KDJ11-B KDJ11-B CLUSTER DIAG
COKDABO

COPYRIGHT (c) 1984
AH-T854B-MC
FICHE 01 OF 03

JUL 1984
digital
Made In USA

KDJ11-B KDJ11-B CLUSTER DIAG
COKDABO

COPYRIGHT (c) 1984
AH-T854B-MC
FICHE 02 OF 03

JUL 1984
digital
Made In USA

KDJ11-B KDJ11-B CLUSTER DIAG
COKDABO

COPYRIGHT (c) 1984
AH-T854B-MC
FICHE 03 OF 03

JUL 1984
digital
Made In USA

```
                                .REM &

                        IDENTIFICATION
                        --------------

PRODUCT CODE:   AC-T853B-MC

PRODUCT NAME:   COKDAB0 KDJ11-B CLUSTER DIAG

PRODUCT DATE:   APRIL, 1984

MAINTAINER:     DIAGNOSTIC ENGINEERING
```

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.

NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS AFFILIATED COMPANIES.

COPYRIGHT (C) 1984 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	

TABLE OF CONTENTS

1. PROGRAM ABSRACT

2. SYSTEM REQUIREMENTS

3. LOADING AND STARTING PROCEDURES

4. SPECIAL ENVIRONMENTS

5. PROGRAM OPTIONS

6. EXECUTION TIMES

7. ERROR INFORMATION

8. EXAMPLES

9. PROGRAM DESCRIPTION

9.1 J11 CODE
9.2 CACHE CODE
9.3 ON-BOARD ROM CODE
9.4 LINE TIME CLOCK CODE
9.5 SERIAL LINE UNIT CODE
9.6 Q22BE CODE

```
                                1.   PROGRAM ABSTRACT


THIS PROGRAM TESTS OUT KDJ11-B CPU BOARD, INCLUDING THE J11 CHIP SET,
ON-BOARD CACHE, ON-BOARD ROM'S, SERIAL LINE UNIT, AND LINE TIME
CLOCKS.
THE KDJ11-B IS A PDP-11 CPU THAT INCORPORATES THE J11 CHIP SET AS THE
HEART OF THE PROCESSOR. IT IS A QUAD HEIGHT Q22 BUS MODULE. IT HAS
ON-BOARD CACHE, SOME OF THE FUNCTIONALITY OF THE CACHE IS HIDDEN INSIDE
THE J11 AND THE REST OF THE FUNCTIONS IMPLEMENTED IN TWO ON-BOARD GATE
ARRAYS. THE STORAGE CAPACITY OF THE CACHE IS 4K BYTES OF RAM, CALLED
DATA RAM'S. THE CACHE IS IMPLEMENTED AS A DIRECT MAPPED CACHE WITH
ADDRESS BITS 21 THROUGH 13 STORED IN A DIFFERENT SET OF RAM'S CALLED
TAG STORE.
THE KDJ11-B ALSO HAS TWO ON-BOARD ROM'S. ONE OF THEM, THE 16-BIT
ADDRESSABLE ROM, CONTAINS THE SELF-TEST AND THE BOOT CODES. THE OTHER
ROM, THE 8-BIT ADDRESSABLE ONE, CONTAINS THE BASE AREA WITH HARDWARE
SELECTION PARAMETERS, OPTIONAL BOOTSTRAPS, OPTIONAL UFD (USER FRIENDLY
DIAGNOSTIC) SYSTEM DESCTIPTION AREA, AND OPTIONAL FOREIGN LANGUAGE
TEXT.
THE SERIAL LINE UNIT IS IMPLEMENTED THRU A DLART CHIP WHICH PROVIDES
THE STANDARD CONSOLE INTERFACE TO THE CPU. IT HAS INTERNAL LOOP BACK
MODE AND PROVIDES WITH THREE CLOCK LINES: 800HZ, 60HZ, AND 50HZ.
THE LINE TIME CLOCK FUNCTIONS ARE IMPLEMENTED USING THOSE THREE LINES
AND THE BEVENT LINE. THE LINE CLOCK STATUS REGISTER IS IMPLEMENTED IN
ONE OF THE GATE ARRAYS.


                                2.   SYSTEM REQUIREMENTS

HARDWARE REQUIREMENTS

TO RUN SUCCESSFULLY THE DIAGNOSTIC NEEDS:

        1. KDJ11-B CPU MODULE
        2. CONSOLE TERMINAL
        3. AT LEAST 28K OF MEMORY

IN DVT,  AND STAGE ONE MANUFACTURING (MODULE ASSEMBLY) THE Q22 BUS
EXERCISER IS NEEDED TO CHECK Q22 BUS LOGIC.


                                3.  LOADING AND STARTING PROCEDURES

TO START UP THIS PROGRAM:

        1. BOOT xxDP.

        2. TYPE "R NAME", WHERE NAME IS THE NAME OF THE BIN OR BIC
           FILE FOR THIS PROGRAM.

THE STARTING ADDRESS OF THE PROGRAM IS 200.
NOTE: IF TRYING TO RESTART THE PROGRAM IN AN ARBITRARY PLACE AFTER
HALT ON BREAK THE FOLLOWING REGISTERS SHOULD BE SET UP:
177772-0        TO DISABLE MEMMORY MANAGEMENT
```

17777520=1000 TO CLEAR DIAGNOSTIC MODE (BIT 8), BUT STILL SAVE
HALT ON BREAK
17777746=400 TO FLUSH THE CACHE

4. SPECIAL ENVIRONMENTS

THE PROGRAM IS APT COMPATIBLE. IT CAN ALSO BE RUN UNDER THE UFD MONITOR.
IN THOSE CASES NONE OF THE STANDARD ERROR PRINTOUTS OCCUR. REFER
TO CORRESPONDING DOCUMENTS ON RUNNING PROCEDURES IN APT AND UNDER
UFD MONITOR.

5. PROGRAM OPTIONS

THE Q22 BUS EXERCISER IS UTILIZED IF IT IS PRESENT IN THE SYSTEM AND
THE DIAGNOSTIC IS NOT RUNNING IN UFD MODE.
STANDARD CAPABILITIES OF LOOPING ON TEST AND ON ERROR ARE PROVIDED.
IN ORDER TO RUN THE EXTENSIVE CACHE DATA RAM TEST BIT 7 HAS TO BE SET
IN THE SOFTWARE SWITCH REGISTER. TO RUN THE EXTENSIVE CACHE TAG RAM TEST
BIT 6 HAS TO BE SET IN THE SOFTWARE SWITCH REGISTER.

SWITCH REGISTER SELECTION:

BIT NUMBER	USE
15	HALT ON ERROR
14	LOOP ON PRESENT TEST
13	INHIBIT ERROR TYPEOUTS
11	INHIBIT ITERATIONS
10	BELL ON ERROR
9	LOOP ON ERROR
8	LOOP ON TEST IN SWR<5-0>
7	DO EXTENSIVE DATA RAM TEST
6	DO EXTENSIVE TAG RAM TEST

6. EXECUTION TIMES

WITHOUT EXTENSIVE RAM TESTS, THE DIAGNOSTIC RUNS IN UNDER 15 SECONDS.
WITH THEM, IT TAKES ABOUT 2 MINUTES.

7. ERROR INFORMATION

IN THE CASE OF ERRORS, A FAILING PC AND TEST NUMBERS ARE GIVEN.
WHERE IT IS POSSIBLE, EXPECTED AND RECEIVED DATA ARE GIVEN.
FOR AN EXAMPLE, SEE SECTION 8.

8. EXAMPLES

AFTER BOOTING XXDP+ AND STARTING THE PROGRAM, THE FOLLOWING
WILL APPEAR ON THE TERMINAL:

```
KDJ11-B CPU DIAGNOSTIC

SWR = XXXXXX    NEW =
```

WHERE XXXXXX CORRESPOND TO PRESENT SOFTWARE SWITCH REGISTER

```
SETTING.
AFTER "NEW" AN OPERATOR CAN DO ONE OF THE FOLLOWING:
1) TYPE IN A NEW SOFTWARE SWITCH REGISTER SETTING FOLLOWED
BY CARRIGE RETURN OR
2) JUST TYPE IN CARRIAGE RETURN IN WHICH CASE THE SOFTWARE
REGISTER WILL REMAIN UNCHANGED.

EXAMPLE OF ERROR PRINTOUT:
        ERROR IN TAG STORE
         TEST   ERROR   ADDRESS ADDRESS
          #      PC     <21-16> <15-0>
          27    105620  66600   000000
NOTE: THIS MAY NOT CORRESPOND TO THE ACTUAL PROGRAM COUNTER.

9. PROGRAM DESCRIPTION

9.1  J11 CODE
THIS PORTION OF THE CODE TESTS OUT THE J11 CHIP SET. IT IS BROKEN
INTO 3 PIECES: CPU TESTS, WHICH VERIFY DIFFERENT INSTRUCTIONS IN
DIFFERENT MODES AND DIFFERENT TRAP CONDITIONS; MMU TESTS, WHICH
VERIFY DIFFERENT FUNCTIONS OF MMU; AND FFP TESTS, WHICH DO
DIFFERENT FLOATING POINT INSTRUCTIONS.
THIS PORTION OF THE CODE HAVE BEEN WRITTEN IN CLOSE RELATIONSHIP
WITH THE J11 MICROCODE. THEREFORE, EVEN THOUGH NOT ALL POSSIBLE
INSTRUCTIONS IN ALL POSSIBLE ADDRESSING MODE HAVE BEEN TESTED, AN
ATTEMPT HAS BEEN MADE TO EXERCISE ALL OF THE MICROCODE.

9.2  CACHE CODE
THIS PORTION OF THE DIAGNOSTIC VERIFIES ALL CACHE FUNCTIONS.
DATA AND TAG RAM'S TESTS ARE ALSO INCLUDED.

*****************************************************************************
NOTE: IN ORDER TO RUN EXTENSIVE CACHE RAM'S TESTS THE CORRESPONDING
BITS IN THE SOFTWARE SWITCH REGISTER SHOULD BE SET. SEE SECTION 5.
*****************************************************************************

9.3  ON-BOARD ROM'S CODE
THESE TESTS VERIFY THE CHECKSUMS OF THE 16-BIT ROM AND THE BASE
AREA OF THE 8-BIT ROM.

9.4  LINE TIME CLOCKS CODE
THIS PART OF THE PROGRAM VERIFIES THE FUNCTIONS OF ALL 4 CLOCK
LINES: 3 OF THEM FROM THE SERIAL LINE CHIP (50HZ, 60HZ, 800HZ)
AND BEVENT LINE.
NOTE: IN UFD MODE ONLY FUNCTIONS CORRESPONDING TO BOOT ROM
SELECTION ARE CHECKED.

9.5  SERIAL LINE UNIT CODE
THESE TESTS VERIFY THE FUNCTIONALITY OF THE SLU CHIP UTILIZING THE
MAINTENANCE MODE OF THE CHIP.
```

```
237
238
239
240
241
242
243
```

9.6 Q22BE CODE
THESE TESTS VERIFY INTERRUPT ARBITRATION OF THE KDJ11-B, DMA
PROTOCOL, AND CACHE FUNCTIONS RELATED TO THE DMA ACTIVITIES,
INCLUDING PMG COUNTER.

&

```
        167400                          $SWR=167400
        000300                          $SWRMK=300
                                .TITLE  COKDABO KDJ11-B CLUSTER
                                ;*COPYRIGHT (C) DEC 83
                                ;*DIGITAL EQUIPMENT CORP.
                                ;*MAYNARD, MASS. 01754
                                ;*
                                ;*PROGRAM BY DIAG. ENG.
                                ;*
                                ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                                ;*PACKAGE (MAINDEC-11-DZQAC-C8), OCT, 1982.
                                ;*
        000001                  $TN=1
                                .SBTTL  OPERATIONAL SWITCH SETTINGS
                                ;*
                                ;*      SWITCH                  USE
                                ;*      ------          --------------------
                                ;*        15            HALT ON ERROR
                                ;*        14            LOOP ON TEST
                                ;*        13            INHIBIT ERROR TYPEOUTS
                                ;*        11            INHIBIT ITERATIONS
                                ;*        10            BELL ON ERROR
                                ;*         9            LOOP ON ERROR
                                ;*         8            LOOP ON TEST IN SWR<5:0>
                                ;*         7            DO EXTENSIVE DATA RAM TEST
                                ;*         6            DO EXTENSIVE TAG RAM TEST
                                .SBTTL  MEMORY MANAGEMENT DEFINITIONS

                                ;*KT11 VECTOR ADDRESS

        000250                  MMVEC=  250

                                ;*KT11 STATUS REGISTER ADDRESSES

        177572                  SR0=    177572
        177574                  SR1=    177574
        177576                  SR2=    177576
        172516                  SR3=    172516

                                ;*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
```

```
                        UDPDR4= 177630
                        UDPDR5= 177632
                        UDPDR6= 177634
                        UDPDR7= 177636

                        ;*USER "I" PAGE ADDRESS REGISTERS

                        UIPAR0= 177640
                        UIPAR1= 177642
                        UIPAR2= 177644
                        UIPAR3= 177646
                        UIPAR4= 177650
                        UIPAR5= 177652
                        UIPAR6= 177654
                        UIPAR7= 177656

                        ;*USER "D" PAGE ADDRESS REGISTERS

                        UDPAR0= 177660
                        UDPAR1= 177662
                        UDPAR2= 177664
                        UDPAR3= 177666
                        UDPAR4= 177670
                        UDPAR5= 177672
                        UDPAR6= 177674
                        UDPAR7= 177676

                        ;*SUPERVISOR "I" PAGE DESCRIPTOR REGISTERS

                        SIPDR0= 172200
                        SIPDR1= 172202
                        SIPDR2= 172204
                        SIPDR3= 172206
                        SIPDR4= 172210
                        SIPDR5= 172212
                        SIPDR6= 172214
                        SIPDR7= 172216

                        ;*SUPERVISOR "D" PAGE DESCRIPTOR REGISTERS

                        SDPDR0= 172220
                        SDPDR1= 172222
                        SDPDR2= 172224
                        SDPDR3= 172226
                        SDPDR4= 172230
                        SDPDR5= 172232
                        SDPDR6= 172234
                        SDPDR7= 172236

                        ;*SUPERVISOR "I" PAGE ADDRESS REGISTERS

                        SIPAR0= 172240
                        SIPAR1= 172242
                        SIPAR2= 172244
                        SIPAR3= 172246
                        SIPAR4= 172250
```

```
                172252          SIPAR5= 172252
                172254          SIPAR6= 172254
                172256          SIPAR7= 172256

                                ;*SUPERVISOR "D" PAGE ADDRESS REGISTERS

                172260          SDPAR0= 172260
                172262          SDPAR1= 172262
                172264          SDPAR2= 172264
                172266          SDPAR3= 172266
                172270          SDPAR4= 172270
                172272          SDPAR5= 172272
                172274          SDPAR6= 172274
                172276          SDPAR7= 172276

                                ;*KERNEL "I" PAGE DESCRIPTOR REGISTERS

                172300          KIPDR0= 172300
                172302          KIPDR1= 172302
                172304          KIPDR2= 172304
                172306          KIPDR3= 172306
                172310          KIPDR4= 172310
                172312          KIPDR5= 172312
                172314          KIPDR6= 172314
                172316          KIPDR7= 172316

                                ;*KERNEL "D" PAGE DESCRIPTOR REGISTERS

                172320          KDPDR0= 172320
                172322          KDPDR1= 172322
                172324          KDPDR2= 172324
                172326          KDPDR3= 172326
                172330          KDPDR4= 172330
                172332          KDPDR5= 172332
                172334          KDPDR6= 172334
                172336          KDPDR7= 172336

                                ;*KERNEL "I" PAGE ADDRESS REGISTERS

                172340          KIPAR0= 172340
                172342          KIPAR1= 172342
                172344          KIPAR2= 172344
                172346          KIPAR3= 172346
                172350          KIPAR4= 172350
                172352          KIPAR5= 172352
                172354          KIPAR6= 172354
                172356          KIPAR7= 172356

                                ;*KERNEL "D" PAGE ADDRESS REGISTERS

                172360          KDPAR0= 172360
                172362          KDPAR1= 172362
                172364          KDPAR2= 172364
                172366          KDPAR3= 172366
                172370          KDPAR4= 172370
                172372          KDPAR5= 172372
```

```
                        KDPAR6= 172374
                        KDPAR7= 172376

                        .SBTTL  BASIC DEFINITIONS

                        ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
000100                  STACK=  1100
104000                  ERROR=  EMT             ;;BASIC DEFINITION OF ERROR CALL
000004                  SCOPE=  IOT             ;;BASIC DEFINITION OF SCOPE CALL

                        ;*MISCELLANEOUS DEFINITIONS
000011                  HT=     11              ;;CODE FOR HORIZONTAL TAB
000012                  LF=     12              ;;CODE FOR LINE FEED
000015                  CR=     15              ;;CODE FOR CARRIAGE RETURN
000200                  CRLF=   200             ;;CODE FOR CARRIAGE RETURN-LINE FEED
177776                  PS=     177776          ;;PROCESSOR STATUS WORD
177776                  PSW=    PS
177774                  STKLMT= 177774          ;;STACK LIMIT REGISTER
177772                  PIRQ=   177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
177570                  DSWR=   177570          ;;HARDWARE SWITCH REGISTER
177570                  DDISP=  177570          ;;HARDWARE DISPLAY REGISTER

                        ;*GENERAL PURPOSE REGISTER DEFINITIONS
000000                  R0=     %0              ;;GENERAL REGISTER
000001                  R1=     %1              ;;GENERAL REGISTER
000002                  R2=     %2              ;;GENERAL REGISTER
000003                  R3=     %3              ;;GENERAL REGISTER
000004                  R4=     %4              ;;GENERAL REGISTER
000005                  R5=     %5              ;;GENERAL REGISTER
000006                  R6=     %6              ;;GENERAL REGISTER
000007                  R7=     %7              ;;GENERAL REGISTER
000006                  SP=     %6              ;;STACK POINTER
000007                  PC=     %7              ;;PROGRAM COUNTER

                        ;*PRIORITY LEVEL DEFINITIONS
000000                  PR0=    0               ;;PRIORITY LEVEL 0
000040                  PR1=    40              ;;PRIORITY LEVEL 1
000100                  PR2=    100             ;;PRIORITY LEVEL 2
000140                  PR3=    140             ;;PRIORITY LEVEL 3
000200                  PR4=    200             ;;PRIORITY LEVEL 4
000240                  PR5=    240             ;;PRIORITY LEVEL 5
000300                  PR6=    300             ;;PRIORITY LEVEL 6
000340                  PR7=    340             ;;PRIORITY LEVEL 7

                        ;*"SWITCH REGISTER" SWITCH DEFINITIONS
100000                  SW15=   100000
040000                  SW14=   40000
020000                  SW13=   20000
010000                  SW12=   10000
004000                  SW11=   4000
002000                  SW10=   2000
001000                  SW09=   1000
000400                  SW08=   400
000200                  SW07=   200
000100                  SW06=   100
000040                  SW05=   40
```

```
000020                  SW04=   20
000010                  SW03=   10
000004                  SW02=   4
000002                  SW01=   2
000001                  SW00=   1
001000                  SW9=    SW09
000400                  SW8=    SW08
000200                  SW7=    SW07
000100                  SW6=    SW06
000040                  SW5=    SW05
000020                  SW4=    SW04
000010                  SW3=    SW03
000004                  SW2=    SW02
000002                  SW1=    SW01
000001                  SW0=    SW00

                        ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
100000                  BIT15=  100000
040000                  BIT14=  40000
020000                  BIT13=  20000
010000                  BIT12=  10000
004000                  BIT11=  4000
002000                  BIT10=  2000
001000                  BIT09=  1000
000400                  BIT08=  400
000200                  BIT07=  200
000100                  BIT06=  100
000040                  BIT05=  40
000020                  BIT04=  20
000010                  BIT03=  10
000004                  BIT02=  4
000002                  BIT01=  2
000001                  BIT00=  1
001000                  BIT9=   BIT09
000400                  BIT8=   BIT08
000200                  BIT7=   BIT07
000100                  BIT6=   BIT06
000040                  BIT5=   BIT05
000020                  BIT4=   BIT04
000010                  BIT3=   BIT03
000004                  BIT2=   BIT02
000002                  BIT1=   BIT01
000001                  BIT0=   BIT00

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

```
                        000240                  PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
                        000001                          UFDSET=         1               ;FLAG FOR UFD
                                                .SBTTL  TRAP CATCHER

                        000000                          .=0
                                                ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
                                                ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
                                                ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
                        000174                          .=174
000174  000000                                  DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
000176  000000                                  SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER
                        000200                  .=200
000200  005037  001160                                  CLR     $TMP0
000204  000137  003764                                  JMP     @#START
                        000220                  .=220
000220  012737  000777  001160                          MOV     #777,$TMP0
000226  000137  003764                                  JMP     @#START

                                                .SBTTL  ACT11 HOOKS

                                                ;;**************************************************************
                                                ;HOOKS REQUIRED BY ACT11
                        000232                          $SVPC=.                 ;SAVE PC
                        000046                          .=46
000046  133610                                          $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
                        000052                          .=52
000052  000000                                          .WORD   0               ;;2)SET LOC.52 TO ZERO
                        000232                          .=$SVPC                 ;; RESTORE PC
                                                .SBTTL  APT PARAMETER BLOCK

                                                ;;**************************************************************
                                                ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                                ;;**************************************************************
                                                        .$X=.   ;;SAVE CURRENT LOCATION
                        000232                          .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
000024  000200                                         200     ;;FOR APT START UP
                        000044                          .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
000044  000232                                          $APTHDR ;;POINT TO APT HEADER BLOCK
                        000232                          .=.$X   ;;RESET LOCATION COUNTER
                                                ;;**************************************************************
                                                ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                                                ;INTERFACE SPEC.

