
015304  005021                        CLR     (R1)+           ;CLEAR LOWER 16 BITS OF (LOWEST + 2)
015306  005021                        CLR     (R1)+           ;CLEAR UPPER 6 BITS OF (LOWEST + 2)
015310  013701  001256                MOV     LOWEST,R1       ;LOAD R1
015314  062701  000200                ADD     #200,R1         ;POINT R1 TO (LOWEST + 1)
015320  010137  172342                MOV     R1,KIPAR1       ;LOAD PAR1
015324  062701  000200                ADD     #200,R1         ;POINT R1 TO (LOWEST + 2)
015330  010137  172340                MOV     R1,KIPAR0       ;LOAD PAR0
```

```
                                .SBTTL  TEST # 14 - MAIN MEM. T.O. THROUGH MAP, CODE RUN OVER U.B.
                                ;;*****************************************************************
                                ;*TEST 14       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.
                                ;*
                                ;;*****************************************************************
     015334                     TST14:
     015334  000004                     SCOPE
     015336  004737  005166             JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     015342  015362  015354  000014     .WORD   TST15,20$,14    ;DATA USED BY PRETST
3565 015350  005037  004312             CLR     LOEFLG          ;CLEAR LOOP ON ERROR FLAG
3566 015354  004737  017616     20$:    JSR     PC,MMTOTM       ;GO DO TEST ON PAGE 60 OF THIS LISTING
3567 015360  104015                     ERROR   +15             ;RETURN HERE IF ERROR - UNIBUS DID NOT TIME OUT
```

```
3581                                    .SBTTL  TEST # 15 - RELOC USING LOWEST USABLE MAP REG USING MAP REG
                                        ;;*****************************************************************
                                        ;*TEST 15       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.
                                        ;*
                                        ;;*****************************************************************
     015362                             TST15:
     015362  000004                             SCOPE
     015364  004737  005166                     JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     015370  015420  015404  000015             .WORD   TST16,20$,15    ;DATA USED BY PRETST
3582 015376  012737  060000  060000             MOV     #060000,060000  ;MAKE SURE ADDRESS 060000 CONTAINS ITS OWN ADDRESS AS DATA
3583 015404  005037  004312          20$:       CLR     LOEFLG          ;CLEAR LOEFLG - USED IN ERROR RETURN
3584 015410  004737  004106          1$:        JSR     PC,MAPADD       ;GO DO THE TEST
3585 015414  104211                             ERROR   +211            ;RETURN IS HERE FOR ERROR
3586 015416  000774                             BR      1$              ;RETURN TO THE SUBROUTINE
```

```
   3595                                 .SBTTL  TEST # 16 - CARRY PROP OF MAP'S RELOC ADDER USING MAP REG
                                        ;;*****************************************************************
                                        ;*TEST 16       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 .
                                        ;*
                                        ;;*****************************************************************
        015420                          TST16:
        015420  000004                          SCOPE
        015422  004737  005166                  JSR     PC,PRETST       ;GO SET UP PRETEST DATA
        015426  015450  015434  000016          .WORD   TST17,20$,16    ;DATA USED BY PRETST
   3596 015434  005037  004312          20$:    CLR     LOEFLG          ;CLEAR LOEFLG - USED AS ERROR RETURN FLAG
   3597 015440  004737  004416          1$:     JSR     PC,TCPMRA       ;GO DO THE TEST
   3598 015444  104211                          ERROR   +211            ;RETURN IS HERE IF AN ERROR
   3599 015446  000774                          BR      1$              ;RETURN TO THE TEST
```

```
                                        .SBTTL  TEST # 17 - VERIFY TRAP DUE TO CACHE PARITY INTERRUPT
                                        ;;*****************************************************************
                                        ;*TEST 17       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
                                        ;;*****************************************************************
     015450                             TST17:
     015450  000004                             SCOPE
     015452  004737  005166                     JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     015456  015610  015554  000017             .WORD   TST20,20$,17    ;DATA USED BY PRETST
3631 015464  005037  177572                     CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
3632 015470  005037  172516                     CLR     MMR3            ;TURN OFF MAP RELOCATION
3633 015474  005037  172340                     CLR     KIPAR0          ;PUT PAR0 BACK WHERE IT SHOULD AND
3634 015500  012737  000200  172342             MOV     #200,KIPAR1     ;PUT PAR1 BACK WHERE IT SHOULD
3635 015506  105737  007136                     TSTB    CPUTYP          ;IS THIS AN 11/24?
3636 015512  001403                             BEQ     1$              ;BRANCH OVER JUMP IF 11/44
3637 015514  105237  001100                     INCB    $TSTNM          ;INCREMENT TEST NUMBER TO SIMULATE SKIPPING TEST
3638 015520  000511                             BR      TST21           ;BRANCH OVER THIS AND NEXT TESTS
3639 015522  132737  000200  020035  1$:       BITB    #200,$ENVM      ;IS APT SIZING?
3640 015530  001405                             BEQ     2$              ;NO;TRY HARDWARE SWITCH REGISTER
3641 015532  032737  000100  020036             BIT     #100,$SWREG     ;YES APT IS SIZING;DOES APT SAY TO DO THIS TEST
3642 015540  001423                             BEQ     TST20           ;;NO - SKIP TEST
3643 015542  000404                             BR      20$             ;YES,DO TEST
3644 015544  032777  000100  163364  2$:       BIT     #100,@SWR       ;DOES HARDWARE SWITCH REGISTER SAY TO DO TEST?
3645 015552  001416                             BEQ     TST20           ;;NO - SKIP TEST
3646 015554  012737  000005  005460  20$:      MOV     #5,CASH1        ;MOVE BYTE TO LOAD TO CACHE TO SUB      ;DPM002
3647 015562  112737  000001  005467             MOVB    #1,CASH2        ;MOVE TEST 1 TO LOCATION IN SUBROUTINE  ;DPM002
3648 015570  004737  005324                     JSR     PC,CASHSR       ;GO DO THE CASH TEST
3649 015574  012737  000000  177746             MOV     #0,CACHE        ;TURN CACHE ON
3650 015602  005703                             TST     R3              ;DID TRAP OCCUR?
3651 015604  001401                             BEQ     TST20           ;;BRANCH TO NEXT TEST IF YES
```

TEST # 17 - VERIFY TRAP DUE TO CACHE PARITY INTERRUPT

```
3652 015606  104020                    ERROR   +20          ;INTERRUPT/ABORT LOGIC TESTS TRAP TO LOC 114 DID NOT OCCUR
```

```
3686                                    .SBTTL  TEST # 20 - VERIFY TRAP DUE TO CACHE PARITY ABORT
                                        ;;*****************************************************************
                                        ;*TEST 20       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
                                        ;*
                                        ;;*****************************************************************
     015610                             TST20:
     015610  000004                             SCOPE
     015612  004737  005166                     JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     015616  015744  015666  000020             .WORD   TST21,20$,20    ;DATA USED BY PRETST
3687 015624  032777  000100  163304             BIT     #BIT6,@SWR      ;DOES SWITCH REGISTER SAY TO DO TEST?
3688 015632  001437                             BEQ     21$             ;NO - SKIP TEST
3689 015634  132737  000200  020035             BITB    #200,$ENVM      ;IS APT SIZING?
3690 015642  001405                             BEQ     12$             ;NO;TRY HARDWARE SWITCH REGISTER
3691 015644  032737  000100  020036             BIT     #100,$SWREG     ;YES APT IS SIZING;DOES APT SAY TO DO THIS TEST
3692 015652  001427                             BEQ     21$             ;NO - SKIP TEST
3693 015654  000404                             BR      20$             ;YES,DO TEST
3694 015656  032777  000100  163252  12$:       BIT     #100,@SWR       ;DOES HARDWARE SWITCH REGISTER SAY TO DO TEST?
3695 015664  001422                             BEQ     21$             ;NO - SKIP TEST
3696 015666  012737  000004  005460  20$:       MOV     #4,CASH1        ;MOVE CACHE LOAD VALUE TO LOC IN SUB    ;DPM002
3697 015674  112737  000002  005467             MOVB    #2,CASH2        ;MOVE 2ND TEST FLAG TO LOCATION IN SUB  ;DPM002
3698 015702  004737  005324                     JSR     PC,CASHSR       ;GO DO CACHE TEST
3699 015706  022704  177777                     CMP     #-1,R4          ;WAS INSTRUCTION ABORTED LEAVING R4 INTACT?
3700 015712  001401                             BEQ     1$              ;YES
3701 015714  104021                             ERROR   +21             ;INTERRUPT/ABORT TESTS R4 WAS OVERWRITTEN WITH
3702                                                                    ;DATA INDICATING THAT INSTRUCTION WAS NOT ABORTED
3703 015716  005703                     1$:     TST     R3              ;DID TRAP OCCUR
3704 015720  001401                             BEQ     2$              ;YES, PASS
```

TEST # 20 - VERIFY TRAP DUE TO CACHE PARITY ABORT

```
3705 015722  104022                          ERROR   +22             ;INTERRUPT/ABORT TESTS TRAP DID NOT OCCUR DUE TO ABORT
3706 015724  012737  000000  177746  2$:     MOV     #0,CACHE        ;TURN CACHE ON
3707 015732  062737  000007  001100  21$:    ADD     #7,$TSTNM       ;ADD 7 TO $TSTNM TO COMPENSATE FOR 7 TESTS SKIPPED
3708 015740  000137  017110                  JMP     TST30           ;JUMP OVER NEXT 7 TESTS - THEY ARE FOR AN 11/24 ONLY
```

TEST # 21 - LMA REGISTER PHYSICAL ADDRESS CHECK

```
                                        .SBTTL  TEST # 21 - LMA REGISTER PHYSICAL ADDRESS CHECK
                                        ;;*****************************************************************
                                        ;*TEST 21       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.
                                        ;*
                                        ;;*****************************************************************
     015744                             TST21:
     015744  000004                             SCOPE
     015746  004737  005166                     JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     015752  016144  016054  000021             .WORD   TST22,20$,21    ;DATA USED BY PRETST
3722 015760  042737  000001  177572             BIC     #BIT0,MMR0      ;TURN OFF MEMORY MANAGEMENT
3723 015766  042737  000060  172516             BIC     #60,MMR3        ;TURN OFF 22-BIT AND MAP RELOCATION
3724 015774  012737  016054  005762             MOV     #20$,EADRES     ;LOAD EADRES WITH LOWER 16 BITS OF THE PHYSICAL ADDRESS
3725 016002  005037  005764                     CLR     EADRES+2        ;LOAD UPPER 6 BITS WITH PHYSICAL BITS EXPECTED
3726 016006  012700  056054                     MOV     #20$+BIT14,R0   ;MOVE ADDRESS +40000 (TO REFERENCE PAR2) TO R0
3727 016012  005077  163262                     CLR     @LREGL          ;CLEAR LOWEST USEABLE MAP REGISTER LOW WORD
3728 016016  005077  163260                     CLR     @LREGU          ;CLEAR LOWEST USEABLE MAP REGISTER HIGH WORD
3729 016022  013746  172344                     MOV     KIPAR2,-(SP)    ;SAVE KIPAR2
3730 016026  013737  001256  172344             MOV     LOWEST,KIPAR2   ;PUT PAR VALUE IN PAR2 TO ACCESS LOWEST MAP REGISTER
3731 016034  052737  000060  172516             BIS     #60,MMR3        ;TURN ON 22-BIT AND MAP RELOCATION
3732 016042  005037  172340                     CLR     KIPAR0          ;CLEAR PAR0 FOR PAGE ACCESSING THIS TEST
3733 016046  052737  000001  177572             BIS     #BIT0,MMR0      ;TURN ON  MEMORY MANAGEMENT
3734 016054  011001                     20$:    MOV     (R0),R1         ;DO THE MAP REGISTER READ THROUGH THE MAP
3735 016056  023737  005762  177734             CMP     EADRES,LMALOW   ;SEE IF LMA LOWER 16 WERE LOADED PROPERLY
3736 016064  001007                             BNE     1$              ;BRANCH TO CALL ERROR IF NOT
3737 016066  013737  177736  001174             MOV     LMAHI,$TMP0     ;MOVE HI 6 BITS TO $TMP0 FOR PREPARATION
3738 016074  042737  177700  001174             BIC     #177700,$TMP0   ;CLEAR ALL BUT LOWER 6 BITS
3739 016102  001416                             BEQ     3$              ;BRANCH AROUND ERROR IF OK
3740 016104  013737  177734  005766     1$:     MOV     LMALOW,EADRS2   ;MOVE LOWER 16 BITS OF RECEIVED DATA TO EADRS2 FOR ERROR
3741 016112  013737  177736  005770             MOV     LMAHI,EADRS2+2  ;MOVE UPPER 6 BITS OF RECEIVED DATA TO EADRS2+2 FOR ERROR
3742 016120  012637  172344                     MOV     (SP)+,KIPAR2    ;RESTORE KIPAR2
3743 016124  104023                             ERROR   +23             ;LMA NOT LOADED PROPERLY
3744 016126  000406                             BR      TST22           ;;BRANCH OVER PAR RESTORATION - NOT NEEDED
3745 016130  012700  004000                     MOV     #4000,R0        ;MOVE 4000 TO WAIT LOOP COUNTER
3746 016134  077001                     2$:     SOB     R0,2$           ;WAIT A LITTLE BEFORE HITTING THE RESET IN NEXT TEST
3747 016136  000402                             BR      TST22           ;;BRANCH TO NEXT TEST
3748 016140  012637  172344             3$:     MOV     (SP)+,KIPAR2    ;RESTORE KIPAR2
```

```
3755                                    .SBTTL  TEST # 22 - LMA FORCE JUMPER BIT TEST
                                        ;;**********************************************************
                                        ;*TEST 22        LMA FORCE JUMPER BIT TEST
                                        ;*
                                        ;*      THIS TEST DETERMINES THAT THE FORCE JUMPER BIT OF THE LMA IS ZERO AFTER
                                        ;*      A SYSTEM RESET.
                                        ;*
                                        ;;**********************************************************
     016144                             TST22:
     016144  000004                             SCOPE
     016146  004737  005166                     JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     016152  016212  016160  000022             .WORD   TST23,20$,22    ;DATA USED BY PRETST
3756         000100                     FJBIT   =100                    ;FORCE JUMPER BIT IS BIT 6
3757 016160  042737  000060  172516    20$:    BIC     #60,MMR3        ;TURN OFF 22-BIT AND MAP RELOCATION
3758 016166  000005                             RESET                   ;RESET THE WORLD, CLEARING THE FJBIT
3759 016170  032737  000100  177736             BIT     #FJBIT,LMAHI    ;CHECK THE BIT FOR BEING ZERO
3760 016176  001405                             BEQ     TST23           ;;BRANCH TO NEXT TEST IF OK
3761 016200  013701  177736                     MOV     LMAHI,R1        ;MOVE LMAHI TO R1 FOR ERROR CALL
3762 016204  042701  000100                     BIC     #FJBIT,R1       ;CLEAR THE BIT THAT SHOULD HAVE BEEN CLEAR
3763 016210  104024                             ERROR   +24             ;LMA FORCE JUMPER BIT NOT ZERO
```

TEST # 23 - SETTING LMA FORCE JUMPER BIT TEST

```
3773                                    .SBTTL  TEST # 23 - SETTING LMA FORCE JUMPER BIT TEST
                                        ;;*****************************************************************
                                        ;*TEST 23       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.
                                        ;*
                                        ;;*****************************************************************
     016212                             TST23:
     016212  000004                             SCOPE
     016214  004737  005166                     JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     016220  016420  016226  000023             .WORD   TST24,20$,23    ;DATA USED BY PRETST
3774 016226  052737  000100  177736  20$:       BIS     #FJBIT,LMAHI    ;SET THE BIT
3775 016234  032737  000100  177736             BIT     #FJBIT,LMAHI    ;SEE IF IT WAS SET
3776 016242  001005                             BNE     1$              ;BRANCH IF SET
3777 016244  013701  177736                     MOV     LMAHI,R1        ;MOVE LMAHI TO R1 FOR ERROR CALL
3778 016250  052701  000100                     BIS     #FJBIT,R1       ;SET THE BIT THAT SHOULD HAVE BEEN SET
3779 016254  104025                             ERROR   +25             ;LMA FORCE JUMPER BIT NOT SET
3780 016256  022737  170000  001256  1$:        CMP     #170000,LOWEST  ;SEE IF MAP REGISTER 0 IS LOWEST
3781 016264  001004                             BNE     2$              ;BRANCH AROUND UPPER LIMIT CHECK IF NOT
3782 016266  022737  177400  001260             CMP     #177400,HIGEST  ;SEE IF MAP REGISTER 31 IS HIGHEST
3783 016274  001451                             BEQ     TST24           ;;BRANCH TO NEXT TEST IF SO
3784 016276  052737  000060  172516  2$:        BIS     #60,MMR3        ;TURN ON 22-BIT AND MAP RELOCATION
3785 016304  012737  016364  001106             MOV     #6$,$LPERR      ;RESET LOOP ON ERROR TO 6$
3786 016312  013746  172350                     MOV     KIPAR4,-(SP)    ;SAVE PAR4
3787 016316  013737  001262  172350             MOV     UBMLOW,KIPAR4   ;MOVE LOWEST PAGE OF MEMORY WINDOW TO PAR4
3788 016324  012700  117776                     MOV     #117776,R0      ;THIS WILL BE USED TO SELECT PAR4, ADDRESS 17776
3789 016330  012702  125252                     MOV     #125252,R2      ;LOAD CONSTANT TO R2
3790 016334  000407                             BR      5$              ;BRANCH OVER LOOP SETUP
3791 016336  062737  000200  172350  4$:        ADD     #200,KIPAR4     ;MAP TO NEXT REGISTER
3792 016344  023737  001264  172350             CMP     UBMHI,KIPAR4    ;SEE IF HIGHEST HAS BEEN REACHED
3793 016352  001415                             BEQ     9$              ;BRANCH TO RESTORE PAR4 AND LEAVE TEST IF SO
3794 016354  004737  003474          5$:        JSR     PC,TSTLOC       ;GO TEST LOCATION FOR WRITEABILITY
3795 016360  000766                             BR      4$              ;BRANCH BACK FOR ANOTHER TEST IF LOCATION LOADED
3796 016362  000403                             BR      7$              ;BRANCH AROUND LOOP ON ERROR SETUP
3797 016364  004737  003474          6$:        JSR     PC,TSTLOC       ;GO TEST LOCATION FOR WRITEABILITY
3798 016370  000401                             BR      8$              ;GO TEST FOR LOOP ON ERROR IF OK NOW
3799 016372  104027                  7$:        ERROR   +27             ;FORCE JUMPER BIT FAILS TO REVERT MAP REGISTER STATUS TO DEF
3800 016374  032777  001000  162534  8$:        BIT     #BIT9,@SWR      ;SEE IF LOOP ON ERROR IS STILL SET
3801 016402  001370                             BNE     6$              ;BRANCH BACK TO LOOP SECTION IF SO
3802 016404  000754                             BR      4$              ;BRANCH BACK FOR ANOTHER TEST
3803 016406  012637  172350          9$:        MOV     (SP)+,KIPAR4    ;RESTORE KIPAR4
3804 016412  042737  000060  172516             BIC     #60,MMR3
```

```
3810                                    .SBTTL  TEST # 24 - CLEARING THE FORCE JUMPER BIT
                                        ;;*****************************************************************
                                        ;*TEST 24       CLEARING THE FORCE JUMPER BIT
                                        ;*
                                        ;*      THIS TEST CLEARS THE FJ BIT AND INSURES THAT IT IS SUCCESSFULLY CLEARED.
                                        ;*
                                        ;;*****************************************************************
     016420                             TST24:
     016420  000004                             SCOPE
     016422  004737  005166                     JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     016426  016464  016434  000024             .WORD   TST25,20$,24    ;DATA USED BY PRETST
3811 016434  042737  000100  177736  20$:       BIC     #FJBIT,LMAHI    ;CLEAR THE BIT
3812 016442  032737  000100  177736             BIT     #FJBIT,LMAHI    ;CHECK TO SEE THAT IT WAS CLEARED
3813 016450  001405                             BEQ     TST25           ;;BRANCH IF CLEARED
3814 016452  013701  177736                     MOV     LMAHI,R1        ;MOVE LMAHI TO R1 FOR ERROR CALL
3815 016456  042701  000100                     BIC     #FJBIT,R1       ;CLEAR THE BIT THAT SHOULD HAVE BEEN CLEAR
3816 016462  104024                             ERROR   +24             ;LMA FORCE JUMPER BIT NOT ZERO
```

```
                                        .SBTTL  TEST # 25 - LMA CONTROL BITS TEST - DATI
                                        ;;*****************************************************************
                                        ;*TEST 25       LMA CONTROL BITS TEST - DATI
                                        ;*
                                        ;*      THIS TEST INSURES THE CONTROL BITS 14 AND 15 LOAD PROPERLY DOING A
                                        ;*      DATI.
                                        ;*
                                        ;;*****************************************************************
016464                                  TST25:
016464  000004                                  SCOPE
016466  004737  005166                          JSR     PC,PRETST       ;GO SET UP PRETEST DATA
016472  016626  016570  000025                  .WORD   TST26,20$,25    ;DATA USED BY PRETST
3824 016500  013746  172344                     MOV     KIPAR2,-(SP)    ;SAVE KIPAR2
3825 016504  012737  170000  172344            MOV     #170000,KIPAR2  ;LOAD KIPAR2
3826 016512  012737  000001  177572            MOV     #1,MMR0         ;TURN ON MEMORY MANAGEMENT
3827 016520  012737  000020  172516            MOV     #20,MMR3        ;TURN ON 22-BIT ADDRESSING
3828 016526  012700  056610                     MOV     #1$+BIT14,R0    ;MOVE LOCATION ADDRESS +40000 (TO REF PAR2) TO R0
3829 016532  011001                             MOV     (R0),R1         ;DO A DATI
3830 016534  013737  177736  001174            MOV     LMAHI,$TMP0     ;MOVE LMAHI TO $TMP0 FOR PREPARATION OF EXPECTED
3831 016542  013737  001174  001200            MOV     $TMP0,$TMP2     ;MOVE IT TO $TMP2 ALSO
3832 016550  042737  037777  001200            BIC     #37777,$TMP2    ;CLEAR ALL BUT THE CONTROL BITS
3833 016556  023737  001174  001200            CMP     $TMP0,$TMP2     ;SEE IF EXPECTED DATA CAME UP
3834 016564  001416                             BEQ     3$              ;BRANCH TO FINISH TEST IF ALL CLEAR
3835 016566  000410                             BR      1$              ;BRANCH OVER LOOP ON ERROR SETUP
3836 016570  011001                     20$:    MOV     (R0),R1         ;DO A DATI
3837 016572  013737  177736  001174            MOV     LMAHI,$TMP0     ;MOVE LMAHI TO $TMP0 FOR COMPARE
3838 016600  023737  001174  001200            CMP     $TMP0,$TMP2     ;SEE IF EXPECTED DATA CAME UP
3839 016606  001401                             BEQ     2$              ;BRANCH AROUND ERROR IF IT DID
3840 016610  104026                     1$:     ERROR   +26             ;LMA CONTROL BITS INCORRECT
3841 016612  032777  001000  162316     2$:     BIT     #BIT9,@SWR      ;SEE IF LOOP ON ERROR IS SET
3842 016620  001363                             BNE     20$             ;BRANCH BACK FOR ANOTHER TRY IF SET
3843 016622  012637  172344             3$:     MOV     (SP)+,KIPAR2    ;RESTORE KIPAR2
```

TEST # 26 - LMA CONTROL BITS TEST - DATO