000232                                          $APTHD:
000232  000000                                  $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
000234  001200                                  $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
000236  000000                                  $TSTM:  .WORD           ;;RUN TIM OF LONGEST TEST
000240  000000                                  $PASTM: .WORD           ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
000242  000000                                  $UNITM: .WORD           ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
000244  000052                                          .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

```
                                .SBTTL  COMMON TAGS

                                ;;*****************************************************************
                                ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
                                ;*USED IN THE PROGRAM.

                        001100          .=1100
001100                  $CMTAG:                         ;;START OF COMMON TAGS
001100  000000                  .WORD   0
001102  000             $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
001103  000             $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
001104  000000          $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
001106  000000          $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
001110  000000          $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
001112  000000          $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
001114  000             $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
001115  001             $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
001116  000000          $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001120  000000          $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
001122  000000          $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
001124  000000          $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
001126  000000          $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
001130  000000                  .WORD   0               ;;RESERVED--NOT TO BE USED
001132  000000                  .WORD   0
001134  000             $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
001135  000             $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
001136  000000                  .WORD   0
001140  177570          SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
001142  177570          DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
001144  177560          $TKS:   177560                  ;;TTY KBD STATUS
001146  177562          $TKB:   177562                  ;;TTY KBD BUFFER
001150  177564          $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
001152  177566          $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
001154  000             $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
001155  002             $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001156  012             $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001157  000             $TPFLG: .BYTE   0               ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
001160  000000          $TMP0:  .WORD   0               ;;USER DEFINED
001162  000000          $TMP1:  .WORD   0               ;;USER DEFINED
001164  000000          $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
001166  000000          $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
001170  177607  000377  $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
001174  077             $QUES:  .ASCII  /?/             ;;QUESTION MARK
001175  015             $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001176  000012          $LF:    .ASCIZ  <12>            ;;LINE FEED
                                ;;*****************************************************************
                                .SBTTL  APT MAILBOX ETABLE

                                ;;*****************************************************************
                                .EVEN
001200                  $MAIL:                          ;;APT MAILBOX
001200  000000          $MSGTY: .WORD   AMSGTY          ;;MESSAGE TYPE CODE
001202  000000          $FATAL: .WORD   AFATAL          ;;FATAL ERROR NUMBER
001204  000000          $TESTN: .WORD   ATESTN          ;;TEST NUMBER
001206  000000          $PASS:  .WORD   APASS           ;;PASS COUNT
001210  000000          $DEVCT: .WORD   ADEVCT          ;;DEVICE COUNT
```

```
                                ;APT MAILBOX-ETABLE
                                ;
                                ;
                                ;
```

```
686  001324                          $ETEND:
687
```

```
                                .SBTTL  ERROR POINTER TABLE

                                ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
                                ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
                                ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
                                ;*NOTE1:        IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
                                ;*NOTE2:        EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

                                ;*      EM              ;;POINTS TO THE ERROR MESSAGE
                                ;*      DH              ;;POINTS TO THE DATA HEADER
                                ;*      DT              ;;POINTS TO THE DATA
                                ;*      DF              ;;POINTS TO THE DATA FORMAT


001324                          $ERRTB:


                                .SBTTL  ERROR DEFINITIONS
                                ;ITEM 1
001324  122614                          EM1             ;CPU ERROR
001326  130360                          DH1             ;TEST #, ERROR PC
001330  131430                          DT1             ;$TMP1,$ERRPC
001332  000000                          0
                                ;ITEM 2
001334  122650                          EM2             ;MMU ERROR
001336  130360                          DH1             ;TEST #, ERROR PC
001340  131430                          DT1             ;$TMP1,$ERRPC
001342  000000                          0
                                ;ITEM 3
001344  122662                          EM3             ;FPP ERROR
001346  130360                          DH1             ;TEST #, ERROR PC
001350  131430                          DT1             ;$TMP1,$ERRPC
001352  000000                          0
                                ;ITEM 4
001354  122674                          EM4             ;ERROR IN READ-WRITE BITS OF CCR
001356  130405                          DH4             ;TEST #, PC, EXPECTED DATA, RECEIVED DATA
001360  131436                          DT4             ;$TMP1,$ERRPC,R1,CCR
001362  000000                          0
                                ;ITEM 5
001364  122734                          EM5             ;FORCE MISS WRITES TO CACHE
001366  130472                          DH5             ;TEST #, PC, HIT/MISS, DATA IN CACHE, DATA IN MEMORY
001370  131450                          DT5             ;$TMP1,$ERRPC, R2, R1, $GDDAT
001372  000000                          0
                                ;ITEM 6
001374  122767                          EM6             ;FORCE MISS WRITE INVALIDATES CACHE
001376  130472                          DH5             ;TEST #, PC, HIT/MISS, DATA IN CACHE, DATA IN MEMORY
001400  131450                          DT5             ;$TMP1,$ERRPC, R2, R1, $GDDAT
001402  000000                          0
                                ;ITEM 7
001404  123032                          EM7             ;UNEXPECTED PARITY INTERRUPT
001406  130571                          DH7             ;TEST #, PC, ADDRESS ACCESSED, MSER
001410  131464                          DT7             ;$TMP1,$ERRPC,$BDADR,MSER
001412  000000                          0
                                ;ITEM 10
001414  123066                          EM10            ;TAG PARITY ERROR
001416  130571                          DH7             ;TEST #, PC, ADDRESS ACCESSED, MSER
```

```
001420  131464                  DT7             ;$TMP1,$ERRPC,$BDADR,MSER
001422  000000                  0
                        ;ITEM 11
001424  123107                  EM11            ;DATA PARITY ERROR
001426  130571                  DH7             ;TEST #, PC, ADDRESS ACCESSED, MSER
001430  131464                  DT7             ;$TMP1,$ERRPC,$BDADR,MSER
001432  000000                  0
                        ;ITEM 12
001434  123131                  EM12            LOW BYTE PARITY ERROR
001436  130571                  DH7             ;TEST #, PC, ADDRESS ACCESSED, MSER
001440  131464                  DT7             ;$TMP1,$ERRPC,$BDADR,MSER
001442  000000                  0
                        ;ITEM 13
001444  123157                  EM13            ;HIGH BYTE PARITY ERROR
001446  130571                  DH7             ;TEST #, PC, ADDRESS ACCESSED, MSER
001450  131464                  DT7             ;$TMP1,$ERRPC,$BDADR,MSER
001452  000000                  0
                        ;ITEM 14
001454  123206                  EM14            ;ERROR DATA PATH
001456  130405                  DH4             ;TEST #, PC, EXPECTED DATA, RECEIVED DATA
001460  131476                  DT14            ;$TMP1,$ERRPC,$GDDAT,TSTLOC
001462  000000                  0
                        ;ITEM 15
001464  123231                  EM15            ;FORCE MISS READS FROM CACHE
001466  130472                  DH5             ;TEST #, PC, HIT/MISS, DATA IN CACHE, DATA IN MEMORY
001470  131450                  DT5             ;$TMP1,$ERRPC, R2, R1, $GDDAT
001472  000000                  0
                        ;ITEM 16
001474  123265                  EM16            ;FORCE MISS READS FROM CACHE AND MISS
001476  130472                  DH5             ;TEST #, PC, HIT/MISS, DATA IN CACHE, DATA IN MEMORY
001500  131450                  DT5             ;$TMP1,$ERRPC, R2, R1, $GDDAT
001502  000000                  0
                        ;ITEM 17
001504  123332                  EM17            ;ERROR IN RECORDING HITS IN HIT/MISS
001506  130405                  DH4             ;TEST #, PC, EXPECTED DATA, RECEIVED DATA
001510  131510                  DT17            ;$TMP1,$ERRPC,$GDDAT,RECDAT
001512  000000                  0
                        ;ITEM 20
001514  123376                  EM20            ;WRITE BYTE ALLOCATES CACHE
001516  130360                  DH1             ;TEST #, PC
001520  131430                  DT1             ;$TMP1,$ERRPC
001522  000000                  0
                        ;ITEM 21
001524  123431                  EM21            ;WRITE BYTE HIT DOES NOT RECORD HIT
001526  130360                  DH1             ;TEST #, PC
001530  131430                  DT1             ;$TMP1,$ERRPC
001532  000000                  0
                        ;ITEM 22
001534  123474                  EM22            ;BYTES REVERSED ON WRITE CYCLES
001536  130360                  DH1             ;TEST #, PC
001540  131430                  DT1             ;$TMP1,$ERRPC
001542  000000                  0
                        ;ITEM 23
001544  123533                  EM23            ;CONDITIONAL BYPASS DOESN'T INVALIDATE CACHE
001546  130472                  DH5             ;TEST #, PC, HIT/MISS, DATA IN CACHE, DATA IN MEMORY
001550  131450                  DT5             ;$TMP1,$ERRPC, R2, R1, $GDDAT
```

```
001552  000000                  0
                        ;ITEM 24
001554  123607                  EM24            ;HITS RECORDED AFTER FLUSHING CACHE
001556  130644                  DH24            ;TEST #, PC, NUMBER OF HITS
001560  131522                  DT24            ;$TMP1,$ERRPC,R3
001562  000000                  0
                        ;ITEM 25
001564  123652                  EM25            ;BYPASS DOESN'T INVALIDATE CACHE
001566  130360                  DH1             ;TEST #, PC
001570  131430                  DT1             ;$TMP1,$ERRPC
001572  000000                  0
                        ;ITEM 26
001574  123712                  EM26            ;MSER DOES NOT CLEAR ON WRITE REFERENCE
001576  130360                  DH1             ;TEST #, PC
001600  131430                  DT1             ;$TMP1,$ERRPC
001602  000000                  0
                        ;ITEM 27
001604  123761                  EM27            ;PARITY ERROR DON'T CAUSE A MISS
001606  130710                  DH27            ;TEST #, PC, MSER, HIT/MISS
001610  131532                  DT27            ;$TMP1,$ERRPC,MSER,R3,0
001612  000000                  0
                        ;ITEM 30
001614  124021                  EM30            ;PARITY ERROR DON'T SET MSER WITH CCR<7>=0
001616  130710                  DH27            ;TEST #, PC, MSER, HIT/MISS
001620  131532                  DT27            ;$TMP1,$ERRPC,MSER,R3,0
001622  000000                  0
                        ;ITEM 31
001624  124073                  EM31            ;PARITY ERROR IGNORED
001626  130360                  DH1             ;TEST #, PC
001630  131430                  DT1             ;$TMP1,$ERRPC
001632  000000                  0
                        ;ITEM 32
001634  124120                  EM32            ;PARITY ERROR IGNORED ON LOW BYTE
001636  130360                  DH1             ;TEST #, PC
001640  131430                  DT1             ;$TMP1,$ERRPC
001642  000000                  0
                        ;ITEM 33
001644  124161                  EM33            ;PARITY ERROR IGNORED ON HIGH BYTE
001646  130360                  DH1             ;TEST #, PC
001650  131430                  DT1             ;$TMP1,$ERRPC
001652  000000                  0
                        ;ITEM 34
001654  124223                  EM34            ;PARITY ABORT LOGIC DOESN'T WORK
001656  130360                  DH1             ;TEST #, PC
001660  131430                  DT1             ;$TMP1,$ERRPC
001662  000000                  0
                        ;ITEM 35
001664  124263                  EM35            ;MSER NOT SET PROPERLY
001666  130405                  DH4             ;TEST #, PC, EXPECTED DATA, RECEIVED DATA
001670  131544                  DT35            ;$TMP1,$ERRPC,$GDDAT,MSER,0
001672  000000                  0
                        ;ITEM 36
001674  124311                  EM36            ;PARITY INTERRUPT DOESN'T WORK
001676  130360                  DH1             ;TEST #, PC
001700  131430                  DT1             ;$TMP1,$ERRPC
001702  000000                  0
```

```
856
857  001704  124355                ;ITEM 37
858  001706  130360                        EM37            ;NXM AND PARITY ABORT DIN'T HAPPEN
859  001710  131430                        DH1             ;TEST #, PC
860  001712  000000                        DT1             ;$TMP1,$ERRPC
861                                         0
862  001714  124417                ;ITEM 40
863  001716  130360                        EM40            ;PARITY ABORT NOT BLOCKED BY NXM TRAP
864  001720  131430                        DH1             ;TEST #, PC
865  001722  000000                        DT1             ;$TMP1,$ERRPC
866                                         0
867  001724  124464                ;ITEM 41
868  001726  130753                        EM41            ;MULTI-PROCESSOR HOOK INSTRUCTION DOESN'T CAUSE MISS
869  001730  131556                        DH41            ;TEST #, PC, INSTRUCTION OPCODE
870  001732  000000                        DT41            ;$TMP1,$ERRPC,$BDDAT
871                                         0
872  001734  124550                ;ITEM 42
873  001736  130360                        EM42            ;ERROR IN PARITY LOGIC
874  001740  131430                        DH1             ;TEST #, PC
875  001742  000000                        DT1             ;$TMP1,$ERRPC
876                                         0
877  001744  124576                ;ITEM 43
878  001746  131024                        EM43            ;ERROR IN CACHE DATA RAMS
879  001750  131566                        DH43            ;TEST #, PC, EXPECTED DATA, RECEIVED DATA, CACHE LOCATION
880  001752  000000                        DT43            ;$TMP1,$ERRPC,R1,RECDAT,$BDADR
881                                         0
882  001754  124627                ;ITEM 44
883  001756  130360                        EM44            ;ERROR IN NXM IN STANDALONE MODE
884  001760  131430                        DH1             ;TEST #, PC
885  001762  000000                        DT1             ;$TMP1,$ERRPC
886                                         0
887  001764  124667                ;ITEM 45
888  001766  130360                        EM45            ;HITS NOT RECORDED PROPERLY THRU HIT/MISS
889  001770  131430                        DH1             ;TEST #, PC
890  001772  000000                        DT1             ;$TMP1,$ERRPC
891                                         0
892  001774  124751                ;ITEM 46
893  001776  131124                        EM46            ;HIT RECORDED FOR A LOCATION NOT IN CACHE
894  002000  131602                        DH47            ;TEST #, PC, LOCATION ACCESSED
895  002002  000000                        DT47            ;$TMP1,$ERRPC,KIPAR6,$BDADR
896                                         0
897  002004  125041                ;ITEM 47
898  002006  131124                        EM47            ;MISS RECORDED FOR A LOCATION THAT SHOULD BE IN CACHE
899  002010  131602                        DH47            ;TEST #, PC, LOCATION ACCESSED
900  002012  000000                        DT47            ;$TMP1,$ERRPC,KIPAR6,$BDADR
901                                         0
902  002014  125126                ;ITEM 50
903  002016  131124                        EM50            ;ERROR IN TAG STORE
904  002020  131614                        DH47            ;TEST #, PC, ADDRESS
905  002022  000000                        DT50            ;$TMP1,$ERRPC,R1,$BDADR
906                                         0
907  002024  125151                ;ITEM 51
908  002026  130405                        EM51            ;ERROR PCR READ-WRITE BITS
909  002030  131626                        DH4             ;TEST #, PC, EXPECTED DATA, RECEIVED DATA
910  002032  000000                        DT51            ;$TMP1,$ERRPC,$GDDAT,PCR
911                                         0
                                    ;ITEM 52
```

```
912  002034  125203                    EM52          ;ERROR IN BCSR READ-WRITE BITS
913  002036  130405                    DH4           ;TEST #, PC, EXPECTED DATA, RECEIVED DATA
914  002040  131640                    DT52          ;$TMP1,$ERROC,$GDDAT,BCSR
915  002042  000000                    0
916                              ;ITEM 53
917  002044  125241                    EM53          ;RESET DOESN'T CLEAR BCSR<4>
918  002046  130360                    DH1           ;TEST #, PC
919  002050  131430                    DT1           ;$TMP1,$ERRPC
920  002052  000000                    0
921                              ;ITEM 54
922  002054  125275                    EM54          ;CHECKSUM ERROR IN 16-BIT ROM
923  002056  130360                    DH1           ;TEST #, PC
924  002060  131430                    DT1           ;$TMP1,$ERRPC
925  002062  000000                    0
926                              ;ITEM 55
927  002064  125333                    EM55          ;CHCKSUM ERROR IN 8-BIT ROM
928  002066  130360                    DH1           ;TEST #, PC
929  002070  131430                    DT1           ;$TMP1,$ERRPC
930  002072  000000                    0
931                              ;ITEM 56
932  002074  125367                    EM56          ;TIMEOUT READING LKS
933  002076  130360                    DH1           ;TEST #, PC
934  002100  131430                    DT1           ;$TMP1,$ERRPC
935  002102  000000                    0
936                              ;ITEM 57
937  002104  125413                    EM57          ;LKS<07> DOES NOT BECOME 1
938  002106  130360                    DH1           ;TEST #, PC
939  002110  131430                    DT1           ;$TMP1,$ERRPC
940  002112  000000                    0
941                              ;ITEM 60
942  002114  125445                    EM60          ;WRITE REFERENCE DOESN'T CLEAR LKS<07>
943  002116  130360                    DH1           ;TEST #, PC
944  002120  131430                    DT1           ;$TMP1,$ERRPC
945  002122  000000                    0
946                              ;ITEM 61
947  002124  125513                    EM61          ;ILLEGAL LKS INTERRUPTS
948  002126  130360                    DH1           ;TEST #, PC
949  002130  131430                    DT1           ;$TMP1,$ERRPC
950  002132  000000                    0
951                              ;ITEM 62
952  002134  125542                    EM62          ;PROCESSOR INTERRUPTS DON'T CLEAR LKS<07>
953  002136  130360                    DH1           ;TEST #, PC
954  002140  131430                    DT1           ;$TMP1,$ERRPC
955  002142  000000                    0
956                              ;ITEM 63
957  002144  125613                    EM63          ;LKS READY DOESN'T GO LOW
958  002146  130360                    DH1           ;TEST #, PC
959  002150  131430                    DT1           ;$TMP1,$ERRPC
960  002152  000000                    0
961                              ;ITEM 64
962  002154  125644                    EM64          ;WRONG NUMBER OF LKS INTERRUPTS
963  002156  130360                    DH1           ;TEST #, PC
964  002160  131430                    DT1           ;$TMP1,$ERRPC
965  002162  000000                    0
966                              ;ITEM 65
967  002164  125703                    EM65          ;LKS INTERRUPTS HAPPEN AT WRONG PRIORITY
```

```
 968  002166  131211                    DM65            ;TEST #, PC, PRIORITY
 969  002170  131664                    DT65            ;$TMP1,$ERRPC,$GDDAT
 970  002172  000000                    0
 971                              ;ITEM 66
 972  002174  125753                    EM66            ;BCSR<12> DOES NOT DISABLES LKS
 973  002176  130360                    DH1             ;TEST #, PC
 974  002200  131430                    DT1             ;$TMP1,$ERRPC
 975  002202  000000                    0
 976                              ;ITEM 67
 977  002204  126012                    EM67            ;BCSR<13> DOESN'T SET LKS<06>
 978  002206  130360                    DH1             ;TEST #, PC
 979  002210  131430                    DT1             ;$TMP1,$ERRPC
 980  002212  000000                    0
 981                              ;ITEM 70
 982  002214  126047                    EM70            ;RESET DOESN'T SET LKS<7>
 983  002216  130360                    DH1             ;TEST #, PC
 984  002220  131430                    DT1             ;$TMP1,$ERRPC
 985  002222  000000                    0
 986                              ;ITEM 71
 987  002224  126100                    EM71            ;RESET DOESN'T CLEAR LKS<06>
 988  002226  130360                    DH1             ;TEST #, PC
 989  002230  131430                    DT1             ;$TMP1,$ERRPC
 990  002232  000000                    0
 991                              ;ITEM 72
 992  002234  126134                    EM72            ;TIMEOUT READING SLU REGISTERS
 993  002236  131255                    DH72            ;TEST #, PC, ADDRESS FAILED
 994  002240  131556                    DT41            ;$TMP1,$ERRPC,$BDDAT
 995  002242  000000                    0
 996                              ;ITEM 73
 997  002244  126172                    EM73            ;XMIT READY DIDN'T GO LOW
 998  002246  130360                    DH1             ;TEST #, PC
 999  002250  131430                    DT1             ;$TMP1,$ERRPC
1000  002252  000000                    0
1001                              ;ITEM 74
1002  002254  126216                    EM74            ;RCSR DOESN'T BECOME 1
1003  002256  130360                    DH1             ;TEST #, PC
1004  002260  131430                    DT1             ;$TMP1,$ERRPC
1005  002262  000000                    0
1006                              ;ITEM 75
1007  002264  126247                    EM75            ;WRONG CHARACTER RECEIVED
1008  002266  130405                    DH4             ;TEST #, PC, EXPECTED DATA, RECEIVED DATA
1009  002270  131674                    DT75            ;$TMP1,$ERRPC,$GDDAT,$BDDAT
1010  002272  000000                    0
1011                              ;ITEM 76
1012  002274  126300                    EM76            ;RCSR<07> NOT CLEARED AFTER READING RBUF
1013  002276  130360                    DH1             ;TEST #, PC
1014  002300  131430                    DT1             ;$TMP1,$ERRPC
1015  002302  000000                    0
1016                              ;ITEM 77
1017  002304  126350                    EM77            ;XCSR<07> NOT SET ON RESET
1018  002306  130360                    DH1             ;TEST #, PC
1019  002310  131430                    DT1             ;$TMP1,$ERRPC
1020  002312  000000                    0
1021                              ;ITEM 100
1022  002314  126402                    EM100           ;RCSR<07> NOT CLEARED ON RESET
1023  002316  130360                    DH1             ;TEST #, PC
```

```
002320  131430                  DT1             ;$TMP1,$ERRPC
002322  000000                  0
                        ;ITEM 101
002324  126440                  EM101           ;SLU INTERRUPTS HAPPEN AT 4
002326  130360                  DH1             ;TEST #, PC
002330  131430                  DT1             ;$TMP1,$ERRPC
002332  000000                  0
                        ;ITEM 102
002334  126473                  EM102           ;RESET DOES NOT CLEAR XCSR<6> AND RSCR<6>
002336  130360                  DH1             ;TEST #, PC
002340  131430                  DT1             ;$TMP1,$ERRPC
002342  000000                  0
                        ;ITEM 103
002344  126555                  EM103           ;TRANSMIT INTERRUPT DOES NOT CLEAR XCSR<07>
002346  130360                  DH1             ;TEST #, PC
002350  131430                  DT1             ;$TMP1,$ERRPC
002352  000000                  0
                        ;ITEM 104
002354  126630                  EM104           ;RECEIVE INTERRUPTS DON'T CLEAR RCSR<07>
002356  130360                  DH1             ;TEST #, PC
002360  131430                  DT1             ;$TMP1,$ERRPC
002362  000000                  0
                        ;ITEM 105
002364  126700                  EM105           ;BREAK CONDITION DOES NOT SET RBUF PROPERLY
002366  131321                  DH105           ;TEST #, PC, RBUF
002370  131706                  DT105           ;$TMP1,$ERRPC,RBUF
002372  000000                  0
                        ;ITEM 106
002374  126753                  EM106           ;RBUF WASN'T CLEARED ON NEXT CHAR.
002376  131321                  DH105           ;TEST #, PC, RBUF
002400  131706                  DT105           ;$TMP1,$ERRPC,RBUF
002402  000000                  0
                        ;ITEM 107
002404  127031                  EM107           ;ERROR IN WRITING TO XCSR<0>
002406  130360                  DH1             ;TEST #, PC
002410  131430                  DT1             ;$TMP1,$ERRPC
002412  000000                  0
                        ;ITEM 110
002414  127066                  EM110           ;RESET DOES NOT CLEAR XCSR<00>
002416  130360                  DH1             ;TEST #, PC
002420  131430                  DT1             ;$TMP1,$ERRPC
002422  000000                  0
                        ;ITEM 111
002424  127130                  EM111           ;FIRST CHARACTER WAS NOT OVERRUN BY THE SECOND
002426  130405                  DH4             ;TEST #, PC, EXPECTED DATA, RECEIVED DATA
002430  131674                  DT75            ;$TMP1,$ERRPC,$GDDAT,$BDDAT
002432  000000                  0
                        ;ITEM 112
002434  127206                  EM112           ;OVERRUN CONDITION DOES NOT SET PROPER BITS IN RBUF
002436  131321                  DH105           ;TEST #, PC, RBUF
002440  131706                  DT105           ;$TMP1,$ERRPC,RBUF
002442  000000                  0
                        ;ITEM 113
002444  127271                  EM113           ;RBUF WAS NOT CLEARED ON THE NEXT CHARACTER
002446  131321                  DH105           ;TEST #, PC, RBUF
002450  131706                  DT105           ;$TMP1,$ERRPC,RBUF
```

K2


```
002452  000000                  0
                        ;ITEM 114
002454  127355                  EM114           ;ERROR IN XCSR<2>
002456  130360                  DH1             ;TEST #, PC
002460  131430                  DT1             ;$TMP1,$ERRPC
002462  000000                  0
                        ;ITEM 115
002464  127376                  EM115           ;ERROR IN TAG STORE FROM STANDALONE MODE
002466  131353                  DH115           ;TEST #, PC, MSER, ADDRESS ACCESSED
002470  131716                  DT115           ;$TMP1,$ERRPC,$BDDAT,KIPAR6,$BDADR
002472  000000                  0
                        ;ITEM 116
002474  127446                  EM116           ;DMA TAG PARITY DOES NOT SET MSER<4>
002476  130360                  DH1             ;TEST #, PC
002500  131430                  DT1             ;$TMP1,$ERRPC
002502  000000                  0
                        ;ITEM 117
002504  127527                  EM117           ;MSER<13>NOT SET IN STANDALONE MODE
002506  130360                  DH1             ;TEST #, PC
002510  131430                  DT1             ;$TMP1,$ERRPC
002512  000000                  0
                        ;ITEM 120
002514  127572                  EM120           ;DMA WRITE HITS DON'T INVALIDATE CACHE
002516  130360                  DH1             ;TEST #, PC
002520  131430                  DT1             ;$TMP1,$ERRPC
002522  000000                  0
                        ;ITEM 121
002524  127640                  EM121           ;IN BLOCK MODE ON WRITE DMA HITS NOT EVERYTHING IS INVALIDATED
002526  130360                  DH1             ;TEST #, PC
002530  131430                  DT1             ;$TMP1,$ERRPC
002532  000000                  0
                        ;ITEM 122
002534  127736                  EM122           ;READ DMA HIT IS MESSED UP
002536  130360                  DH1             ;TEST #, PC
002540  131430                  DT1             ;$TMP1,$ERRPC
002542  000000                  0
                        ;ITEM 123
002544  127764                  EM123           ;ERROR IN DMA CYCLES FROM Q22BE
002546  130360                  DH1             ;TEST #, PC
002550  131430                  DT1             ;$TMP1,$ERRPC
002552  000000                  0
                        ;ITEM 124
002554  130016                  EM124           ;PIRQ INTERRUPTS DON'T TAKE PRIORITY OVER Q BUS INTERRUPTS
002556  130360                  DH1             ;TEST #, PC
002560  131430                  DT1             ;$TMP1,$ERRPC
002562  000000                  0
                        ;ITEM 125
002564  130110                  EM125           ;NO POWER DOWN TRAP TO 24 OCCUR
002566  130360                  DH1             ;TEST #, PC
002570  131430                  DT1             ;$TMP1,$ERRPC
002572  000000                  0
                        ;ITEM 126
002574  130147                  EM126           ;ERROR IN INTERRUPTS FROM Q22BE
002576  130360                  DH1             ;TEST #, PC
002600  131430                  DT1             ;$TMP1,$ERRPC
002602  000000                  0
```

```
1136
1137  002604  130204                    ;ITEM 127
1138  002606  130360                          EM127           ;ERROR IN PMG COUNTER
1139  002610  131430                          DH1             ;TEST #, PC
1140  002612  000000                          DT1             ;$TMP1,$ERRPC
1141                                          0
1142  002614  130246                    ;ITEM 130
1143  002616  130360                          EM130           ;UNEXPECTED TIMEOUT
1144  002620  131732                          DH1             ;TEST #, PC
1145  002622  000000                          DT130           ;$TMP1,$ERRPC
1146                                          0
1147  002624  130273                    ;ITEM 131
1148  002626  130360                          EM131           ;ERROR WRITING TO LKS<6>
1149  002630  131430                          DH1             ;TEST #, PC
1150  002632  000000                          DT1             ;$TMP1,$ERRPC
1151                                          0
1152  002634  130323                    ;ITEM 132
1153  002636  130360                          EM132           ;MAINTENANCE REGISTER ERROR
1154  002640  131430                          DH1             ;TEST #, PC
1155  002642  000000                          DT1             ;$TMP1,$ERRPC
1156                                          0
1157
1158                                    .SBTTL  GLOBAL VARIABLES AND REGISTER NAMES
1159
1160  002644  000000                    ;REGISTERS FOR THE FIRST Q22BE
1161  002646  000000                    CSR1:   .WORD   0               ;CONTROL REGISTER 1 FOR Q22BE
1162  002650  000000                    CSR2:   .WORD   0               ;CONTROL/STATUS REGISTER 2
1163  002652  000000                    BA:     .WORD   0               ;DMA ADDRESS FOR Q22BE
1164  002654  000000                    WC:     .WORD   0               ;WORD COUNT REGISTER
1165  002656  000000                    DATA:   .WORD   0               ;DMA DATA FOR Q22BE
1166  002660  000000                    VQBE1:  .WORD   0               ;ADDRESS OF VECTOR FOR Q22BE
1167  002662  000000                    VQPR1:  .WORD   0               ;PRIORITY
1168                                    SIMGOA: .WORD   0               ;SIMULTANEUOS GO ADDRESS REGISTER
1169
1170  002664  000000                    ;REGISTERS FOR THE SECOND Q22BE
1171  002666  000000                    CSR12:  .WORD   0               ;CONTROL REGISTER 1 FOR Q22BE
1172  002670  000000                    CSR22:  .WORD   0               ;CONTROL/STATUS REGISTER 2
1173  002672  000000                    BA2:    .WORD   0               ;DMA ADDRESS FOR Q22BE
1174  002674  000000                    WC2:    .WORD   0               ;WORD COUNT REGISTER
1175  002676  000000                    DATA2:  .WORD   0               ;DMA DATA FOR Q22BE
1176  002700  000000                    VQBE2:  .WORD   0               ;ADDRESS OF VECTOR FOR Q22BE
1177                                    VQPR2:  .WORD   0               ;PRIORITY
1178  002702  000000
1179  002704  000000                    LKSFL:  .WORD   0
1180  002706  000000                    ACTCHS: .WORD   0               ;ACTUAL CHECKSUM
1181  002710  000000                    SAVPCR: .WORD   0
1182          002710                    SAVBR:  .WORD   0
1183  002710  000000                    .=2710
1184  002712  000017                    TEMP:   .WORD   0
1185  002750  000000                            .BLKW   15.             ;RESERVED FOR BLOCK MODE TRANSFER
1186  002752  000032  000012  000006    TIMOUT: .WORD   0
1187  002760  000002                    Q22EN:  .WORD   32,12,6,2        ;PRIORITY 7-4 FOR Q22BE
1188
1189
1190          177524                            BCR=            177524  ;BOOT/DIAGNOSTICS CONFIGURATION
1191          177524                            BDR=            177524  ;BOOT/DIAGNOSTICS DISPLAY
```

M2


```
        177520                  BCSR=           177520          ;BOOT/DIAGNOSTICS STATUS
        177746                  CCR=            177746          ;CACHE CONTROL REGISTER
        177752                  HITMIS=         177752          ;HIT OR MISS REGISTER
        177734                  KMCR=           177734          ;UNIBUS CONFIGURATION REGISTER
        177546                  LKS=            177546          ;CLOCK STATUS REGISTER
        177750                  MAIREG=         177750          ;MAINTENANCE REGISTER
        177744                  MSER=           177744          ;MEMORY SYSTEM ERROR
        177522                  PCR=            177522          ;PAGE CONTROL REGISTER
        177772                  PIR=            177772          ;PROGRAM INTERRUPT REQUEST
        177562                  RBUF=           177562          ;RECEIVER DATA BUFFER
        177560                  RCSR=           177560          ;RECEIVER STATUS REGISTER
        177566                  XBUF=           177566          ;TRANSMITTER DATA BUFFER
        177564                  XCSR=           177564          ;TRANSMITTER STATUS REGISTER

        177766                  CPEREG=         177766          ;CPU ERROR REGISTER

        177572                  MMR0=SR0                        ;MEMORY MANAGEMENT REG. 0
        177574                  MMR1=SR1                        ;MEMORY MANAGEMENT REG. 1
        177576                  MMR2=SR2                        ;MEMORY MANAGEMENT REG. 2
        172516                  MMR3=SR3                        ;MEMORY MANAGEMENT REG. 3
        120001                  POLY= 120001
        000000                  NULL=           0

                        .SBTTL  GLOBAL DATA SECTION

                        ;++
                        ; THE GLOBAL DATA SECTION CONTAINS DATA THAT ARE USED
                        ; IN MORE THAN ONE TEST.
                        ;--

                        ;THESE LOCATIONS ARE USED IN MORE THAN ONE TEST TO STORE VECTOR DATA
                        ;WHEN THE TEST NEEDS TO HAVE AN ERROR CONDITION RESPOND DIFFERENTLY
                        ;FROM THE DEFAULT RESPONCE.
002762  000000          SLOC00: .WORD   0
002764  000000          SLOC01: .WORD   0

                        ;THESE LOCATIONS ARE USED IN MORE THAN ONE TEST TO STORE WORKING DATA.
002766  000000          LOWADD: .WORD   0               ;STORES LOW ADDRESS FOR RAM TESTS
002770  000000          GOODAD: .WORD   0               ;STORES GOOD ADDRESS FOR RAM TESTS
002772  000000          TSTADD: .WORD   0               ;ADDRESS STORE FOR RAM TESTS
002774  000000          NEWADD: .WORD   0               ;ADDRESS STORE FOR RAM TEST
002776  000000          FLAG:   .WORD   0               ;USED TO STORE "FLAG" CONDITIONS
003000  000000          SAVSUP: .WORD   0               ;USED TO STORE SUPERVISOR STACK VALUE
003002  000000          SAVUSE: .WORD   0               ;USED TO STORE USER STACK VALUE
003004  000000          SAVMR0: .WORD   0               ;USED TO STORE MMU STATUS REGISTER 0 DATA
003006  000000          SAVMR1: .WORD   0               ;USED TO STORE MMU STATUS REGISTER 1 DATA
003010  000000          SAVMR2: .WORD   0               ;USED TO STORE MMU STATUS REGISTER 2 DATA
003012  000004          FLOAT:  .BLKW   4               ;USED TO STORE VALUES FOR MMU TESTS
003022  000004          FLO:    .BLKW   4               ;USED TO STORE VALUES FOR MMU TESTS
003032  000000          SEQ:    .WORD   0               ;STORES SEQUENCE NUMBER FOR JUMP TESTS
003034  000000          SPS:    .WORD   0               ;STORES STACK POINTER FOR JUMP TESTS
003036  000000          SPSJ:   .WORD   0               ;STORES STACK POINTER FOR JUMP TESTS
003040  000004          BTEXP:  .BLKW   4               ;STORES EXPONENT DURING BIT TESTS
003050  000004          BTRES:  .BLKW   4               ;STORES RECIEVED DATA FOR BIT TESTS
003060  000000          COUNT:  .WORD   0               ;ERROR INDICATOR FOR FLOATING POINT TESTS
003062  000004          RECFEC: .BLKW   4               ;RECIEVED FLOATING POINT EXCEPTION CODE
```

```
1248  003072  000004                  RECST:  .BLKW   4               ;RECIEVED FLOATING POINT STATUS
1249  003102  000004                  RECDST: .BLKW   4               ;DESTINATION ADDRESS FOR FLOATING POINT TESTS
1250
1251                                  ;THESE LOCATIONS ARE USED BY MORE THAN ONE TEST AS LOOP COUNTERS
1252  003112  000000                  ALLCTR: .WORD   0
1253  003114  000000                  LOOPIN: .WORD   0
1254
1255                                  ;SOME MORE TEMPORARY STORAGE FOR RAM TESTS
1256  003116  000000                  SAVPOS: .WORD   0               ;STORES TEMPORARY BIT POSITIONS FOR RAM TESTS
1257  003120  000000                  MASK:   .WORD   0               ;STORES BIT MASK FOR ERROR ISOLATION
1258
1259                                  ;!!!!!THIS IS IT. THE PROGRAM TEST LOCATION!!!!!!!!!!!!!!!!!!
1260  003122  000000                  TSTLOC: .WORD   0
1261  003124  000024                          .BLKW   20.
1262                                        ;FPP REGISTER DEFINITIONS
1263          000000                        AC0= %0
1264          000001                        AC1= %1
1265          000002                        AC2= %2
1266          000003                        AC3= %3
1267          000004                        AC4= %4
1268          000005                        AC5= %5
1269          000006                        AC6= %6
1270          000007                        AC7= %7
1271
1272                                        ;FPP INTERRUPT VECTOR
1273
1274          000244                        FPVEC=244
1275
1276
1277          001000                        STBOT=  1000
1278
1279
1280  003174  123456                  TAB1:   .WORD   123456
1281  003176  000000                          .WORD   000000
1282  003200  000000                          .WORD   0
1283  003202  000001                          .WORD   1
1284  003204  055555                  TAB2:   .WORD   055555
1285  003206  177777                          .WORD   177777
1286  003210  145671                          .WORD   145671
1287  003212  100000                          .WORD   100000
1288  003214  003000                  TAB3:   .WORD   003000
1289  003216  123456                          .WORD   123456
1290  003220  000000                          .WORD   0
1291  003222  000000                          .WORD   0
1292  003224  055555                  TAB4:   .WORD   55555
1293  003226  177777                          .WORD   -1
1294  003230  000000                          .WORD   0
1295  003232  000000                          .WORD   0
1296  003234  043243                  TAB5:   .WORD   43243
1297  003236  000000                          .WORD   0
1298  003240  000000                          .WORD   0
1299  003242  000000                          .WORD   0
1300  003244  162400                  TAB5A:  .WORD   162400
1301  003246  000000                          .WORD   0
1302  003250  000000                          .WORD   0
1303  003252  000000                          .WORD   0
```

```
1304  003254  000000                    TAB6:   .WORD   0
1305  003256  000000                            .WORD   0
1306  003260  000000                            .WORD   0
1307  003262  000000                            .WORD   0
1308  003264  047050                    TAB6A:  .WORD   47050
1309  003266  010000                            .WORD   10000
1310  003270  000000                            .WORD   0
1311  003272  000000                            .WORD   0
1312  003274  000200                    TAB7:   .WORD   200
1313  003276  000000                            .WORD   0
1314  003300  000000                            .WORD   0
1315  003302  000000                            .WORD   0
1316  003304  000200                    TAB8:   .WORD   200
1317  003306  000000                            .WORD   0
1318  003310  000000                            .WORD   0
1319  003312  000001                            .WORD   1
1320  003314  000400  000000  000000    TAB9:   .WORD   400,0,0,0
1321  003322  000000
1322  003324  030000                    TAB10:  .WORD   30000
1323  003326  003000                            .WORD   3000
1324  003330  000000                            .WORD   0
1325  003332  000000                            .WORD   0
1326  003334  016400                    TAB11:  .WORD   16400
1327  003336  000000                            .WORD   0
1328  003340  000000                            .WORD   0
1329  003342  000000                            .WORD   0
1330  003344  030000  003000  000002    TAB11A: .WORD   30000,3000,2,0
1331  003352  000000
1332  003354  016100  000000  000000    TAB12:  .WORD   16100,0,0,1
1333  003362  000001
1334  003364  016200                    TAB13:  .WORD   16200
1335  003366  000000                            .WORD   0
1336  003370  000000                            .WORD   0
1337  003372  000001                            .WORD   1
1338  003374  030000  003000  000000    TAB13B: .WORD   30000,3000,0,140000
1339  003402  140000
1340  003404  030000                    TAB14:  .WORD   30000
1341  003406  000000                            .WORD   0
1342  003410  000000                            .WORD   0
1343  003412  000000                            .WORD   0
1344  003414  024700                    TAB15:  .WORD   24700
1345  003416  000000                            .WORD   0
1346  003420  000000                            .WORD   0
1347  003422  000000                            .WORD   0
1348  003424  025000                    TAB16:  .WORD   25000
1349  003426  175363                            .WORD   175363
1350  003430  123456                            .WORD   123456
1351  003432  123456                            .WORD   123456
1352  003434  030000                    TAB17:  .WORD   30000
1353  003436  007020                            .WORD   7020
1354  003440  000000  000000                    .WORD   0,0
1355  003444  023456                    TAB18:  .WORD   23456
1356  003446  000000                            .WORD   0
1357  003450  000000                            .WORD   0
1358  003452  000001                            .WORD   1
1359  003454  100200  000000  000000    TAB21:  .WORD   100200,0,0,0
```

```
1360  003462  000000
1361  003464  100400  000000  000000  TAB22:  .WORD   100400,0,0,0
1362  003472  000000
1363  003474  000200  000000  000000  TAB23:  .WORD   200,0,0,1
1364  003502  000001
1365  003504  062400  000000  000000  TAB24:  .WORD   62400,0,0,0
1366  003512  000000
1367  003514  001100  000000  000000  TAB25:  .WORD   1100,0,0,0
1368  003522  000000
1369  003524  100600  000000  000000  TAB26:  .WORD   100600,0,0,0
1370  003532  000000
1371  003534  001000  000000  000000  TAB27:  .WORD   1000,0,0,0
1372  003542  000000
1373  003544  000600  000000  000000  TAB28:  .WORD   600,0,0,0
1374  003552  000000
1375  003554  010100  000000  000000  TAB29:  .WORD   10100,0,0,0
1376  003562  000000
1377  003564  010100  000000  002000  TAB29A: .WORD   10100,0,2000,0
1378  003572  000000
1379                                  ;
1380  003574  000500  000000  000000  TAB30:  .WORD   500,0,0,0
1381  003602  000000
1382  003604  100400  000000  000000  TAB31:  .WORD   100400,0,0,0
1383  003612  000000
1384  003614  016000  000000  000000  TAB32:  .WORD   16000,0,0,0
1385  003622  000000
1386  003624  011600  000000  000000  TAB33:  .WORD   11600,0,0,0
1387  003632  000000
1388  003634  000640  000000  000000  TAB34:  .WORD   640,0,0,0
1389  003642  000000
1390  003644  077600  000000  000000  TAB40:  .WORD   77600,0,0,0
1391  003652  000000
1392  003654  100200  000000  000000  TAB41:  .WORD   100200,0,0,1
1393  003662  000001
1394  003664  000340  000000  000000  TAB42:  .WORD   340,0,0,0
1395  003672  000000
1396  003674  000077  177777  177777  TAB43:  .WORD   77,177777,177777,177776
1397  003702  177776
1398  003704  000577  177777  177777  TAB45:  .WORD   577,-1,-1,-1
1399  003712  177777
1400  003714  000577  177777  000000  TAB46:  .WORD   577,-1,0,0
1401  003722  000000
1402  003724  173737  124242  052525  TAB47:  .WORD   173737,124242,052525,12346
1403  003732  012346
1404  003734  000000  000000  052525  TAB47A: .WORD   0,0,052525,12346
1405  003742  012346
1406  003744  173737  124242  000000  TAB48:  .WORD   173737,124242,0,0
1407  003752  000000
1408  003754  000600  000000  000000  TAB49:  .WORD   600,0,0,0
1409  003762  000000
1410
1411  003764                          START:
1412                                  ;; LCP/ORION ROUTINE TO SAVE EMTULATOR AND PRIORITY
1413
1414  003764  005737  004052          EMTSAV: TST     SAV30                   ;; FIRST TIME THROUGH ?
1415  003770  001034                          BNE     VMKOR                   ;; BRANCH IF BEEN HERE ALREADY
```

```
1416  003772  032737  000040  000052          BIT     @#BIT5,@#52             ;; ARE WE IN UFD MODE ?
1417  004000  001430                          BEQ     VMKOR                   ;; LEAVE IF NOT
1418  004002  012737  177777  004056          MOV     #-1,UFDFLG              ;; SET UFD FLAG
1419  004010  032737  000100  000052          BIT     @#BIT6,@#52             ;; ARE WE IN QUIET MODE ?
1420  004016  001403                          BEQ     1$                      ;; BR IF NOT
1421  004020  012737  177777  004060          MOV     #-1,UQUIET              ;; SET QUIET MODE
1422  004026  104042                  1$:     EMT     42                      ;; GET ADDRESS OF XXDP DCA TABLE
1423  004030  005060  000042                  CLR     42(R0)                  ;; CLR XXDP+ "DRSERR"
1424  004034  013737  000030  004052          MOV     30,SAV30                ;; SAVE EMULATOR ADDRESS
1425  004042  013737  000032  004054          MOV     32,SAV32                ;; SAVE EMULATOR PRIORITY LEVEL
1426  004050  000404                          BR      VMKOR                   ;; GET AROUND TAG AREA
1427  004052  000000                  SAV30:  .WORD   0                       ;; PUT EMULATOR INFO HERE
1428  004054  000000                  SAV32:  .WORD   0                       ;; PUT PRIORITY LOCATION        HERE
1429  004056  000000                  UFDFLG: .WORD   0                       ;; USER FRIENDLY MODE FLAG
1430  004060  000000                  UQUIET: .WORD   0                       ;; UFD QUIET MODE FLAG
1431  004062                          VMKOR:
1432                                  ;*************************************************************************
1433
1434
1435  004062                          1$:
1436                                  .SBTTL  INITIALIZE THE COMMON TAGS
1437                                  ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
1438  004062  012706  001100                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
1439  004066  005026                          CLR     (R6)+           ;;CLEAR MEMORY LOCATION
1440  004070  022706  001140                  CMP     #SWR,R6 ;;DONE?
1441  004074  001374                          BNE     .-6             ;;LOOP BACK IF NO
1442  004076  012706  001100                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
1443                                  ;;INITIALIZE A FEW VECTORS
1444  004102  012737  133644  000020          MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
1445  004110  012737  000340  000022          MOV     #340,@#IOTVEC+2 ;;LEVEL 7
1446  004116  012737  134132  000030          MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
1447  004124  012737  000340  000032          MOV     #340,@#EMTVEC+2 ;;LEVEL 7
1448  004132  012737  136656  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
1449  004140  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
1450  004146  012737  136740  000024          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
1451  004154  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
1452  004162  013737  133556  133550          MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
1453  004170  005037  001164                  CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
1454  004174  005037  001166                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
1455  004200  112737  000001  001115          MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
1456  004206  012737  004206  001106          MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
1457  004214  012737  004214  001110          MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
1458                                  ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
1459                                  ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
1460  004222  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
1461  004226  012737  004262  000004          MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
1462  004234  012737  177570  001140          MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
1463  004242  012737  177570  001142          MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
1464  004250  022777  177777  174662          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
1465  004256  001012                          BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
1466                                                                  ;;AND  THE HARDWARE SWR IS NOT = -1
1467  004260  000403                          BR      65$             ;;BRANCH IF NO TIMEOUT
1468  004262  012716  004270          64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
1469  004266  000002                          RTI
1470  004270  012737  000176  001140  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
1471  004276  012737  000174  001142          MOV     #DISPREG,DISPLAY
```

```
1472  004304  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
1473
1474  004310  005037  001206                  CLR     $PASS           ;;CLEAR PASS COUNT
1475  004314  132737  000200  001221          BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
1476  004322  001403                          BEQ     67$             ;;YES,USE NON-APT SWITCH
1477  004324  012737  001222  001140          MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
1478  004332                          67$:
1479  004332  013737  004056  004060          MOV     UFDFLG,UQUIET   ;ABORT IN UFD ON ERROR
1480  004340  012737  133124  000004          MOV     #TOUT,@#ERRVEC  ;POINT TO TIMEOUT ROUTINE
1481  004346  012737  000340  000006          MOV     #340,@#ERRVEC+2 ;AT PRIORITY 7
1482  004354  012737  132374  000114          MOV     #RAMPAR,@#114   ;POINT PARITY ABORT
1483  004362  012737  000340  000116          MOV     #340,@#116      ;AT PRIORITY7
1484  004370  012737  133306  000250          MOV     #MMUTRP,@#250   ;POINT MMU TRAP VECTOR
1485  004376  012737  000340  000252          MOV     #340,@#252      ;
1486  004404  005037  177766                  CLR     @#177766        ;CLEAR CPU ERROR REGISTER
1487  004410  032737  000100  000052          BIT     #BIT06,@#52     ;IN UFD QUIET MODE ?
1488  004416  001061                          BNE     LOOP            ;IF SO, SKIP PRINTOUT
1489  004420  012701  004432                  MOV     #.+12,R1        ;STORE LOCATION POINTER
1490  004424  012711  177777                  MOV     #-1,(R1)        ;MAKE SURE TO PRINT NOT ONLY AT FIRST TIME
1491                                  .SBTTL  TYPE PROGRAM NAME
1492                                  ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
1493  004430  005227  177777                  INC     #-1             ;;FIRST TIME?
1494  004434  001044                          BNE     68$             ;;BRANCH IF NO
1495  004436  022737  133610  000042          CMP     #$ENDAD,@#42    ;;ACT-11?
1496  004444  001440                          BEQ     68$             ;;BRANCH IF YES
1497  004446  104401  004514                  TYPE    ,69$            ;;TYPE ASCIZ STRING
1498                                  .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
1499  004452  005737  000042                  TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
1500  004456  001012                          BNE     70$             ;;BRANCH IF YES
1501  004460  123727  001220  000001          CMPB    $ENV,#1         ;;ARE WE RUNNING UNDER APT?
1502  004466  001406                          BEQ     70$             ;;BRANCH IF YES
1503  004470  023727  001140  000176          CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
1504  004476  001005                          BNE     71$             ;;BRANCH IF NO
1505  004500  104406                          GTSWR                   ;;GET SOFT-SWR SETTINGS
1506  004502  000403                          BR      71$
1507  004504  112737  000001  001134  70$:    MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
1508  004512                          71$:
1509  004512  000415                          BR      68$             ;;GET OVER THE ASCIZ
1510                                  ;;69$:  .ASCIZ  <CRLF>#KDJ11-B CPU DIAGNOSTIC#<CRLF>
1511  004546                          68$:
1512  004546  032737  000200  000052          BIT     #BIT07,@#52     ;UFD MODE?
1513  004554  001002                          BNE     LOOP            ;IF YES, BRANCH
1514  004556  004737  132452                  JSR     PC,Q22SIZ       ;SIZE FOR Q22BE
1515  004562                          LOOP:
1516                                  .DSABLE AMA
1517                                  ;;*****************************************************************
1518  004562  000004                  TST1:   SCOPE
1519                                  .SBTTL  BASE INSTRUCTION SET TESTS
1520                                  ;*****************************************************************
1521                                  ;*****************************************************************
1522                                  ;                       BEGIN BASE INSTRUCTION SET TESTING
1523                                  ;*****************************************************************
1524                                  ;*****************************************************************
1525  004564                          FRSTST:
1526
1527                                  ;TEST BEQ BNE INSTRUCTIONS
```