```
                                        .SBTTL  TEST # 26 - LMA CONTROL BITS TEST - DATO
                                        ;;*****************************************************************
                                        ;*TEST 26       LMA CONTROL BITS TEST - DATO
                                        ;*
                                        ;*      THIS TEST INSURES THE CONTROL BITS 14 AND 15 LOAD PROPERLY DOING A
                                        ;*      DATO.
                                        ;*
                                        ;;*****************************************************************
016626                                  TST26:
016626  000004                                  SCOPE
016630  004737  005166                          JSR     PC,PRETST       ;GO SET UP PRETEST DATA
016634  016762  016724  000026                  .WORD   TST27,20$,26    ;DATA USED BY PRETST
3851    100000                          DATO    =100000
3852 016642  013746  172344                    MOV     KIPAR2,-(SP)    ;SAVE KIPAR2
3853 016646  012737  170000  172344            MOV     #170000,KIPAR2  ;LOAD KIPAR2
3854 016654  012700  056744                    MOV     #1$+BIT14,R0    ;MOVE LOCATION ADDRESS +40000 (TO REF PAR2) TO R0
3855 016660  010110                            MOV     R1,(R0)         ;DO A DATO
3856 016662  013737  177736  001174            MOV     LMAHI,$TMP0     ;MOVE LMAHI TO $TMP0 FOR CONTROL BIT ANALYSIS
3857 016670  013737  001174  001200            MOV     $TMP0,$TMP2     ;MOVE IT IT $TMP2 ALSO
3858 016676  052737  100000  001200            BIS     #DATO,$TMP2     ;SET AND CLEAR THE CONTROL BITS EXPECTED TO BE SET
3859 016704  042737  040000  001174            BIC     #BIT14,$TMP0    ;AND CLEAR SO $TMP2 WILL CONTAIN THE EXPECTED
3860 016712  023737  001174  001200            CMP     $TMP0,$TMP2     ;SEE IF BIT 15 IS SET AND 14 IS CLEAR
3861 016720  001416                            BEQ     3$              ;BRANCH IF OK
3862 016722  000410                            BR      1$              ;GO CALL ERROR
3863 016724  010110                    20$:    MOV     R1,(R0)         ;DO A DATO
3864 016726  013737  177736  001174            MOV     LMAHI,$TMP0     ;MOVE LMA HIGH REGISTER CONTENTS TO $TMP0 FOR COMPARE
3865 016734  023737  001174  001200            CMP     $TMP0,$TMP2     ;SEE IF EXPECTED CAME UP
3866 016742  001401                            BEQ     2$              ;BRANCH AROUND ERROR IF IT DID
3867 016744  104026                    1$:     ERROR   +26             ;LMA CONTROL BITS INCORRECT
3868 016746  032777  001000  162162    2$:     BIT     #BIT9,@SWR      ;SEE IF LOOP ON ERROR IS SET
3869 016754  001363                            BNE     20$             ;BRANCH BACK FOR ANOTHER TRY IF SET
3870 016756  012637  172344            3$:     MOV     (SP)+,KIPAR2    ;RESTORE KIPAR2
```

```
                                        .SBTTL  TEST # 27 - LMA CONTROL BITS TEST - DATOB
                                        ;;*****************************************************************
                                        ;*TEST 27       LMA CONTROL BITS TEST - DATOB
                                        ;*
                                        ;*      THIS TEST INSURES THE CONTROL BITS 14 AND 15 LOAD PROPERLY DOING A
                                        ;*      DATOB.
                                        ;*
                                        ;;*****************************************************************
016762                                  TST27:
016762  000004                                  SCOPE
016764  004737  005166                          JSR     PC,PRETST       ;GO SET UP PRETEST DATA
016770  017110  017052  000027                  .WORD   TST30,20$,27    ;DATA USED BY PRETST
3878            140000                  DATOB   =140000                 ;DATOB CONTROL BITS STATUS=140000
3879    016776  013746  172344                  MOV     KIPAR2,-(SP)    ;SAVE KIPAR2
3880    017002  012737  170000  172344          MOV     #170000,KIPAR2  ;LOAD KIPAR2
3881    017010  012700  057072                  MOV     #1$+BIT14,R0    ;MOVE LOCATION ADDRESS +40000 (TO REF PAR2) TO R0
3882    017014  110110                          MOVB    R1,(R0)         ;DO A DATOB
3883    017016  013737  177736  001174          MOV     LMAHI,$TMP0     ;MOVE LMAHI TO $TMP0 FOR CONTROL BIT CHECK
3884    017024  013737  001174  001200          MOV     $TMP0,$TMP2     ;MOVE TO $TMP2 ALSO
3885    017032  052737  140000  001200          BIS     #DATOB,$TMP2    ;SET THE EXPECTED DATA BITS INTO $TMP0
3886    017040  023737  001174  001200          CMP     $TMP0,$TMP2     ;SEE IF EXPECTED DATA CAME UP
3887    017046  001416                          BEQ     3$              ;BRANCH IF OK
3888    017050  000410                          BR      1$              ;BRANCH TO CALL ERROR
3889    017052  110110                  20$:    MOVB    R1,(R0)         ;DO A DATOB
3890    017054  013737  177736  001174          MOV     LMAHI,$TMP0     ;MOVE LMA HIGH REGISTER CONTENTS TO $TMP0 FOR COMPARE
3891    017062  023737  001174  001200          CMP     $TMP0,$TMP2     ;SEE IF EXPECTED DATA CAMPE UP
3892    017070  001401                          BEQ     2$              ;BRANCH IF OK
3893    017072  104026                  1$:     ERROR   +26             ;LMA CONTROL BITS INCORRECT
3894    017074  032777  001000  162034  2$:     BIT     #BIT9,@SWR      ;SEE IF LOOP ON ERROR IS SET
3895    017102  001363                          BNE     20$             ;BRANCH BACK FOR ANOTHER TRY IF SET
3896    017104  012637  172344          3$:     MOV     (SP)+,KIPAR2    ;RESTORE KIPAR2
3897                                    ;;*****************************************************************
3898                                    ;*
3899                                    ;*      >>NOTE<<: 'DATIP' CANNOT BE CHECKED IN THE 11/24 BECAUSE THE LMA IS
3900                                    ;*                WRITTEN TWICE WHEN A 'DATIP' IS EXECUTED, DESTROYING THE
3901                                    ;*                'DATIP' STATE THAT WAS WRITTEN FIRST.
3902                                    ;*
3903                                    ;;*****************************************************************
```

```
                        .SBTTL  MEMORY ON UNIBUS TESTS HEADER
                ;;*****************************************************************
                ;*
                ;*      THE NEXT TWO TESTS WILL EXECUTE IF A '1' IN BIT 5 OF THE SWITCH
                ;*      REGISTER IS FOUND SET.  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.
                ;;*****************************************************************
```

```
3949                                    .SBTTL  TEST # 30 - MEMORY ON UNIBUS TEST
                                        ;*********************************************************************
                                        ;*TEST 30       MEMORY ON UNIBUS TEST
                                        ;*
                                        ;*      THIS TEST FIRST CHECKS TO SEE IF THE UNIBUS MEMORY TESTS HAVE BEEN
                                        ;*      SELECTED.  IF NOT, THE EOP IS EZECUTED.  IT THEN CHECKS FOR UNIBUS
                                        ;*      MEMORY EXISTENCE BY CHECKING THAT THE CONTENTS OF LOWEST OR HIGHEST
                                        ;*      LOCATIONS IS NOT IN ITS DEFAULT STATE.  IF BOTH ARE, AN ERROR IS
                                        ;*      CALLED AND THE EOP IS EZECUTED.  IF EITHER ARE NOT, THE NEXT SECTION
                                        ;*      IS EXECUTED THAT DETERMINES THE SIZE OF THE UB MEMORY AND TELLS
                                        ;*      USER OF THE RESULTS ON THE FIRST PASS ONLY.
                                        ;*
                                        ;*********************************************************************
     017110                             TST30:
     017110  000004                             SCOPE
     017112  004737  005166                     JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     017116  017336  017124  000030             .WORD   TST31,20$,30    ;DATA USED BY PRETST
3950 017124  042737  000001  177572  20$:       BIC     #1,MMR0         ;TURN OFF MEMORY MANAGEMENT
3951 017132  032777  000040  161776             BIT     #BIT5,@SWR      ;SEE IF THIS TEST HAS BEEN SELECTED
3952 017140  001425                             BEQ     2$              ;BRANCH TO JUMP IF TEST NOT SELECTED
3953 017142  012737  021450  001362             MOV     #$EOP,NXTTST    ;POINT TO ESCAPE VECTOR
3954 017150  022737  170000  001256  1$:        CMP     #170000,LOWEST  ;SEE IF LOWEST IS LOWEST
3955 017156  001020                             BNE     3$              ;GO DO TEST IF IT ISN'T - UNIBUS MEMORY EXISTS
3956 017160  022737  177400  001260             CMP     #177400,HIGEST  ;SEE IF HIGEST IS HIGHEST
3957 017166  001014                             BNE     3$              ;GO DO TEST IF IT ISN'T - UNIBUS MEMORY EXISTS
3958 017170  012737  012744  001106             MOV     #SIZEJ0,$LPERR  ;MOVE SIZE JUMPER ROUTINE TO LOOP ON ERROR
3959 017176  112737  000007  001100             MOVB    #7,$TSTNM       ;EXECUTING AFTER TEST 7 ON LOOPBACK
3960 017204  112737  000007  020020             MOVB    #7,$TESTN       ;EXECUTING AFTER TEST 7 ON LOOPBACK
3961 017212  104017                             ERROR   +17             ;NO UNIBUS MEMORY EXISTS
3962 017214  000137  021450          2$:        JMP     $EOP            ;JUMP TO END OF PASS
3963 017220  005037  001320          3$:        CLR     ERRCNT          ;CLEAR THE ERROR COUNT INDICATOR
3964 017224  005037  001330                     CLR     PCPUER          ;CLEAR THE ERROR REGISTER RECEIVER
3965 017230  005037  172516                     CLR     MMR3            ;CLEAR MEMORY MANAGEMENT REGISTER MMR3
3966 017234  052737  000020  172516             BIS     #20,MMR3        ;TURN ON 22-BIT MAPPING
3967 017242  005037  172340                     CLR     KIPAR0          ;MAP PAR0 TO 0-4K
3968 017246  012737  000200  172342             MOV     #200,KIPAR1     ;MAP PAR1 TO 4-8K
3969 017254  013700  001274                     MOV     UBRHI,R0        ;MOVE UBRHI TO R0
3970 017260  163700  001272                     SUB     UBRLOW,R0       ;SUBTRACT UBRLOW FROM IT, AND
3971 017264  005200                             INC     R0              ;ADD LAST BLOCK OF 4K TO LOOP COUNTER
3972 017266  010037  001316                     MOV     R0,NUMOFK       ;SAVE R0 IN NUMOFK
3973 017272  005046                  4$:        CLR     -(SP)           ;CLEAR THE MAJOR LOOP INDICATOR ON STACK
3974 017274  012746  000200                     MOV     #200,-(SP)      ;MOVE PAR CHANGE TO STACK
3975 017300  013746  001262                     MOV     UBMLOW,-(SP)    ;MOVE STARTING PAR VALUE TO STACK
3976 017304  012746  000002                     MOV     #2,-(SP)        ;MOVE INCREMENT VALUE TO STACK
3977 017310  005737  020022                     TST     $PASS           ;SEE IF THIS IS FIRST PASS
3978 017314  001010                             BNE     TST31           ;;BRANCH TO NEXT TEST IF NOT
3979 017316  010046                             MOV     R0,-(SP)        ;MOVE LOOP COUNTER TO THE STACK AND
3980 017320  006316                             ASL     (SP)            ;ROTATE THIS TO THE LEFT 2 PLACES
3981 017322  006316                             ASL     (SP)            ;TO INDICATE NUMBER OF K IN OCTAL
3982 017324  104401  006123                     TYPE    ,UBMAVA         ;GO TYPE THE UNI-BUS MEMORY AVAILABLE MESSAGE
3983 017330  104405                             TYPDS                   ;GO TYPE THE NUMBER IN DECIMAL
3984 017332  104401  006172                     TYPE    ,UBMEND         ;TYPE A " K" AND <CRLF>
3985         021450                  TST32=$EOP
```

```
                                .SBTTL  TEST # 31 - USING MARCH ALGORITHM, CHECK UB MEMORY
                                ;;*****************************************************************
                                ;*TEST 31       USING MARCH ALGORITHM, CHECK UB MEMORY
                                ;*
                                ;*      THIS TEST LOADS PATTERN 125252 INTO ALL LOCATIONS IN UB MEMORY, THEN
                                ;*      USING THE MARCH ALGORITHM, CHECKS THE MEMORY
                                ;*
                                ;;*****************************************************************
     017336                     TST31:
     017336  000004                     SCOPE
     017340  004737  005166             JSR     PC,PRETST       ;GO SET UP PRETEST DATA
     017344  021450  017454  000031     .WORD   TST32,20$,31    ;DATA USED BY PRETST
3993 017352  012704  125252             MOV     #125252,R4      ;MOVE FIRST TEST PATTERN TO R4
3994 017356  012705  052525             MOV     #52525,R5       ;MOVE SECOND TEST PATTERN TO R5
3995 017362  013737  001262  172354  1$: MOV     UBMLOW,KIPAR6   ;INITIALIZE PAR6
3996 017370  013700  001316             MOV     NUMOFK,R0       ;REINITIALIZE LOOP COUNTER R0
3997 017374  052737  000001  177572     BIS     #1,MMR0         ;TURN ON MEMORY MANAGEMENT
3998 017402  012702  010000         2$: MOV     #10000,R2       ;ACCESS ALL WORDS IN THIS 4K BLOCK
3999 017406  012703  140000             MOV     #140000,R3      ;FIRST ADDRESS OF THIS PAGE
4000 017412  010423                 3$: MOV     R4,(R3)+        ;MOVE THE PATTERN TO THE LOCATION
4001 017414  077202                     SOB     R2,3$           ;SUBTRACT 1 AND BRANCH IF 4K NOT DONE
4002 017416  062737  000200  172354     ADD     #200,KIPAR6     ;MAP TO NEXT 4K BLOCK
4003 017424  077012                     SOB     R0,2$           ;SUBTRACT 1 AND BRANCH IF BLOCKS OF 4K NOT DONE
4004 017426  016637  000002  172354     MOV     2(SP),KIPAR6    ;REINITIALIZE KIPAR6 TO POINT AT BEGINNING
4005 017434  010100                     MOV     R1,R0           ;REINITIALIZE LOOP COUNTER R0
4006 017436  012702  010000         4$: MOV     #10000,R2       ;ACCESS ALL WORDS IN THIS 4K BLOCK
4007 017442  012703  140000             MOV     #140000,R3      ;FIRST ADDRESS OF THIS PAGE
4008 017446  066603  000006             ADD     6(SP),R3        ;ADD OFFSET FOR THIS MAJOR PASS
4009 017452  000401                     BR      5$              ;BRANCH OVER LOOP ON ERROR PREPARATION
4010 017454  010413                20$: MOV     R4,(R3)         ;REWRITE 1ST PATTERN TO LOCATION FOR LOOP ON ERROR
4011 017456  020413                 5$: CMP     R4,(R3)         ;SEE IF IT WAS LOADED PROPERLY
4012 017460  001403                     BEQ     6$              ;BRANCH AROUND ERROR CALL IF OK
4013 017462  010437  001204             MOV     R4,$TMP4        ;MOVE EXPECTED DATA TO $TMP4
4014 017466  000405                     BR      7$              ;GO COMPLETE DATA FETCHING AND CALL ERROR
4015 017470  005113                 6$: COM     (R3)            ;COMPLEMENT THAT LOCATION TO PRODUCE SECOND TEST PATTERN
4016 017472  020513                     CMP     R5,(R3)         ;SEE IF IT IS THE COMPLEMENT
4017 017474  001417                     BEQ     11$             ;BRANCH AROUND ERROR CALL IF IT IS
4018 017476  010537  001204             MOV     R5,$TMP4        ;MOVE EXPECTED DATA TO $TMP4
4019 017502  011337  001206         7$: MOV     (R3),$TMP5      ;MOVE RECEIVED DATA TO $TMP5
4020 017506  005737  001330             TST     PCPUER          ;SEE IF THIS ACCESS TIMED OUT - IF IT DID, BRANCH
4021 017512  001010                     BNE     11$             ;AROUND ERROR CALLS - TIMEOUT ROUTINE LOGGED ERROR
4022 017514  010337  005762             MOV     R3,EADRES       ;MOVE ADDRESS IN R3 TO EADRES FOR ERROR CALL
4023 017520  104206                 8$: ERROR   +206            ;DATA PATTERN NOT CORRECT
4024 017522  000404                     BR      11$             ;BRANCH AROUND INCREMENT AND CLEAR INSTRUCTIONS
4025 017524  005237  001110         9$: INC     $ERTTL          ;STILL COUNT THIS AS AN ERROR
4026 017530  005037  001330        10$: CLR     PCPUER          ;CLEAR TIMEOUT RECEIVER
4027 017534  061603                11$: ADD     (SP),R3         ;ADD INCREMENT/DECREMENT VALUE TO R3
4028 017536  077231                     SOB     R2,5$           ;SUBTRACT 1 AND BRANCH IF 4K NOT CHECKED
4029 017540  066637  000004  172354     ADD     4(SP),KIPAR6    ;MAP TO NEXT 4K BLOCK
4030 017546  077045                     SOB     R0,4$           ;BRANCH BACK IF MORE BLOCKS TO CHECK
4031 017550  005766  000006             TST     6(SP)           ;TEST TO SEE IF THIS IS SECOND PASS
4032 017554  001404                     BEQ     12$             ;BRANCH TO 2ND PASS SETUP IF NOT
4033 017556  062706  000010             ADD     #10,SP          ;CLEAN UP STACK
4034 017562  000137  021450             JMP     $EOP            ;JUMP TO END OF PASS
4035 017566  012766  017776  000006 12$: MOV     #17776,6(SP)    ;MOVE 2K WORDS -2 (ALSO 2ND PASS INDICATOR) TO STACK
4036 017574  012766  177600  000004     MOV     #-200,4(SP)     ;MOVE REVERSE PAR STEP TO STACK
4037 017602  013766  001264  000002     MOV     UBMHI,2(SP)     ;MOVE LAST PAR VALUE TO STACK
```

TEST # 31 - USING MARCH ALGORITHM, CHECK UB MEMORY

```
4038 017610  012716  177776          MOV     #-2,(SP)        ;MOVE DECREMENT VALUE TO STACK
4039 017614  000662                  BR      1$              ;BRANCH BACK FOR SECOND PASS
```

```
4040                                            .SBTTL  SUBROUTINE TO TEST TIMEOUT THROUGH UNIBUS MAP
4041                                            ;;*************************************************************
4042 017616  005737  004312          MMTOTM: TST     LOEFLG          ;SEE IF THIS ENTRY IS FROM ERROR LOOPING
4043 017622  001402                          BEQ     1$              ;BRANCH IF NOT
4044 017624  062706  000002                  ADD     #2,SP           ;CLEAN EXTRA RETURN OFF STACK
4045 017630  005037  001320          1$:     CLR     ERRCNT          ;CLEAR ERRCNT FOR THIS TEST
4046 017634  013737  001256  172350          MOV     LOWEST,KIPAR4   ;LOAD PAR 4 WITH LOWEST USABLE MAP REG
4047 017642  012777  000074  161432          MOV     #74,@LREGU      ;LOAD UPPER 6 BITS OF LOWEST MAP REG
4048 017650  005077  161424                  CLR     @LREGL          ;LOAD LOWER 16 BITS OF LOWEST MAP REG
4049 017654  032777  004000  161254          BIT     #BIT11,@SWR     ;SEE IF AN 11/24 WITH UB MEMORY ONLY
4050 017662  001403                          BEQ     20$             ;BRANCH IF NOT
4051 017664  012737  177600  172350          MOV     #177600,KIPAR4  ;RESET KIPAR4 SO A TIMEOUT THROUGH MAP CAN BE EXPECTED
4052 017672  005037  001330          20$:    CLR     PCPUER          ;CPU ERROR REGISTER LOCATION
4053 017676  012737  000020  001326          MOV     #TIMOUT,CPUEXP  ;EXPECTING TIMEOUT IN THIS TEST
4054 017704  013703  100000                  MOV     100000,R3       ;TRY TO READ THROUGH PAGE 4 THIS REFERENCE WILL GO OUT
4055                                    ;ON THE UNIBUS TO SELECT THE LOWEST USABLE MAP REGISTER (DEFAULT MAP REG. 0).  PHYSICAL
4056                                    ;ADDRESS 17700000 IS THEN GENERATED, WHICH SHOULD TIME OUT SINCE IT IS THE FIRST
4057                                    ;NON-EXISTENT LOCATION.
4058 017710  005037  001326                  CLR     CPUEXP          ;CLEAR LOCATION - NO MORE TIMEOUTS FOR A WHILE
4059 017714  022737  000020  001330          CMP     #TIMOUT,PCPUER  ;THE UNIBUS SHOULD HAVE TIMED OUT
4060 017722  001405                          BEQ     3$              ;BRANCH IF CONDITION WAS CORRECT
4061 017724  011646                  2$:     MOV     (SP),-(SP)      ;PUSH ANOTHER RETURN ONTO THE STACK FOR POSSIBLE ERROR LOOP
4062 017726  012737  000001  004312          MOV     #1,LOEFLG       ;SET ERROR LOOP FLAG
4063 017734  000207                          RTS     PC              ;EXIT
4064 017736  062716  000002          3$:     ADD     #2,(SP)         ;CORRECT RETURN PC OVER ERROR CALL
4065 017742  000207                          RTS     PC              ;EXIT
```

```
4066          020000                          .=20000         ;THE APT TABLES NEED TO START AT 20000 - THIS STATEMENT DOES THAT
4067
4068          177777                  ADDW0=  177777
4069          177777                  ADDW1=  177777
4070                                  .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)
```

```
 4072                                   .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:
```

```
                                                                    H 9
 4074                                .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                             $SCOPE:
 020140  005037  001360             CLR     RETRY           ;CLEAR RETRY FLAG AN THE START OF EACH TEST
 020144  005037  001320             CLR     ERRCNT          ;CLEAR THE MULTIPLE ERROR COUNTER
 020150  005037  001246             CLR     DATAOR          ;LOCATION FOR LOGICAL OR OF BAD DATA
 020154  005037  001236             CLR     ADDROR          ;LOCATION FOR LOGICAL OR OF ADDRESS
 020160  005037  001240             CLR     ADDROR+2        ;LOCATION FOR UPPER 6 BITS OF LOGICAL OR OF ADDRESS
 020164  005037  001254             CLR     PATTOR          ;LOCATION FOR LOGICAL OR OF PATTERN LOADED
 020170  012737  177777  001242     MOV     #-1,DATAND      ;LOCATION FOR LOGICAL AND OF BAD DATA
 020176  012737  177777  001232     MOV     #-1,ADRAND      ;LOCATION FOR LOGICAL AND OF ADDRESS
 020204  012737  000077  001234     MOV     #77,ADRAND+2    ;LOCATION FOR UPPER 6 BITS OF LOGICAL AND OF ADDRESS
 020212  012737  177777  001252     MOV     #-1,PATAND      ;LOCATION FOR LOGICAL AND OF PATTERN LOADED
 020220  012737  000077  005764     MOV     #77,EADRES+2    ;RESTORE UPPER 6 BIT LOCATION OF EADRES+2
 020226  012737  000077  005770     MOV     #77,EADRS2+2    ;RESTORE UPPER 6 BIT LOCATION OF EADRS2+2
 020234  032777  040000  160674  1$: BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
 020242  001134                     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  000503                             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  000443                             BR      7$              ;;LOOP ON THE PRESENT TEST
 020302                             6$:;#####END OF CODE FOR THE XOR TESTER#####
 020302  032777  000400  160626             BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
 020310  001407                             BEQ     2$              ;;BR IF NO
 020312  017746  160620                     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  001502                             BEQ     $OVER           ;;BR IF YES
 020330  013737  177766  020552     2$:     MOV     177766,CPSAVE   ;MOVE CPU ERR REG VALUE TO LOC FOR TST  ;DPM001
 020336  032737  000001  020552             BIT     #BIT00,CPSAVE   ;SEE IF THE POWER MONITOR BIT IS ON     ;DPM001
 020344  001406                             BEQ     2000$           ;BRANCH TO CONTINUE ROUTINE IF CLEAR    ;DPM001
 020346  042737  000001  177766             BIC     #BIT00,177766   ;CLEAR THE BIT FOUND TO BE SET          ;DPM001
 020354  104177                             EMT     +177            ;CALL SPECIAL POWER FAIL BIT ERROR CALL ;DPM001
 020356  105037  001101                     CLRB    $ERFLG          ;CLEAR THE ERROR FLAG                   ;DPM001
 020362  105737  001101             2000$:  TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
 020366  001421                             BEQ     3$              ;;BR IF NO
 020370  123737  001113  001101             CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
 020376  101015                             BHI     3$              ;;BR IF NO
```