F3


```
                                        ;;*****************************************************************
                                        ;THESE TWO INSTRUCTIONS ARE FUNDAMENTAL TO RECOGNIZING ERROR CONDITIONS
004564  000277                                  SCC
004566  000244                                  CLZ                     ;CC=0100 - Z BIT CLEARED
004570  001401                                  BEQ     1$              ;*TEST INSTR -TRY TO CAUSE A BEQ ERROR
004572  001001                                  BNE     2$              ;BRANCH IF GOOD
                                                                        ;THE Z FLAG DIDNT CLEAR OR BRANCH FAILED.
                                                                        ;FAILURE AT THIS LOCATION
                                                                        ;COULD MEAN A BUS PROBLEM, MICRO-CODE PROBLEM
                                                                        ;CONDITION CODE PROBLEM OR JUST ABOUT ANYTHING
                                                                        ;ELSE.
004574  104001                          1$:     ERROR   +1              ;CPU ERROR
004576  000257                          2$:     CCC                     ;
004600  000264                                  SEZ                     ;COND CODES = 0100 (ZERO)
004602  001001                                  BNE     3$              ;*TEST INSTR* TRY TO BRANCH ON ZERO FLAG
004604  001401                                  BEQ     4$              ;*TEST INSTR* BRANCH IF GOOD
                                                                        ;BRANCH FAILURE WITH Z BIT SET
004606  104001                          3$:     ERROR   +1              ;CPU ERROR
004610                                  4$:                             ;END OF TEST
004610                                  M2:

                                        ;       TEST BRANCH ON CARRY
                                        ;;*****************************************************************
                                        ;THIS IS A TEST TO SEE IF THE MODULE FORM ANTICIPATED IS FEASIBLE.
004610  000257                                  CCC                     ;CC=0000
004612  103001                                  BCC     2$              ;*TEST INSTR*
                                                                        ;BRANCH CARRY CLEAR FAILED
004614  104001                          1$:     ERROR   +1              ;CPU ERROR
004616  000261                          2$:     SEC                     ;CC=1111
004620  103401                                  BCS     4$              ;*TEST INSTR*
                                                                        ; BRANCH CARRY SET FAILED
004622  104001                          3$:     ERROR   +1              ;CPU ERROR
004624                                  4$:

004624                                  M3:

                                        ;       TEST DATA PATHS
004624  005000                                  CLR     R0
004626  001005                                  BNE     1$              ;TRY TO INSURE WE ARE TESTING
                                                                        ;THE DATA PATH AND NOT THE "CLR R0" INSTRUCTION
004630  005010                                  CLR     (R0)            ;FORCE LOCATION TO ZERO
004632  001003                                  BNE     1$              ;TRY TO INSURE 0=0
004634  005737  000000                          TST     @#0             ;AGAIN, TRY TO INSURE THAT 0=0
004640  001401                                  BEQ     2$              ;BRANCH IF GOOD
                                                                        ;LOCATION 0 NOT SETUP PROPERLY
004642  104001                          1$:     ERROR   +1              ;CPU ERROR
004644                                  2$:
004644                                  M4:

                                        ;       TEST DATA PATHS - ONES AND ZEROS
004644  012737  125252  000000                  MOV     #125252,@#0             ;0=125252
004652  022737  125252  000000                  CMP     #125252,@#0             ;SEE IF DATA MADE IT
004660  001401                                  BEQ     2$                      ;BRANCH IF IF DATA IS GOOD
                                                                                ;ERROR! EITHER THE BUS IS BAD,
                                                                                ;OR THE MOV OR COMPARE
                                                                                ;INSTRUCTIONS FAILED
```

```
1584  004662  104001                  1$:     ERROR   ;1              ;CPU ERROR
1585  004664                          2$:                                     ;END OF TEST
1586
1587                                  ;
1588  004664                          M5:
1589
1590                                  ;       TEST DATA PATHS - DATA 0'S AND 1'S
1591  004664  012737  052525  000000          MOV     #052525,@#0             ;SETUP DATA
1592  004672  023727  000000  052525          CMP     @#0,#052525             ; TEST FOR CORRECT DATA
1593  004700  001401                          BEQ     2$
1594  004702  104001                  1$:     ERROR   ;1              ;CPU ERROR
1595  004704                          2$:
1596                                  ;
1597  004704                          M6:
1598                                  ;       TEST DATA PATHS - 1'S
1599  004704  005037  000000                  CLR     @#0
1600  004710  005137  000000                  COM     @#0             ;SET UP MEMORY LOCATION 0 = 111111
1601  004714  023727  000000  177777          CMP     @#0,#177777     ; TEST DATA
1602  004722  001401                          BEQ     2$              ;BRANCH IF NO ERROR
1603  004724  104001                  1$:     ERROR   ;1              ;CPU ERROR
1604  004726                          2$:
1605
1606                                  ;
1607  004726                          GPROTS:
1608                                  ;       R0 BIT TESTS
1609  004726  012700  177777                  MOV     #177777,R0              ;R0=177777
1610  004732  020027  177777                  CMP     R0,#177777              ;DOES R0=177777
1611  004736  001401                          BEQ     1$                      ;YES GO ON
1612                                                                          ;NO GO TO ERROR
1613  004740  104001                          ERROR   ;1              ;CPU ERROR
1614  004742  005000                  1$:     CLR     R0                      ;R0=0
1615  004744  020027  000000                  CMP     R0,#0                   ;DOES R0=0
1616  004750  001401                          BEQ     2$                      ;YES GO ON
1617                                                                          ;NO GO TO ERROR
1618  004752  104001                          ERROR   ;1              ;CPU ERROR
1619  004754  012700  125252          2$:     MOV     #125252,R0              ;R0=125252
1620  004760  020027  125252                  CMP     R0,#125252              ;DOES R0=125252
1621  004764  001401                          BEQ     3$                      ;YES GO ON
1622                                                                          ;NO GO TO ERROR
1623  004766  104001                          ERROR   ;1              ;CPU ERROR
1624  004770  012700  052525          3$:     MOV     #52525,R0               ;R0=52525
1625  004774  020027  052525                  CMP     R0,#52525               ;DOES R0=52525
1626  005000  001401                          BEQ     4$                      ;YES GO ON
1627                                                                          ;NO GO TO ERROR
1628  005002  104001                          ERROR   ;1              ;CPU ERROR
1629  005004                          4$:
1630                                  ;
1631  005004                          GPR1TS:
1632                                  ;       R1 BIT TESTS
1633  005004  012701  177777                  MOV     #177777,R1              ;R1=177777
1634  005010  020127  177777                  CMP     R1,#177777              ;DOES R1=177777
1635  005014  001401                          BEQ     1$                      ;YES GO ON
1636                                                                          ;NO GO TO ERROR
1637  005016  104001                          ERROR   ;1              ;CPU ERROR
1638  005020  005001                  1$:     CLR     R1                      ;R1=0
1639  005022  020127  000000                  CMP     R1,#0                   ;DOES R1=0
```

```
1640  005026  001401                        BEQ     2$                      ;YES GO ON
1641                                                                          ;NO GO TO ERROR
1642  005030  104001                        ERROR   +1              ;CPU ERROR
1643  005032  012701  125252        2$:     MOV     #125252,R1              ;R1=125252
1644  005036  020127  125252                CMP     R1,#125252              ;DOES R1=125252
1645  005042  001401                        BEQ     3$                      ;YES GO ON
1646                                                                          ;NO GO TO ERROR
1647  005044  104001                        ERROR   +1              ;CPU ERROR
1648  005046  012701  052525        3$:     MOV     #52525,R1               ;R1=52525
1649  005052  020127  052525                CMP     R1,#52525               ;DOES R1=52525
1650  005056  001401                        BEQ     4$                      ;YES GO ON
1651                                                                          ;NO GO TO ERROR
1652  005060  104001                        ERROR   +1              ;CPU ERROR
1653  005062                        4$:
1654                                ;
1655  005062                        GPR2TS:
1656                                ;       R2 BIT TESTS
1657  005062  012702  177777                MOV     #177777,R2              ;R2=177777
1658  005066  020227  177777                CMP     R2,#177777              ;DOES R2=177777
1659  005072  001401                        BEQ     1$                      ;YES GO ON
1660                                                                          ;NO GO TO ERROR
1661  005074  104001                        ERROR   +1              ;CPU ERROR
1662  005076  005002        1$:     CLR     R2                      ;R2=0
1663  005100  020227  000000                CMP     R2,#0                   ;DOES R2=0
1664  005104  001401                        BEQ     2$                      ;YES GO ON
1665                                                                          ;NO GO TO ERROR
1666  005106  104001                        ERROR   +1              ;CPU ERROR
1667  005110  012702  125252        2$:     MOV     #125252,R2              ;R2=125252
1668  005114  020227  125252                CMP     R2,#125252              ;DOES R2=125252
1669  005120  001401                        BEQ     3$                      ;YES GO ON
1670                                                                          ;NO GO TO ERROR
1671  005122  104001                        ERROR   +1              ;CPU ERROR
1672  005124  012702  052525        3$:     MOV     #52525,R2               ;R2=52525
1673  005130  020227  052525                CMP     R2,#52525               ;DOES R2=52525
1674  005134  001401                        BEQ     4$                      ;YES GO ON
1675                                                                          ;NO GO TO ERROR
1676  005136  104001                        ERROR   +1              ;CPU ERROR
1677  005140                        4$:
1678                                ;
1679  005140                        GPR3TS:
1680                                ;       R3 BIT TESTS
1681  005140  012703  177777                MOV     #177777,R3              ;R3=177777
1682  005144  020327  177777                CMP     R3,#177777              ;DOES R3=177777
1683  005150  001401                        BEQ     1$                      ;YES GO ON
1684                                                                          ;NO GO TO ERROR
1685  005152  104001                        ERROR   +1              ;CPU ERROR
1686  005154  005003        1$:     CLR     R3                      ;R3=0
1687  005156  020327  000000                CMP     R3,#0                   ;DOES R3=0
1688  005162  001401                        BEQ     2$                      ;YES GO ON
1689                                                                          ;NO GO TO ERROR
1690  005164  104001                        ERROR   +1              ;CPU ERROR
1691  005166  012703  125252        2$:     MOV     #125252,R3              ;R3=125252
1692  005172  020327  125252                CMP     R3,#125252              ;DOES R3=125252
1693  005176  001401                        BEQ     3$                      ;YES GO ON
1694                                                                          ;NO GO TO ERROR
1695  005200  104001                        ERROR   +1              ;CPU ERROR
```

```
1696  005202  012703  052525          3$:     MOV     #52525,R3               ;R3=52525
1697  005206  020327  052525                  CMP     R3,#52525               ;DOES R3=52525
1698  005212  001401                         BEQ     4$                      ;YES GO ON
1699                                                                          ;NO GO TO ERROR
1700  005214  104001                         ERROR   .1              ;CPU ERROR
1701  005216                          4$:
1702                                  ;
1703  005216                          GPR4TS:
1704                                  ;       R4 BIT TESTS
1705  005216  012704  177777                  MOV     #177777,R4              ;R4=177777
1706  005222  020427  177777                  CMP     R4,#177777              ;DOES R4=177777
1707  005226  001401                         BEQ     1$                      ;YES GO ON
1708                                                                          ;NO GO TO ERROR
1709  005230  104001                         ERROR   .1              ;CPU ERROR
1710  005232  005004                  1$:     CLR     R4                      ;R4=0
1711  005234  020427  000000                  CMP     R4,#0                   ;DOES R4=0
1712  005240  001401                         BEQ     2$                      ;YES GO ON
1713                                                                          ;NO GO TO ERROR
1714  005242  104001                         ERROR   .1              ;CPU ERROR
1715  005244  012704  125252          2$:     MOV     #125252,R4              ;R4=125252
1716  005250  020427  125252                  CMP     R4,#125252              ;DOES R4=125252
1717  005254  001401                         BEQ     3$                      ;YES GO ON
1718                                                                          ;NO GO TO ERROR
1719  005256  104001                         ERROR   .1              ;CPU ERROR
1720  005260  012704  052525          3$:     MOV     #52525,R4               ;R4=52525
1721  005264  020427  052525                  CMP     R4,#52525               ;DOES R4=52525
1722  005270  001401                         BEQ     4$                      ;YES GO ON
1723                                                                          ;NO GO TO ERROR
1724  005272  104001                         ERROR   .1              ;CPU ERROR
1725  005274                          4$:
1726                                  ;
1727  005274                          GPR5TS:
1728                                  ;       R5 BIT TESTS
1729  005274  012705  177777                  MOV     #177777,R5              ;R5=177777
1730  005300  020527  177777                  CMP     R5,#177777              ;DOES R5=177777
1731  005304  001401                         BEQ     1$                      ;YES GO ON
1732                                                                          ;NO GO TO ERROR
1733  005306  104001                         ERROR   .1              ;CPU ERROR
1734  005310  005005                  1$:     CLR     R5                      ;R5=0
1735  005312  020527  000000                  CMP     R5,#0                   ;DOES R5=0
1736  005316  001401                         BEQ     2$                      ;YES GO ON
1737                                                                          ;NO GO TO ERROR
1738  005320  104001                         ERROR   .1              ;CPU ERROR
1739  005322  012705  125252          2$:     MOV     #125252,R5              ;R5=125252
1740  005326  020527  125252                  CMP     R5,#125252              ;DOES R5=125252
1741  005332  001401                         BEQ     3$                      ;YES GO ON
1742                                                                          ;NO GO TO ERROR
1743  005334  104001                         ERROR   .1              ;CPU ERROR
1744  005336  012705  052525          3$:     MOV     #52525,R5               ;R5=52525
1745  005342  020527  052525                  CMP     R5,#52525               ;DOES R5=52525
1746  005346  001401                         BEQ     4$                      ;YES GO ON
1747                                                                          ;NO GO TO ERROR
1748  005350  104001                         ERROR   .1              ;CPU ERROR
1749  005352                          4$:
1750                                  ;
1751  005352                          GPR6TS:
```

```
1752                                    ;       R6 BIT TESTS
1753  005352  012706  177777            ;       MOV     #177777,R6              ;R6=177777
1754  005356  020627  177777                    CMP     R6,#177777              ;DOES R6=177777
1755  005362  001401                            BEQ     1$                      ;YES GO ON
1756                                                                            ;NO GO TO ERROR
1757  005364  104001                            ERROR   .1              ;CPU ERROR
1758  005366  005006                    1$:     CLR     R6                      ;R6=0
1759  005370  020627  000000                    CMP     R6,#0                   ;DOES R6=0
1760  005374  001401                            BEQ     2$                      ;YES GO ON
1761                                                                            ;NO GO TO ERROR
1762  005376  104001                            ERROR   .1              ;CPU ERROR
1763  005400  012706  125252            2$:     MOV     #125252,R6              ;R6=125252
1764  005404  020627  125252                    CMP     R6,#125252              ;DOES R6=125252
1765  005410  001401                            BEQ     3$                      ;YES GO ON
1766                                                                            ;NO GO TO ERROR
1767  005412  104001                            ERROR   .1              ;CPU ERROR
1768  005414  012706  052525            3$:     MOV     #52525,R6               ;R6=52525
1769  005420  020627  052525                    CMP     R6,#52525               ;DOES R6=52525
1770  005424  001401                            BEQ     4$                      ;YES GO ON
1771                                                                            ;NO GO TO ERROR
1772  005426  104001                            ERROR   .1              ;CPU ERROR
1773  005430  012706  001000            4$:     MOV     #STBOT,R6               ;RESTORE SP
1774
1775                                    ;
1776  005434                            PSWBTS:
1777                                    ;       PSW LOW BYTE BIT TESTS
1778  005434  012737  000377  177776            MOV     #377,@#177776           ;PS=357 T BIT SHOULDN'T SET
1779  005442  022737  000357  177776            CMP     #357,@#177776           ;DOES PS=357
1780  005450  001401                            BEQ     1$                      ;YES GO ON
1781                                                                            ;NO GO TO ERROR
1782  005452  104001                            ERROR   .1              ;CPU ERROR
1783  005454  005037  177776            1$:     CLR     @#177776                ;PS=0
1784  005460  022737  000000  177776            CMP     #0,@#177776             ;DOES PS=0
1785  005466  001401                            BEQ     2$                      ;YES GO ON
1786                                                                            ;NO GO TO ERROR
1787  005470  104001                            ERROR   .1              ;CPU ERROR
1788  005472  012737  000105  177776    2$:     MOV     #105,@#177776           ;PS=105
1789  005500  022737  000105  177776            CMP     #105,@#177776           ;DOES PS=105
1790  005506  001401                            BEQ     3$                      ;YES GO ON
1791                                                                            ;NO GO TO ERROR
1792  005510  104001                            ERROR   .1              ;CPU ERROR
1793  005512  012737  000252  177776    3$:     MOV     #252,@#177776           ;PS=252
1794  005520  022737  000252  177776            CMP     #252,@#177776           ;DOES PS=252
1795  005526  001401                            BEQ     4$                      ;YES GO ON
1796                                                                            ;NO GO TO ERROR
1797  005530  104001                            ERROR   .1              ;CPU ERROR
1798  005532                            4$:
1799
1800  005532                            MSPO:
1801
1802                                    ;       TEST SINGLE OPERAND INSTRUCTIONS- MODE 0
1803                                    ;;**************************************************************
1804                                    ;THE INC, COM, CLR, AND DECREMENT INSTRUCTIONS ARE VERIFED.
1805  005532  005004                            CLR     R4                      ;INITIALIZE R4 WITH DATA
1806  005534  005104                            COM     R4                      ;
1807  005536  005004                            CLR     R4                      ;*TEST INSTRUCTION
```

```
1808  005540  001401                    BEQ     2$              ;BRANCH IF R4 CLEARED
1809  005542  104001          1$:       ERROR   +1              ;CPU ERROR
1810  005544  005104          2$:       COM     R4              ;*TEST COMPLIMENT INSTRUCTION
1811  005546  005204                    INC     R4              ;*TEST INCREMENT INSTRUCTION
1812  005550  001401                    BEQ     4$              ;BRANCH IF R4 =0
1813                                                            ;COMPLIMENT OR INCREMENT FAILED
1814  005552  104001          3$:       ERROR   +1              ;CPU ERROR
1815  005554                  4$:
1816
1817                          ;
1818  005554                  MSPB:
1819
1820                          ;       TEST SINGLE OPS - EVEN BYTE OF CLRB, DECB, AND COMB
1821  005554  005004                    CLR     R4
1822  005556  005104                    COM     R4              ;SETUP TEST REGISTER
1823  005560  105004                    CLRB    R4              ;*TEST CLEAR BYTE INSTRUCTION
1824  005562  001401                    BEQ     2$              ;BRANCH IF GOOD
1825                                                            ;CLEAR EVEN BYTE FAILED
1826  005564  104001          1$:       ERROR   +1              ;CPU ERROR
1827  005566  105304          2$:       DECB    R4              ;*TEST DECREMENT BYTE
1828  005570  100002                    BPL     3$              ;DECREMENT BYTE FAILED
1829  005572  105104                    COMB    R4              ;*TEST COMPLIMENT BYTE
1830  005574  001401                    BEQ     4$              ;BRANCH IF GOOD
1831                                                            ;COMPLIMENT OR DECREMENT FAILED TO WORK
1832  005576  104001          3$:       ERROR   +1              ;CPU ERROR
1833  005600                  4$:
1834
1835                          ;
1836  005600                  MSPC:
1837                          ;       TEST SINGLE OPS - MODE 1 CLRB, COMB, AND INCB
1838  005600  005004                    CLR     R4
1839  005602  005014                    CLR     (R4)