```
020400  032777  001000  160530          BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
020406  001404                          BEQ     4$              ;;BR IF NO
020410  013737  001106  001104  7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
020416  000446                          BR      $OVER
020420  105037  001101          4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
020424  005037  001212                  CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
020430  000415                          BR      1$              ;;ESCAPE TO THE NEXT TEST
020432  032777  004000  160476  3$:     BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
020440  001011                          BNE     1$              ;;BR IF YES
020442  005737  020022                  TST     $PASS           ;;IF FIRST PASS OF PROGRAM
020446  001406                          BEQ     1$              ;;      INHIBIT ITERATIONS
020450  005237  001102                  INC     $ICNT           ;;INCREMENT ITERATION COUNT
020454  023737  001212  001102          CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
020462  002024                          BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
020464  012737  000001  001102  1$:     MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
020472  013737  020550  001212          MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
020500  105237  001100          $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
020504  113737  001100  020020          MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
020512  011637  001104                  MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
020516  011637  001106                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
020522  005037  001214                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
020526  112737  000001  001113          MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
020534  013777  001100  160376  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
020542  013716  001104                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
020546  000002                          RTI                     ;;FIXES PS
020550  000002                  $MXCNT: 2.                      ;;MAX. NUMBER OF ITERATIONS
020552  000000                  CPSAVE: .WORD   0               ;LOCATION TO SAVE CPU ERR REG CONTENTS  ;DPM001
```

```
 4076                                   .SBTTL  ERROR HANDLER ROUTINE

                                        ;;*****************************************************************
                                        ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
                                        ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
                                        ;*AND GO TO ERTYPE ON ERROR
                                        ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                        ;*SW15=1         HALT ON ERROR
                                        ;*SW13=1         INHIBIT ERROR TYPEOUTS
                                        ;*SW10=1         BELL ON ERROR
                                        ;*SW09=1         LOOP ON ERROR
                                        ;*CALL
                                        ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER

020554  000000                  IBSAVE: .WORD   0               ;LOC'N TO HOLD $ITEMB DURING DUAL ERR   ;DPM001
020556  105037  020554          $ERROR: CLRB    IBSAVE          ;CLEAR THE ITEM BYTE SAVE LOCATION      ;DPM001
020562  113737  001100  020020        MOVB    $TSTNM,$TESTN   ;SAVE TEST NUMBER FOR ERROR TYPE OUT
020570  005237  001320                INC     ERRCNT          ;COUNT ALL MULTIPLE ERRORS
020574  010037  001160                MOV     R0,$REG0        ;SAVE R0 FOR POSSIBLE TYPE OUT
020600  010137  001162                MOV     R1,$REG1        ;SAVE R1 FOR POSSIBLE TYPE OUT
020604  010237  001164                MOV     R2,$REG2        ;SAVE R2 FOR POSSIBLE TYPE OUT
020610  010337  001166                MOV     R3,$REG3        ;SAVE R3 FOR POSSIBLE TYPE OUT
020614  010437  001170                MOV     R4,$REG4        ;SAVE R4 FOR POSSIBLE TYPE OUT
020620  010537  001172                MOV     R5,$REG5        ;SAVE R5 FOR POSSIBLE TYPE OUT
020624  105237  001101          7$:   INCB    $ERFLG          ;;SET THE ERROR FLAG
020630  001775                      BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
020632  013777  001100  160300        MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
020640  032777  002000  160270        BIT     #BIT10,@SWR     ;;BELL ON ERROR?
020646  001402                      BEQ     1$              ;;NO - SKIP
020650  104401  001216                TYPE    ,$BELL          ;;RING BELL
020654  005237  001110          1$:   INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
020660  011637  001114                MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
020664  162737  000002  001114        SUB     #2,$ERRPC
020672  117737  160216  001112        MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
020700  122737  000177  001112        CMPB    #177,$ITEMB     ;SEE IF THIS IS THE POWER FAIL CALL      ;DPM001
020706  001426                      BEQ     2001$           ;BRANCH AROUND ROUTINE IF IT IS          ;DPM001
020710  105737  020554                TSTB    IBSAVE          ;SEE IF THIS IS THE 2ND ERROR CALL       ;DPM001
020714  001021                      BNE     2000$           ;BRANCH IF SO                            ;DPM001
020716  013737  177766  020552        MOV     177766,CPSAVE   ;MOVE CPU ERR REG TO CPSAVE FOR TEST     ;DPM001
020724  032737  000001  020552        BIT     #BIT00,CPSAVE   ;SEE IF POWER MONITOR BIT IS SET         ;DPM001
020732  001414                      BEQ     2001$           ;BRANCH IF OK                            ;DPM001
020734  042737  000001  177766        BIC     #BIT00,177766   ;CLEAR THE BIT FOUND SET                 ;DPM001
020742  113737  001112  020554        MOVB    $ITEMB,IBSAVE   ;MAKE IBSAVE NON-ZERO FOR DUAL CALL      ;DPM001
020750  112737  000177  001112        MOVB    #177,$ITEMB     ;SET $ITEMB TO SPECIAL POWER FAIL PNTR   ;DPM001
020756  000402                      BR      2001$           ;BRANCH OVER IBSAVE CLEARING             ;DPM001
020760  105037  020554        2000$:  CLRB    IBSAVE          ;CLEAR IBSAVE SO AFTER 2ND ERROR, EXIT   ;DPM001
02076?                        2001$:
020764  032777  020000  160144        BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
020772  001004                      BNE     20$             ;;SKIP TYPEOUTS
020774  004737  002026                JSR     PC,ERTYPE       ;;GO TO USER ERROR ROUTINE
021000  104401  001223                TYPE    ,$CRLF
021004                        20$:
021004  122737  000001  020034        CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
021012  001007                      BNE     2$              ;;NO,SKIP APT ERROR REPORT
021014  113737  001112  021026        MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
021022  004737  021220                JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
021026     000                21$:   .BYTE   0
```

```
021027  000                     .BYTE   0
021030  000777          22$:    BR      22$             ;;APT ERROR LOOP
021032  105737  020554  2$:     TSTB    !BSAVE          ;SEE IF POWER FAIL ERROR CALL           ;DPM001
021036  001004                  BNE     3$              ;BRANCH IF NOT - HALT NOT ALLOWED       ;DPM001
021040  005777  160072          TST     @SWR            ;;HALT ON ERROR
021044  100001                  BPL     3$              ;;SKIP IF CONTINUE
021046  000000                  HALT                    ;;HALT ON ERROR!
021050  032777  001000  160060  3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
021056  001405                  BEQ     4$              ;;BR IF NO
021060  105737  020554          TSTB    !BSAVE          ;SEE IF THIS IS THE PWR MNTR BIT ERROR  ;DPM001
021064  001257                  BNE     7$              ;BRANCH BACK IF SO - FUDGING NOT ALLOWED;DPM001
021066  013716  001106          MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
021072  005737  001214  4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
021076  001405                  BEQ     5$              ;;BR IF NONE
021100  105737  020554          TSTB    !BSAVE          ;SEE IF THIS IS THE PWR MNTR BIT ERROR  ;DPM001
021104  001247                  BNE     7$              ;BRANCH BACK IF SO - FUDGING NOT ALLOWED;DPM001
021106  013716  001214          MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
021112                  5$:
021112  022737  021676  000042  CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
021120  001001                  BNE     6$              ;;BRANCH IF NO
021122  000000                  HALT                    ;;YES
021124                  6$:
021124  105737  020554          TSTB    !BSAVE          ;SEE IF THIS IS THE PWR FAIL ERROR CALL ;DPM001
021130  001235                  BNE     7$              ;BRANCH BACK TO CALL ORIGINAL ERR IF SO ;DPM001
021132  032777  001000  157776  BIT     #SW9,@SWR       ;ARE WE LOOPING ON THIS ERROR?
021140  001417                  BEQ     1000$           ;BRANCH IF NOT
021142  012737  177777  177766  MOV     #-1,CPUERR      ;CLEAR CPU ERROR REGISTER
021150  042737  177776  177572  BIC     #177776,MMR0    ;CLEAR MEMORY MANAGEMENT STATUS REGISTER
021156  012737  177777  003032  MOV     #-1,TOFLAG      ;INITIALIZE TRAP FLAG
021164  012737  177777  003224  MOV     #-1,CPFLAG      ;INITIALIZE CP TRAP FLAG
021172  012737  177777  003374  MOV     #-1,MMFLAG      ;INITIALIZE MEMORY MANAGEMENT TRAP FLAG
021200  000002          1000$:  RTI                     ;RETURN TO TEST
```

```
4078                                    .SBTTL  APT COMMUNICATIONS ROUTINE
                                        ;;*****************************************************************
021202  112737  000001  021446  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
021210  112737  000001  021444  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
021216  000403                          BR      $ATYC
021220  112737  000001  021446  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
021226                          $ATYC:
021226  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
021230  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
021232  105737  021444                  TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
021236  001450                          BEQ     5$              ;;IF NOT:  BR
021240  122737  000001  020034          CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
021246  001031                          BNE     3$              ;;IF NOT:  BR
021250  132737  000100  020035          BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
021256  001425                          BEQ     3$              ;;IF NOT:  BR
021260  017600  000004                  MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
021264  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
021272  005737  020014          1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
021276  001375                          BNE     1$              ;;IF NOT:  WAIT
021300  010037  020030                  MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
021304  105720                  2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
021306  001376                          BNE     2$
021310  163700  020030                  SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
021314  006200                          ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
021316  010037  020032                  MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
021322  012737  000004  020014          MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
021330  000413                          BR      5$
021332  017637  000004  021356  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
021340  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
021346  013746  177776                  MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
021352  004737  022052                  JSR     PC,$TYPE        ;;CALL TYPE MACRO
021356  000000                  4$:     .WORD   0
021360                          5$:
021360  105737  021446          10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
021364  001416                          BEQ     12$             ;;IF NOT:  BR
021366  005737  020034                  TST     $ENV            ;;RUNNING UNDER APT?
021372  001413                          BEQ     12$             ;;IF NOT:  BR
021374  005737  020014          11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
021400  001375                          BNE     11$             ;;IF NOT:  WAIT
021402  017637  000004  020016          MOV     @4(SP),$FATAL   ;;GET ERROR #
021410  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
021416  005237  020014                  INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
021422  105037  021446          12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
021426  105037  021445                  CLRB    $LFLG           ;;CLEAR LOG FLAG
021432  105037  021444                  CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
021436  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
021440  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
021442  000207                          RTS     PC              ;;RETURN
021444     000          $MFLG:  .BYTE   0               ;;MESSG. FLAG
021445     000          $LFLG:  .BYTE   0               ;;LOG FLAG
021446     000          $FFLG:  .BYTE   0               ;;FATAL FLAG
                                .EVEN
            000200      APTSIZE=200
            000001      APTENV=001
            000100      APTSPOOL=100
            000040      APTCSUP=040
```

```
4080                                    .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

021450                                  $EOP:
021450  000004                                SCOPE                   ;LOOP ON LAST TEST
021452  005037  177572                      CLR     MMR0            ;TURN OFF FULL RELOCATION
021456  005037  172516                      CLR     MMR3            ;DISABLE THE UNIBUS MAP
021462  104414                              TBITR                   ;RESTORE THE T BIT IF IT WAS ON
021464  005037  001100                      CLR     $TSTNM          ;;ZERO THE TEST NUMBER
021470  005037  001212                      CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
021474  005237  020022                      INC     $PASS           ;;INCREMENT THE PASS NUMBER
021500  042737  100000  020022              BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
021506  005327                              DEC     (PC)+           ;;LOOP?
021510  000001                  $EOPCT:     .WORD   1
021512  003075                              BGT     $DOAGN          ;;YES
021514  012737                              MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
021516  000001                  $ENDCT:     .WORD   1
021520  021510                              $EOPCT
021522  104401  021530                      TYPE    ,65$            ;;TYPE ASCIZ STRING
021526  000407                              BR      64$             ;;GET OVER THE ASCIZ
                                        ;;65$:  .ASCIZ  <12><15>/END PASS #/
021546                                  64$:
021546  013746  020022                      MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
                                                                    ;;TYPE PASS NUMBER
021552  104405                              TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
021554  005737  001110                      TST     $ERTTL          ;SEE IF THERE ARE ANY ERRORS TO REPORT  ;DPM001
021560  001427                              BEQ     1000$           ;BRANCH AROUNT MESSAGE PRINT IF NOT     ;DPM001
021562  104401  021570                      TYPE    ,67$            ;;TYPE ASCIZ STRING
021566  000421                              BR      66$             ;;GET OVER THE ASCIZ
                                        ;;67$:  .ASCIZ  /  TOTAL ERRORS SINCE LAST REPORT /
021632                                  66$:
021632  013746  001110                      MOV     $ERTTL,-(SP)    ;;SAVE $ERTTL FOR TYPEOUT
                                                                    ;;TOTAL NUMBER OF ERRORS
021636  104405                              TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
021640  104401  001223          1000$:      TYPE    ,$CRLF          ;;TYPE CARRIAGE RETURN, LINE FEED
021644  005037  001110                      CLR     $ERTTL          ;;CLEAR ERROR TOTAL
021650  013700  000042          $GET42:     MOV     @#42,R0         ;;GET MONITOR ADDRESS
021654  001414                              BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
021656  005046                              CLR     -(SP)           ;;INSURE THE "T" BIT IS CLEAR
021660  012746  021666                      MOV     #$CLR.T,-(SP)   ;;SETUP FOR AN RTI OR RTT
021664  000426                              BR      $RTRN           ;;GO DO AN RTI OR RTT TO LOAD THE PSW
                                                                    ;;WITH A CLEARED "T" BIT
021666                                  $CLR.T:
021666  013700  000042                      MOV     @#42,R0         ;;INSURE R0 CONTAINS THE MONITORS
021672  001405                              BEQ     $DOAGN          ;;RETURN ADDRESS
021674  000005                              RESET                   ;;CLEAR THE WORLD
021676  004710                  $ENDAD:     JSR     PC,(R0)         ;;GO TO MONITOR
021700  000240                              NOP                     ;;SAVE ROOM
021702  000240                              NOP                     ;;FOR
```

END OF PASS ROUTINE

```
021704  000240                          NOP                     ;;ACT11
021706                          $DOAGN:
021706  104400                          TRAP                    ;;PUSH OLD PSW AND PC ON STACK
021710  042716  000020                  BIC     #20,(SP)        ;;CLEAR THE ''T'' BIT
021714  032777  010000  157214          BIT     #BIT12,@SWR     ;;RUN WITH TRACE TRAP?
021722  001005                          BNE     1$              ;;BR IF NO
021724  005137  021750                  COM     $TBIT           ;;IS IT TIME FOR TRACE TRAP
021730  100402                          BMI     1$              ;;BR IF NO
021732  052716  000020                  BIS     #20,(SP)        ;;SET TRACE TRAP
021736  012746  021744          1$:     MOV     #$LOOP,-(SP)    ;;JUMP TO START OF TEST
021742  000002                  $RTRN:  RTI                     ;;RETURN--THIS IS CHANGED TO
                                                                ;;AN 'RTT'' IF 'RTT'' IS A LEGAL
                                                                ;;INSTRUCTION
021744                          $LOOP:
021744  000137                          JMP     @(PC)+          ;;RETURN
021746  010606                  $RTNAD: .WORD   LOOP
021750  000000                  $TBIT.  .WORD   0               ;;''T'' BIT STATE INDICATOR
021752     377     377     000  $ENULL: .BYTE   -1,-1,0          ;;NULL CHARACTER STRING
                                        .EVEN
```

```
  4082                                  .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
        021756                          $SAVREG:
        021756  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
        021760  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
        021762  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
        021764  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
        021766  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
        021770  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
        021772  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
        021776  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
        022002  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
        022006  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
        022012  000002                          RTI
                                        ;*RESTORE R0-R5
                                        ;*CALL:
                                        ;*      RESREG
        022014                          $RESREG:
        022014  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
        022020  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
        022024  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
        022030  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
        022034  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
        022036  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
        022040  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
        022042  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
        022044  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
        022046  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
        022050  000002                          RTI
```

```
4084                                    .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
                                        ;*
022052  105737  001155          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
022056  100002                          BPL     1$              ;;BR IF YES
022060  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
022062  000430                          BR      3$              ;;LEAVE
022064  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
022066  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
022072  122737  000001  020034          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
022100  001011                          BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
022102  132737  000100  020035          BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
022110  001405                          BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
022112  010037  022122                  MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
022116  004737  021210                  JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
022122  000000                  61$:    .WORD   0               ;;MESSAGE ADDRESS
022124  132737  000040  020035  62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
022132  001003                          BNE     60$             ;;YES,SKIP TYPE OUT
022134  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
022136  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
022140  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
022142  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
022144  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
022150  000002                          RTI                     ;;RETURN
022152  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
022156  001430                          BEQ     8$
022160  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
022164  001006                          BNE     5$
022166  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
022170  104401                          TYPE                    ;;TYPE A CR AND LF
022172  001223                          $CRLF
022174  105037  022412                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
022200  000755                          BR      2$              ;;GET NEXT CHARACTER
022202  004737  022264          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
022206  123726  001154          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
022212  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
022214  013746  001152                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                ;;AND THE NULL CHAR.
022220  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
022224  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
022226  004737  022264                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
022232  105337  022412                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
022236  000770                          BR      7$              ;;LOOP
                                        ;HORIZONTAL TAB PROCESSOR
022240  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
```

```
022244  004737  022264          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
022250  132737  000007  022412          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
022256  001372                          BNE     9$              ;;TAB STOP
022260  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
022262  000724                          BR      2$              ;;GET NEXT CHARACTER
022264                          $TYPEC:
022264  105777  156652                  TSTB    @$TKS           ;;CHAR IN KYBD BUFFER?          ;MJD001
022270  100022                          BPL     10$             ;;BR IF NOT                     ;MJD001
022272  017746  156646                  MOV     @$TKB,-(SP)     ;;GET CHAR                      ;MJD001
022276  042716  177600                  BIC     #177600,(SP)    ;;STRIP EXTRANEOUS BITS         ;MJD001
022302  122716  000023                  CMPB    #$XOFF,(SP)     ;;WAS CHAR XOFF                 ;MJD001
022306  001012                          BNE     102$            ;;BR IF NOT                     ;MJD001
022310                          101$:                                                           ;MJD001
022310  105777  156626                  TSTB    @$TKS           ;;WAIT FOR CHAR                 ;MJD001
022314  100375                          BPL     101$                                            ;MJD001
022316  117716  156622                  MOVB    @$TKB,(SP)      ;;GET CHAR                      ;MJD001
022322  042716  177600                  BIC     #177600,(SP)    ;;STRIP IT                      ;MJD001
022326  122716  000021                  CMPB    #$XON,(SP)      ;;WAS IT XON?                   ;MJD001
022332  001366                          BNE     101$            ;;BR IF NOT                     ;MJD001
022334                          102$:                                                           ;MJD001
022334  005726                          TST     (SP)+           ;;FIX STACK                     ;MJD001
022336                          10$:                                                            ;MJD001
022336  105777  156604                  TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
022342  100375                          BPL     10$                                             ;MJD001
022344  126627  000002  000021          CMPB    2(SP),#$XON     ;;IS CHARACTER A RANDOM XON?    ;RAN001
022352  001420                          BEQ     $TYPEX          ;;BRANCH IF YES                 ;RAN001
022354  116677  000002  156566          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
022362  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
022370  001003                          BNE     1$              ;;BRANCH IF NO
022372  105037  022412                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
022376  000406                          BR      $TYPEX          ;;EXIT
022400  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
022406  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
022410  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
022412  000000                  $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
022414  000207                  $TYPEX: RTS     PC
```

```
 4086                                   .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
 022416  017646  000000         $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
 022422  116637  000001  022641         MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
 022430  112637  022643                 MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
 022434  062716  000002                 ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
 022440  000406                         BR      $TYPON
 022442  112737  000001  022641 $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
 022450  112737  000006  022643         MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
 022456  112737  000005  022640 $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
 022464  010346                         MOV     R3,-(SP)        ;;SAVE R3
 022466  010446                         MOV     R4,-(SP)        ;;SAVE R4
 022470  010546                         MOV     R5,-(SP)        ;;SAVE R5
 022472  113704  022643                 MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
 022476  005404                         NEG     R4
 022500  062704  000006                 ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
 022504  110437  022642                 MOVB    R4,$OMODE       ;;SAVE IT FOR USE
 022510  113704  022641                 MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
 022514  016605  000012                 MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
 022520  005003                         CLR     R3              ;;CLEAR THE OUTPUT WORD
 022522  006105                 1$:     ROL     R5              ;;ROTATE MSB INTO "C"
 022524  000404                         BR      3$              ;;GO DO MSB
 022526  006105                 2$:     ROL     R5              ;;FORM THIS DIGIT
 022530  006105                         ROL     R5
 022532  006105                         ROL     R5
 022534  010503                         MOV     R5,R3
 022536  006103                 3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
 022540  105337  022642                 DECB    $OMODE          ;;TYPE THIS DIGIT?
 022544  100016                         BPL     7$              ;;BR IF NO
 022546  042703  177770                 BIC     #177770,R3      ;;GET RID OF JUNK
 022552  001002                         BNE     4$              ;;TEST FOR 0
 022554  005704                         TST     R4              ;;SUPPRESS THIS 0?
 022556  001403                         BEQ     5$              ;;BR IF YES
 022560  005204                 4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
 022562  052703  000060                 BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
 022566  052703  000040         5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
```

BINARY TO OCTAL (ASCII) AND TYPE

```
022572  110337  022636                  MOVB    R3,8$           ;;SAVE FOR TYPING
022576  104401  022636                  TYPE    ,8$             ;;GO TYPE THIS DIGIT
022602  105337  022640          7$:     DECB    $OCNT           ;;COUNT BY 1
022606  003347                          BGT     2$              ;;BR IF MORE TO DO
022610  002402                          BLT     6$              ;;BR IF DONE
022612  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
022614  000744                          BR      2$              ;;GO DO THE LAST DIGIT
022616  012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
022620  012604                          MOV     (SP)+,R4        ;;RESTORE R4
022622  012603                          MOV     (SP)+,R3        ;;RESTORE R3
022624  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
022632  012616                          MOV     (SP)+,(SP)
022634  000002                          RTI                     ;;RETURN
022636     000                  8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
022637     000                          .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
022640     000                  $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
022641     000                  $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
022642  000000                  $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
```

```
4088                                    .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
022644                                  $TYPDS:
022644  010046                                 MOV     R0,-(SP)        ;;PUSH R0 ON STACK
022646  010146                                 MOV     R1,-(SP)        ;;PUSH R1 ON STACK
022650  010246                                 MOV     R2,-(SP)        ;;PUSH R2 ON STACK
022652  010346                                 MOV     R3,-(SP)        ;;PUSH R3 ON STACK
022654  010546                                 MOV     R5,-(SP)        ;;PUSH R5 ON STACK
022656  012746  020200                         MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
022662  016605  000020                         MOV     20(SP),R5       ;;GET THE INPUT NUMBER
022666  100004                                 BPL     1$              ;;BR IF INPUT IS POS.
022670  005405                                 NEG     R5              ;;MAKE THE BINARY NUMBER POS.
022672  112766  000055  000001                 MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
022700  005000                          1$:    CLR     R0              ;;ZERO THE CONSTANTS INDEX
022702  012703  023060                         MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
022706  112723  000040                         MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
022712  005002                          2$:    CLR     R2              ;;CLEAR THE BCD NUMBER
022714  016001  023050                         MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
022720  160105                          3$:    SUB     R1,R5           ;;FORM THIS BCD DIGIT
022722  002402                                 BLT     4$              ;;BR IF DONE
022724  005202                                 INC     R2              ;;INCREASE THE BCD DIGIT BY 1
022726  000774                                 BR      3$
022730  060105                          4$:    ADD     R1,R5           ;;ADD BACK THE CONSTANT
022732  005702                                 TST     R2              ;;CHECK IF BCD DIGIT=0
022734  001002                                 BNE     5$              ;;FALL THROUGH IF 0
022736  105716                                 TSTB    (SP)            ;;STILL DOING LEADING 0'S?
022740  100407                                 BMI     7$              ;;BR IF YES
022742  106316                          5$:    ASLB    (SP)            ;;MSD?
022744  103003                                 BCC     6$              ;;BR IF NO
022746  116663  000001  177777                 MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
022754  052702  000060                  6$:    BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
022760  052702  000040                  7$:    BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
022764  110223                                 MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
022766  005720                                 TST     (R0)+           ;;JUST INCREMENTING
022770  020027  000010                         CMP     R0,#10          ;;CHECK THE TABLE INDEX
022774  002746                                 BLT     2$              ;;GO DO THE NEXT DIGIT
022776  003002                                 BGT     8$              ;;GO TO EXIT
023000  010502                                 MOV     R5,R2           ;;GET THE LSD
023002  000764                                 BR      6$              ;;GO CHANGE TO ASCII
023004  105726                          8$:    TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
023006  100003                                 BPL     9$              ;;BR IF NO
023010  116663  177777  177776                 MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
023016  105013                          9$:    CLRB    (R3)            ;;SET THE TERMINATOR
023020  012605                                 MOV     (SP)+,R5        ;;POP STACK INTO R5
023022  012603                                 MOV     (SP)+,R3        ;;POP STACK INTO R3
023024  012602                                 MOV     (SP)+,R2        ;;POP STACK INTO R2
023026  012601                                 MOV     (SP)+,R1        ;;POP STACK INTO R1
023030  012600                                 MOV     (SP)+,R0        ;;POP STACK INTO R0
023032  104401  023060                         TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
```

```
023036  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
023044  012616                          MOV     (SP)+,(SP)
023046  000002                          RTI                     ;;RETURN TO USER
023050  023420                  $DTBL:  10000.
023052  001750                          1000.
023054  000144                          100.
023056  000012                          10.
023060                          $DBLK:  .BLKW   4
```

```
 4090                                   .SBTTL  TTY INPUT ROUTINE
                                        ;;*****************************************************************
                                        .ENABL  LSB
                                        .DSABL  LSB
                                        ;;*****************************************************************
                                        ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
                                        ;*CALL:
                                        ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
                                        ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
                                        ;*                              ;;WITH PARITY BIT STRIPPED OFF
                                        ;
023070  011646                  $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
023072  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
023100  105777  156036          1$:     TSTB    @$TKS           ;;WAIT FOR
023104  100375                          BPL     1$              ;;A CHARACTER
023106  117766  156032  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
023114  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
023122  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
023130  001013                          BNE     3$              ;;BRANCH IF NO
023132  105777  156004          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
023136  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
023140  117746  156000                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
023144  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
023150  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
023154  001366                          BNE     2$              ;;IF NOT DISCARD IT
023156  000750                          BR      1$              ;;YES, RESUME
023160  026627  000004  000021  3$:     CMP     4(SP),#$XON     ;;IS IT A RANDOM XON?           ;RAN001
023166  001744                          BEQ     1$              ;;BRANCH IF YES                 ;RAN001
023170  026627  000004  000140          CMP     4(SP),#140      ;;IS IT UPPER CASE?
023176  002407                          BLT     4$              ;;BRANCH IF YES
023200  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
023206  003003                          BGT     4$              ;;BRANCH IF YES
023210  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
023216  000002                  4$:     RTI                     ;;GO BACK TO USER
                                        ;;*****************************************************************
                                        ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
                                        ;*CALL:
                                        ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
                                        ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
                                        ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
023220  010346                  $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
023222  012703  023326          1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
023226  022703  023336          2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
023232  101405                          BLOS    4$              ;;BR IF YES
023234  104406                          RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
023236  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
023240  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
023244  001003                          BNE     3$              ;;SKIP IF NOT
023246  104401  001222          4$:     TYPE    ,$QUES          ;;TYPE A '?'
023252  000763                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
023254  111337  023324          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
023260  104401  023324                  TYPE    ,9$
023264  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
023270  001356                          BNE     2$              ;;LOOP IF NOT RETURN
023272  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
023276  104401  001224                  TYPE    ,$LF            ;;TYPE A LINE FEED
023302  012603                          MOV     (SP)+,R3        ;;RESTORE R3
```

```
023304  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
023306  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
023314  012766  023326  000004          MOV     #$TTYIN,4(SP)
023322  000002                          RTI                     ;;RETURN
023324     000                  9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
023325     000                          .BYTE   0               ;;TERMINATOR
023326                          $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
023336     136      125     015 $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL "U"
023343     136      107     015 $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL "G"
023350     015      012     123 $MSWR:  .ASCIZ  <15><12>/SWR = /
023361     040      040     116 $MNEW:  .ASCIZ  /  NEW = /
```

```
                                .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.
023372  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
023374  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
023400  005740                          TST     -(R0)           ;;BACKUP BY 2
023402  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
023404  006300                          ASL     R0              ;;POSITION FOR INDEXING
023406  016000  023426                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
023412  000200                          RTS     R0              ;;GO TO ROUTINE
                                ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
023414  011646                  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
023416  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
023424  000002                          RTI                     ;;RESTORE THE PSW
                                .SBTTL  TRAP TABLE
                                ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
                                ;*BY THE "TRAP" INSTRUCTION.
                                ;       ROUTINE
                                ;       -------
023426  023414                  $TRPAD: .WORD   $TRAP2
023430  022052                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
023432  022442                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
023434  022416                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
023436  022456                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
023440  022644                          $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
023442  023070                          $RDCHR  ;;CALL=RDCHR    TRAP+6(104406)  TTY TYPEIN CHARACTER ROUTINE
023444  023220                          $RDLIN  ;;CALL=RDLIN    TRAP+7(104407)  TTY TYPEIN STRING ROUTINE
023446  005626                          $RDOCT  ;;CALL=RDOCT    TRAP+10(104410) READ AN OCTAL NUMBER FROM TTY
023450  021756                          $SAVREG ;;CALL=SAVREG   TRAP+11(104411) SAVE R0-R5 ROUTINE
023452  022014                          $RESREG ;;CALL=RESREG   TRAP+12(104412) RESTORE R0-R5 ROUTINE
4093 023454  002726                     TBITOF  ;;CALL=TBITO    TRAP+13(104413) THIS WILL TURN OFF T BIT TRAPPING
4094 023456  002754                     TBITRE  ;;CALL=TBITR    TRAP+14(104414) THIS WILL RETURN THE T BIT TO PREVIOUS CONDI
4095 023460  005626                     $RDOCT  ;;CALL=RDOCT    TRAP+15(104415) READ OCTAL NUMBER
```

```
4097                                    .SBTTL  POWER DOWN AND UP ROUTINES
                                        ;;******************************************************************
                                        ;POWER DOWN ROUTINE
023462  012737  023640  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
023470  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
023476  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
023500  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
023502  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
023504  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
023506  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
023510  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
023512  017746  155420                  MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
023516  010637  023644                  MOV     SP,$SAVR6       ;;SAVE SP
023522  012737  023534  000024          MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
023530  000000                          HALT
023532  000776                          BR      .-2             ;;HANG UP
                                        ;;******************************************************************
                                        ;POWER UP ROUTINE
023534  012737  023640  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
023542  013706  023644                  MOV     $SAVR6,SP       ;;GET SP
023546  005037  023644                  CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
023552  005237  023644          1$:     INC     $SAVR6          ;;WAIT FOR THE INC
023556  001375                          BNE     1$              ;;OF  WORD
023560  012677  155352                  MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
023564  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
023566  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
023570  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
023572  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
023574  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
023576  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
023600  012737  023462  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
023606  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
023614  104401                          TYPE                    ;;REPORT THE POWER FAILURE
023616  023646          $PWRMG: .WORD   PWRMSG          ;;POWER FAIL MESSAGE POINTER
023620  012716                  MOV     (PC)+,(SP)      ;;RESTART AT START
023622  010000          $PWRAD: .WORD   START           ;;RESTART ADDRESS
023624  042766  000020  000002          BIC     #20,2(SP)       ;;CLEAR "T" BIT
023632  005037  021750                  CLR     $TBIT           ;;CLEAR THE "T" BIT FLAG
023636  000002                          RTI
023640  000000          $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
023642  000776                  BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
023644  000000          $SAVR6: 0                       ;;PUT THE SP HERE
4098 023646     012     015     120  PWRMSG: .ASCIZ  <12><15>?POWER FAILURE, RESTARTING PROGRAM?
4099                                    .EVEN
```

```
4101                                    .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
023712  104411                  $DB20:  SAVREG                  ;;SAVE ALL REGISTERS
023714  016601  000002                  MOV     2(SP),R1        ;;PICKUP THE POINTER TO LOW WORD
023720  012705  024031                  MOV     #$OCTVL+13.,R5  ;;POINTER TO DATA TABLE
023724  012704  000014                  MOV     #12.,R4         ;;DO ELEVEN CHARACTERS
023730  012703  177770                  MOV     #^C7,R3         ;;MASK
023734  012100                          MOV     (R1)+,R0        ;;LOWER WORD
023736  012101                          MOV     (R1)+,R1        ;;HIGH WORD
023740  005002                          CLR     R2              ;;TERMINATOR
023742  110245                  1$:     MOVB    R2,-(R5)        ;;PUT CHARACTER IN DATA TABLE
023744  010002                          MOV     R0,R2           ;;GET THIS DIGIT
023746  005304                          DEC     R4              ;;COUNT THIS CHARACTER
023750  003007                          BGT     3$              ;;BR IF NOT THE LAST DIGIT
023752  001405                          BEQ     2$              ;;BR IF IT IS THE LAST DIGIT
023754  005205                          INC     R5              ;;ALL DIGITS DONE-ADJUST POINTER FOR FIRST
023756  010566  000002                  MOV     R5,2(SP)        ;;ASCIZ CHAR. & PUT IT ON THE STACK
023762  104412                          RESREG                  ;;RESTORE ALL REGISTERS
023764  000207                          RTS     PC              ;;RETURN TO USER
023766  006203                  2$:     ASR     R3              ;;POSITION THE MASK FOR THE LAST DIGIT
023770  006001                  3$:     ROR     R1              ;;POSITION THE BINARY NUMBER FOR
023772  006000                          ROR     R0              ;;      THE NEXT OCTAL DIGIT
023774  006001                          ROR     R1
023776  006000                          ROR     R0
024000  006001                          ROR     R1
024002  006000                          ROR     R0
024004  040302                          BIC     R3,R2           ;;MASK OUT ALL JUNK
024006  062702  000060                  ADD     #'0,R2          ;;MAKE THIS CHAR. ASCII
024012  000753                          BR      1$              ;;GO PUT IT IN THE DATA TABLE
024014                          $OCTVL: .BLKB   14.             ;;RESERVE DATA TABLE
```

ERROR MESSAGES

```
4103                                    .SBTTL  ERROR MESSAGES
4104 024032  116  117  124  EM1:    .ASCIZ  ?NOT THE CORRECT TRAP CONDITION THROUGH ERRVEC (#004)?
4105 024117  125  116  105  EM2:    .ASCIZ  ?UNEXPECTED CPU TRAP THROUGH ERRVEC (#004)?
4106 024171  125  116  105  EM3:    .ASCIZ  ?UNEXPECTED MEMORY MANAGEMENT TRAP, MEMORY MANAGEMENT STATUS REGISTERS?
4107 024277  123  125  115  EM4:    .ASCIZ  ?SUMMARY OF MAP REGISTERS THAT TIMED OUT ON READ?
4108 024357  123  125  115  EM5:    .ASCIZ  ?SUMMARY OF DUAL ADDRESSING ERRORS ON LOADING MAP REGISTERS?
4109 024452  123  125  115  EM6:    .ASCIZ  ?SUMMARY OF BIT PATTERN FAILURES IN LOWER 16 BITS OF MAP REGISTERS?
4110 024554  123  125  115  EM7:    .ASCIZ  ?SUMMARY OF BIT PATTERN FAILURES IN UPPER 6 BITS OF MAP REGISTERS?
4111 024655  103  101  116  EM10:   .ASCII  ?CAN'T GET TO MAIN MEMORY FROM UNIBUS WITH THE MAP OFF?<CRLF>
4112 024743  123  117  040          .ASCIZ  ?SO JUMPING TO THE SIZE JUMPER TEST FOR VERIFICATION?
4113 025027  123  125  115  EM11:   .ASCIZ  ?SUMMARY OF COUNT PATTERN FAILURES ON THE UNIBUS DATA PATH?
4114 025121  125  116  111  EM12:   .ASCIZ  ?UNIBUS MAP IS RELOCATING WHEN NOT ENABLED?
4115 025173  103  101  116  EM13:   .ASCII  ?CANNOT USE ANY OF THE MAP REGISTERS OR PHYSICAL?<CRLF>
4116 025253  101  104  104          .ASCII  ?ADDRESS BIT14 IS STUCK LOW, MUST RESTART PROGRAM?<CRLF>
4117 025334  111  106  040          .ASCIZ  ?IF YOU DON'T LOOP ON THIS PROBLEM.?
4118 025377  123  125  115  EM14:   .ASCIZ  ?SUMMARY OF UNIBUS ADDRESS ERRORS, WITH THE MAP RELOCATION DISABLED?
4119 025502  115  101  111  EM15:   .ASCIZ  ?MAIN MEMORY TIMEOUT OVER THE UNIBUS DID NOT OCCUR PROPERLY?
4120 025575  122  105  114  EM16:   .ASCIZ  ?RELOCATION THROUGH THE MAP WAS NOT CORRECT, CARRY PROPAGATION?
4121 025673  116  117  040  EM17:   .ASCIZ  ?NO UNIBUS MEMORY EXISTS?
4122 025723  111  116  124  EM20:   .ASCIZ  ?INTERRUPT/ABORT LOGIC TESTS TRAP TO LOCATION 114 DID NOT OCCUR?
4123 026022  111  116  124  EM21:   .ASCIZ  ?INTERRUPT/ABORT TESTS R4 WAS OVERWRITTEN WITH?
4124 026100  111  116  124  EM22:   .ASCIZ  ?INTERRUPT/ABORT TESTS TRAP DID NOT OCCUR DUE TO ABORT?
4125 026166  114  115  101  EM23:   .ASCIZ  ?LMA NOT LOADED PROPERLY?
4126 026216  114  115  101  EM24:   .ASCIZ  ?LMA FORCE JUMPER BIT NOT ZERO?
4127 026254  114  115  101  EM25:   .ASCIZ  ?LMA FORCE JUMPER BIT NOT SET?
4128 026311  114  115  101  EM26:   .ASCIZ  ?LMA CONTROL BITS INCORRECT?
4129 026344  106  117  122  EM27:   .ASCIZ  ?FORCE JUMPER BIT FAILS TO REVERT MAP REGISTER STATUS TO DEFAULT?
4130 026444  113  111  120  EM30:   .ASCIZ  ?KIPAR5 NOT LOADED PROPERLY?
4131 026477  124  110  105  EM201:  .ASCIZ  ?THE FOLLOWING REGISTERS TIMED OUT WHEN READ?
4132 026553  124  110  105  EM202:  .ASCIZ  ?THE FOLLOWING ARE DUAL ADDRESSING ERRORS IN THE UNIBUS MAP?
4133 026646  124  110  105  EM203:  .ASCIZ  ?THE BIT PATTERN THROUGH THE MAP REGISTERS FAILED?
4134 026727  125  116  111  EM204:  .ASCIZ  ?UNIBUS DATA PATH COUNT PATTERN FAILURE?
4135 026776  125  116  111  EM205:  .ASCIZ  ?UNIBUS ADDRESSING ERRORS, MAP RELOCATION DISABLED?
4136 027060  104  101  124  EM206:  .ASCIZ  ?DATA PATTERN NOT CORRECT?
4137 027111  122  105  106  EM207:  .ASCIZ  ?REFERENCED MAP REGISTER 0 WITH ADDRESS ONE BIT DIFFERENT THAN 17770200?
4138 027220  115  101  120  EM210:  .ASCIZ  ?MAP REGISTER(S) UNDER TEST DID NOT RESPOND IN DUAL MAPPING TEST?
4139 027320  122  105  114  EM211:  .ASCII  ?RELOCATION THROUGH THE MAP WAS NOT CORRECT, CARRY PROPAGATION?<CRLF>
4140 027416  124  105  123          .ASCIZ  ?TEST CODE BEING RUN OVER UNIBUS?
4141 027456  115  101  111  EM212:  .ASCII  ?MAIN MEMORY TIMEOUT OVER THE UNIBUS DID NOT OCCUR PROPERLY?<CRLF>
4142 027551  124  105  123          .ASCIZ  ?TEST CODE BEING RUN OVER UNIBUS?
4143 027611  115  101  120  EM213:  .ASCIZ  ?MAP REGISTER ENABLED WHEN DDW SAYS IT SHOULD BE DISABLED?
4144 027702  115  101  120  EM214:  .ASCIZ  ?MAP REGISTER DISABLED WHEN DDW SAYS IT SHOULD BE ENABLED?
```

```
                                                .SBTTL  DATA HEADERS
027773  122  105  103  DH1:    .ASCIZ  ?RECEIVD EXPECTD TESTNO  ERR PC?
030032  122  105  103  DH2:    .ASCIZ  ?RECEIVD TESTNO  ERR PC?
030061  123  124  101  DH3:    .ASCII  ?STATUS  AUTOI/D VIRTUAL?<CRLF>
030111  122  105  107          .ASCIZ  ?REGISTR REGISTR ADDRESS TESTNO  ERR PC?
030160  122  105  107  DH4:    .ASCII  ?REGADRS   REGADRS?<CRLF>
030202  040  042  117          .ASCIZ  ? 'OR''     'AND''    #ERRORS TESTNO  ERR PC?
030255  122  105  107  DH5:    .ASCII  ?REGLOAD   REGLOAD   REGDUAL   REGDUAL?<CRLF>
030323  040  042  117          .ASCIZ  ? 'OR''     'AND''    'OR''     'AND''    #ERRORS TESTNO?
030412  115  101  120  DH6:    .ASCII  ?MAPREG    MAPREG    EXPECTD EXPECTD RECEIVD RECEIVD?<CRLF>
030476  040  042  117          .ASCIZ  ? 'OR''     'AND''    'OR''     'AND''    'OR''     'AND''  #ERRORS TESTNO?
030601  124  105  123  DH10:   .ASCIZ  ?TESTNO  ERR PC  MMR3?
030626  105  130  120  DH11:   .ASCII  ?EXPECTD   EXPECTD   RECEIVD   RECEIVD?<CRLF>
030674  040  042  117          .ASCIZ  ? 'OR''     'AND''    'OR''     'AND''    #ERRORS TESTNO?
030763  103  117  116  DH15:   .ASCII  ?CONDITN CONDITN?<CRLF>
031003  105  130  120          .ASCIZ  ?EXPECTD RECEIVD TESTNO  ERR PC?
031042  124  105  123  DH23:   .ASCIZ  ?TESTNO  ERR PC  LMAEXP    LMARCV?
031103  124  105  123  DH24:   .ASCIZ  ?TESTNO  ERR PC  LMAEXP  LMARCV?
031142  124  105  123  DH27:   .ASCIZ  ?TESTNO  ERR PC  LMARCV  KIPAR4?
031201  124  105  123  DH30:   .ASCIZ  ?TESTNO  ERR PC  PR5EXP  PR5RCV?
031240  122  105  107  DH201:  .ASCIZ  ?REGADRS TESTNO  ERR PC?
031267  115  101  120  DH202:  .ASCII  ?MAPREG    MAPREG    NON-ZER?<CRLF>
031323  124  105  123          .ASCIZ  ?TESTING   DUALED    CONTNTS TESTNO  ERR PC?
031376  122  105  107  DH203:  .ASCIZ  ?REGADRS   PATTRN  EXPCTD  RECEVD  TESTNO  ERR PC?
031457  105  130  120  DH204:  .ASCIZ  ?EXPECTD RECEIVD ADDRSLOAD TESTNO  ERR PC?
031530  101  104  104  DH205:  .ASCII  ?ADDRESS   ADDRESS?<CRLF>
031552  105  130  120          .ASCIZ  ?EXPECTD   RECEIVD   TESTNO  ERR PC?
031615  101  104  104  DH206:  .ASCIZ  ?ADDRESS  EXPCTD  RECVED  TESTNO  ERR PC?
031665  101  104  104  DH207:  .ASCIZ  ?ADDRUSED  BITDIFF TESTNO  ERR PC?
031726  124  105  123  DH210:  .ASCIZ  ?TESTNO  ERR PC  MAPREGADR?
031760  103  117  122  DH211:  .ASCII  ?CORRECT   EXPECTD RECEIVD?<CRLF>
032012  101  104  104          .ASCIZ  ?ADDRESS   DATA    FROM UB TESTNO  ERR PC?
032063  103  117  116  DH212:  .ASCII  ?CONDITN CONDITN?<CRLF>
032103  105  130  120          .ASCIZ  ?EXPECTD RECEIVD TESTNO  ERR PC?
032142  124  105  123  DH213:  .ASCIZ  ?TESTNO  ERR PC  REG NO  DDWDAT  DDWADR?
                                .EVEN
```

DATA TABLES

```
4181                                        .SBTTL  DATA TABLES
4182 032212  001330  001326  020020  DT1:    .WORD   PCPUER,CPUEXP,$TESTN,BADPC,0
4183 032224  001330  020020  001340  DT2:    .WORD   PCPUER,$TESTN,BADPC,0
4184 032234  001350  001352  001354  DT3:    .WORD   PMMR0,PMMR1,PMMR2,$TESTN,BADPC,0
4185 032250  001236  001232  001320  DT4:    .WORD   ADDROR,ADRAND,ERRCNT,$TESTN,$ERRPC,0
4186 032264  001236  001232  001246  DT5:    .WORD   ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,$TESTN,0
4187 032302  001236  001232  001254  DT6:    .WORD   ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,ERRCNT,$TESTN,0
4188 032324  020020  001114  172516  DT10:   .WORD   $TESTN,$ERRPC,MMR3,0
4189 032334  001254  001252  001246  DT11:   .WORD   PATTOR,PATAND,DATAOR,DATAND,ERRCNT,$TESTN,0
4190 032352  001236  001232  001246  DT14:   .WORD   ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,$TESTN,0
4191 032370  001326  001330  020020  DT15:   .WORD   CPUEXP,PCPUER,$TESTN,$ERRPC,0
4192 032402  020020  001114  005762  DT23:   .WORD   $TESTN,$ERRPC,EADRES,EADRS2,0
4193 032414  020020  001114  001162  DT24:   .WORD   $TESTN,$ERRPC,$REG1,LMAHI,0
4194 032426  020020  001114  001174  DT26:   .WORD   $TESTN,$ERRPC,$TMP0,$REG2,0
4195 032440  020020  001114  001174  DT27:   .WORD   $TESTN,$ERRPC,$TMP0,KIPAR4,0
4196 032452  020020  001114  001206  DT30:   .WORD   $TESTN,$ERRPC,$TMP5,KIPAR5,0
4197 032464  005762  020020  001114  DT201:  .WORD   EADRES,$TESTN,$ERRPC,0
4198 032474  005766  005762  001202  DT202:  .WORD   EADRS2,EADRES,$TMP3,$TESTN,$ERRPC,0
4199 032510  005766  001174  001170  DT203:  .WORD   EADRS2,$TMP0,$REG4,$REG3,$TESTN,$ERRPC,0
4200 032526  001174  001176  001164  DT204:  .WORD   $TMP0,$TMP1,$REG2,$TESTN,$ERRPC,0
4201 032542  005762  005766  020020  DT205:  .WORD   EADRES,EADRS2,$TESTN,$ERRPC,0
4202 032554  005762  001204  001206  DT206:  .WORD   EADRES,$TMP4,$TMP5,$TESTN,$ERRPC,0
4203 032570  005762  001160  020020  DT207:  .WORD   EADRES,$REG0,$TESTN,$ERRPC,0
4204 032602  020020  001114  005762  DT210:  .WORD   $TESTN,$ERRPC,EADRES,0
4205 032612  005762  001166  001164  DT211:  .WORD   EADRES,$REG3,$REG2,$TESTN,$ERRPC,0
4206 032626  001326  001330  020020  DT212:  .WORD   CPUEXP,PCPUER,$TESTN,$ERRPC,0
4207 032640  020020  001114  001174  DT213:  .WORD   $TESTN,$ERRPC,$TMP0,$TMP1,$REG5,0
```

NON-DEFAULT UNIBUS MAP JUMPER MESSAGE