1840  005604  005114                    COM     (R4)            ;SETUP TEST DATA
1841  005606  005014                    CLR     (R4)            ;*TEST INSTRUCTION
1842  005610  001401                    BEQ     2$              ;BRANCH IF GOOD
1843                                                            ;MODE 1 FAILED
1844  005612  104001          1$:       ERROR   +1              ;CPU ERROR
1845  005614  005114          2$:       COM     (R4)            ;*TEST INSTRUCTION
1846  005616  001403                    BEQ     3$              ;(0)SHOULD = -1
1847  005620  100002                    BPL     3$              ;
1848  005622  005214                    INC     (R4)            ;*TEST INSTRUCTION
1849  005624  001401                    BEQ     4$              ;BRANCH IF GOOD
1850                                                            ;COM OR INC FAILED TO ALTER LOC 0 CORRECTLY
1851  005626  104001          3$:       ERROR   +1              ;CPU ERROR
1852  005630                  4$:
1853
1854                          ;
1855  005630                  MSPD:
1856
1857                          ;       TEST SINGLE OPS MODE1-EVEN BYTE-CLRB,COMB,INCB
1858  005630  005004                    CLR     R4
1859  005632  005014                    CLR     (R4)
1860  005634  005114                    COM     (R4)            ;SETUP TEST DATA
1861  005636  105014                    CLRB    (R4)            ;*TEST INSTRUCTION
1862  005640  105014                    CLRB    (R4)            ;*TEST INSTRUCTION
1863  005642  001401                    BEQ     2$              ;BRANCH IF GOOD
```

```
1864
1865  005644  104001                  1$:     ERROR   +1              ;CLEAR (0) EVEN BYTE FAILED
1866  005646  105214                  2$:     INCB    (R4)            ;CPU ERROR
1867  005650  100405                          BMI     3$              ;*TEST INSTRUCTION
1868  005652  001404                          BEQ     3$              ; TEST FLAGS
1869  005654  105114                          COMB    (R4)            ;*TEST INSTRUCTION
1870  005656  105214                          INCB    (R4)
1871  005660  105214                          INCB    (R4)
1872  005662  001401                          BEQ     4$              ;BRANCH IF GOOD
1873                                                                  ;COMB OR INCB FAILED
1874  005664  104001                  3$:     ERROR   +1              ;CPU ERROR
1875  005666                          4$:
1876
1877                                  ;
1878  005666                          MSPEO:
1879
1880                                  ;       TEST SINGLE OPS - ODD BYTE - CLRB, COMB, DECB
1881  005666  005004                          CLR     R4
1882  005670  005014                          CLR     (R4)
1883  005672  005114                          COM     (R4)            ;SETUP TEST DATA
1884  005674  005204                          INC     R4              ;POINT TO ODD BYTE
1885  005676  105014                          CLRB    (R4)            ;*TEST INSTRUCTION
1886  005700  001401                          BEQ     1$              ;BRANCH IF GOOD
1887                                                                  ;CLEAR ODD BYTE FAILED
1888  005702  104001                          ERROR   +1              ;CPU ERROR
1889  005704  005304                  1$:     DEC     R4              ;POINT TO EVEN BYTE
1890  005706  005214                          INC     (R4)            ;LOC 0=1 0
1891  005710  005204                          INC     R4              ;POINT TO ODD BYTE
1892  005712  105114                          COMB    (R4)            ;*TEST INSTRUCTION
1893  005714  105214                          INCB    (R4)            ;LOC 0=-1 0
1894  005716  100003                          BPL     2$              ;BRANCH IF ERROR
1895  005720  001402                          BEQ     2$              ;
1896  005722  105214                          INCB    (R4)            ;*TEST INSTRUCTION
1897  005724  001401                          BEQ     3$              ;BRANCH IF GOOD
1898                                                                  ;MODE 1, ODD BYTE FAILED
1899  005726  104001                  2$:     ERROR   +1              ;CPU ERROR
1900  005730                          3$:
1901
1902                                  ;
1903  005730                          MSPF:
1904                                  ;       TEST SINGLE OP - MODE 2 - CLR, COM, INC
1905  005730  005004                          CLR     R4
1906  005732  105104                          COMB    R4
1907  005734  005204                          INC     R4              ;R4=400
1908  005736  005014                          CLR     (R4)            ;400=0
1909  005740  005114                          COM     (R4)            ;400=-1
1910  005742  005024                          CLR     (R4)+           ;*TEST INSTRUCTION
1911  005744  001401                          BEQ     1$              ;BRANCH IF GOOD
1912                                                                  ;MODE 2 CLEAR FAILED
1913  005746  104001                          ERROR   +1              ;CPU ERROR
1914  005750  005304                  1$:     DEC     R4
1915  005752  005304                          DEC     R4              ;R4=400
1916  005754  005124                          COM     (R4)+           ;*TEST INSTRUCTION
1917  005756  100004                          BPL     2$              ;BRANCH IF FAILURE
1918  005760  005304                          DEC     R4
1919  005762  005304                          DEC     R4              ;R4=400
```

```
1920  005764  005224                  INC     (R4)+           ;*TEST INSTRUCTION
1921  005766  001401                  BEQ     3$              ;BRANCH IF GOOD
1922                                                          ;MODE 2 FAILURE
1923  005770  104001            2$:   ERROR   +1              ;CPU ERROR
1924  005772                    3$:
1925
1926                            ;
1927  005772                    MSPG:
1928
1929                            ;       TEST CLRB, COMB, DECB, MODE 2 - EVEN BYTE
1930  005772  005004                    CLR     R4
1931  005774  105104                    COMB    R4
1932  005776  005204                    INC     R4              ;R4=400
1933  006000  005014                    CLR     (R4)
1934  006002  005114                    COM     (R4)            ;400=-1
1935  006004  105024                    CLRB    (R4)+           ;*TEST INSTRUCTION
1936  006006  001401                    BEQ     1$              ;BRANCH IF GOOD
1937                                                          ;MODE 2 EVEN BYTE FAILED
1938  006010  104001                    ERROR   +1              ;CPU ERROR
1939  006012  005304            1$:     DEC     R4
1940  006014  105324                    DECB    (R4)+           ;*TEST INSTRUCTION
1941  006016  100003                    BPL     2$              ;BRANCH IF BAD
1942  006020  005304                    DEC     R4              ;POINT TO EVEN BYTE
1943  006022  105124                    COMB    (R4)+           ;*TEST INSTRUCTION
1944  006024  001401                    BEQ     3$              ;BRANCH IF GOOD
1945                                                          ;MODE 2, EVEN BYTE FAILED
1946  006026  104001            2$:     ERROR   +1              ;CPU ERROR
1947  006030                    3$:
1948
1949
1950  006030                    MSPH:
1951
1952                            ;       TEST CLRB, COMB, INCB MODE 2 - ODD BYTE
1953  006030  005004                    CLR     R4
1954  006032  105104                    COMB    R4
1955  006034  005204                    INC     R4              ;R4=400
1956  006036  005014                    CLR     (R4)
1957  006040  005114                    COM     (R4)            ;400=-1 -1
1958  006042  005214                    INC     (R4)            ;POINT TO ODD BYTE
1959  006044  105024                    CLRB    (R4)+           ;*TEST INSTRUCTION
1960  006046  001401                    BEQ     1$              ;BRANCH IF GOOD
1961                                                          ;MODE 2,ODD BYTE FAILED
1962  006050  104001                    ERROR   +1              ;CPU ERROR
1963  006052  005304            1$:     DEC     R4              ;POINT TO ODD BYTE
1964  006054  005304                    DEC     R4
1965  006056  105224                    INCB    (R4)+           ;400=1 0
1966  006060  105124                    COMB    (R4)+           ;*TEST INSTRUCTION
1967  006062  100003                    BPL     2$              ;BRANCH IF MODE 2 FAILED
1968  006064  005304                    DEC     R4              ;POINT TO ODD BYTE
1969  006066  105224                    INCB    (R4)+
1970  006070  001401                    BEQ     3$              ;BRANCH IF GOOD
1971                                                          ;MODE 2,ODD BYTE FAILED
1972  006072  104001            2$:     ERROR   +1              ;CPU ERROR
1973  006074                    3$:
1974
1975                            ;
```

```
1976  006074                        MSPI:
1977
1978                                ;       TEST CLR, COM, INC - MODE 3
1979  006074  005004                       CLR     R4              ;
1980  006076  005014                       CLR     (R4)            ;0=0
1981  006100  105114                       COMB    (R4)            ;
1982  006102  005214                       INC     (R4)            ;0=400
1983  006104  005034                       CLR     @(R4)+          ;*TEST INSTRUCTION
1984  006106  001401                       BEQ     1$              ;BRANCH IF GOOD
1985                                                                ;MODE 3 FAILED, 400 SHOULD=0
1986  006110  104001                       ERROR   +1              ;CPU ERROR
1987  006112  005304                1$:    DEC     R4
1988  006114  005304                       DEC     R4              ;R4=0
1989  006116  005134                       COM     @(R4)+          ;*TEST INSTRUCTION
1990  006120  100004                       BPL     2$              ;BRANCH IF BAD
1991  006122  005304                       DEC     R4
1992  006124  005304                       DEC     R4              ;REPOSITION POINTER
1993  006126  005234                       INC     @(R4)+          ;*TEST INSTRUCTION
1994  006130  001401                       BEQ     3$              ;BRANCH IF GOOD
1995                                                                ;MODE 3 FAILED
1996  006132  104001                2$:    ERROR   +1              ;CPU ERROR
1997  006134                        3$:
1998
1999                                ;
2000  006134                        MSPJ:
2001
2002                                ;       TEST CLRB, COMB, INCB - MODE 3, EVEN/ODD BYTE
2003  006134  005004                       CLR     R4              ;R4=0
2004  006136  005001                       CLR     R1
2005  006140  105101                       COMB    R1
2006  006142  005201                       INC     R1              ;R1=400
2007  006144  005011                       CLR     (R1)
2008  006146  005121                       COM     (R1)+           ;400=-1
2009  006150  005011                       CLR     (R1)            ;
2010  006152  105111                       COMB    (R1)            ;402=000 377
2011  006154  005014                       CLR     (R4)            ;
2012  006156  105114                       COMB    (R4)            ;
2013  006160  005214                       INC     (R4)            ;0=400
2014  006162  105034                       CLRB    @(R4)+          ;*TEST INSTRUCTION 400=377 000
2015  006164  001401                       BEQ     1$              ;BRANCH IF MODE 3 EVEN BYTE CLEARED
2016                                                                ; TEST INSTRUCTION FAILED
2017  006166  104001                       ERROR   +1              ;CPU ERROR
2018  006170  005304                1$:    DEC     R4              ;REPOSTION POINTER
2019  006172  005304                       DEC     R4
2020  006174  105134                       COMB    @(R4)+          ;*TEST INSTRUCTION
2021  006176  005304                       DEC     R4
2022  006200  005304                       DEC     R4              ;REPOSITION POINTER
2023  006202  105234                       INCB    @(R4)+          ;*TEST INSTRUCTION
2024  006204  001401                       BEQ     3$              ;BRANCH IF GOOD
2025                                                                ;MODE 3, EVEN BYTE FAILED
2026  006206  104001                2$:    ERROR   +1              ;CPU ERROR
2027  006210  005304                3$:    DEC     R4
2028  006212  005304                       DEC     R4
2029  006214  005214                       INC     (R4)            ;R4=401
2030  006216  105234                       INCB    @(R4)+          ;*TEST INSTRUCTION
2031  006220  001004                       BNE     4$              ;BRANCH IF 402 NEQ 0
```

```
2032  006222  005304                    DEC     R4
2033  006224  005304                    DEC     R4              ;R4=401
2034  006226  105034                    CLRB    @(R4)+          ;401=0
2035  006230  001401                    BEQ     5$              ;BRANCH IF GOOD
2036                                                            ;ODD BYTE FAILED
2037  006232  104001              4$:   ERROR   .1              ;CPU ERROR
2038  006234  005304              5$:   DEC     R4
2039  006236  005304                    DEC     R4              ;R4=401
2040  006240  105134                    COMB    @(R4)+          ;403=377
2041  006242  005304                    DEC     R4
2042  006244  005304                    DEC     R4
2043  006246  105234                    INCB    @(R4)+          ;*TEST INSTRUCTION
2044  006250  001401                    BEQ     7$              ;BRANCH IF GOOD
2045                                                            ;MODE3 ODD BYTE FAILED.
2046  006252  104001              6$:   ERROR   .1              ;CPU ERROR
2047  006254                      7$:
2048
2049                              ;
2050  006254                      MSPL:
2051
2052                              ;       TEST CLR, COM, DEC - MODE 4
2053  006254  005004                    CLR     R4
2054  006256  105104                    COMB    R4
2055  006260  005204                    INC     R4              ;R4=400
2056  006262  005014                    CLR     (R4)            ;
2057  006264  005124                    COM     (R4)+           ;400= 1
2058  006266  005014                    CLR     (R4)            ;
2059  006270  005224                    INC     (R4)+           ;402=1
2060  006272  005044                    CLR     -(R4)           ;*TEST INSTRUCTION
2061  006274  001401                    BEQ     1$              ;BRANCH IF GOOD
2062                                                            ;MODE 4 FAILED
2063  006276  104001                    ERROR   .1              ;CPU ERROR
2064  006300  005344              1$:   DEC     -(R4)           ;*TEST INSTRUCTION 400=2
2065  006302  005114                    COM     (R4)            ;400=1
2066  006304  001405                    BEQ     2$              ;BRANCH IF BAD
2067  006306  100404                    BMI     2$              ;BRANCH IF BAD
2068  006310  005204                    INC     R4
2069  006312  005204                    INC     R4              ;R4=400
2070  006314  005344                    DEC     -(R4)           ;*TEST INSTRUCTION
2071  006316  001401                    BEQ     3$              ;BRANCH IF GOOD
2072                                                            ;MODE 4 FAILED
2073  006320  104001              2$:   ERROR   .1              ;CPU ERROR
2074  006322                      3$:
2075
2076                              ;
2077  006322                      MSPM:
2078
2079                              ;       TEST COMB, INCB, CLRB - MODE 4, ODD BYTE
2080  006322  005004                    CLR     R4
2081  006324  105104                    COMB    R4              ;
2082  006326  005204                    INC     R4              ;R4=400
2083  006330  005044                    CLR     -(R4)           ;376=0
2084  006332  105114                    COMB    (R4)
2085  006334  005224                    INC     (R4)+           ;376=001 000
2086  006336  005014                    CLR     (R4)
2087  006340  005124                    COM     (R4)+           ;400=1
```

```
2088  006342  005204                  INC     R4              ;R4=403
2089  006344  105044                  CLRB    -(R4)           ; TEST INST. CLEAR ODD BYTE (401)
2090  006346  001401                  BEQ     2$              ;BRANCH IF GOOD
2091                                                          ;MODE 4 BYTE FAILED
2092  006350  104001            1$:   ERROR   .1              ;CPU ERROR
2093  006352  005204            2$:   INC     R4
2094  006354  005204                  INC     R4              ;R4=403
2095  006356  105144                  COMB    -(R4)           ; TEST INST. 401=377
2096  006360  005304                  DEC     R4
2097  006362  005304                  DEC     R4
2098  006364  105244                  INCB    -(R4)           ; TEST INST. 401=0
2099  006366  001401                  BEQ     4$              ;BRANCH IF GOOD
2100                                                          ;MODE 4 ODD BYTE FAILED
2101  006370  104001            3$:   ERROR   .1              ;CPU ERROR
2102  006372  105344            4$:   DECB    -(R4)           ;*TEST INST.
2103  006374  001401                  BEQ     6$              ;BRANCH IF GOOD
2104                                                          ;MODE 4 DECREMENT ODD BYTE FAILED
2105  006376  104001            5$:   ERROR   .1              ;CPU ERROR
2106  006400                     6$:
2107
2108                             ;
2109  006400                     MSPN:
2110
2111                             ;       TEST CLR, COM, INC - MODE 5
2112  006400  005004                    CLR     R4
2113  006402  005014                    CLR     (R4)
2114  006404  105114                    COMB    (R4)
2115  006406  005224                    INC     (R4)+           ;0=400
2116  006410  005054                    CLR     @-(R4)          ;*TEST INST. 400=0
2117  006412  001401                    BEQ     1$              ;BRANCH IF GOOD
2118                                                          ;MODE 5 FAILED
2119  006414  104001                    ERROR   .1              ;CPU ERROR
2120  006416  005204            1$:     INC     R4
2121  006420  005204                    INC     R4              ;RESET POINTER TO 0
2122  006422  005154                    COM     @-(R4)          ;*TEST INST. 376=1
2123  006424  001407                    BEQ     2$              ;BRANCH IF BAD
2124  006426  005204                    INC     R4
2125  006430  005204                    INC     R4              ;REPOSITION POINTER
2126  006432  005354                    DEC     @-(R4)          ;*TEST INST. 376=0
2127  006434  001403                    BEQ     2$              ;BRANCH IF BAD
2128  006436  005224                    INC     (R4)+           ;0=401 R4=2
2129  006440  105254                    INCB    @-(R4)          ;*TEST INST.,400= 0 376
2130  006442  001401                    BEQ     3$              ;BRANCH IF GOOD
2131                                                          ;MODE 5 FAILED
2132  006444  104001            2$:     ERROR   .1              ;CPU ERROR
2133  006446                     3$:
2134
2135                             ;
2136  006446                     MSPO:
2137
2138                             ;       TEST NEG MODE 5
2139  006446  005004                    CLR     R4
2140  006450  105104                    COMB    R4
2141  006452  005204                    INC     R4              ;R4=400
2142  006454  005001                    CLR     R1
2143  006456  105101                    COMB    R1
```

```
2144  006460  005301                    DEC     R1              ;R1=376
2145  006462  005002                    CLR     R2              ;R2=0
2146  006464  005012                    CLR     (R2)            ;0=0
2147  006466  005014                    CLR     (R4)            ;
2148  006470  005114                    COM     (R4)            ;400=-1
2149  006472  005011                    CLR     (R1)            ;376=0
2150  006474  005454                    NEG     @-(R4)          ;0=0
2151  006476  001401                    BEQ     2$              ;BRANCH IF GOOD
2152                                                            ;NEG FAILED
2153  006500  104001              1$:   ERROR   .1              ;CPU ERROR
2154  006502  005334              2$:   DEC     @(R4).          ;0=-1
2155  006504  005454                    NEG     @-(R4)          ;0=1
2156  006506  001403                    BEQ     3$              ;BRANCH IF BAD
2157  006510  102402                    BVS     3$              ;BRANCH IF BAD
2158  006512  100401                    BMI     3$              ;BRANCH IF BAD
2159  006514  103401                    BCS     4$              ;BRANCH IF GOOD
2160                                                            ;NEG FAILED
2161  006516  104001              3$:   ERROR   .1              ;CPU ERROR
2162  006520  005334              4$:   DEC     @(R4).          ; TEST RESULT OF NEGATE
2163  006522  001401                    BEQ     6$              ;BRANCH IF GOOD
2164                                                            ;RESULT OF NEGATE BAD
2165  006524  104001              5$:   ERROR   .1              ;CPU ERROR
2166  006526  105212              6$:   INCB    (R2)            ;0=1
2167  006530  005454                    NEG     @-(R4)          ;0=-1
2168  006532  001403                    BEQ     7$              ;
2169  006534  102402                    BVS     7$
2170  006536  103001                    BCC     7$              ;
2171  006540  100401                    BMI     8$              ;BRANCH IF GOOD
2172                                                            ;BAD NEGATE
2173  006542  104001              7$:   ERROR   .1              ;CPU ERROR
2174  006544  105212              8$:   INCB    (R2)            ;0=0
2175  006546  001401                    BEQ     10$             ;BRANCH IF GOOD
2176  006550  104001              9$:   ERROR   .1              ;CPU ERROR
2177  006552                    10$:
2178
2179                              ;
2180
2181  006552                    MSPP:
2182
2183                              ;       TEST CLR, COM, INC - MODE 6
2184  006552  005004                    CLR     R4
2185  006554  005204                    INC     R4
2186  006556  005204                    INC     R4              ;R4=2
2187  006560  005001                    CLR     R1
2188  006562  105101                    COMB    R1
2189  006564  005201                    INC     R1              ;R1=400
2190  006566  005011                    CLR     (R1)            ;
2191  006570  005121                    COM     (R1).           ;400=-1
2192  006572  005011                    CLR     (R1)            ;R1=402
2193  006574  005211                    INC     (R1)            ;402=1
2194  006576  005002                    CLR     R2              ;R2=0
2195  006600  005012                    CLR     (R2)            ;0=0
2196  006602  005064  000376            CLR     376(R4)         ;400=0
2197  006606  001401                    BEQ     2$              ;BRANCH IF GOOD
2198  006610  104001              1$:   ERROR   .1              ;CPU ERROR
2199  006612  005364  000376      2$:   DEC     376(R4)         ;400=-1
```

```
2200  006616  005164  000400            COM     400(R4)         ;402=-1
2201  006622  001405                    BEQ     3$              ;BRANCH IF BAD
2202  006624  005264  000400            INC     400(R4)         ;402=-2
2203  006630  005264  000400            INC     400(R4)
2204  006634  001401                    BEQ     4$              ;BRANCH IF GOOD
2205                                                            ;MODE 6 FAILED
2206  006636  104001              3$:   ERROR   .1              ;CPU ERROR
2207  006640  005261  177776      4$:   INC     -2(R1)          ;400=0
2208  006644  001401                    BEQ     6$              ;BRANCH IF GOOD
2209                                                            ;INC MODE 6 FAILED