```
4208                                            .SBTTL  NON-DEFAULT UNIBUS MAP JUMPER MESSAGE
4209 032654  200  123  111  JMPMSG:  .ASCII  <CRLF>?SIZE JUMPERS ON UNIBUS MAP ARE NOT IN THEIR DEFAULT?<CRLF>
4210 032741  120  117  123           .ASCII  ?POSITION.  MAP REGISTERS BETWEEN THE LOWEST AND HIGHEST?<CRLF>
4211 033031  125  123  105           .ASCII  ?USEABLE, AND ABOVE THE UNIBUS END NUMBER WILL BE TESTED.?<CRLF>
4212 033122  125  116  111           .ASCII  ?UNIBUS MEMORY WILL BE ASSUMED TO BE BETWEEN UNIBUS BEGIN?<CRLF>
4213 033213  101  116  104           .ASCII  ?AND END REGISTER NUMBERS IF BIT <5> IS SET IN THE SWR,?<CRLF>
4214 033302  105  116  101           .ASCII  ?ENABLING TEST #23 TO EXECUTE.?<CRLF><CRLF>
4215 033341  040  040  114           .ASCII  ?  LOWEST  HIGEST  UNIBUS  UNIBUS?<CRLF>
4216 033402  040  040  125           .ASCII  ?  USABLE  USABLE  BEGIN    END?<CRLF>
4217 033441  040  040  040           .ASCIZ  ?   REG#    REG #   REG #   REG #  TEST #?<CRLF>
```

```
                                                .SBTTL  DATA FIELDS
033513      000     000     000     DF1:        .BYTE   0,0,0,0,0
033520      002     002     001     DF4:        .BYTE   2,2,1,0,0
033525      002     002     002     DF5:        .BYTE   2,2,2,2,1,0
033533      002     002     000     DF6:        .BYTE   2,2,0,0,0,0,1,0
033543      000     000     000     DF11:       .BYTE   0,0,0,0,1,0
033551      002     002     000     DF14:       .BYTE   2,2,0,0,1,0
033557      000     000     002     DF23:       .BYTE   0,0,2,2
033563      002     000     000     DF201:      .BYTE   2,0,0,0,0,0
033571      000     000     003     DF204:      .BYTE   0,0,3,0,0
033576      000     000     002     DF210:      .BYTE   0,0,2
          000001                                .END
```

SYMBOL TABLE

```
ABASE = 000000          BIT0  = 000001          DDISP = 177570          DT212  032626          FJBIT = 000100
ACDW1 = 000000          BIT00 = 000001          DFMSG   006116          DT213  032640          FLAG    001324
ACDW2 = 000000          BIT01 = 000002          DF1     033513          DT23   032402          FLOATR  005772
ACPUOP= 000000          BIT02 = 000004          DF11    033543          DT24   032414          FTTHRU  003532
ADDROR  001236          BIT03 = 000010          DF14    033551          DT26   032426          GMRMD1  007041
ADDW0 = 177777          BIT04 = 000020          DF201   033563          DT27   032440          GMRMOD  007U44
ADDW1 = 177777          BIT05 = 000040          DF204   033571          DT3    032234          HIADRS= 177742
ADDW10= 000000          BIT06 = 000100          DF210   033576          DT30   032452          HIGEST  001260
ADDW11= 000000          BIT07 = 000200          DF23    033557          DT4    032250          HITMIS= 177752
ADDW12= 000000          BIT08 = 000400          DF4     033520          DT5    032264          HT    = 000011
ADDW13= 000000          BIT09 = 001000          DF5     033525          DT6    032302          IBSAVE  020554
ADDW14= 000000          BIT1  = 000002          DF6     033533          EADRES 005762          IOTVEC= 000020
ADDW15= 000000          BIT10 = 002000          DH1     027773          EADRS2 005766          JMPMSG  032654
ADDW2 = 000000          BIT11 = 004000          DH10    030601          EMTVEC= 000030         KDPAR0= 172360
ADDW3 = 000000          BIT12 = 010000          DH11    030626          EM1    024032          KDPAR1= 172362
ADDW4 = 000000          BIT13 = 020000          DH15    030763          EM10   024655          KDPAR2= 172364
ADDW5 = 000000          BIT14 = 040000          DH2     030032          EM11   025027          KDPAR3= 172366
ADDW6 = 000000          BIT15 = 100000          DH201   031240          EM12   025121          KDPAR4= 172370
ADDW7 = 000000          BIT2  = 000004          DH202   031267          EM13   025173          KDPAR5= 172372
ADDW8 = 000000          BIT3  = 000010          DH203   031376          EM14   025377          KDPAR6= 172374
ADDW9 = 000000          BIT4  = 000020          DH204   031457          EM15   025502          KDPAR7= 172376
ADEVCT= 000000          BIT5  = 000040          DH205   031530          EM16   025575          KDPDR0= 172320
ADEVM = 000000          BIT6  = 000100          DH206   031615          EM17   025673          KDPDR1= 172322
ADRAND  001232          BIT7  = 000200          DH207   031665          EM2    024117          KDPDR2= 172324
ADREXT  003674          BIT8  = 000400          DH210   031726          EM20   025723          KDPDR3= 172326
AENV  = 000000          BIT9  = 001000          DH211   031760          EM201  026477          KDPDR4= 172330
AENVM = 000000          BPTVEC= 000014          DH212   032063          EM202  026553          KDPDR5= 172332
AFATAL= 000000          BUPWIN  001276          DH213   032142          EM203  026646          KDPDR6= 172334
AMADR1= 000000          CACHE = 177746          DH23    031042          EM204  026727          KDPDR7= 172336
AMADR2= 000000          CACHVE= 000114          DH24    031103          EM205  026776          KERSTK= 001100
AMADR3= 000000          CASHSR  005324          DH27    031142          EM206  027060          KIPAR0= 172340
AMADR4= 000000          CASH1   005460          DH3     030061          EM207  027111          KIPAR1= 172342
AMAMS1= 000000          CASH2   005467          DH30    031201          EM21   026022          KIPAR2= 172344
AMAMS2= 000000          CHARCT  005756          DH4     030166          EM210  027220          KIPAR3= 172346
AMAMS3= 000000          CHKLMA  005076          DH5     030255          EM211  027320          KIPAR4= 172350
AMAMS4= 000000          CHKPAT  005576          DH6     030412          EM212  027456          KIPAR5= 172352
AMSGAD= 000000          CLRMAP  002772          DISPLA  001140          EM213  027611          KIPAR6= 172354
AMSGLG= 000000          CMPE  = 177744          DISPRE  000174          EM214  027702          KIPAR7= 172356
AMSGTY= 000000          CNTR    001322          DSABLD  005230          EM22   026100          KIPDR0= 172300
AMTYP1= 000000          CONTRL= 177746          DSWR  = 177570          EM23   026166          KIPDR1= 172302
AMTYP2= 000000          CPFLAG  003224          DTMS    005736          EM24   026216          KIPDR2= 172304
AMTYP3= 000000          CPSAVE  020552          DTMSG   005742          EM25   026254          KIPDR3= 172306
AMTYP4= 000000          CPUER   003222          DT1     032212          EM26   026311          KIPDR4= 172310
APASS = 000000          CPUERR= 177766          DT10    032324          EM27   026344          KIPDR5= 172312
APRIOR= 000000          CPUEXP  001326          DT11    032334          EM3    024171          KIPDR6= 172314
APTCSU= 000040          CPUMSG  006176          DT14    032352          EM30   026444          KIPDR7= 172316
APTENV= 000001          CPUTYP  007136          DT15    032370          EM4    024277          KSP   =%000006
APTSIZ= 000200          CR    = 000015          DT2     032224          EM5    024357          LF    = 000012
APTSPO= 000100          CRLF  = 000200          DT201   032464          EM6    024452          LKS   = 177546
ASWREG= 000000          CTRAPS= 000116          DT202   032474          EM7    024554          LKVEC = 000100
ATESTN= 000000          CTRAPV= 000114          DT203   032510          ENABLD 005266          LMAH    001304
AUNIT = 000000          DATA    001364          DT204   032526          ERRCNT 001320          LMAHI = 177736
AUSWR = 000000          DATAND  001242          DT205   032542          ERROR = 104000         LMAL    001306
AVECT1= 000000          DATAOR  001246          DT206   032554          ERRVEC= 000004         LMALOW= 177734
AVECT2= 000000          DATEXT  003624          DT207   032570          ERTYPE 002026          LOADRS= 177740
BADCPU  006605          DATO  = 100000          DT210   032602          ER200  001666          LOEFLG  004312
BADPC   001340          DATOB = 140000          DT211   032612          EXTOUT 005760          LOOP    010606
```

SYMBOL TABLE

```
LOWEST  001256
LREGL   001300
LREGU   001302
MAINT = 177750
MAPADD  004106
MAPH0 = 170202
MAPH00= 170202
MAPH01= 170206
MAPH02= 170212
MAPH03= 170216
MAPH04= 170222
MAPH05= 170226
MAPH06= 170232
MAPH07= 170236
MAPH1 = 170206
MAPH10= 170242
MAPH11= 170246
MAPH12= 170252
MAPH13= 170256
MAPH14= 170262
MAPH15= 170266
MAPH16= 170272
MAPH17= 170276
MAPH2 = 170212
MAPH20= 170302
MAPH21= 170306
MAPH22= 170312
MAPH23= 170316
MAPH24= 170320
MAPH25= 170326
MAPH26= 170332
MAPH27= 170336
MAPH3 = 170216
MAPH30= 170342
MAPH31= 170346
MAPH32= 170352
MAPH33= 170356
MAPH34= 170362
MAPH35= 170366
MAPH36= 170372
MAPH37= 170376
MAPH4 = 170222
MAPH5 = 170226
MAPH6 = 170232
MAPH7 = 170236
MAPL0 = 170200
MAPL00= 170200
MAPL01= 170204
MAPL02= 170210
MAPL03= 170214
MAPL04= 170220
MAPL05= 170224
MAPL06= 170230
MAPL07= 170234
MAPL1 = 170204
MAPL10= 170240
MAPL11= 170244
MAPL12= 170250
MAPL13= 170254
MAPL14= 170260
MAPL15= 170264
MAPL16= 170270
MAPL17= 170274
MAPL2 = 170210
MAPL20= 170300
MAPL21= 170304
MAPL22= 170310
MAPL23= 170314
MAPL24= 170320
MAPL25= 170324
MAPL26= 170330
MAPL27= 170334
MAPL3 = 170214
MAPL30= 170340
MAPL31= 170344
MAPL32= 170350
MAPL33= 170354
MAPL34= 170360
MAPL35= 170364
MAPL36= 170370
MAPL37= 170374
MAPL4 = 170220
MAPL5 = 170224
MAPL6 = 170230
MAPL7 = 170234
MASK1   005774
MASK2   005776
MEMERR= 177744
MFPT  = 000007
MMFLAG  003374
MMRHI   001270
MMRLOW  001266
MMR0  = 177572
MMR1  = 177574
MMR2  = 177576
MMR3  = 172516
MMTOTM  017616
MMTRAP  003372
MMVEC = 000250
MRQUES  006752
NEWSWR  006157
NEXMEM= 000040
NEXT    003552
NOMORE  006417
NOMSG   013646
NUMOFK  001316
NXTTST  001362
OLDPC   001342
OLDPS   001344
OLDPSW  001346
PADRSH  001230
PADRSL  001226
PATAND  001252
PATCH   007140
PATEXT  003650
PATRNS  006100
PATTOR  001254
PCONTR  001334
PCPUER  001330
PIRQ  = 177772
PIRQVE= 000240
PMAINT  001336
PMBECD  002526
PMBECF  002536
PMBECH  002476
PMBECM  002442
PMBECW  002432
PMMR0   001350
PMMR1   001352
PMMR2   001354
PPARER  001332
PRETST  005166
PR0   = 000000
PR1   = 000040
PR2   = 000100
PR3   = 000140
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
PTMP2   005000
PWRMSG  023646
PWRVEC= 000024
RDCHR = 104406
RDLIN = 104407
RDOCT = 104415
RELC22  011672
RESREG= 104412
RESVEC= 000010
RETRY   001360
RSIZE   001356
R10   =%000000
R11   =%000001
R12   =%000002
R13   =%000003
R14   =%000004
R15   =%000005
R3STAK  004314
R6    =%000006
R7    =%000007
SAVREG= 104411
SCOPE = 000004
SDPAR0= 172260
SDPAR1= 172262
SDPAR2= 172264
SDPAR3= 172266
SDPAR4= 172270
SDPAR5= 172272
SDPAR6= 172274
SDPAR7= 172276
SDPDR0= 172220
SDPDR1= 172222
SDPDR2= 172224
SDPDR3= 172226
SDPDR4= 172230
SDPDR5= 172232
SDPDR6= 172234
SDPDR7= 172236
SIPAR0= 172240
SIPAR1= 172242
SIPAR2= 172244
SIPAR3= 172246
SIPAR4= 172250
SIPAR5= 172252
SIPAR6= 172254
SIPAR7= 172256
SIPDR0= 172200
SIPDR1= 172202
SIPDR2= 172204
SIPDR3= 172206
SIPDR4= 172210
SIPDR5= 172212
SIPDR6= 172214
SIPDR7= 172216
SIZEHI= 177762
SIZEJ0  012744
SIZEJ1  013154
SIZEJ2  013402
SIZELO= 177760
SPECST  006000
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
SSP   =%000006
STACK = 001100
START   010000
STKLMT= 177774
SUPSTK= 000700
SWR     001136
SWREG   000176
SW0   = 000001
SW00  = 000001
SW01  = 000002
SW02  = 000004
SW03  = 000010
SW04  = 000020
SW05  = 000040
SW06  = 000100
SW07  = 000200
SW08  = 000400
SW09  = 001000
SW1   = 000002
SW10  = 002000
SW11  = 004000
SW12  = 010000
SW13  = 020000
SW14  = 040000
SW15  = 100000
SW2   = 000004
SW3   = 000010
SW4   = 000020
SW5   = 000040
SW6   = 000100
SW7   = 000200
SW8   = 000400
SW9   = 001000
SYSTID= 177764
TBIT  = 000020
TBITO = 104413
TBITOF  002726
TBITR = 104414
TBITRE  002754
TBITVE= 000014
TCPMRA  004416
TIMEOU  003030
TIMOUT= 000020
TKVEC = 000060
TOFLAG  003032
TOMSG   006257
TPVEC = 000064
TRAPVE= 000034
TRTVEC= 000014
TSTLOC  003474
TST1    010676
TST10   014616
TST11   015072
TST12   015140
TST13   015174
TST14   015334
TST15   015362
TST16   015420
TST17   015450
TST2    011122
TST20   015610
TST21   015744
TST22   016144
TST23   016212
TST24   016420
TST25   016464
TST26   016626
TST27   016762
TST3    011444
TST30   017110
TST31   017336
TST32 = 021450
TST4    011762
TST5    012242
TST6    012630
TST7    013712
TYPDAT= 002244
TYPDS = 104405
TYPE  = 104401
```

SYMBOL TABLE

```
TYPOC = 104402          USESTK= 000600          $DOAGN  021706          $MBADR  020002          $SVPC = 000204
TYPON = 104404          USP   =%000006          $DTBL   023050          $MFLG   021444          $SWR  = 177400
TYPOS = 104403          YESMSG  013542          $ENDAD  021676          $MNEW   023361          $SWREG  020036
TYPVAD  002542          $APTHD  020000          $ENDCT  021516          $MSGAD  020030          $SWRMK= 000340
UBADDR  002650          $ATYC   021226          $ENULL  021752          $MSGLG  020032          $TBIT   021750
UBMAVA  006123          $ATY1   021202          $ENV    020034          $MSGTY  020014          $TESTN  020020
UBMEND  006172          $ATY3   021210          $ENVM   020035          $MSWR   023350          $TIMES  001212
UBMHI   001264          $ATY4   021220          $EOP    021450          $MTYP1  020045          $TKB    001144
UBMLOW  001262          $AUTOB  001132          $EOPCT  021510          $MTYP2  020051          $TKS    001142
UBMSU   015236          $BASE   020070          $ERFLG  001101          $MTYP3  020055          $TMP0   001174
UBM24L  001310          $BDADR  001120          $ERMAX  001113          $MTYP4  020061          $TMP1   001176
UBM24P  001314          $BDDAT  001124          $ERROR  020556          $MXCNT  020550          $TMP2   001200
UBM24U  001312          $BELL   001216          $ERRPC  001114          $NULL   001152          $TMP3   001202
UBRHI   001274          $CDW1   020074          $ERRTB  001366          $NWTST= 000001          $TMP4   001204
UBRLOW  001272          $CDW2   020076          $ERTTL  001110          $OCNT   022640          $TMP5   001206
UDPAR0= 177660          $CHARC  022412          $ESCAP  001214          $OCTVL  024014          $TMP6   001210
UDPAR1= 177662          $CLR.T  021666          $ETABL  020034          $OMODE  022642          $TN   = 000032
UDPAR2= 177664          $CMTAG  001100          $ETEND  020140          $OVER   020534          $TPB    001150
UDPAR3= 177666          $CM1   = 000006         $FATAL  020016          $PASS   020022          $TPFLG  001155
UDPAR4= 177670          $CM2   = 000014         $FFLG   021446          $PASTM  020006          $TPS    001146
UDPAR5= 177672          $CM3   = 000006         $FILLC  001154          $PWRAD  023622          $TRAP   023372
UDPAR6= 177674          $CM4   = 000007         $FILLS  001153          $PWRDN  023462          $TRAP2  023414
UDPAR7= 177676          $CNTLG  023343          $GDADR  001116          $PWRMG  023616          $TRP  = 000016
UDPDR0= 177620          $CNTLU  023336          $GDDAT  001122          $PWRUP  023534          $TRPAD  023426
UDPDR1= 177622          $CPUOP  020042          $GET42  021650          $QUES   001222          $TSTM   020004
UDPDR2= 177624          $CRLF   001223          $HD   = 000000          $RDCHR  023070          $TSTNM  001100
UDPDR3= 177626          $DBLK   023060          $HIBTS  020000          $RDLIN  023220          $TTYIN  023326
UDPDR4= 177630          $DB20   023712          $HIOCT  005734          $RDOCT  005626          $TYPDS  022644
UDPDR5= 177632          $DDW0   020100          $ICNT   001102          $RDSZ = 000010          $TYPE   022052
UDPDR6= 177634          $DDW1   020102          $ILLUP  023640          $REGAD  001156          $TYPEC  022264
UDPDR7= 177636          $DDW10  020124          $INTAG  001133          $REG0   001160          $TYPEX  022414
UIPAR0= 177640          $DDW11  020126          $ITEMB  001112          $REG1   001162          $TYPOC  022442
UIPAR1= 177642          $DDW12  020130          $LF     001224          $REG2   001164          $TYPON  022456
UIPAR2= 177644          $DDW13  020132          $LFLG   021445          $REG3   001166          $TYPOS  022416
UIPAR3= 177646          $DDW14  020134          $LOOP   021744          $REG4   001170          $UNIT   020026
UIPAR4= 177650          $DDW15  020136          $LPADR  001104          $REG5   001172          $UNITM  020010
UIPAR5= 177652          $DDW2   020104          $LPERR  001106          $RESRE  022014          $USWR   020040
UIPAR6= 177654          $DDW3   020106          $MADR1  020046          $RTNAD  021746          $VECT1  020064
UIPAR7= 177656          $DDW4   020110          $MADR2  020052          $RTRN   021742          $VECT2  020066
UIPDR0= 177600          $DDW5   020112          $MADR3  020056          $SAVRE  021756          $XOFF = 000023
UIPDR1= 177602          $DDW6   020114          $MADR4  020062          $SAVR6  023644          $XON  = 000021
UIPDR2= 177604          $DDW7   020116          $MAIL   020014          $SCOPE  020140          $XTSTR  020244
UIPDR3= 177606          $DDW8   020120          $MAMS1  020044          $SETUP= 000037          $$GET4= 000001
UIPDR4= 177610          $DDW9   020122          $MAMS2  020050          $SSWR   000176          $OFILL  022641
UIPDR5= 177612          $DEVCT  020024          $MAMS3  020054          $STUP = 177777          .$X   = 020000
UIPDR6= 177614          $DEVM   020072          $MAMS4  020060          $SVLAD  020500          .$Y   = 002026
UIPDR7= 177616