2210  006646  104001              5$:   ERROR   .1              ;CPU ERROR
2211  006650                      6$:
2212
2213                              ;
2214                              ;
2215                              ;
2216  006650                      MSPQ:
2217
2218                              ;       TEST NEG MODE 6
2219  006650  005001                      CLR     R1              ;R1=0
2220  006652  005004                      CLR     R4
2221  006654  105104                      COMB    R4
2222  006656  005204                      INC     R4              ;R4=400
2223  006660  005014                      CLR     (R4)            ;
2224  006662  005114                      COM     (R4)            ;400=-1
2225  006664  005044                      CLR     -(R4)           ;376=0
2226  006666  005044                      CLR     -(R4)           ;
2227  006670  005224                      INC     (R4)+           ;374=1 R4=376
2228  006672  005464  000002              NEG     2(R4)           ;400=1
2229  006676  001403                      BEQ     1$              ;NEGATE FAILED
2230  006700  102402                      BVS     1$
2231  006702  100401                      BMI     1$
2232  006704  103401                      BCS     2$              ;BRANCH IF GOOD
2233                                                            ;NEGATE FAILED
2234  006706  104001              1$:   ERROR   .1              ;CPU ERROR
2235  006710  005364  000002      2$:   DEC     2(R4)           ; TEST RESULT OF NEGATE
2236  006714  001401                    BEQ     4$              ;BRANCH IF GOOD
2237                                                            ;RESULT OF NEGATE FAILED
2238  006716  104001              3$:   ERROR   .1              ;CPU ERROR
2239  006720  005464  000000      4$:   NEG     0(R4)           ;+0=0
2240  006724  001401                    BEQ     5$              ; BRANCH IF GOOD
2241                                                            ;NEGATE FAILED
2242  006726  104001                      ERROR   .1              ;CPU ERROR
2243  006730  105461  000374      5$:   NEGB    374(R1)         ;374=0 377
2244  006734  102403                      BVS     6$              ;
2245  006736  001402                      BEQ     6$
2246  006740  100001                      BPL     6$
2247  006742  103401                      BCS     7$              ;BRANCH IF GOOD
2248                                                            ;NEGATE FAILED
2249  006744  104001              6$:   ERROR   .1              ;CPU ERROR
2250  006746  105261  000374      7$:   INCB    374(R1)         ;374=0
2251  006752  001401                    BEQ     9$              ;BRANCH IF GOOD
2252                                                            ;NEGATE FAILED
2253  006754  104001              8$:   ERROR   .1              ;CPU ERROR
2254  006756                      9$:
2255
```

```
2256
2257  006756                          MSPR:
2258
2259                                  ;       TEST CLR, COM, INC - MODE 7
2260  006756  005001                          CLR     R1              ;R1=0
2261  006760  005004                          CLR     R4              ;
2262  006762  105104                          COMB    R4              ;
2263  006764  005204                          INC     R4              ;R4=400
2264  006766  005011                          CLR     (R1)
2265  006770  105111                          COMB    (R1)
2266  006772  005211                          INC     (R1)
2267  006774  005211                          INC     (R1)
2268  006776  005211                          INC     (R1)            ;;0=402
2269  007000  005014                          CLR     (R4)            ;400=0
2270  007002  005064  000002                  CLR     2(R4)           ;
2271  007006  005164  000002                  COM     2(R4)           ;402=-1
2272  007012  005074  177400                  CLR     @-400(R4)       ;402=0
2273  007016  001401                          BEQ     2$              ;BRANCH IF GOOD
2274  007020  104001                  1$:     ERROR   +1              ;CPU ERROR
2275                                                                  ;INSTRUCTION FAILED
2276  007022  005171  000000          2$:     COM     @0(R1)          ;402=-1
2277  007026  100401                          BMI     4$              ;BRANCH IF GOOD
2278  007030  104001                  3$:     ERROR   +1              ;CPU ERROR
2279
2280  007032  005104                  4$:     COM     R4
2281  007034  005274  000401                  INC     @401(R4)                ;402=0
2282  007040  001401                          BEQ     6$              ;BRANCH IF GOOD
2283  007042  104001                  5$:     ERROR   +1              ;CPU ERROR
2284                                                                  ;MODE 7 FAILED
2285  007044                          6$:
2286
2287                                  ;
2288  007044                          MSPS:
2289
2290                                  ;       TEST NEG MODE 7
2291  007044  005004                          CLR     R4
2292  007046  005014                          CLR     (R4)            ;0=0
2293  007050  005002                          CLR     R2              ;
2294  007052  105102                          COMB    R2
2295  007054  005202                          INC     R2              ;R2=400
2296  007056  005012                          CLR     (R2)            ;400=0
2297  007060  005472  177400                  NEG     @-400(R2)               ;NEG OF 0=0
2298  007064  103401                          BCS     1$              ;****
2299  007066  001401                          BEQ     2$              ;BRANCH IF GOOD
2300  007070  104001                  1$:     ERROR   +1              ;CPU ERROR
2301                                                                  ;
2302  007072  005314                  2$:     DEC     (R4)            ;0=-1
2303  007074  005474  000400                  NEG     @+400(R4)               ;0=1
2304  007100  001403                          BEQ     3$              ;BRANCH IF ERROR
2305  007102  102402                          BVS     3$              ;
2306  007104  100401                          BMI     3$              ;
2307  007106  103401                          BCS     4$              ;BRANCH IF GOOD
2308  007110  104001                  3$:     ERROR   +1              ;CPU ERROR
2309                                                                  ;NEGATE MODE 7 FAILED
2310  007112                          4$:
2311
```

```
2312                                    ;
2313  007112                            MSPT:
2314
2315                                    ;       TEST SINGLE OPERAND MODE 2 REG 7
2316  007112  005004                           CLR     R4
2317  007114  105104                           COMB    R4
2318  007116  005204                           INC     R4              ;R4=400
2319  007120  005027                           CLR     (R7)+           ;CLEAR NEXT LOCATION
2320  007122  177777                   1$:     .WORD   -1              ;SETUP INITIAL DATA
2321  007124  001401                           BEQ     3$              ;BRANCH IF GOOD
2322  007126  104001                   2$:     ERROR   +1              ;CPU ERROR
2323  007130                            3$:
2324
2325                                    ;
2326
2327  007130                            MSPU:
2328
2329                                    ;       TEST TST MODE 0
2330  007130  005004                           CLR     R4              ;R4=0
2331  007132  000277                           SCC                     ;CONDITION CODES =1111
2332  007134  000244                           CLZ                     ;CC=1011
2333  007136  005704                           TST     R4              ;*TEST INSTRUCTION
2334  007140  103403                           BCS     1$
2335  007142  102402                           BVS     1$
2336  007144  100401                           BMI     1$              ;BRANCH IF ERROR
2337  007146  001401                           BEQ     2$              ;BRANCH IF GOOD
2338  007150  104001                   1$:     ERROR   +1              ;CPU ERROR
2339                                                                   ;TST MODE 0 FAILED
2340  007152  005304                   2$:     DEC     R4              ;R4=-1
2341  007154  000277                           SCC
2342  007156  000250                           CLN                     ;CC=0111
2343  007160  005704                           TST     R4              ;*TEST INSTRUCTION MODE 0
2344  007162  103403                           BCS     3$              ;BRANCH IF ERROR
2345  007164  102402                           BVS     3$
2346  007166  001401                           BEQ     3$
2347  007170  100401                           BMI     4$              ;BRANCH IF GOOD
2348  007172  104001                   3$:     ERROR   +1              ;CPU ERROR
2349                                                                   ;TST FAILED
2350  007174                            4$:
2351
2352                                    ;
2353  007174                            MSPV0:
2354
2355                                    ;       TEST TST MODE 0 BYTE
2356  007174  005004                           CLR     R4
2357  007176  105104                           COMB    R4              ;0=000 377
2358  007200  000277                           SCC
2359  007202  000250                           CLN                     ;CC=0111
2360  007204  105704                           TSTB    R4              ;*TEST INSTRUCTION ON EVEN BYTE
2361  007206  102403                           BVS     1$              ;BRANCH IF ERROR
2362  007210  103402                           BCS     1$              ;
2363  007212  102401                           BVS     1$              ;
2364  007214  100401                           BMI     2$              ;BRANCH IF GOOD
2365  007216  104001                   1$:     ERROR   +1              ;CPU ERROR
2366                                                                   ;
2367  007220  005204                   2$:     INC     R4              ;POINT TO 1
```

```
007222  105704                          TSTB    R4              ; TEST INSTRUCTION
007224  001401                          BEQ     4$              ;BRANCH IF GOOD
007226  104001                  3$:     ERROR   +1              ;CPU ERROR
                                                                ;TST FAILED ON BYTE
007230                          4$:

                                ;
007230                          MSPV:

                                ;       TEST TST MODE 1
007230  005004                          CLR     R4
007232  005014                          CLR     (R4)            ;0=0
007234  000277                          SCC
007236  000244                          CLZ                     ;CC=1011
007240  005714                          TST     (R4)            ;*TEST INSTRUCTION IN MODE 1
007242  103403                          BCS     1$              ;BRANCH IF ERROR
007244  102402                          BVS     1$
007246  100401                          BMI     1$              ;
007250  001401                          BEQ     2$              ;BRANCH IF GOOD
007252  104001                  1$:     ERROR   +1              ;CPU ERROR

007254  005214                  2$:     INC     (R4)            ;0=1
007256  000277                          SCC
007260  005714                          TST     (R4)            ; TEST INSTRUCTION
007262  001403                          BEQ     3$              ;BRANCH IF ERROR
007264  102402                          BVS     3$
007266  103401                          BCS     3$
007270  100001                          BPL     4$              ;BRANCH IF GOOD
007272  104001                  3$:     ERROR   +1              ;CPU ERROR
                                                                ;TST FAILED MODE 1
007274                          4$:


                                ;
007274                          MSPX:

                                ;       TEST TST MODE 1 BYTE
007274  005004                          CLR     R4              ;R4=0
007276  005014                          CLR     (R4)
007300  105114                          COMB    (R4)
007302  005214                          INC     (R4)            ;0=001 000
007304  000277                          SCC
007306  000244                          CLZ                     ;CC=1011
007310  105714                          TSTB    (R4)            ;*TEST INTRUCTION
007312  103403                          BCS     1$              ;BRANCH IF ERROR
007314  102402                          BVS     1$              ;
007316  100401                          BMI     1$
007320  001401                          BEQ     2$              ;BRANCH IF GOOD
007322  104001                  1$:     ERROR   +1              ;CPU ERROR

007324  005204                  2$:     INC     R4              ;R4=1
007326  000277                          SCC
007330  105714                          TSTB    (R4)            ; TEST INSTRUCTION
007332  001403                          BEQ     3$              ;BRANCH IF ERROR
007334  100402                          BMI     3$
007336  102401                          BVS     3$
007340  103001                          BCC     4$              ;BRANCH IF GOOD
```

```
2424  007342  104001                  3$:     ERROR   ·1              ;CPU ERROR
2425                                                                    ;
2426  007344                          4$:
2427
2428                                  ;
2429  007344                          MSPY:
2430
2431                                  ;       TEST TST MODE 2
2432  007344  005004                          CLR     R4              ;
2433  007346  005024                          CLR     (R4)·           ;0=0
2434  007350  005014                          CLR     (R4)
2435  007352  005114                          COM     (R4)            ;2=-1
2436  007354  005004                          CLR     R4              ;R4=0
2437  007356  000277                          SCC             ;
2438  007360  000244                          CLZ             ;CC=1011
2439  007362  005724                          TST     (R4)·           ; TEST INSTRUCTION
2440  007364  103403                          BCS     1$              ;BRANCH IF ERROR
2441  007366  102402                          BVS     1$
2442  007370  100401                          BMI     1$
2443  007372  001401                          BEQ     2$              ;BRANCH IF GOOD
2444  007374  104001                  1$:     ERROR   ·1              ;CPU ERROR
2445                                                                  ;MODE 2 TEST FAILED
2446  007376  005724                  2$:     TST     (R4)·           ;TST LOC2
2447  007400  103403                          BCS     3$
2448  007402  102402                          BVS     3$
2449  007404  001401                          BEQ     3$
2450  007406  100401                          BMI     4$
2451  007410  104001                  3$:     ERROR   ·1              ;CPU ERROR
2452                                                                  ;MODE 2 FAILED
2453  007412                          4$:
2454
2455                                  ;
2456  007412                          MSPZ:
2457
2458                                  ;       TEST TST MODE 2 BYTE
2459  007412  005004                          CLR     R4
2460  007414  005024                          CLR     (R4)·           ;
2461  007416  105144                          COMB    -(R4)           ;0=377 000
2462  007420  005304                          DEC     R4              ;R4=0
2463  007422  000277                          SCC
2464  007424  000244                          CLZ                     ;CC=1011
2465  007426  105724                          TSTB    (R4)·           ;
2466  007430  102403                          BVS     1$              ;BRANCH IF ERROR
2467  007432  103402                          BCS     1$
2468  007434  100401                          BMI     1$
2469  007436  001401                          BEQ     2$              ;BRANCH IF GOOD
2470  007440  104001                  1$:     ERROR   ·1              ;CPU ERROR
2471                                                                  ;MODE 2 EVEN BYTE FAILED
2472  007442  000277                  2$:     SCC
2473  007444  000250                          CLN                     ;CC=0111
2474  007446  105724                          TSTB    (R4)·           ;
2475  007450  001403                          BEQ     3$              ;
2476  007452  103402                          BCS     3$
2477  007454  102401                          BVS     3$
2478  007456  100401                          BMI     4$              ;BRANCH IF GOOD
2479  007460  104001                  3$:     ERROR   ·1              ;CPU ERROR
```

```
                                                                        ;MODE 2 ODD BYTE FAILED
007462                          4$:

                                ;
007462                          MSPAA:

                                ;       TEST TST MODE 3
007462  005004                          CLR     R4
007464  005014                          CLR     (R4)
007466  105114                          COMB    (R4)
007470  005214                          INC     (R4)                    ;0=400
007472  005034                          CLR     @(R4)+                  ;400=0
007474  005004                          CLR     R4                      ;R4=0
007476  000277                          SCC
007500  000244                          CLZ                             ;CC=1011
007502  005734                          TST     @(R4)+                  ; TEST MODE 3
007504  103403                          BCS     1$                      ;BRANCH IF ERROR
007506  102402                          BVS     1$                      ;
007510  100401                          BMI     1$
007512  001401                          BEQ     2$                      ;BRANCH IF GOOD
007514  104001                  1$:     ERROR   +1                      ;CPU ERROR
                                                                        ;MODE 3 FAILED
007516  005304                  2$:     DEC     R4
007520  005304                          DEC     R4                      ;R4=0
007522  005334                          DEC     @(R4)+                  ;400=-1
007524  005004                          CLR     R4
007526  000277                          SCC
007530  000250                          CLN                             ;CC=0111
007532  005734                          TST     @(R4)+                  ; TEST INSTRUCTION
007534  103403                          BCS     3$                      ;
007536  001402                          BEQ     3$                      ;BRANCH IF ERROR
007540  102401                          BVS     3$                      ;
007542  100401                          BMI     4$                      ;BRANCH IF GOOD
                                                                        ;ERROR MODE 3
007544  104001                  3$:     ERROR   +1                      ;CPU ERROR
007546                          4$:

                                ;
007546                          MSPBB:

                                ;       TEST TST MODE 3 AUTO-INC
007546  005004                          CLR     R4
007550  005014                          CLR     (R4)                    ;0=0
007552  105114                          COMB    (R4)                    ;
007554  005214                          INC     (R4)                    ;0=400
007556  005001                          CLR     R1
007560  105101                          COMB    R1
007562  005201                          INC     R1                      ;R1=400
007564  005011                          CLR     (R1)                    ;400=0
007566  000277                          SCC
007570  005734                          TST     @(R4)+                  ;400=0
007572  103403                          BCS     1$                      ;ERROR IF CARRY
007574  102402                          BVS     1$                      ;ERROR IF OVERFLOW
007576  100401                          BMI     1$                      ;ERROR IF MINUS
007600  001401                          BEQ     2$                      ;ERROR IF NOT EQUAL
007602  104001                  1$:     ERROR   +1                      ;CPU ERROR
```

```
2536
2537  007604  005304                    2$:     DEC     R4                      ;CC SHOULD = 0100
2538  007606  005304                            DEC     R4
2539  007610  005704                            TST     R4                      ;SEE IF AUTO-INC WORKED
2540  007612  001401                            BEQ     4$                      ;ERROR IF R4 NE 0
2541  007614  104001                    3$:     ERROR   +1                      ;CPU ERROR
2542                                                                    ;AUTO-INC FIALED
2543  007616                            4$:
2544
2545                                    ;
2546  007616                            MSTB3:
2547
2548                                    ;       TEST TST MODE 3 BYTE
2549  007616  005004                            CLR     R4
2550  007620  005014                            CLR     (R4)
2551  007622  105114                            COMB    (R4)
2552  007624  005214                            INC     (R4)
2553  007626  005214                            INC     (R4)                    ;0=401
2554  007630  005001                            CLR     R1
2555  007632  105101                            COMB    R1
2556  007634  005201                            INC     R1                      ;R1=400
2557  007636  005011                            CLR     (R1)
2558  007640  005111                            COM     (R1)
2559  007642  105011                            CLRB    (R1)                    ;400=377 000
2560  007644  105734                            TSTB    @(R4)+                  ;** TEST INSTRUCTION
2561  007646  001403                            BEQ     1$                      ;ERROR IF EQUAL
2562  007650  103402                            BCS     1$                      ;ERROR IF CARRY SET
2563  007652  102401                            BVS     1$                      ;ERROR IR OVERFLOW
2564  007654  100401                            BMI     2$                      ;BRANCH IF MINUS
2565  007656  104001                    1$:     ERROR   +1                      ;CPU ERROR
2566                                                                    ;CC ERROR
2567  007660  005304                    2$:     DEC     R4
2568  007662  005304                            DEC     R4
2569  007664  001401                            BEQ     4$                      ;BRANCH IF AUTO-INC WORKED
2570  007666  104001                    3$:     ERROR   +1                      ;CPU ERROR
2571                                                                    ;AUTO-INC FAILED
2572  007670                            4$:
2573
2574                                    ;
2575  007670                            MST4:
2576
2577                                    ;       TEST TST MODE 4
2578  007670  005004                            CLR     R4
2579  007672  005014                            CLR     (R4)                    ;0=0
2580  007674  005204                            INC     R4
2581  007676  005204                            INC     R4                      ;R4=2
2582  007700  000277                            SCC
2583  007702  000244                            CLZ                             ;CC=1011
2584  007704  005744                            TST     -(R4)                   ;**TEST INTRUCTION
2585  007706  103403                            BCS     1$                      ;ERROR IF CARRY
2586  007710  102402                            BVS     1$                      ;ERROR IF OVERFLOW
2587  007712  100401                            BMI     1$                      ;ERROR IF MINUS
2588  007714  001401                            BEQ     2$                      ;BRANCH IF GOOD
2589  007716  104001                    1$:     ERROR   +1                      ;CPU ERROR
2590                                                                    ;CC WRONG
2591  007720  005704                    2$:     TST     R4                      ;INSURE CORRECT AUTO-DEC
```

```
2592  007722  001401                    BEQ     4$              ;BRANCH IF GOOD AUTO-DEC
2593                                                            ;BAD AUTO-DEC
2594  007724  104001              3$:   ERROR   ·1              ;CPU ERROR
2595  007726                      4$:
2596
2597                              ;
2598  007726                      MST4B:
2599
2600                              ;       TEST TST MODE 4 BYTE
2601  007726  005004                      CLR     R4
2602  007730  005014                      CLR     (R4)
2603  007732  005114                      COM     (R4)
2604  007734  105114                      COMB    (R4)            ;0=377 000
2605  007736  000277                      SCC
2606  007740  005204                      INC     R4
2607  007742  005204                      INC     R4              ;R4=2
2608  007744  105744                      TSTB    -(R4)           ;**TEST INSTRUCTION
2609  007746  001403                      BEQ     1$              ;ERROR IF EQUAL TO 0
2610  007750  103402                      BCS     1$              ;ERROR IF CARRY
2611  007752  102401                      BVS     1$              ;ERROR IF OVERFLOW
2612  007754  100401                      BMI     2$              ;BRANCH IF MINUS
2613  007756  104001              1$:     ERROR   ·1              ;CPU ERROR
2614                                                  ;CC SHOULD EQUAL 0100
2615  007760  105744              2$:     TSTB    -(R4)           ;**TEST EVEN BYTE
2616  007762  001401                      BEQ     4$              ;BRANCH IF GOOD
2617  007764  104001              3$:     ERROR   ·1              ;CPU ERROR
2618                                                  ;CC SHOULD EQUAL 0100 AND R4=-1
2619  007766                      4$:
2620                              ;
2621  007766                      MST5:
2622
2623                              ;       TEST TST MODE 5
2624  007766  005004                      CLR     R4              ;
2625  007770  005024                      CLR     (R4)+           ;0=0, R4=2
2626  007772  000277                      SCC
2627  007774  000244                      CLZ                     ;CC=1011
2628  007776  005754                      TST     @-(R4)          ; TEST INSTRUCTION
2629  010000  103403                      BCS     1$              ;ERROR IF CARRY
2630  010002  102402                      BVS     1$              ;ERROR IF OVERFLOW
2631  010004  100401                      BMI     1$              ;ERROR IF MINUS
2632  010006  001401                      BEQ     2$              ;BRANCH IF GOOD
2633  010010  104001              1$:     ERROR   ·1              ;CPU ERROR
2634                                                  ;CC WRONG, SHOULD = 0100
2635  010012  005704              2$:     TST     R4
2636  010014  001401                      BEQ     4$              ;BRANCH IF AUTO-DEC WORKED
2637  010016  104001              3$:     ERROR   ·1              ;CPU ERROR
2638                                                  ;AUTO-DEC FAILED
2639  010020                      4$:
2640
2641                              ;
2642  010020                      MST5B:
2643
2644                              ;       TEST TST MODE 5 BYTE
2645  010020  005004                      CLR     R4
2646  010022  005014                      CLR     (R4)
2647  010024  105114                      COMB    (R4)
```

```
2648  010026  005214                        INC     (R4)            ;0=400
2649  010030  005034                        CLR     @(R4)+          ;400=0, R4=2
2650  010032  005154                        COM     @-(R4)          ;
2651  010034  105134                        COMB    @(R4)+          ;400=377 000 R4=2
2652  010036  105754                        TSTB    @-(R4)          ;**TEST INSTRUCTION
2653  010040  103403                        BCS     1$              ;ERROR IF CARRY
2654  010042  100402                        BMI     1$              ;ERROR IF MINUS
2655  010044  102401                        BVS     1$              ;ERROR IF OVERFLOW
2656  010046  001401                        BEQ     2$              ;BRANCH IF GOOD
2657  010050  104001              1$:       ERROR   +1              ;CPU ERROR
2658                                                  ;CC SHOULD = 0100
2659  010052  005224              2$:       INC     (R4)+           ;0=401
2660  010054  105754                        TSTB    @-(R4)          ;**TEST INSTRUCTION
2661  010056  100401                        BMI     4$              ;BRANCH IF GOOD
2662                                                                ;EVEN BYTE FAILURE
2663  010060  104001              3$:       ERROR   +1              ;CPU ERROR
2664  010062                      4$:
2665
2666                              ;
2667  010062                      MST6:
2668
2669                              ;       TEST TST MODE 6
2670  010062  005004                        CLR     R4
2671  010064  005014                        CLR     (R4)            ;0=0
2672  010066  105104                        COMB    R4
2673  010070  005204                        INC     R4              ;R4=400
2674  010072  005014                        CLR     (R4)
2675  010074  005114                        COM     (R4)            ;400=-1
2676  010076  005764  177400               TST     -400(R4)        ;**TEST LOCATION 0
2677  010102  103403                        BCS     1$              ;ERROR IF CARRY
2678  010104  102402                        BVS     1$              ;ERROR IF OVERFLOW
2679  010106  100401                        BMI     1$              ;ERROR IF MINUS
2680  010110  001401                        BEQ     2$              ;BRANCH IF ZERO
2681  010112  104001              1$:       ERROR   +1              ;CPU ERROR
2682                                                  ;CC ARE WRONG
2683  010114  005004              2$:       CLR     R4              ;
2684  010116  005764  000400               TST     400(R4)         ;TST LOCATION 400
2685  010122  001401                        BEQ     3$              ;ERROR IF EQUAL
2686  010124  100401                        BMI     4$              ;BRANCH IF MINUS
2687  010126  104001              3$:       ERROR   +1              ;CPU ERROR
2688                                                  ;CC ERROR
2689  010130                      4$:
2690
2691                              ;
2692  010130                      MST7:
2693
2694                              ;       TEST TST MODE 7
2695  010130  005004                        CLR     R4
2696  010132  005014                        CLR     (R4)
2697  010134  005124                        COM     (R4)+           ;0=-1
2698  010136  005014                        CLR     (R4)            ;2=0
2699  010140  005002                        CLR     R2              ;R2=0
2700  010142  005004                        CLR     R4
2701  010144  105104                        COMB    R4
2702  010146  005204                        INC     R4              ;R4=400
2703  010150  005014                        CLR     (R4)            ;400=0
```

```
010152  005774  177402                  TST     @-376(R4)       ;**TEST LOCATION 0
010156  103403                          BCS     1$              ;ERROR IF CARRY
010160  102402                          BVS     1$              ;ERROR IF OVERFLOW
010162  001401                          BEQ     1$              ;ERROR IF ZERO
010164  100401                          BMI     2$              ;BRANCH IF GOOD
010166  104001                  1$:     ERROR   +1              ;CPU ERROR
                                                        ;ERROR IN CC
010170  005222                  2$:     INC     (R2)+           ;0=-2 R2=2
010172  005774  177402                  TST     @-376(R4)       ;** CHECK CONTENTS OF LOCATION 2
010176  100401                          BMI     3$              ;ERROR IF MINUS
010200  001401                          BEQ     4$              ;BRANCH IF GOOD
010202  104001                  3$:     ERROR   +1              ;CPU ERROR
                                                        ;CC SHOULD = 0100
010204                          4$:

                                ;
                                ;
                                ;
                                ;
                                ;
                                .SBTTL ************ DOUBLE OPERAND TESTS *****************
                                ;
010204                          MDM0:

                                ;       TEST MOVE MODE 0
010204  005004                          CLR     R4              ;R4=0
010206  005001                          CLR     R1
010210  005101                          COM     R1              ;R1=-1
010212  010104                          MOV     R1,R4           ;**TEST MOVE INSTRUCTION
010214  001402                          BEQ     1$              ;ERROR IF R4=0
010216  102401                          BVS     1$              ;ERROR IF OBERFLOW
010220  100401                          BMI     2$              ;BRANCH IF MINUS
010222  104001                  1$:     ERROR   +1              ;CPU ERROR
                                                        ;CC SHOULD = 100-, R4 SHOULD=-1
010224  005204                  2$:     INC     R4              ;R4 SHOULD =0
010226  001375                          BNE     1$              ;ERROR IF R4 NE 0

                                ;
010230                          MDA0:

                                ;       TEST ADD MODE 0
010230  005004                          CLR     R4              ;R4=0
010232  005001                          CLR     R1
010234  005101                          COM     R1              ;R1=-1
010236  060104                          ADD     R1,R4           ;** TEST ADD OF R1 TO R4
010240  001403                          BEQ     1$              ;ERROR IF ZERO
010242  103402                          BCS     1$              ;ERROR IF CARRY
010244  102401                          BVS     1$              ;ERROR IF OVERFLOW
010246  100401                          BMI     2$              ;BRANCH IF MINUS
010250  104001                  1$:     ERROR   +1              ;CPU ERROR
                                                        ;CC SHOULD = 1000 , R4=-1
010252  005204                  2$:     INC     R4              ;R4 SHOULD =0
010254  001401                          BEQ     4$              ;BRANCH IF R4=0
010256  104001                  3$:     ERROR   +1              ;CPU ERROR
                                                        ;R4 SHOULD = 0
010260                          4$:
```

```
2760
2761                                    ;
2762  010260                            MDS0:
2763
2764                                    ;       TEST SUB MODE 0
2765  010260  005004                            CLR     R4
2766  010262  005001                            CLR     R1
2767  010264  005201                            INC     R1                      ;R1=1 R4=0
2768  010266  160104                            SUB     R1,R4                   ;**TEST OF R4-R1, R4=-1
2769  010270  102403                            BVS     1$                      ;ERROR IF V SET
2770  010272  103002                            BCC     1$                      ;ERROR IF NO CARRY
2771  010274  001401                            BEQ     1$                      ;ERROR IF =0
2772  010276  100401                            BMI     2$                      ;BRANCH IF MINUS
2773  010300  104001                    1$:     ERROR   ;1              ;CPU ERROR
2774                                                                    ;CC SHOULD = 1001
2775  010302  005101                    2$:     COM     R1                      ;R1=1
2776  010304  005201                            INC     R1                      ;GET TWO'S COMPLIMENT, R1=-1
2777  010306  160104                            SUB     R1,R4                   ;**TEST R4-R1 (-1-1= 0
2778  010310  001401                            BEQ     4$                      ;BRANCH IF ZERO
2779  010312  104001                    3$:     ERROR   ;1              ;CPU ERROR
2780                                                                    ;CC SHOULD = 0100
2781  010314                            4$:
2782
2783                                    ;
2784  010314                            MDM27:
2785
2786                                    ;       TEST MOV MODE 27,00
2787  010314  000257                            CCC                             ;CC=0000
2788  010316  012704  125252                    MOV     #125252,R4              ;**TEST MOVE
2789  010322  001401                            BEQ     1$                      ;ERROR IF = 0
2790  010324  100401                            BMI     2$                      ;BRANCH IF MINUS
2791  010326  104001                    1$:     ERROR   ;1              ;CPU ERROR
2792                                                                    ;CC SHOULD = 1000
2793  010330  012701  052525            2$:     MOV     #052525,R1              ;**TEST MOVE
2794  010334  100401                            BMI     3$                      ;ERROR IF MINUS
2795  010336  001001                            BNE     4$                      ;BRANCH IF NE 0
2796  010340  104001                    3$:     ERROR   ;1              ;CPU ERROR
2797                                                                    ;CC SHOULD = 0000
2798  010342  060104                    4$:     ADD     R1,R4                   ;R1+R4=-1
2799  010344  100401                            BMI     6$                      ;BRANCH IF MINUS
2800  010346  104001                    5$:     ERROR   ;1              ;CPU ERROR
2801                                                                    ;MOV FAILED
2802  010350  005204                    6$:     INC     R4                      ;R4+1=0
2803  010352  001375                            BNE     5$                      ;ERROR IF NOT ZERO
2804
2805                                    ;
2806  010354                            MBI00:
2807
2808                                    ;       TEST BIC, BIS MODE 0,0
2809  010354  005004                            CLR     R4
2810  010356  005104                            COM     R4                      ;R4=-1
2811  010360  012701  125252                    MOV     #125252,R1              ;SETUP R1 TEST DATA
2812  010364  012702  052525                    MOV     #052525,R2              ;R2=COMPLIMENT OF R1
2813  010370  000261                            SEC
2814  010372  040104                            BIC     R1,R4                   ;**TEST BIC WITH CARRY SET
2815  010374  103003                            BCC     1$                      ;ERROR IF NO CARRY
```

```
2816  010376  102402                        BVS     1$                      ;ERROR IF OVERFLOW
2817  010400  001401                        BEQ     1$                      ;ERROR IF 0
2818  010402  100001                        BPL     2$                      ;BRANCH IF PLUS
2819  010404  104001              1$:       ERROR   ·1              ;CPU ERROR
2820                                                                ;CC SHOULD = 0001
2821  010406  020402              2$:       CMP     R4,R2                   ;COMPARE CONTENTS OF R4 AND R2
2822  010410  001401                        BEQ     4$                      ;BRANCH IF EQUAL
2823  010412  104001              3$:       ERROR   ·1              ;CPU ERROR
2824                                                                ;R4 AND R2 SHOULD BE EQUAL
2825  010414  005301              4$:       DEC     R1                      ;R1=125251
2826  010416  050201                        BIS     R2,R1                   ;BIS 052525 AND 125251=177775
2827  010420  100401                        BMI     6$                      ;BRANCH IF MIUS VALUE
2828  010422  104001              5$:       ERROR   ·1              ;CPU ERROR
2829                                                                ;BAD BIS OPERATION
2830  010424  005201              6$:       INC     R1
2831  010426  005201                        INC     R1
2832  010430  005201                        INC     R1                      ;R1=0
2833  010432  001373                        BNE     5$                      ;ERROR IF NE 0
2834
2835                                    ;
2836  010434                        MBC00:
2837
2838                                    ;       TEST BIT, CMP MODE 0,0
2839  010434  012701  125252                MOV     #125252,R1              ;R1=125252
2840  010440  012704  100000                MOV     #100000,R4              ;R4=100000
2841  010444  012702  052525                MOV     #052525,R2              ;R2=052525
2842  010450  030401                        BIT     R4,R1                   ;**TEST OF BIT ,CC=1000
2843  010452  001401                        BEQ     1$                      ;ERROR IF EQ 0
2844  010454  100401                        BMI     2$                      ;BRANCH IF GOOD
2845  010456  104001              1$:       ERROR   ·1              ;CPU ERROR
2846                                                                ;CC SHOULD = 1000
2847  010460  020401              2$:       CMP     R4,R1                   ;*TEST 100000-125252=25252
2848  010462  001402                        BEQ     3$                      ;ERROR IF EQUAL 0
2849  010464  103001                        BCC     3$                      ;ERROR IF CARRY CLEARED
2850  010466  100401                        BMI     4$                      ;BRANCH IF GOOD
2851  010470  104001              3$:       ERROR   ·1              ;CPU ERROR
2852                                                                ;CC SHOULD = 0010
2853  010472  020104              4$:       CMP     R1,R4                   ;125252-100000 = 25252
2854  010474  001403                        BEQ     5$                      ;ERROR IF EQUAL
2855  010476  103402                        BCS     5$                      ;ERROR IF CARRY
2856  010500  102401                        BVS     5$                      ;ERROR IF OVERFLOW
2857  010502  100001                        BPL     6$                      ;BRANCH IF GOOD
2858  010504  104001              5$:       ERROR   ·1              ;CPU ERROR
2859                                                                ;CC SHOULD =0001
2860  010506  005004              6$:       CLR     R4
2861  010510  005204                        INC     R4                      ;R4=1
2862  010512  000277                        SCC
2863  010514  030401                        BIT     R4,R1                   ;R4 ? R1 = 2
2864  010516  001401                        BEQ     8$                      ;BRANCH IF GOOD
2865  010520  104001              7$:       ERROR   ·1              ;CPU ERROR
2866                                                                ;CC SHOULD = 0101
2867  010522                        8$:
2868
2869                                    ;
2870  010522                        MM11:
2871
```

```
2872                                    ;       TEST MOV, MOVB MODE 1,1 AND SIGN EXT ON MOVB TO GPR
2873  010522  012704  000400                      MOV     #400,R4         ;R4=400
2874  010526  012701  000402                      MOV     #402,R1         ;R1=402
2875  010532  005014                              CLR     (R4)            ;
2876  010534  005114                              COM     (R4)            ;400=-1
2877  010536  005011                              CLR     (R1)            ;
2878  010540  105111                              COMB    (R1)            ;402=000 377
2879  010542  005002                              CLR     R2              ;R2=0
2880  010544  012703  000405                      MOV     #405,R3         ;R3=405
2881  010550  000277                              SCC                     ;CC=1111
2882  010552  011412                              MOV     (R4),(R2)       ;MOV    400 TO 0 ,0=-1
2883  010554  001403                              BEQ     1$              ;ERROR IF 0
2884  010556  102402                              BVS     1$              ;ERROR IF OVERFLOW
2885  010560  103001                              BCC     1$              ;ERROR IF NO CARRY
2886  010562  100401                              BMI     2$              ;BRANCH IF GOOD
2887  010564  104001                  1$:         ERROR   #1              ;CPU ERROR
2888                                                              ;CC SHOULD =1001
2889  010566  005212                  2$:         INC     (R2)            ;0=0
2890  010570  001004                              BNE     3$              ;ERROR IF NOT 0
2891  010572  000257                              CCC                     ;CC=0000
2892  010574  111113                              MOVB    (R1),(R3)               ;405=377
2893  010576  001401                              BEQ     3$              ;ERROR IF EQUAL
2894  010600  100401                              BMI     4$              ;BRANCH IF GOOD
2895  010602  104001                  3$:         ERROR   #1              ;CPU ERROR
2896                                                          ;
2897  010604  105213                  4$:         INCB    (R3)            ;405=0
2898  010606  001375                              BNE     3$              ;ERROR IF 405 NOT 0
2899                                  ;CHECK THAT SIGN EXTENSION OCCURS ON A MOVB TO GENERAL REGISTER.
2900  010610  005002                              CLR     R2              ;INIT R2 TO ZERO.
2901  010612  111102                              MOVB    (R1),R2         ;MOVE 377 TO R2
2902  010614  100005                              BPL     5$              ;ERROR! BIT 15 SHOULD BE SET.
2903  010616  102404                              BVS     5$              ;V BIT SHOULD BE CLEARED
2904  010620  103403                              BCS     5$              ;CARRY BIT SHOULD BE UNAFFECTED
2905  010622  022702  177777                      CMP     #177777,R2      ;TEST R2
2906  010626  001401                              BEQ     6$              ;SIGN EXTENDED THROUGH UPPER BYTE
2907                                                                      ;ERROR! BYTE SHOULD HAVE
2908                                                                      ;SIGN EXTENDED THROUGH UPPER BYTE
2909  010630  104001                  5$:         ERROR   #1              ;CPU ERROR
2910  010632                          6$:
2911
2912                                  ;
2913  010632                          MA11:
2914
2915                                  ;       TEST ADD MODE 1,1
2916  010632  012704  000400                      MOV     #400,R4         ;R4=400
2917  010636  012701  000402                      MOV     #402,R1         ;R1=402
2918  010642  012714  177753                      MOV     #-25,(R4)       ;400=-25
2919  010646  012711  000024                      MOV     #24,(R1)        ;402=24
2920  010652  061114                              ADD     (R1),(R4)       ;-25+24=-1
2921  010654  001404                              BEQ     1$              ;ERROR IF 0
2922  010656  103403                              BCS     1$              ;ERROR IF CARRY
2923  010660  100002                              BPL     1$              ;ERROR IF POSITIVE RESULT
2924  010662  005214                              INC     (R4)            ;-1+1=0
2925  010664  001401                              BEQ     2$              ;BRANCH IF GOOD
2926  010666  104001                  1$:         ERROR   #1              ;CPU ERROR
2927                                                              ;CC SHOULD = 1000
```

```
010670                          2$:

                                ;
010670                          MS11:

                                ;       TEST SUB MODE 1,1
010670  012704  000400                  MOV     #400,R4         ;R4=400
010674  012701  000404                  MOV     #404,R1         ;R1=404
010700  012714  000003                  MOV     #3,(R4)         ;400=3
010704  012711  000006                  MOV     #6,(R1)         ;406=6
010710  000277                          SCC                     ;CC=1111
010712  161411                          SUB     (R4),(R1)       ;6-3=3
010714  001402                          BEQ     1$              ;ERROR IF 0