. ABS.  033601      000
        000000      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  56416 WORDS  ( 221 PAGES)
DYNAMIC MEMORY:  20034 WORDS  ( 77 PAGES)
ELAPSED TIME:  00:28:05
CKKUAD.BIN,CKKUAD/CR/-SP/NL:TOC=CKKUAD.MLB/ML,CKKUAD.P11
```

CKKUAD CREATED BY MACRO ON 26-OCT-81 AT 11:20 PAGE 1
SEQUENCE 138

SYMBOL CROSS REFERENCE CREF V01

SYMBOL	VALUE	REFERENCES								
ABASE	= 000000	91-4072	91-4072							
ACDW1	= 000000	91-4072	91-4072							
ACDW2	= 000000	91-4072	91-4072							
ACPUOP	= 000000	91-4072	91-4072							
ADDROR	001236	#21-1493	*41-2225	*41-2240	*92-4074	*92-4074	106-4185	106-4186	106-4187	106-4190
ADDW0	= 177777	#90-4068	91-4072	91-4072						
ADDW1	= 177777	#90-4069	91-4072	91-4072						
ADDW10	= 000000	91-4072	91-4072							
ADDW11	= 000000	91-4072	91-4072							
ADDW12	= 000000	91-4072	91-4072							
ADDW13	= 000000	91-4072	91-4072							
ADDW14	= 000000	91-4072	91-4072							
ADDW15	= 000000	91-4072	91-4072							
ADDW2	= 000000	91-4072	91-4072							
ADDW3	= 000000	91-4072	91-4072							
ADDW4	= 000000	91-4072	91-4072							
ADDW5	= 000000	91-4072	91-4072							
ADDW6	= 000000	91-4072	91-4072							
ADDW7	= 000000	91-4072	91-4072							
ADDW8	= 000000	91-4072	91-4072							
ADDW9	= 000000	91-4072	91-4072							
ADEVCT	= 000000	91-4072	91-4072							
ADEVM	= 000000	91-4072	91-4072							
ADRAND	001232	#21-1493	*41-2227	*41-2242	*92-4074	*92-4074	106-4185	106-4186	106-4187	106-4190
ADREXT	003674	35-2054	#41-2217	56-2825	57-2894	59-2972	68-3329	68-3371	69-3442	
AENV	= 000000	91-4072	91-4072							
AENVM	= 000000	91-4072	91-4072							
AFATAL	= 000000	91-4072	91-4072							
AMADR1	= 000000	91-4072	91-4072							
AMADR2	= 000000	91-4072	91-4072							
AMADR3	= 000000	91-4072	91-4072							
AMADR4	= 000000	91-4072	91-4072							
AMAMS1	= 000000	91-4072	91-4072							
AMAMS2	= 000000	91-4072	91-4072							
AMAMS3	= 000000	91-4072	91-4072							
AMAMS4	= 000000	91-4072	91-4072							
AMSGAD	= 000000	91-4072	91-4072							
AMSGLG	= 000000	91-4072	91-4072							
AMSGTY	= 000000	91-4072	91-4072							
AMTYP1	= 000000	91-4072	91-4072							
AMTYP2	= 000000	91-4072	91-4072							
AMTYP3	= 000000	91-4072	91-4072							
AMTYP4	= 000000	91-4072	91-4072							
APASS	= 000000	91-4072	91-4072							
APRIOR	= 000000	91-4072								
APTCSU	= 000040	#94-4078	97-4084							
APTENV	= 000001	93-4076	94-4078	#94-4078	97-4084					
APTSIZ	= 000200	#94-4078								
APTSPO	= 000100	94-4078	#94-4078	97-4084						
ASWREG	= 000000	91-4072	91-4072							
ATESTN	= 000000	91-4072	91-4072							
AUNIT	= 000000	91-4072	91-4072							

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES								
AUSWR	= 000000	91-4072	91-4072							
AVECT1	= 000000	91-4072	91-4072							
AVECT2	= 000000	91-4072	91-4072							
BADCPU	006605	#53-2622	54-2710							
BADPC	001340	#21-1493	*36-2089	*37-2125	106-4182	106-4183	106-4184			
BIT0	= 000001	#17-1486	38-2152	38-2157	58-2933	62-3087	63-3125	79-3722	79-3733	
BIT00	= 000001	#17-1486	17-1486	50-2484	92-4074	92-4074	93-4076	93-4076		
BIT01	= 000002	#17-1486	17-1486							
BIT02	= 000004	#17-1486	17-1486							
BIT03	= 000010	#17-1486	17-1486							
BIT04	= 000020	#17-1486	17-1486							
BIT05	= 000040	#17-1486	17-1486							
BIT06	= 000100	#17-1486	17-1486	50-2502	50-2506					
BIT07	= 000200	#17-1486	17-1486	50-2521						
BIT08	= 000400	#17-1486	17-1486	50-2485	50-2537	92-4074				
BIT09	= 001000	#17-1486	17-1486	55-2778	57-2906	59-2985	92-4074	93-4076		
BIT1	= 000002	#17-1486								
BIT10	= 002000	#17-1486	93-4076							
BIT11	= 004000	#17-1486	42-2281	44-2361	46-2426	59-2954	68-3317	68-3336	73-3523	89-4049
		92-4074								
BIT12	= 010000	#17-1486	50-2486	50-2538	95-4080					
BIT13	= 020000	#17-1486	93-4076							
BIT14	= 040000	#17-1486	79-3726	83-3828	84-3854	84-3859	85-3881	92-4074		
BIT15	= 100000	#17-1486	45-2401							
BIT2	= 000004	#17-1486								
BIT3	= 000010	#17-1486								
BIT4	= 000020	16-1457	#17-1486	32-1974	58-2934					
BIT5	= 000040	#17-1486	34-2008	63-3124	69-3409	70-3470	72-3507	87-3951		
BIT6	= 000100	#17-1486	78-3687							
BIT7	= 000200	#17-1486								
BIT8	= 000400	#17-1486								
BIT9	= 001000	#17-1486	56-2831	61-3057	68-3346	69-3453	81-3800	83-3841	84-3868	85-3894
BPTVEC	= 000014	#17-1486								
BUPWIN	001276	#21-1493	*65-3215	68-3364	68-3392					
CACHE	= 177746	#16-1461	*50-2485	50-2486	*50-2499	*50-2502	*50-2506	50-2510	*50-2521	*50-2532
		*50-2537	50-2538	*77-3649	*78-3706					
CACHVE	= 000114	#17-1486								
CASHSR	005324	#50-2484	77-3648	78-3698						
CASH1	005460	#50-2509	*77-3646	*78-3696						
CASH2	005467	#50-2513	*77-3647	*78-3697						
CHARCT	005756	#52-2596								
CHKLMA	005076	38-2165	42-2278	44-2358	#46-2422	68-3323	68-3342	68-3351		
CHKPAT	005576	#50-2544	56-2814	56-2828						
CLRMAP	002772	#34-2005	57-2879	60-3009	68-3281					
CMPE	= 177744	#16-1462	*50-2507	*50-2531						
CNTR	001322	#21-1493								
CONTRL	= 177746	#17-1486								
CPFLAG	003224	#36-2080	*36-2096	*36-2099	*36-2104	*54-2737	*93-4076			
CPSAVE	020552	29-1897	*92-4074	92-4074	#92-4074	*93-4076	93-4076			
CPUER	003222	#36-2079	54-2731	55-2784						
CPUERR	= 177766	#17-1486	35-2039	*35-2040	36-2088	*36-2105	*54-2740	*93-4076		
CPUEXP	001326	#21-1493	35-2036	35-2041	35-2050	36-2090	36-2094	36-2103	*44-2342	*44-2347

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES								
		*44-2350	*44-2352	*54-2733	*59-2959	*59-2961	*59-2964	*59-2966	*59-2974	*59-2976
		*59-2987	*59-2989	*60-3014	*60-3016	*60-3029	*60-3031	*60-3041	*61-3061	*62-3095
		*62-3098	*63-3134	*63-3137	*63-3145	*64-3170	*64-3176	*64-3179	*65-3206	*65-3209
		*65-3212	*68-3309	*68-3311	*89-4053	*89-4058	106-4182	106-4191	106-4206	
CPUMSG	006176	#53-2617	*54-2689	54-2690	*54-2695	54-2696				
CPUTYP	007136	35-2033	35-2046	36-2092	#53-2627	*54-2681	*54-2682	54-2692	55-2752	77-3635
CR	= 000015	#17-1486	97-4084	97-4084						
CRLF	= 000200	#17-1486	52-2614	52-2616	53-2617	53-2617	53-2618	53-2619	53-2620	53-2621
		53-2622	53-2623	53-2625	53-2626	97-4084	97-4084	104-4111	104-4115	104-4116
		104-4139	104-4141	105-4148	105-4150	105-4152	105-4154	105-4157	105-4159	105-4166
		105-4170	105-4175	105-4177	107-4209	107-4209	107-4210	107-4211	107-4212	107-4213
		107-4214	107-4214	107-4215	107-4216	107-4217				
CTRAPS	= 000116	#16-1465	50-2493	*50-2495	*50-2536					
CTRAPV	= 000114	#16-1464	50-2492	*50-2494	*50-2535					
DATA	001364	#21-1493	44-2343	*44-2378	*44-2385	*44-2386	*44-2387			
DATAND	001242	#21-1493	*39-2184	*56-2858	*92-4074	106-4186	106-4187	106-4189	106-4190	
DATAOR	001246	#21-1493	*39-2182	*56-2856	*92-4074	106-4186	106-4187	106-4189	106-4190	
DATEXT	003624	#39-2182	56-2821	57-2898	61-3048	68-3332	69-3445			
DATO	= 100000	#84-3851	84-3859							
DATOB	= 140000	#85-3878	85-3885							
DDISP	= 177570	#17-1486	21-1493							
DFMSG	006116	52-2594	#52-2613							
DF1	033513	22-1498	22-1504	22-1511	23-1545	23-1558	23-1566	23-1580	24-1592	24-1598
		24-1605	24-1611	24-1623	24-1629	24-1635	24-1641	25-1646	27-1720	27-1726
		#108-4219								
DF11	033543	23-1552	#108-4223							
DF14	033551	23-1573	26-1681	#108-4224						
DF201	033563	26-1655	26-1668	26-1687	26-1693	27-1706	27-1714	#108-4226		
DF204	033571	26-1674	#108-4227							
DF210	033576	26-1699	#108-4228							
DF23	033557	24-1617	#108-4225							
DF4	033520	22-1518	#108-4220							
DF5	033525	22-1525	23-1587	#108-4221						
DF6	033533	22-1532	23-1538	26-1662	#108-4222					
DH1	027773	22-1496	#105-4146							
DH10	030601	23-1543	23-1556	23-1564	24-1590	24-1596	24-1603	24-1609	#105-4156	
DH11	030626	23-1549	23-1570	23-1584	#105-4157					
DH15	030763	23-1577	#105-4159							
DH2	030032	22-1502	#105-4147							
DH201	031240	26-1653	#105-4165							
DH202	031267	26-1659	#105-4166							
DH203	031376	26-1666	#105-4168							
DH204	031457	26-1672	#105-4169							
DH205	031530	26-1678	#105-4170							
DH206	031615	26-1685	#105-4172							
DH207	031665	26-1691	#105-4173							
DH210	031726	26-1697	#105-4174							
DH211	031760	27-1703	27-1711	#105-4175						
DH212	032063	#105-4177								
DH213	032142	27-1718	27-1724	#105-4179						
DH23	031042	24-1615	#105-4161							
DH24	031103	24-1621	24-1627	24-1633	#105-4162					

SYMBOL CROSS REFERENCE CREF V01

SYMBOL	VALUE	REFERENCES							
DH27	031142	24-1639	#105-4163						
DH3	030061	22-1508	#105-4148						
DH30	031201	25-1644	#105-4164						
DH4	030160	22-1515	#105-4150						
DH5	030255	22-1522	#105-4152						
DH6	030412	22-1529	23-1535	#105-4154					
DISPLA	001140	#21-1493	47-2444	92-4074	93-4076				
DISPRE	000174	#18-1489							
DSABLD	005230	#48-2456	63-3138	64-3180	65-3204				
DSWR	= 177570	#17-1486	21-1493						
DTMS	005736	#52-2594	66-3224						
DTMSG	005742	52-2594	#52-2595						
DT1	032212	22-1497	#106-4182						
DT10	032324	23-1544	23-1557	23-1565	24-1591	24-1597	24-1604	24-1610	#106-4188
DT11	032334	23-1551	#106-4189						
DT14	032352	23-1572	23-1586	#106-4190					
DT15	032370	23-1579	#106-4191						
DT2	032224	22-1503	#106-4183						
DT201	032464	26-1654	#106-4197						
DT202	032474	26-1661	#106-4198						
DT203	032510	26-1667	#106-4199						
DT204	032526	26-1673	#106-4200						
DT205	032542	26-1680	#106-4201						
DT206	032554	26-1686	#106-4202						
DT207	032570	26-1692	#106-4203						
DT210	032602	26-1698	#106-4204						
DT211	032612	27-1705	27-1713	#106-4205					
DT212	032626	#106-4206							
DT213	032640	27-1719	27-1725	#106-4207					
DT23	032402	24-1616	#106-4192						
DT24	032414	24-1622	24-1628	#106-4193					
DT26	032426	24-1634	#106-4194						
DT27	032440	24-1640	#106-4195						
DT3	032234	22-1510	#106-4184						
DT30	032452	25-1645	#106-4196						
DT4	032250	22-1517	#106-4185						
DT5	032264	22-1524	#106-4186						
DT6	032302	22-1531	23-1537	#106-4187					
EADRES	005762	*41-2232	*41-2233	41-2244	*41-2246	*41-2247	*42-2289	*42-2290	*44-2368 *44-2369
		#52-2598	*59-2984	*79-3724	*79-3725	79-3735	*88-4022	*92-4074	106-4192 106-4197
		106-4198	106-4201	106-4202	106-4203	106-4204	106-4205		
EADRS2	005766	#52-2599	*57-2899	*69-3446	*69-3447	*79-3740	*79-3741	*92-4074	106-4192 106-4198
		106-4199	106-4201						
EMTVEC	= 000030	#17-1486	54-2652	54-2652					
EM1	024032	22-1495	#104-4104						
EM10	024655	23-1541	#104-4111						
EM11	025027	23-1548	#104-4113						
EM12	025121	23-1555	#104-4114						
EM13	025173	23-1561	#104-4115						
EM14	025377	23-1569	#104-4118						
EM15	025502	23-1576	#104-4119						
EM16	025575	23-1583	#104-4120						

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES
EM17	025673	24-1589 #104-4121
EM2	024117	22-1501 #104-4105
EM20	025723	24-1595 #104-4122
EM201	026477	26-1652 #104-4131
EM202	026553	26-1658 #104-4132
EM203	026646	26-1665 #104-4133
EM204	026727	26-1671 #104-4134
EM205	026776	26-1677 #104-4135
EM206	027060	26-1684 #104-4136
EM207	027111	26-1690 #104-4137
EM21	026022	24-1601 #104-4123
EM210	027220	26-1696 #104-4138
EM211	027320	27-1701 #104-4139
EM212	027456	27-1709 #104-4141
EM213	027611	27-1717 #104-4143
EM214	027702	27-1723 #104-4144
EM22	026100	24-1608 #104-4124
EM23	026166	24-1614 #104-4125
EM24	026216	24-1620 #104-4126
EM25	026254	24-1626 #104-4127
EM26	026311	24-1632 #104-4128
EM27	026344	24-1638 #104-4129
EM3	024171	22-1507 #104-4106
EM30	026444	25-1643 #104-4130
EM4	024277	22-1514 #104-4107
EM5	024357	22-1521 #104-4108
EM6	024452	22-1528 #104-4109
EM7	024554	23-1534 #104-4110
ENABLD	005266	#49-2471 63-3146 64-3174 65-3213
ERRCNT	001320	#21-1493 29-1788 29-1795 *35-2048 *36-2102 *55-2754 55-2756 *55-2775 55-2780 55-2786 *55-2789 *56-2806 56-2838 *56-2842 *56-2859 57-2912 *57-2915 61-3062 *61-3065 *68-3288 *68-3312 68-3395 *68-3398 69-3456 *69-3459 *87-3963 *89-4045 *92-4074 *93-4076 106-4185 106-4186 106-4187 106-4189 106-4190
ERROR =	104000	#16-1470 #17-1486 35-2044 35-2056 36-2097 36-2100 37-2131 55-2790 56-2830 56-2845 56-2847 57-2905 57-2916 59-2983 60-3022 61-3056 61-3066 61-3069 62-3102 63-3139 63-3143 63-3147 64-3175 64-3182 65-3205 65-3214 68-3345 68-3372 68-3399 69-3452 69-3460 70-3475 71-3490 72-3512 74-3567 75-3585 76-3598 77-3652 78-3701 78-3705 79-3743 80-3763 81-3779 81-3799 82-3816 83-3840 84-3867 85-3893 87-3961 88-4023
ERRVEC =	000004	#17-1486 *54-2731 *54-2732 *55-2751 *55-2784 92-4074 *92-4074 *92-4074 *92-4074
ERTYPE	002026	#29-1774 93-4076
ER200	001666	#26-1647 29-1803 29-1806
EXTOUT	005760	#52-2597
FJBIT =	000100	#80-3756 80-3759 80-3762 81-3774 81-3775 81-3778 82-3811 82-3812 82-3815
FLAG	001324	#21-1493
FLOATR	005772	38-2147 *38-2152 *38-2157 *38-2159 48-2460 49-2475 #52-2600
FTTHRU	003532	#38-2155 *63-3130
GMRMD1	007041	#53-2625 *55-2765 55-2766 55-2767
GMRMOD	007044	#53-2626 55-2769
GNS =	******	18-1489 18-1489 95-4080 95-4080 101-4092 101-4093 101-4093 101-4094

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES							
		101-4094	101-4095	101-4095					
HIADRS	= 177742	#17-1486							
HIGEST	001260	#21-1493	*64-3164	*64-3185	64-3186	65-3196	68-3359	68-3385	81-3782	87-3956
HITMIS	= 177752	#17-1486							
HT	= 000011	#17-1486	97-4084	97-4084					
IBSAVE	020554	#93-4076	*93-4076	93-4076	*93-4076	*93-4076	93-4076	93-4076	93-4076	93-4076
IOTVEC	= 000020	#17-1486	54-2652	54-2652						
JMPMSG	032654	66-3223	#107-4209							
KDPAR0	= 172360	#17-1486								
KDPAR1	= 172362	#17-1486								
KDPAR2	= 172364	#17-1486								
KDPAR3	= 172366	#17-1486								
KDPAR4	= 172370	#17-1486								
KDPAR5	= 172372	#17-1486								
KDPAR6	= 172374	#17-1486								
KDPAR7	= 172376	#17-1486								
KDPDR0	= 172320	#17-1486								
KDPDR1	= 172322	#17-1486								
KDPDR2	= 172324	#17-1486								
KDPDR3	= 172326	#17-1486								
KDPDR4	= 172330	#17-1486								
KDPDR5	= 172332	#17-1486								
KDPDR6	= 172334	#17-1486								
KDPDR7	= 172336	#17-1486								
KERSTK	= 001100	#17-1486								
KIPAR0	= 172340	#17-1486	30-1919	58-2926	*73-3552	*77-3633	*79-3732	*87-3967		
KIPAR1	= 172342	#17-1486	*73-3550	*77-3634	*87-3968					
KIPAR2	= 172344	#17-1486	69-3444	79-3729	*79-3730	*79-3742	*79-3748	83-3824	*83-3825	*83-3843
		84-3852	*84-3853	*84-3870	85-3879	*85-3880	*85-3896			
KIPAR3	= 172346	#17-1486	45-2398	*45-2399	*45-2404	54-2716	*54-2717	*54-2722	54-2723	*54-2725
KIPAR4	= 172350	#17-1486	*44-2335	44-2375	*44-2379	57-2893	*63-3127	*63-3140	63-3141	64-3155
		64-3162	*64-3168	*64-3171	64-3172	64-3185	*65-3201	65-3202	65-3215	65-3216
		*68-3289	*68-3302	*68-3303	68-3306	68-3321	68-3328	68-3340	68-3355	68-3357
		68-3359	*68-3361	68-3362	68-3364	*69-3410	*69-3417	69-3418	81-3786	*81-3787
		*81-3791	81-3792	*81-3803	*89-4046	*89-4051	106-4195			
KIPAR5	= 172352	#17-1486	*60-3024	*60-3026	61-3067	*61-3070	*62-3091	106-4196		
KIPAR6	= 172354	#17-1486	*42-2272	*44-2336	59-2950	*59-2951	*59-2956	59-2958	59-2971	*59-2990
		*59-2992	*59-2995	*60-3011	*60-3019	60-3024	*60-3036	62-3088	69-3423	69-3426
		*69-3436	69-3437	69-3441	*88-3995	*88-4002	*88-4004	*88-4029		
KIPAR7	= 172356	#17-1486	35-2053	56-2824	57-2896	68-3331	68-3370			
KIPDR0	= 172300	#17-1486	58-2921							
KIPDR1	= 172302	#17-1486								
KIPDR2	= 172304	#17-1486								
KIPDR3	= 172306	#17-1486								
KIPDR4	= 172310	#17-1486								
KIPDR5	= 172312	#17-1486								
KIPDR6	= 172314	#17-1486								
KIPDR7	= 172316	#17-1486								
KSP	=%000006	#17-1486	*29-1774	*29-1778	29-1791	29-1799	*29-1828	*29-1836	*29-1842	*29-1848
		29-1850	*29-1851	*29-1859	*29-1875	*29-1882	*29-1888	*29-1889	30-1912	*30-1927
		30-1929	*30-1930	*30-1935	30-1935	31-1945	*31-1950	31-1952	*31-1953	*31-1957
		31-1957	32-1975	32-1977	32-1978	33-1990	*35-2031	*35-2032	*35-2058	*35-2059

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES								
		*36-2086	*36-2087	*36-2106	*36-2107	37-2125	*37-2126	*37-2127	*37-2134	*37-2135
LF	= 000012	#17-1486	97-4084	97-4084						
LKS	= 177546	#17-1486								
LKVEC	= 000100	#17-1486								
LMAH	001304	#21-1493	*68-3319	68-3324	*68-3338	68-3343	68-3352			
LMAHI	= 177736	#16-1472	38-2166	42-2279	44-2359	45-2395	68-3319	68-3338	79-3737	79-3741
		80-3759	80-3761	*81-3774	81-3775	81-3777	*82-3811	82-3812	82-3814	83-3830
		83-3837	84-3856	84-3864	85-3883	85-3890	106-4193			
LMAL	001306	#21-1493	*68-3320	68-3324	*68-3339	68-3343	68-3352			
LMALOW	= 177734	#16-1471	38-2166	42-2279	44-2359	45-2391	45-2400	68-3320	68-3339	79-3735
		79-3740								
LOADRS	= 177740	#17-1486								
LOEFLG	004312	42-2263	*42-2292	#42-2299	44-2326	*44-2370	*70-3471	*71-3487	*72-3508	*74-3565
		*75-3583	*76-3596	89-4042	*89-4062					
LOOP	010606	54-2727	#54-2729	95-4080						
LOWEST	001256	#21-1493	42-2272	44-2336	*64-3155	64-3156	67-3251	68-3286	68-3289	73-3548
		79-3730	81-3780	87-3954	89-4046					
LREGL	001300	#21-1493	42-2271	44-2332	44-2380	*67-3255	67-3257	68-3285	79-3727	89-4048
LREGU	001302	#21-1493	42-2270	44-2331	44-2382	44-2383	*67-3257	*67-3258	73-3542	79-3728
		89-4047								
MAINT	= 177750	#16-1463	#17-1486	*50-2496	*50-2533					
MAPADD	004106	#42-2263	71-3489	75-3584						
MAPH0	= 170202	#17-1486								
MAPH00	= 170202	#17-1486	17-1486	56-2852						
MAPH01	= 170206	#17-1486	17-1486							
MAPH02	= 170212	#17-1486	17-1486							
MAPH03	= 170216	#17-1486	17-1486							
MAPH04	= 170222	#17-1486	17-1486							
MAPH05	= 170226	#17-1486	17-1486							
MAPH06	= 170232	#17-1486	17-1486							
MAPH07	= 170236	#17-1486	17-1486							
MAPH1	= 170206	#17-1486								
MAPH10	= 170242	#17-1486								
MAPH11	= 170246	#17-1486								
MAPH12	= 170252	#17-1486								
MAPH13	= 170256	#17-1486								
MAPH14	= 170262	#17-1486								
MAPH15	= 170266	#17-1486								
MAPH16	= 170272	#17-1486								
MAPH17	= 170276	#17-1486								
MAPH2	= 170212	#17-1486								
MAPH20	= 170302	#17-1486								
MAPH21	= 170306	#17-1486								
MAPH22	= 170312	#17-1486								
MAPH23	= 170316	#17-1486								
MAPH24	= 170320	#17-1486								
MAPH25	= 170326	#17-1486								
MAPH26	= 170332	#17-1486								
MAPH27	= 170336	#17-1486								
MAPH3	= 170216	#17-1486								
MAPH30	= 170342	#17-1486								
MAPH31	= 170346	#17-1486								

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES
MAPH32	= 170352	#17-1486
MAPH33	= 170356	#17-1486
MAPH34	= 170362	#17-1486
MAPH35	= 170366	#17-1486
MAPH36	= 170372	#17-1486
MAPH37	= 170376	#17-1486 34-2012 56-2843 56-2849 57-2877 57-2885
MAPH4	= 170222	#17-1486
MAPH5	= 170226	#17-1486
MAPH6	= 170232	#17-1486
MAPH7	= 170236	#17-1486
MAPL0	= 170200	#17-1486 34-2005 55-2755 55-2773 *59-2953 59-2967 *59-2969 59-2980 *59-2982 63-3116
MAPL00	= 170200	#17-1486 17-1486 56-2808 57-2875
MAPL01	= 170204	#17-1486 17-1486 68-3290 68-3374
MAPL02	= 170210	#17-1486 17-1486
MAPL03	= 170214	#17-1486 17-1486
MAPL04	= 170220	#17-1486 17-1486
MAPL05	= 170224	#17-1486 17-1486
MAPL06	= 170230	#17-1486 17-1486
MAPL07	= 170234	#17-1486 17-1486
MAPL1	= 170204	#17-1486
MAPL10	= 170240	#17-1486
MAPL11	= 170244	#17-1486
MAPL12	= 170250	#17-1486
MAPL13	= 170254	#17-1486
MAPL14	= 170260	#17-1486
MAPL15	= 170264	#17-1486
MAPL16	= 170270	#17-1486
MAPL17	= 170274	#17-1486
MAPL2	= 170210	#17-1486
MAPL20	= 170300	#17-1486
MAPL21	= 170304	#17-1486
MAPL22	= 170310	#17-1486
MAPL23	= 170314	#17-1486
MAPL24	= 170320	#17-1486
MAPL25	= 170324	#17-1486
MAPL26	= 170330	#17-1486
MAPL27	= 170334	#17-1486
MAPL3	= 170214	#17-1486
MAPL30	= 170340	#17-1486
MAPL31	= 170344	#17-1486
MAPL32	= 170350	#17-1486
MAPL33	= 170354	#17-1486
MAPL34	= 170360	#17-1486
MAPL35	= 170364	#17-1486
MAPL36	= 170370	#17-1486
MAPL37	= 170374	#17-1486
MAPL4	= 170220	#17-1486
MAPL5	= 170224	#17-1486
MAPL6	= 170230	#17-1486
MAPL7	= 170234	#17-1486
MASK1	005774	50-2545 #52-2601 *56-2809 56-2822 *56-2853

SYMBOL CROSS REFERENCE CREF V01

SYMBOL VALUE REFERENCES

```
MASK2     005776   50-2548  #52-2602  *56-2810  *56-2854
MEMERR  = 177744  #17-1486
MFPT    = 000007  #16-1469   54-2680
MMFLAG    003374  #37-2119  *37-2133  *54-2739  *93-4076
MMRHI     001270  #21-1493   52-2595  *64-3165  *64-3183  *64-3184
MMRLOW    001266  #21-1493   52-2595  *64-3154
MMR0    = 177572  #17-1486   17-1486   37-2128  *37-2132  *50-2484  *54-2734  *57-2874  *58-2933  *62-3087
                  *63-3125  *68-3282  *68-3287  *77-3631  *79-3722  *79-3733  *83-3826  *87-3950  *88-3997
                  *93-4076  *95-4080
MMR1    = 177574  #17-1486   17-1486   37-2129
MMR2    = 177576  #17-1486   17-1486   37-2130
MMR3    = 172516  #17-1486   17-1486   34-2008  *54-2735  *57-2881  *58-2934  *63-3124  *68-3278  *68-3280
                  *69-3409  *70-3470  *77-3632  *79-3723  *79-3731  *80-3757  *81-3784  *81-3804  *83-3827
                  *87-3965  *87-3966  *95-4080   106-4188
MMTOTM    017616   70-3474   74-3566  #89-4042
MMTRAP    003372  #37-2118   54-2729
MMVEC   = 000250  #17-1486  *54-2729  *54-2730
MRQUES    006752  #53-2624   55-2763
NEWSWR    006157  #52-2615   54-2702
NEXMEM  = 000040   44-2350   44-2353  #72-3507
NEXT      003552   38-2150   38-2153   38-2158  #38-2160
NOMORE    006417   29-1793  #53-2620
NOMSG     013646   64-3187   66-3222   66-3232  #67-3250
NUMOFK    001316  #21-1493  *87-3972   88-3996
NXTTST    001362  #21-1493  *47-2441  *87-3953
OLDPC     001342  #21-1493  *35-2031   35-2059  *36-2086   36-2089   36-2107  *37-2126   37-2135
OLDPS     001344  #21-1493  *35-2032   35-2058  *36-2087   36-2106  *37-2127   37-2134
OLDPSW    001346  #21-1493  *32-1977   33-1990  *33-1991
PADRSH    001230  #21-1493  *30-1925  *31-1949
PADRSL    001226  #21-1493  *30-1926   30-1927  *31-1948   31-1950
PATAND    001252  #21-1493  *40-2198  *56-2857  *92-4074   106-4187  106-4189
PATCH     007140  #53-2631
PATEXT    003650  #40-2196   56-2819   61-3050
PATRNS    006100  #52-2605   56-2813   60-3038
PATTOR    001254  #21-1493  *40-2196  *56-2855  *92-4074   106-4187  106-4189
PCONTR    001334  #21-1493
PCPUER    001330  #21-1493  *35-2035  *35-2039   35-2040   35-2041  *36-2088   36-2094  *36-2103   36-2105
                  *38-2161   38-2168  *44-2338   44-2348   44-2353  *60-3013   60-3017  *68-3304   68-3313
                  *87-3964   88-4020  *88-4026  *89-4052   89-4059   106-4182  106-4183   106-4191  106-4206
PIRQ    = 177772  #17-1486
PIRQVE  = 000240  #17-1486
PMAINT    001336  #21-1493
PMBECD    002526   29-1893  #29-1897
PMBECF    002536   29-1893  #29-1898
PMBECH    002476   29-1893  #29-1895
PMBECM    002442   29-1893  #29-1894
PMBECW    002432   29-1783  #29-1893
PMMR0     001350  #21-1493  *37-2128   106-4184
PMMR1     001352  #21-1493  *37-2129   106-4184
PMMR2     001354  #21-1493  *37-2130   106-4184
PPARER    001332  #21-1493
PRETST    005166  #47-2440   55-2750   56-2805   57-2873   59-2949   60-3008   62-3085   68-3277   69-3408
```

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES								
		70-3469	71-3486	72-3501	74-3564	75-3581	76-3595	77-3630	78-3686	79-3721
		80-3755	81-3773	82-3810	83-3823	84-3850	85-3877	87-3949	88-3992	
PR0	= 000000	#17-1486								
PR1	= 000040	#17-1486								
PR2	= 000100	#17-1486								
PR3	= 000140	#17-1486								
PR4	= 000200	#17-1486								
PR5	= 000240	#17-1486								
PR6	= 000300	#17-1486								
PR7	= 000340	#17-1486								
PS	= 177776	#17-1486	17-1486	*54-2639						
PSW	= 177776	#17-1486								
PTMP2	005000	42-2284	44-2363	#45-2391						
PWRMSG	023646	102-4097	#102-4098							
PWRVEC	= 000024	#17-1486	54-2653	54-2653	*102-4097	*102-4097	*102-4097	*102-4097	*102-4097	*102-4097
RDCHR	= 104406	55-2764	100-4090	#101-4092						
RDLIN	= 104407	51-2567	#101-4092							
RDOCT	= 104415	#16-1473	54-2703	#101-4092	#101-4095					
RELC22	011672	57-2913	#58-2917							
RESREG	= 104412	30-1934	31-1956	#101-4092	103-4101					
RESVEC	= 000010	#17-1486	*54-2658	54-2664	*54-2664					
RETRY	001360	#21-1493	*92-4074							
RSIZE	001356	#21-1493	*44-2375							
R10	=%000000	#17-1486								
R11	=%000001	#17-1486								
R12	=%000002	#17-1486								
R13	=%000003	#17-1486								
R14	=%000004	#17-1486								
R15	=%000005	#17-1486								
R3STAK	004314	42-2267	#43-2306							
R6	=%000006	#17-1486	*50-2530	*50-2530						
R7	=%000007	#17-1486								
SAVREG	= 104411	30-1911	31-1944	#101-4092	103-4101					
SCOPE	= 000004	#17-1486	55-2750	56-2805	57-2873	59-2949	60-3008	62-3084	68-3277	69-3408
		70-3469	71-3486	72-3500	73-3520	74-3564	75-3581	76-3595	77-3630	78-3686
		79-3721	80-3755	81-3773	82-3810	83-3823	84-3850	85-3877	87-3949	88-3992
		95-4080								
SDPAR0	= 172260	#17-1486								
SDPAR1	= 172262	#17-1486								
SDPAR2	= 172264	#17-1486								
SDPAR3	= 172266	#17-1486								
SDPAR4	= 172270	#17-1486								
SDPAR5	= 172272	#17-1486								
SDPAR6	= 172274	#17-1486								
SDPAR7	= 172276	#17-1486								
SDPDR0	= 172220	#17-1486								
SDPDR1	= 172222	#17-1486								
SDPDR2	= 172224	#17-1486								
SDPDR3	= 172226	#17-1486								
SDPDR4	= 172230	#17-1486								
SDPDR5	= 172232	#17-1486								
SDPDR6	= 172234	#17-1486								

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES							
SDPDR7	= 172236	#17-1486							
SIPAR0	= 172240	#17-1486							
SIPAR1	= 172242	#17-1486							
SIPAR2	= 172244	#17-1486							
SIPAR3	= 172246	#17-1486							
SIPAR4	= 172250	#17-1486							
SIPAR5	= 172252	#17-1486							
SIPAR6	= 172254	#17-1486							
SIPAR7	= 172256	#17-1486							
SIPDR0	= 172200	#17-1486							
SIPDR1	= 172202	#17-1486							
SIPDR2	= 172204	#17-1486							
SIPDR3	= 172206	#17-1486							
SIPDR4	= 172210	#17-1486							
SIPDR5	= 172212	#17-1486							
SIPDR6	= 172214	#17-1486							
SIPDR7	= 172216	#17-1486							
SIZEHI	= 177762	#17-1486							
SIZEJ0	012744	60-3023	62-3086	62-3101	#63-3115	87-3958			
SIZEJ1	013154	#64-3154							
SIZEJ2	013402	#65-3196							
SIZELO	= 177760	#17-1486							
SPECST	006000	#52-2603	63-3118	63-3123	66-3231	66-3234			
SR0	= 177572	#17-1486							
SR1	= 177574	#17-1486							
SR2	= 177576	#17-1486							
SR3	= 172516	#17-1486							
SSP	=%000006	#17-1486							
STACK	= 001100	#17-1486	17-1486	17-1486	17-1486	54-2645	54-2663		
START	010000	18-1489	#54-2639	63-3144	102-4097				
STKLM1	= 177774	#17-1486							
SUPSTK	= 000700	#17-1486							
SWR	001136	#21-1493	29-1797	42-2281	44-2361	46-2426	*54-2686	*54-2688	*54-2694	*54-2699
		*54-2728	55-2778	56-2831	57-2906	59-2985	61-3057	68-3317	68-3336	68-3346
		69-3453	73-3523	77-3644	78-3687	78-3694	81-3800	83-3841	84-3868	85-3894
		87-3951	89-4049	92-4074	92-4074	92-4074	92-4074	92-4074	93-4076	93-4076
		93-4076	93-4076	93-4076	95-4080	102-4097	102-4097			
SWREG	000176	#18-1489							
SW0	= 000001	#17-1486							
SW00	= 000001	#17-1486	17-1486						
SW01	= 000002	#17-1486	17-1486						
SW02	= 000004	#17-1486	17-1486						
SW03	= 000010	#17-1486	17-1486						
SW04	= 000020	#17-1486	17-1486						
SW05	= 000040	#17-1486	17-1486						
SW06	= 000100	#17-1486	17-1486						
SW07	= 000200	#17-1486	17-1486						
SW08	= 000400	#17-1486	17-1486						
SW09	= 001000	#17-1486	17-1486						
SW1	= 000002	#17-1486							
SW10	= 002000	#17-1486							
SW11	= 004000	#17-1486							

SYMBOL CROSS REFERENCE CREF V01

SYMBOL	VALUE	REFERENCES							
SW12	= 010000	#17-1486							
SW13	= 020000	#17-1486							
SW14	= 040000	#17-1486							
SW15	= 100000	#17-1486							
SW2	= 000004	#17-1486							
SW3	= 000010	#17-1486							
SW4	= 000020	#17-1486							
SW5	= 000040	#17-1486							
SW6	= 000100	#17-1486							
SW7	= 000200	#17-1486	29-1797						
SW8	= 000400	#17-1486							
SW9	= 001000	#17-1486	93-4076						
SYSTID	= 177764	#17-1486							
TBIT	= 000020	#32-1974	32-1975	32-1978	33-1991				
TBITO	= 104413	#101-4093							
TBITOF	002726	#32-1975	101-4093						
TBITR	= 104414	58-2917	73-3522	95-4080	#101-4094				
TBITRE	002754	#33-1990	101-4094						
TBITVE	= 000014	#17-1486	54-2653	54-2653					
TCPMRA	004416	#44-2326	72-3511	76-3597					
TIMEOU	003030	#35-2024	55-2751						
TIMOUT	= 000020	#16-1457	35-2050	59-2959	59-2964	59-2974	59-2987	60-3014	60-3029	60-3041
		62-3093	63-3134	64-3176	65-3206	68-3309	89-4053	89-4059		
TKVEC	= 000060	#17-1486							
TOFLAG	003032	#35-2025	*35-2043	*35-2055	*35-2057	*54-2738	*93-4076		
TOMSG	006257	#53-2618	66-3233						
TPVEC	= 000064	#17-1486							
TRAPVE	= 000034	#17-1486	54-2652	54-2653					
TRTVEC	= 000014	#17-1486							
TSTLOC	003474	#38-2146	63-3135	64-3177	65-3207	81-3794	81-3797		
TST1	010676	54-2666	54-2667	#55-2750					
TST10	014616	68-3277	68-3396	#69-3408					
TST11	015072	69-3408	69-3425	69-3457	#70-3469				
TST12	015140	70-3469	#71-3486						
TST13	015174	71-3486	#72-3500						
TST14	015334	#74-3564							
TST15	015362	74-3564	#75-3581						
TST16	015420	75-3581	#76-3595						
TST17	015450	76-3595	#77-3630						
TST2	011122	55-2750	55-2787	#56-2805					
TST20	015610	77-3630	77-3642	77-3645	77-3651	#78-3686			
TST21	015744	73-3526	77-3638	78-3686	#79-3721				
TST22	016144	79-3721	79-3744	79-3747	#80-3755				
TST23	016212	80-3755	80-3760	#81-3773					
TST24	016420	81-3773	81-3783	#82-3810					
TST25	016464	82-3810	82-3813	#83-3823					
TST26	016626	83-3823	#84-3850						
TST27	016762	84-3850	#85-3877						
TST3	011444	56-2805	56-2848	56-2850	#57-2873				
TST30	017110	78-3708	85-3877	#87-3949					
TST31	017336	87-3949	87-3978	#88-3992					
TST32	= 021450	#87-3985	88-3992						

SYMBOL CROSS REFERENCE

SYMBOL		VALUE	REFERENCES								
TST4		011762	57-2873	#59-2949							
TST5		012242	59-2949	#60-3008							
TST6		012630	60-3008	61-3068	#62-3084						
TST7		013712	#68-3277								
TYPDAT	=	002244	#29-1829	66-3228							
TYPDS	=	104405	29-1843	29-1883	66-3236	87-3983	95-4080	95-4080	#101-4092		
TYPE	=	104401	29-1793	29-1810	29-1814	29-1816	29-1821	29-1823	29-1852	29-1885	30-1931
			31-1954	54-2670	54-2690	54-2696	54-2702	54-2710	55-2763	55-2766	55-2769
			66-3223	66-3229	66-3230	66-3233	66-3237	66-3242	87-3982	87-3984	93-4076
			93-4076	95-4080	95-4080	95-4080	97-4084	98-4086	99-4088	100-4090	100-4090
			100-4090	#101-4092	102-4097						
TYPOC	=	104402	29-1779	29-1837	#101-4092						
TYPON	=	104404	#101-4092								
TYPOS	=	104403	29-1876	#101-4092							
TYPVAD		002542	29-1860	#30-1911							
UBADDR		002650	29-1868	#31-1944							
UBMAVA		006123	#52-2614	87-3982							
UBMEND		006172	#52-2616	87-3984							
UBMHI		001264	#21-1493	*64-3162	*64-3163	*65-3216	81-3792	88-4037			
UBMLOW		001262	#21-1493	*64-3159	*65-3196	*65-3197	81-3787	87-3975	88-3995		
UBMSU		015236	#73-3520								
UBM24L		001310	#21-1493	*42-2275	42-2276	42-2283	42-2289	*44-2339	44-2368	46-2433	*63-3131
			*68-3305	*68-3308	*70-3473	*72-3510					
UBM24P		001314	#21-1493	*68-3297	*68-3301	68-3308					
UBM24U		001312	#21-1493	*44-2341	44-2369	46-2431	*63-3132	*68-3279	*70-3472	*71-3488	*72-3509
UBRHI		001274	#21-1493	52-2595	*64-3160	*64-3161	*65-3217	*65-3218	87-3969		
UBRLOW		001272	#21-1493	52-2595	*64-3158	*65-3198	87-3970				
UDPAR0	=	177660	#17-1486								
UDPAR1	=	177662	#17-1486								
UDPAR2	=	177664	#17-1486								
UDPAR3	=	177666	#17-1486								
UDPAR4	=	177670	#17-1486								
UDPAR5	=	177672	#17-1486								
UDPAR6	=	177674	#17-1486								
UDPAR7	=	177676	#17-1486								
UDPDR0	=	177620	#17-1486								
UDPDR1	=	177622	#17-1486								
UDPDR2	=	177624	#17-1486								
UDPDR3	=	177626	#17-1486								
UDPDR4	=	177630	#17-1486								
UDPDR5	=	177632	#17-1486								
UDPDR6	=	177634	#17-1486								
UDPDR7	=	177636	#17-1486								
UIPAR0	=	177640	#17-1486								
UIPAR1	=	177642	#17-1486								
UIPAR2	=	177644	#17-1486								
UIPAR3	=	177646	#17-1486								
UIPAR4	=	177650	#17-1486								
UIPAR5	=	177652	#17-1486								
UIPAR6	=	177654	#17-1486								
UIPAR7	=	177656	#17-1486								
UIPDR0	=	177600	#17-1486								

SYMBOL CROSS REFERENCE CREF V01

```
SYMBOL  VALUE           REFERENCES
UIPDR1  = 177602        #17-1486
UIPDR2  = 177604        #17-1486
UIPDR3  = 177606        #17-1486
UIPDR4  = 177610        #17-1486
UIPDR5  = 177612        #17-1486
UIPDR6  = 177614        #17-1486
UIPDR7  = 177616        #17-1486
USESTK  = 000600        #17-1486
USP     =%000006        #17-1486
YESMSG    013542         64-3166   #66-3221
$APTHD    020000         90-4070   #90-4070
$ASTAT  = ******         94-4078    94-4078
$ATYC     021226         94-4078   #94-4078
$ATY1     021202        #94-4078
$ATY3     021210        #94-4078    97-4084
$ATY4     021220         93-4076   #94-4078
$AUTOB    001132        #21-1493
$BASE     020070        #91-4072
$BDADR    001120        #21-1493
$BDDAT    001124        #21-1493
$BELL     001216        #21-1493    93-4076    93-4076    93-4076
$CDW1     020074        #91-4072
$CDW2     020076        #91-4072
$CHARC    022412        *97-4084   *97-4084    97-4084   *97-4084   #97-4084
$CKSWR  = ******         101-4092
$CLR.T    021666         95-4080   #95-4080
$CMTAG    001100        #21-1493    54-2640    54-2641
$CM1    = 000006        #21-1493    21-1493    21-1493   #21-1493    21-1493    21-1493   #21-1493    21-1493    21-1493
                        #21-1493    21-1493    21-1493   #21-1493    21-1493    21-1493   #21-1493    21-1493    21-1493
                        #21-1493
$CM2    = 000014        #21-1493    21-1493    21-1493   #21-1493    21-1493    21-1493   #21-1493    21-1493    21-1493
                        #21-1493    21-1493    21-1493   #21-1493    21-1493    21-1493   #21-1493    21-1493    21-1493
                        #21-1493
$CM3    = 000006        #21-1493    21-1493    21-1493
$CM4    = 000007        #21-1493    21-1493    21-1493   #21-1493    21-1493    21-1493   #21-1493    21-1493    21-1493
                        #21-1493    21-1493    21-1493   #21-1493    21-1493    21-1493   #21-1493    21-1493    21-1493
                        #21-1493    21-1493    21-1493   #21-1493
$CNTLG    023343        #100-4090
$CNTLU    023336        #100-4090
$CPUOP    020042        #91-4072
$CRLF     001223        #21-1493    29-1810    29-1816    29-1823    66-3229    66-3230    66-3237    66-3242    93-4076
                         93-4076    93-4076    95-4080    97-4084    97-4084    97-4084   100-4090   100-4090
$DBLK     023060         99-4088    99-4088   #99-4088
$DB20     023712         29-1849    30-1928    31-1951   #103-4101
$DDW0     020100         63-3115   #91-4072
$DDW1     020102         38-2149    38-2151   #91-4072
$DDW10    020124        #91-4072
$DDW11    020126        #91-4072
$DDW12    020130        #91-4072
$DDW13    020132        #91-4072
$DDW14    020134        #91-4072
$DDW15    020136        #91-4072
```

SYMBOL	VALUE	REFERENCES								
$DDW2	020104	#91-4072								
$DDW3	020106	#91-4072								
$DDW4	020110	#91-4072								
$DDW5	020112	#91-4072								
$DDW6	020114	#91-4072								
$DDW7	020116	#91-4072								
$DDW8	020120	#91-4072								
$DDW9	020122	#91-4072								
$DEVCT	020024	#91-4072								
$DEVM	020072	#91-4072								
$DOAGN	021706	95-4080	95-4080	95-4080	#95-4080					
$DTBL	023050	99-4088	#99-4088							
$ENDAD	021676	19-1491	93-4076	#95-4080						
$ENDCT	021516	54-2654	#95-4080							
$ENULL	021752	#95-4080								
$ENV	020034	48-2456	49-2471	55-2758	#91-4072	93-4076	94-4078	94-4078	97-4084	
$ENVM	020035	48-2458	49-2473	54-2697	54-2726	77-3639	78-3689	#91-4072	94-4078	97-4084
		97-4084								
$EOP	021450	87-3953	87-3962	87-3985	88-4034	#95-4080				
$EOPCT	021510	*54-2654	#95-4080	95-4080						
$ERFLG	001101	#21-1493	92-4074	*92-4074	92-4074	92-4074	*92-4074	92-4074	92-4074	*93-4076
		93-4076	93-4076							
$ERMAX	001113	#21-1493	*54-2656	92-4074	*92-4074	92-4074	92-4074			
$ERROR	020556	54-2652	#93-4076							
$ERRPC	001114	#21-1493	29-1778	29-1897	*93-4076	*93-4076	93-4076	93-4076	93-4076	106-4185
		106-4188	106-4191	106-4192	106-4193	106-4194	106-4195	106-4196	106-4197	106-4198
		106-4199	106-4200	106-4201	106-4202	106-4203	106-4204	106-4205	106-4206	106-4207
$ERRTB	001366	#22-1493	29-1806	29-1811						
$ERTTL	001110	#21-1493	*55-2788	*56-2841	*57-2914	*61-3064	*68-3397	*69-3458	*88-4025	*93-4076
		93-4076	93-4076	95-4080	95-4080	*95-4080				
$ESCAP	001214	#21-1493	*54-2655	*92-4074	93-4076	93-4076	93-4076	93-4076		
$ETABL	020034	#91-4072								
$ETEND	020140	90-4070	#91-4072							
$FATAL	020016	#91-4072	*94-4078							
$FFLG	021446	*94-4078	*94-4078	94-4078	*94-4078	#94-4078				
$FILLC	001154	#21-1493	97-4084	97-4084	97-4084					
$FILLS	001153	#21-1493	97-4084	97-4084						
$GDADR	001116	#21-1493								
$GDDAT	001122	#21-1493								
$GET42	021650	#95-4080								
$GTSWR	= ******	101-4092								
$HD	= 000000	16-1482	16-1482	16-1482						
$HIBTS	020000	#90-4070								
$HIOCT	005734	*51-2587	#51-2592							
$ICNT	001102	#21-1493	*92-4074	92-4074	*92-4074	92-4074	92-4074			
$ILLUP	023640	102-4097	102-4097	#102-4097						
$INTAG	001133	#21-1493								
$ITEMB	001112	#21-1493	29-1776	*93-4076	93-4076	93-4076	*93-4076	93-4076	93-4076	93-4076
$LF	001224	#21-1493	93-4076	93-4076	97-4084	97-4084	100-4090	100-4090	100-4090	
$LFLG	021445	*94-4078	#94-4078							
$LOOP	021744	95-4080	#95-4080							
$LPADR	001104	#21-1493	*47-2446	*54-2666	*92-4074	*92-4074	92-4074	92-4074	92-4074	

SYMBOL CROSS REFERENCE CREF V01

SYMBOL	VALUE	REFERENCES
$LPERR	001106	#21-1493 *42-2269 *44-2337 *47-2442 *54-2667 *55-2776 *55-2785 *56-2807 *56-2833 *56-2840 *56-2851 *61-3051 *63-3133 *64-3167 *65-3199 *81-3785 *87-3958 92-4074 *92-4074 92-4074 92-4074 93-4076
$MADR1	020046	#91-4072
$MADR2	020052	#91-4072
$MADR3	020056	#91-4072
$MADR4	020062	#91-4072
$MAIL	020014	90-4070 90-4070 #91-4072 92-4074 93-4076 97-4084
$MAMS1	020044	#91-4072
$MAMS2	020050	#91-4072
$MAMS3	020054	#91-4072
$MAMS4	020060	#91-4072
$MBADR	020002	#90-4070
$MFLG	021444	*94-4078 94-4078 *94-4078 #94-4078
$MNEW	023361	#100-4090
$MSGAD	020030	#91-4072 *94-4078 94-4078
$MSGLG	020032	#91-4072 *94-4078
$MSGTY	020014	*35-2027 *36-2082 *37-2121 *54-2709 *55-2760 #91-4072 94-4078 *94-4078 94-4078 *94-4078
$MSWR	023350	#100-4090
$MTYP1	020045	#91-4072
$MTYP2	020051	#91-4072
$MTYP3	020055	#91-4072
$MTYP4	020061	#91-4072
$MXCNT	020550	92-4074 92-4074 92-4074 #92-4074
$NULL	001152	#21-1493 97-4084 97-4084 97-4084
$NWTST	= 000001	#54-2750 54-2750 #55-2750 55-2750 #55-2805 55-2805 #56-2805 56-2805 #56-2873 56-2873 #57-2873 57-2873 #58-2949 58-2949 #59-2949 59-2949 #59-3008 59-3008 #60-3008 60-3008 #61-3084 61-3084 #62-3084 62-3084 #67-3277 67-3277 #68-3277 68-3277 #68-3408 68-3408 #69-3408 69-3408 #69-3469 69-3469 #70-3469 70-3469 #70-3486 70-3486 #71-3486 71-3486 #71-3500 71-3500 #72-3500 72-3500 #73-3564 73-3564 #74-3564 74-3564 #74-3581 74-3581 #75-3581 75-3581 #75-3595 75-3595 #76-3595 76-3595 #76-3630 76-3630 #77-3630 77-3630 #77-3686 77-3686 #78-3686 78-3686 #78-3721 78-3721 #79-3721 79-3721 #79-3755 79-3755 #80-3755 80-3755 #80-3773 80-3773 #81-3773 81-3773 #81-3810 81-3810 #82-3810 82-3810 #82-3823 82-3823 #83-3823 83-3823 #83-3850 83-3850 #84-3850 84-3850 #84-3877 84-3877 #85-3877 85-3877 #86-3949 86-3949 #87-3949 87-3949 #87-3992 87-3992 #88-3992 88-3992
$OCNT	022640	*98-4086 *98-4086 #98-4086
$OCTVL	024014	103-4101 #103-4101
$OMODE	022642	*98-4086 *98-4086 98-4086 *98-4086 *98-4086 #98-4086
$OVER	020534	92-4074 92-4074 92-4074 92-4074 #92-4074
$PASS	020022	*54-2644 54-2674 66-3221 87-3977 #91-4072 92-4074 92-4074 92-4074 *95-4080 *95-4080 95-4080 95-4080 95-4080
$PASTM	020006	#90-4070
$PWRAD	023622	#102-4097
$PWRDN	023462	54-2653 #102-4097 102-4097
$PWRMG	023616	#102-4097
$PWRUP	023534	102-4097 #102-4097
$QUES	001222	#21-1493 93-4076 93-4076 97-4084 97-4084 100-4090 100-4090 100-4090
$RDCHR	023070	#100-4090 101-4092 101-4092
$RDDEC	= ******	101-4092

SYMBOL CROSS REFERENCE CREF V01

SYMBOL	VALUE	REFERENCES								
$RDLIN	023220	#100-4090	101-4092	101-4092						
$RDOCT	005626	#51-2562	101-4092	101-4092	101-4095					
$RDSZ	= 000010	#100-4090	100-4090							
$REGAD	001156	#21-1493								
$REG0	001160	#21-1493	*93-4076	106-4203						
$REG1	001162	#21-1493	*93-4076	106-4193						
$REG2	001164	#21-1493	*93-4076	106-4194	106-4200	106-4205				
$REG3	001166	#21-1493	*93-4076	106-4199	106-4205					
$REG4	001170	#21-1493	*93-4076	106-4199						
$REG5	001172	#21-1493	*93-4076	106-4207						
$RESRE	022014	#96-4082	101-4092							
$RTNAD	021746	#95-4080								
$RTRN	021742	54-2653	*54-2657	*54-2662	95-4080	#95-4080				
$R2A	= ******	101-4092								
$SAVRE	021756	#96-4082	101-4092	101-4092						
$SAVR6	023644	*102-4097	102-4097	*102-4097	*102-4097	#102-4097				
$SCOPE	020140	54-2652	#92-4074							
$SETUP	= 000037	#18-1488	18-1488	#18-1488	18-1488	#18-1488	18-1488	#18-1488	18-1488	#18-1488
		18-1488	#18-1488	92-4074	93-4076	93-4076	93-4076	93-4076	95-4080	95-4080
		100-4090	100-4090	10[illegible]-4097						
$SSWR	000176	#28-1730	54-2694	54-2699	54-2700	*54-2704	*54-2706			
$STUP	= 177777	#18-1488	#18-1488	18-1488	#18-1488	#18-1488	18-1488	#18-1488	#18-1488	18-1488
		#18-1488	#18-1488	18-1488	#18-1488	#18-1488	18-1488			
$SVLAD	020500	92-4074	#92-4074							
$SVPC	= 000204	#19-1491	19-1491							
$SWR	= 177400	#16-1467	16-1482	16-1483	16-1483	16-1483	16-1483	16-1483	16-1483	16-1483
		16-1483	21-1493	21-1493	21-1493	55-2750	56-2805	57-2873	59-2949	60-3008
		62-3084	68-3277	69-3408	70-3469	71-3486	72-3500	74-3564	75-3581	76-3595
		77-3630	78-3686	79-3721	80-3755	81-3773	82-3810	83-3823	84-3850	85-3877
		87-3949	88-3992	92-4074	92-4074	92-4074	92-4074	92-4074	92-4074	92-4074
		92-4074	92-4074	92-4074	92-4074	92-4074	92-4074	92-4074	92-4074	92-4074
		92-4074	92-4074	92-4074	92-4074	93-4076	93-4076	93-4076	93-4076	93-4076
		93-4076	93-4076	93-4076	93-4076	93-4076	93-4076	95-4080	95-4080	95-4080
		95-4080	95-4080	102-4097						
$SWREG	020036	54-2728	77-3641	78-3691	#91-4072					
$SWRMK	= 000340	#16-1466	16-1483	16-1483	16-1483	16-1483	16-1483	16-1483	16-1483	16-1483
		16-1483	92-4074	92-4074	92-4074	92-4074	92-4074	92-4074	92-4074	92-4074
		92-4074	92-4074	92-4074						
$TBIT	021750	*54-2665	*95-4080	95-4080	95-4080	#95-4080	*102-4097			
$TESTN	020020	29-1897	52-2595	*87-3960	#91-4072	*92-4074	*93-4076	106-4182	106-4183	106-4184
		106-4185	106-4186	106-4187	106-4188	106-4189	106-4190	106-4191	106-4192	106-4193
		106-4194	106-4195	106-4196	106-4197	106-4198	106-4199	106-4200	106-4201	106-4202
		106-4203	106-4204	106-4205	106-4206	106-4207				
$TIMES	001212	#21-1493	*47-2447	*92-4074	92-4074	*92-4074	92-4074	92-4074	*95-4080	
$TKB	001144	#21-1493	97-4084	97-4084	97-4084	97-4084	100-4090	100-4090	100-4090	100-4090
$TKS	001142	#21-1493	54-2641	97-4084	97-4084	97-4084	97-4084	100-4090	100-4090	100-4090
		100-4090								
$TMPO	001174	#21-1493	*38-2146	38-2170	*56-2817	*56-2822	*57-2878	*57-2883	57-2884	57-2901
		*61-3045	*63-3129	64-3154	64-3160	64-3183	65-3198	65-3217	*68-3283	*68-3349
		68-3367	*68-3373	*79-3737	*79-3738	*83-3830	83-3831	83-3833	*83-3837	83-3838
		*84-3856	84-3857	*84-3859	84-3860	*84-3864	84-3865	*85-3883	85-3884	85-3886
		*85-3890	85-3891	106-4194	106-4195	106-4199	106-4200	106-4207		

SYMBOL CROSS REFERENCE CREF V01

SYMBOL	VALUE	REFERENCES								
$TMP1	001176	#21-1493	*42-2283	*42-2286	*44-2366	*48-2462	*49-2477	*61-3046	106-4200	106-4207
$TMP2	001200	#21-1493	*42-2287	*44-2365	*45-2400	*45-2401	*45-2402	45-2403	*45-2403	*59-2950
		59-2995	*83-3831	*83-3832	83-3833	83-3838	*84-3857	*84-3858	84-3860	84-3865
		*85-3884	*85-3885	85-3886	85-3891					
$TMP3	001202	#21-1493	*57-2891	57-2892	*68-3326	68-3327	106-4198			
$TMP4	001204	#21-1493	*88-4013	*88-4018	106-4202					
$TMP5	001206	#21-1493	*41-2224	41-2229	41-2230	*60-3025	*60-3027	61-3067	61-3070	*88-4019
		106-4196	106-4202							
$TMP6	001210	#21-1493	*60-3012	*60-3020	60-3025					
$TN	= 000032	#16-1468	16-1482	54-2750	55-2750	55-2750	55-2750	#55-2750	55-2787	55-2805
		56-2805	56-2805	56-2805	#56-2805	56-2848	56-2850	56-2873	57-2873	57-2873
		57-2873	#57-2873	58-2949	59-2949	59-2949	59-2949	#59-2949	59-3008	60-3008
		60-3008	60-3008	#60-3008	61-3068	61-3084	62-3084	#62-3084	67-3277	68-3277
		68-3277	68-3277	#68-3277	68-3396	68-3408	69-3408	69-3408	69-3408	#69-3408
		69-3425	69-3457	69-3469	70-3469	70-3469	70-3469	#70-3469	70-3486	71-3486
		71-3486	71-3486	#71-3486	71-3500	72-3500	#72-3500	73-3564	74-3564	74-3564
		74-3564	#74-3564	74-3581	75-3581	75-3581	75-3581	#75-3581	75-3595	76-3595
		76-3595	76-3595	#76-3595	76-3630	77-3630	77-3630	77-3630	#77-3630	77-3642
		77-3645	77-3651	77-3686	78-3686	78-3686	78-3686	#78-3686	78-3708	78-3721
		79-3721	79-3721	79-3721	#79-3721	79-3744	79-3747	79-3755	80-3755	80-3755
		80-3755	#80-3755	80-3760	80-3773	81-3773	81-3773	81-3773	#81-3773	81-3783
		81-3810	82-3810	82-3810	82-3810	#82-3810	82-3813	82-3823	83-3823	83-3823
		83-3823	#83-3823	83-3850	84-3850	84-3850	84-3850	#84-3850	84-3877	85-3877
		85-3877	85-3877	#85-3877	86-3949	87-3949	87-3949	87-3949	#87-3949	87-3978
		87-3992	88-3992	88-3992	88-3992	#88-3992				
$TPB	001150	#21-1493	97-4084	97-4084	97-4084					
$TPFLG	001155	#21-1493	97-4084	97-4084	97-4084					
$TPS	001146	#21-1493	97-4084	97-4084	97-4084					
$TRAP	023372	54-2652	#101-4092							
$TRAP2	023414	#101-4092	101-4092							
$TRP	= 000016	#101-4092	101-4092	101-4092	101-4092	101-4092	#101-4092	101-4092	101-4092	101-4092
		101-4092	#101-4092	101-4092	101-4092	101-4092	101-4092	#101-4092	101-4092	101-4092
		101-4092	101-4092	#101-4092	101-4092	101-4092	101-4092	101-4092	#101-4092	101-4092
		101-4092	101-4092	101-4092	#101-4092	101-4092	101-4092	101-4092	101-4092	#101-4092
		101-4092	101-4092	101-4092	101-4092	#101-4092	101-4092	101-4092	101-4092	101-4092
		#101-4092	101-4092	101-4092	101-4092	101-4092	#101-4092	101-4093	101-4093	101-4093
		101-4093	#101-4093	101-4094	101-4094	101-4094	101-4094	#101-4094	101-4095	101-4095
		101-4095	101-4095	#101-4095						
$TRPAD	023426	101-4092	#101-4092							
$TSTM	020004	#90-4070								
$TSTNM	001100	#21-1493	*47-2443	47-2444	*73-3521	*73-3525	*77-3637	*78-3707	*87-3959	92-4074
		92-4074	*92-4074	92-4074	92-4074	92-4074	92-4074	93-4076	93-4076	93-4076
		93-4076	*95-4080							
$TTYIN	023326	100-4090	100-4090	100-4090	#100-4090					
$TYPBN	= ******	101-4092								
$TYPDS	022644	#99-4088	101-4092	101-4092						
$TYPE	022052	94-4078	#97-4084	101-4092	101-4092					
$TYPEC	022264	97-4084	97-4084	97-4084	#97-4084					
$TYPEX	022414	97-4084	97-4084	97-4084	#97-4084					
$TYPOC	022442	#98-4086	101-4092	101-4092						
$TYPON	022456	98-4086	#98-4086	101-4092						
$TYPOS	022416	#98-4086	101-4092							

SYMBOL	VALUE	REFERENCES			
$UNIT	020026	#91-4072			
$UNITM	020010	#90-4070			
$USWR	020040	#91-4072			
$VECT1	020064	#91-4072			
$VECT2	020066	#91-4072			
$XOFF	= 000023	97-4084	97-4084		
$XON	= 000021	97-4084	97-4084	97-4084	100-4090
$XTSTR	020244	#92-4074			
$$GET4	= 000001	#95-4080	#95-4080	95-4080	
$OFILL	022641	*98-4086	*98-4086	98-4086	#98-4086
$40CAT	= ******	92-4074	93-4076		
.$ASTA	= ******	94-4078	94-4078		
.$X	= 020000	#90-4070	90-4070		
.$Y	= 002026	#28-1728	28-1731		

MACRO CROSS REFERENCE CREF V01

MACRO NAME	REFERENCES									
COMMEN	#17-1486									
DONE	#16-1343	95-4080								
ENDCOM	#17-1486									
ESCAPE	#17-1486									
GETPRI	#17-1486	#95-4080								
GETSWR	#17-1486									
MSG	#80-3764	#81-3773	#81-3805	#82-3810	#82-3817	#83-3823	#83-3844	#84-3850	#84-3871	#85-3877
	#87-3986	#88-3992								
MSG1	#54-2741	55-2750								
MSG11	#67-3259	68-3277								
MSG12	#68-3400	#69-3408								
MSG13	#69-3461	70-3469								
MSG14	#70-3476	71-3486								
MSG15	#71-3492	#72-3500								
MSG16	#73-3553	#74-3564								
MSG17	#74-3568	75-3581								
MSG2	#55-2791	#56-2805								
MSG20	#75-3587	76-3595								
MSG21	#76-3600	#77-3630								
MSG22	#77-3653	78-3686								
MSG220	#78-3709	79-3721								
MSG221	#79-3749	80-3755								
MSG23	#86-3938	#87-3949								
MSG4	#56-2861	#57-2873								
MSG5	#58-2939	#59-2949								
MSG6	#59-2996	#60-3008								
MSG7	#61-3071	#62-3084								
MULT	#17-1486									
NEWTST	#16-1459	#17-1486	#54-2750	#55-2805	#56-2873	#58-2949	#59-3008	#61-3084	#67-3277	#68-3408
	#69-3469	#70-3486	#71-3500	#73-3564	#74-3581	#75-3595	#76-3630	#77-3686	#78-3721	#79-3755
	#80-3773	#81-3810	#82-3823	#83-3850	#84-3877	#86-3949	#87-3992			
POP	#17-1486	#94-4078	#94-4078	#96-4082	#99-4088	#102-4097	#102-4097			
PUSH	#17-1486	94-4078	94-4078	94-4078	96-4082	99-4088	102-4097	102-4097		
REPORT	#17-1486									
SAVTST	#16-1381	#93-4076								
SETPRI	#17-1486									
SETTRA	#101-4092	101-4092	101-4092	101-4092	101-4092	101-4092	101-4092	101-4092	101-4092	101-4092
	101-4092	101-4093	101-4094	101-4095						
SETUP	#17-1486									
SKIP	#17-1486	#55-2787	#56-2848	#56-2850	#61-3068	#68-3396	#69-3425	#69-3457	#77-3642	#77-3645
	#77-3651	#79-3744	#79-3747	#80-3760	#81-3783	#82-3813	#87-3978			
SKIPJ	#16-1334	78-3708								
SLASH	#17-1486									
SPACE	#16-1340	#17-1486								
SSCOPE	#16-1359	#92-4074								
STARS	#17-1486	19-1491	21-1493	21-1493	29-1733	29-1773	30-1900	30-1910	31-1938	31-1943
	32-1960	32-1961	32-1963	32-1964	32-1966	32-1973	33-1981	33-1989	34-1995	34-2004
	35-2016	35-2023	36-2062	36-2064	36-2066	36-2078	37-2110	37-2117	38-2138	38-2145
	39-2174	39-2181	40-2188	40-2195	41-2202	41-2216	42-2252	42-2262	44-2318	44-2325
	46-2407	46-2421	47-2439	48-2450	48-2451	49-2467	50-2482	50-2543	51-2554	55-2750
	55-2750	56-2805	56-2805	57-2873	57-2873	59-2949	59-2949	60-3008	60-3008	62-3084
	62-3084	63-3104	63-3114	64-3149	64-3153	65-3189	65-3195	66-3220	67-3244	67-3249

C 13


```
MACRO CROSS REFERENCE                                            CREF    V01
MACRO NAME      REFERENCES
                 68-3277    68-3277    69-3408    69-3408    70-3469    70-3469    71-3486    71-3486    72-3500    72-3500
                 72-3502    72-3505    73-3516    73-3518    73-3528    73-3540    74-3564    74-3564    75-3581    75-3581
                 76-3595    76-3595    77-3630    77-3630    77-3630    77-3630    77-3630    78-3686    78-3686    79-3721
                 79-3721    80-3755    80-3755    81-3773    81-3773    82-3810    82-3810    83-3823    83-3823    84-3850
                 84-3850    85-3877    85-3877    85-3897    85-3903    86-3906    86-3937    87-3949    87-3949    88-3992
                 88-3992    89-4041    90-4070    90-4070    90-4070    91-4072    92-4074    93-4076    94-4078    95-4080
                 96-4082    97-4084    98-4086    99-4088    100-4090   100-4090   100-4090   101-4092   102-4097   102-4097
                 103-4101
SWRSU           #17-1486
TRACK           #16-1373    55-2750    56-2805    57-2873    59-2949    60-3008    68-3277    69-3408    70-3469    71-3486
                 74-3564    75-3581    76-3595    77-3630    78-3686    79-3721    80-3755    81-3773    82-3810    83-3823
                 84-3850    85-3877    87-3949    88-3992
TRACK1          #16-1376    55-2750    56-2805    57-2873    59-2949    60-3008    68-3277    69-3408    70-3469    71-3486
                 74-3564    75-3581    76-3595    77-3630    78-3686    79-3721    80-3755    81-3773    82-3810    83-3823
                 84-3850    85-3877    87-3949    88-3992
TRMTRP          #101-4092
TYPBIN          #17-1486
TYPDEC          #17-1486    95-4080    95-4080
TYPNAM          #17-1486
TYPNUM          #17-1486
TYPOCS          #17-1486
TYPOCT          #17-1486
TYPTXT          #16-1456   #17-1486   #95-4080   #95-4080
USER            #16-1401   #21-1493
ZEROER          #16-1349   #16-1391   #93-4076
$$CMRE          #20-1493   #21-1493   #21-1493   #21-1493   #21-1493   #21-1493   #21-1493
$$CMTM          #20-1493   #21-1493   #21-1493   #21-1493   #21-1493   #21-1493   #21-1493   #21-1493
$$ESCA          #17-1486
$$NEWT          #16-1459   #17-1486   #54-2750   #55-2805   #56-2873   #58-2949   #59-3008   #61-3084   #67-3277   #68-3408
                #69-3469   #70-3486   #71-3500   #73-3564   #74-3581   #75-3595   #76-3630   #77-3686   #78-3721   #79-3755
                #80-3773   #81-3810   #82-3823   #83-3850   #84-3877   #86-3949   #87-3992
$$SET           #101-4092  #101-4092  #101-4092  #101-4092  #101-4092  #101-4092  #101-4092  #101-4092  #101-4092  #101-4092
                #101-4092  #101-4093  #101-4094  #101-4095
$$SKIP          #17-1486   #55-2787   #56-2848   #56-2850   #61-3068   #68-3396   #69-3425   #69-3457   #77-3642   #77-3645
                #77-3651   #79-3744   #79-3747   #80-3760   #81-3783   #82-3813   #87-3978
.EQUIV          #16-1337    16-1457    17-1486    17-1486    17-1486    17-1486    17-1486    17-1486    17-1486    17-1486
                 17-1486    17-1486    17-1486    17-1486    17-1486    17-1486    17-1486    17-1486    17-1486    17-1486
                 17-1486    17-1486    17-1486    17-1486    17-1486    17-1486    17-1486    17-1486    17-1486    17-1486
                 17-1486    17-1486    17-1486    17-1486    17-1486    17-1486    17-1486    17-1486    17-1486    17-1486
                 17-1486    17-1486    17-1486    17-1486    17-1486    17-1486    17-1486    17-1486    17-1486    17-1486
                 17-1486    17-1486    17-1486    17-1486    32-1974    72-3507
.HEADE          #16-1456    16-1482
.SETUP          #16-1456    18-1488
.SWRHI          #16-1270   #16-1483
.SWRLO          #16-1483    16-1484
.$ACT1          #16-1456    19-1491
.$APTB          #16-1460   #91-4072
.$APTH          #16-1460   #90-4070
.$APTY          #16-1460   #94-4078
.$CATC          #16-1456    18-1489
.$CMTA          #16-689    #16-1456   #20-1493
.$DB20          #16-1459    103-4101
```

MACRO CROSS REFERENCE

MACRO NAME	REFERENCES	
.$EOP	#16-472	95-4080
.$ERRO	#16-860	#93-4076
.$POWE	#16-1459	#102-4097
.$READ	#16-1458	100-4090
.$SAVE	#16-1458	96-4082
.$SCOP	#16-1039	92-4074
.$TRAP	#16-1459	#101-4092
.$TYPD	#16-1458	#99-4088
.$TYPE	#16-1458	97-4084
.$TYPO	#16-1458	98-4086
.1170	#16-1456	17-1486