010716  100401                          BMI     1$              ;ERROR IF MINUS
010720  103001                          BCC     2$              ;BRANCH IF GOOD
010722  104001                  1$:     ERROR   +1              ;CPU ERROR
                                                        ;CC SHOULD = 0000
010724  161411                  2$:     SUB     (R4),(R1)       ;3-3=0
010726  001375                          BNE     1$              ;ERROR IF NOT 0

                                ;
010730                          MBB11:

                                ;       TEST BIC, BIS MODE 1,1
010730  012704  000400                  MOV     #400,R4         ;R4=400
010734  012701  000402                  MOV     #402,R1         ;R1=402
010740  012714  052525                  MOV     #052525,(R4)    ;400=052525
010744  012711  125252                  MOV     #125252,(R1)    ;402=125252
010750  051411                          BIS     (R4),(R1)       ;R4 V R1 = -1
010752  001401                          BEQ     1$              ;ERROR IF 0
010754  100401                          BMI     2$              ;BRANCH IF GOOD
010756  104001                  1$:     ERROR   +1              ;CPU ERROR
                                                        ;CC SHOULD = 1000
010760  005211                  2$:     INC     (R1)            ;402=0
010762  001401                          BEQ     4$              ;BRANCH IF GOOD
010764  104001                  3$:     ERROR   +1              ;CPU ERROR
                                                        ;CC SHOULD = 0100
010766  005311                  4$:     DEC     (R1)            ;402=-1
010770  041411                          BIC     (R4),(R1)       ;R1=125252
010772  001401                          BEQ     5$              ;ERROR IF 0
010774  100401                          BMI     6$              ;BRANCH IF GOOD
010776  104001                  5$:     ERROR   +1              ;CPU ERROR
                                                        ;CC SHOULD = 1000
011000  005111                  6$:     COM     (R1)            ;402=052525
011002  041114                          BIC     (R1),(R4)       ;400=0
011004  001401                          BEQ     8$              ;BRANCH IF GOOD
011006  104001                  7$:     ERROR   +1              ;CPU ERROR
                                                        ;CC SHOULD = 0100
011010                          8$:

                                ;
011010                          MBC11:

                                ;       TEST BIT, CMP MODE 1,1
011010  012704  000400                  MOV     #400,R4                 ;R4=400
011014  012714  052525                  MOV     #052525,(R4)            ;400=052525
```

```
 2984  011020  012701  000402                MOV     #402,R1                 ;R1-402
 2985  011024  012711  125252                MOV     #125252,(R1)            ;402=125252
 2986  011030  000241                        CLC                             ;CLEAR CARRY
 2987  011032  031411                        BIT     (R4),(R1)               ;**052525↑125252=0
 2988  011034  103401                        BCS     1$                      ;ERROR IF CARRY
 2989  011036  001401                        BEQ     2$                      ;BRANCH IF GOOD
 2990  011040  104001              1$:       ERROR   +1              ;CPU ERROR
 2991                                                                ;CC SHOULD = 0100
 2992  011042  021411              2$:       CMP     (R4),(R1)               ;400-402=125253
 2993  011044  001403                        BEQ     3$                      ;ERROR IF ZERO
 2994  011046  103002                        BCC     3$                      ;ERROR IF NO CARRY
 2995  011050  102001                        BVC     3$                      ;ERROR IF NO OVERFLOW
 2996  011052  100401                        BMI     4$                      ;BRANCH IF GOOD
 2997  011054  104001              3$:       ERROR   +1              ;CPU ERROR
 2998                                                                ;CC SHOULD = 1000
 2999  011056  005014              4$:       CLR     (R4)
 3000  011060  005214                        INC     (R4)                    ;400=1
 3001  011062  031114                        BIT     (R1),(R4)               ;125252↑1=0
 3002  011064  001401                        BEQ     6$                      ;BRANCH IF GOOD
 3003  011066  104001              5$:       ERROR   +1              ;CPU ERROR
 3004                                                                ;CC SHOULD= 0100
 3005  011070                    6$:
 3006
 3007                            ;
 3008  011070                    MM22:
 3009
 3010                            ;       TEST MOV MODE 2,2
 3011  011070  012704  000400                MOV     #400,R4         ;R4=400
 3012  011074  012701  000402                MOV     #402,R1         ;R1=402
 3013  011100  012714  000005                MOV     #5,(R4)         ;400=5
 3014  011104  005021                        CLR     (R1)+           ;402=0
 3015  011106  005011                        CLR     (R1)            ;
 3016  011110  005111                        COM     (R1)            ;404=-1
 3017  011112  005741                        TST     -(R1)           ;R1=402
 3018  011114  000277                        SCC                     ;CC=1111
 3019  011116  012124                        MOV     (R1)+,(R4)+     ;400=0 R4=402 R1=404
 3020  011120  100403                        BMI     1$              ;ERROR IF MINUS
 3021  011122  103002                        BCC     1$              ;ERROR IF NO CARRY
 3022  011124  102401                        BVS     1$              ;ERROR IF OVERFLOW
 3023  011126  001401                        BEQ     2$              ;BRANCH IF GOOD
 3024  011130  104001              1$:       ERROR   +1              ;CPU ERROR
 3025                                                    ;CC SHOULD= 0101
 3026  011132  005244              2$:       INC     -(R4)           ;400=1 R4=400
 3027
 3028  011134  061411                        ADD     (R4),(R1)       ;1+ -1 =0
 3029  011136  001401                        BEQ     4$              ;BRANCH IF GOOD
 3030  011140  104001              3$:       ERROR   +1              ;CPU ERROR
 3031                                                    ;CC SHOULD = 0100
 3032  011142                    4$:
 3033
 3034                            ;
 3035  011142                    MS22:
 3036
 3037                            ;       TEST SUB MODE 2,2
 3038  011142  012704  000400                MOV     #400,R4                 ;R4=400
 3039  011146  012701  000402                MOV     #402,R1                 ;R1=402
```

```
3040  011152  012714  177760              MOV     #177760,(R4)            ;400=177760
3041  011156  012711  177750              MOV     #177750,(R1)            ;402=177750
3042  011162  162421                      SUB     (R4)+,(R1)+             ;R1=177770
3043  011164  001403                      BEQ     1$                      ;ERROR IF ZERO
3044  011166  102402                      BVS     1$                      ;ERROR IF OVERFLOW
3045  011170  103001                      BCC     1$                      ;ERROR IF NO CARRY
3046  011172  100401                      BMI     2$                      ;BRANCH IF GOOD
3047  011174  104001              1$:     ERROR   +1              ;CPU ERROR
3048                                                            ;CC SHOULD=1000
3049  011176  005241              2$:     INC     -(R1)                   ;R1=1777771
3050  011200  162721  177771            SUB     #177771,(R1)+           ;R1=0
3051  011204  100401                      BMI     3$                      ;ERROR IF MINUS
3052  011206  001401                      BEQ     4$                      ;BRANCH IF GOOD
3053  011210  104001              3$:     ERROR   +1              ;CPU ERROR
3054                                                            ;CCSHOULD = 0100
3055  011212                      4$:
3056
3057                              ;
3058  011212                      MBB22:
3059
3060                              ;       TEST BIC, BICB, BIS, BISB MODE 2,2
3061  011212  012704  000400              MOV     #400,R4                 ;R4=400
3062  011216  012701  000402              MOV     #402,R1                 ;R1=402
3063  011222  012702  000404              MOV     #404,R2                 ;R2=404
3064  011226  012714  141401              MOV     #141401,(R4)            ;400=303 001
3065  011232  012711  177405              MOV     #177405,(R1)            ;402=377 005
3066  011236  012722  000070              MOV     #70,(R2)+               ;404=2070
3067  011242  012722  177777              MOV     #-1,(R2)+               ;406=-1
3068  011246  042421                      BIC     (R4)+,(R1)+             ;402=074004
3069  011250  001401                      BEQ     1$                      ;ERROR IF ZERO
3070  011252  100001                      BPL     2$                      ;BRANCH IF GOOD
3071                                                            ;CC SHOULD = 1000
3072  011254  104001              1$:     ERROR   +1              ;CPU ERROR
3073  011256  052421              2$:     BIS     (R4)+,(R1)+             ;404=074074
3074  011260  142421                      BICB    (R4)+,(R1)+             ;406=074
3075  011262  005301                      DEC     R1                      ;R4=405 R1=406
3076  011264  152421                      BISB    (R4)+,(R1)+             ;406=-1 R4=406 R1=407
3077  011266  100401                      BMI     4$                      ;BRANCH IF GOOD
3078                                                            ;406 SHOULD=-1
3079  011270  104001              3$:     ERROR   +1              ;CPU ERROR
3080  011272  005214              4$:     INC     (R4)                    ;406 SHOULD=0
3081  011274  001401                      BEQ     6$                      ;BRANCH IF GOOD
3082                                                            ;ERROR! 406 NE 0
3083  011276  104001              5$:     ERROR   +1              ;CPU ERROR
3084  011300                      6$:
3085
3086                              ;
3087  011300                      MBC22:
3088
3089                              ;       TEST BIT, CMP MODE 2,2
3090  011300  012704  000400              MOV     #400,R4                 ;R4=400
3091  011304  012701  000402              MOV     #402,R1                 ;R1=402
3092  011310  012714  125252              MOV     #125252,(R4)            ;400=125252
3093  011314  012721  100001              MOV     #100001,(R1)+           ;402=100001
3094  011320  012711  100002              MOV     #100002,(R1)            ;404=100002
3095  011324  005741                      TST     -(R1)                   ;R1=402
```

```
011326  132421                          BITB    (R4)+,(R1)+     ;**ANDED RESULT= 000
011330  100401                          BMI     1$              ;ERROR IF MINUS
011332  001401                          BEQ     2$              ;BRANCH IF GOOD
011334  104001                  1$:     ERROR   +1              ;CPU ERROR
                                                        ;CC SHOULD = 0100
011336  132124                  2$:     BITB    (R1)+,(R4)+     ;** ANDED RESULT = 200
011340  001401                          BEQ     3$              ;ERROR IF EQUAL
011342  100401                          BMI     4$              ;BRANCH IF GOOD
011344  104001                  3$:     ERROR   +1              ;CPU ERROR
                                                        ;CC SHOULD= 1000 R4=402 R1=404
011346  022421                  4$:     CMP     (R4)+,(R1)+     ;RESULT =+1
011350  001402                          BEQ     5$              ;ERROR IF EQUAL
011352  103001                          BCC     5$              ;ERROR IF NO CARRY
011354  100401                          BMI     6$              ;BRANCH IF GOOD
011356  104001                  5$:     ERROR   +1              ;CPU ERROR
                                                        ;CC SHOULD = 0000
011360  005341                  6$:     DEC     -(R1)           ;404=100001
011362  005741                          TST     -(R1)           ;R4=404 R1=402
011364  022124                          CMP     (R1)+,(R4)+     ;RESULT =0
011366  001401                          BEQ     8$              ;BRANCH IF GOOD
                                                                ;CC SHOULD = 0100 R1=404 R4=406
011370  104001                  7$:     ERROR   +1              ;CPU ERROR
011372                          8$:

                                ;
011372                          MS33:

                                ;       TEST SUB MODE 3,3
011372  005004                          CLR     R4              ;R4=0
011374  012701  000002                  MOV     #2,R1           ;R1=2
011400  012702  000400                  MOV     #400,R2         ;R2=400
011404  012714  000400                  MOV     #400,(R4)       ;0=400
011410  012711  000402                  MOV     #402,(R1)       ;2=402
011414  012722  000200                  MOV     #200,(R2)+      ;400=200
011420  012712  054320                  MOV     #54320,(R2)     ;402=54320
011424  163431                          SUB     @(R4)+,@(R1)+   ;54320 - 200=54120
011426  001402                          BEQ     1$              ;ERROR IF ZERO
011430  103401                          BCS     1$              ;ERROR IF CARRY
011432  100001                          BPL     2$              ;BRANCH IF GOOD
011434  104001                  1$:     ERROR   +1              ;CPU ERROR
                                                        ;CC SHOULD =0001
011436  022712  054120          2$:     CMP     #54120,(R2)     ; TEST R4 AUTO-INC AND RESULT
011442  001401                          BEQ     4$              ;BRANCH IF GOOD
011444  104001                  3$:     ERROR   +1              ;CPU ERROR
                                                        ;CC SHOULD = 0100 R4=2 R1=4
011446  005067  166326          4$:     CLR     0               ;RESTORE VECTORS
011452  005067  166324                  CLR     2               ;       "



                                ;
011456                          MCB44:

                                ;       TEST CMP, BIT MODE 4,4
011456  012704  000400                  MOV     #400,R4                 ;R4=400
011462  012701  000402                  MOV     #402,R1                 ;R1=402
011466  012721  125366                  MOV     #125366,(R1)+           ;402=125366 R1=404
```

```
3152  011472  012724  173001                  MOV     #173001,(R4)+           ;400=173001 R4=402
3153  011476  024441                          CMP     -(R4),-(R1)             ;173001 - 125366=045603 CC=0000
3154  011500  103401                          BCS     1$                      ;ERROR IF CARRY
3155  011502  100001                          BPL     2$                      ;BRANCH IF GOOD
3156                                                                           ;BAD COMPARE
3157  011504  104001                  1$:     ERROR   +1              ;CPU ERROR
3158  011506  005204                  2$:     INC     R4
3159  011510  005201                          INC     R1                      ;R1=403
3160  011512  000261                          SEC                             ;SET CARRY
3161  011514  134144                          BITB    -(R1),-(R4)             ;173^1=0
3162  011516  103001                          BCC     3$                      ;C SHOULD REMAIN SET
3163  011520  001401                          BEQ     4$                      ;BRANCH IF GOOD
3164                                                                           ;INCORRECT COMPARE R4=400 R1=402
3165  011522  104001                  3$:     ERROR   +1              ;CPU ERROR
3166  011524  005724                  4$:     TST     (R4)+                   ;R4=402
3167  011526  005201                          INC     R1                      ;R1=403
3168  011530  124441                          CMPB    -(R4),-(R1)             ;173 - 173=0
3169  011532  001401                          BEQ     6$                      ;BRANCH IF GOOD
3170                                                                           ;BAD COMPARE
3171  011534  104001                  5$:     ERROR   +1              ;CPU ERROR
3172  011536                          6$:
3173                                  ;
3174  011536                          MA55:
3175
3176                                  ;       TEST ADD MODE 5,5
3177  011536  012704  000400                  MOV     #400,R4
3178  011542  012724  000001                  MOV     #1,(R4)+                ;400=1
3179  011546  012724  177776                  MOV     #-2,(R4)+               ;402=-2
3180  011552  012724  000400                  MOV     #400,(R4)+              ;404=400
3181  011556  012714  000402                  MOV     #402,(R4)               ;406=402 R4=406
3182  011562  012701  000410                  MOV     #410,R1                 ;R1=410
3183  011566  065451                          ADD     @-(R4),@-(R1)           ;1+ -2= -1
3184  011570  001402                          BEQ     1$                      ;ERROR IF ZERO
3185  011572  100001                          BPL     1$                      ;ERROR IF PLUS
3186  011574  103001                          BCC     2$                      ;BRANCH IF GOOD
3187  011576  104001                  1$:     ERROR   +1              ;CPU ERROR
3188                                                                   ;BAD ADD
3189  011600  062704  000004          2$:     ADD     #4,R4                   ;R4=410
3190  011604  065154                          ADD     @-(R1),@-(R4)           ;-1 + 1 = 0
3191  011606  001401                          BEQ     4$                      ;BRANCH IF GOOD
3192  011610  104001                  3$:     ERROR   +1              ;CPU ERROR
3193                                                                   ;CC SHOULD= 0100 R4=406 R1=402
3194  011612                          4$:
3195
3196                                  ;
3197                                  ;
3198                                  ;
3199                                  ;-----------------------------------
3200                                  ;TEST DOP BIT(B) MODE 6,6
3201                                  ;
3202                                          ;
3203                                          ;
3204                                          ;
3205  011612                          MB66:
3206
3207                                  ;       TEST BIT, BITB MODE 6,6
```

```
3208  011612  005004                          CLR     R4                      ;R4=0
3209  011614  012701  000400                  MOV     #400,R1                 ;
3210  011620  012721  125252                  MOV     #125252,(R1)+           ;400=125252
3211  011624  012721  000001                  MOV     #1,(R1)+                ;402=1
3212  011630  012721  100000                  MOV     #100000,(R1)+           ;404=1000000  R1=406
3213  011634  036461  000400  177774          BIT     400(R4),-4(R1)          ;(400)!(402)=0
3214  011642  001401                          BEQ     2$                      ;BRANCH IF GOOD
3215                                                                          ;CC SHOULD = 0100
3216  011644  104001                  1$:     ERROR   .1              ;CPU ERROR
3217  011646  136461  000405  177772  2$:     BITB    405(R4),-6(R1)          ;(405)!(400)=200
3218  011654  001401                          BEQ     3$                      ;ERROR IF ZERO
3219  011656  100401                          BMI     4$                      ;BRANCH IF GOOD
3220                                                                          ;CC SHOULD = 1000
3221  011660  104001                  3$:     ERROR   .1              ;CPU ERROR
3222  011662                          4$:
3223
3224                                  ;
3225  011662                          MS77:
3226
3227                                  ;       TEST SUB MODE 7,7
3228  011662  012704  000400                  MOV     #400,R4         ;
3229  011666  005001                          CLR     R1
3230  011670  012724  177776                  MOV     #-2,(R4)+               ;400=-2
3231  011674  012724  177777                  MOV     #-1,(R4)+               ;402=-1
3232  011700  012724  000400                  MOV     #400,(R4)+              ;404=400 R4=406
3233  011704  012711  000402                  MOV     #402,(R1)               ;0=402
3234  011710  005201                          INC     R1                      ;R1=1
3235  011712  167471  177372  000403          SUB     @-406(R4),@403(R1)      ;-2 - -1 = -1
3236  011720  001401                          BEQ     1$                      ;ERROR  IF ZERO
3237  011722  100401                          BMI     2$                      ;BRANCH IF GOOD
3238                                                                          ;CC SHOULD=1000
3239  011724  104001                  1$:     ERROR   .1              ;CPU ERROR
3240  011726  167174  177777  177776  2$:     SUB     @-1(R1),@-2(R4)         ;-1 - -1 = 0
3241  011734  001401                          BEQ     4$                      ;BRANCH IF GOOD
3242  011736  104001                  3$:     ERROR   .1              ;CPU ERROR
3243                                                                  ;ERROR ON SUBTRACT 400=0
3244  011740  005067  166034          4$:     CLR     0                       ;RESTORE VECTORS
3245  011744  005067  166032                  CLR     2                       ;       "
3246
3247
3248                                  ;
3249  011750                          MRL0:
3250
3251                                  ;       TEST ROL, ROLB MODE 0
3252  011750  012704  125252                  MOV     #125252,R4              ;R4=125252
3253  011754  000277                          SCC                             ;CC=1111
3254  011756  006104                          ROL     R4                      ;R4=052525 WITH CARRY SET
3255  011760  102004                          BVC     1$                      ;ERROR IF V CLEAR
3256  011762  103003                          BCC     1$                      ;ERROR IF CARRY CLEAR
3257  011764  022704  052525                  CMP     #052525,R4              ;SEE IF R0 = EXPECTED
3258  011770  001401                          BEQ     2$                      ;ERROR IF R4 NE EXPECTD
3259  011772  104001                  1$:     ERROR   .1              ;CPU ERROR
3260                                                                  ;ROL FAILED, CC SHOULD=0011
3261  011774  012704  125252          2$:     MOV     #125252,R4              ;R4=125252
3262  012000  000257                          CCC                             ;CC=0000
3263  012002  106104                          ROLB    R4                      ;ROTATE EVEN BYTE
```

```
3264  012004  103005                  BCC     3$                      ;ERROR IF NO CARRY
3265  012006  102004                  BVC     3$                      ;ERROR IF NO OVERFLOW
3266  012010  100403                  BMI     3$                      ;ERROR IF MINUS
3267  012012  022704  125124          CMP     #125124,R4              ;SEE IF R4 = EXPECTED
3268  012016  001401                  BEQ     4$                      ;BRANCH IF GOOD
3269  012020  104001          3$:     ERROR   +1              ;CPU ERROR
3270                                                          ;ROLB FAILED, CC SHOULD=1011, R4=125125
3271  012022                  4$:
3272
3273                          ;
3274  012022                  MRLB1:
3275
3276                          ;       TEST ROL, ROLB MODE 1
3277  012022  005004                  CLR     R4                      ;R4=0
3278  012024  012714  052525          MOV     #52525,(R4)             ;0=52525
3279  012030  006114                  ROL     (R4)                    ;**TEST INSTRUCTION, 0=125252
3280  012032  100005                  BPL     1$                      ;ERROR IF PLUS
3281  012034  102004                  BVC     1$                      ;ERROR IF NO OVERFLOW
3282  012036  103403                  BCS     1$                      ;ERROR IF CARRY
3283  012040  021427  125252          CMP     (R4),#125252            ;SEE IF R4=EXPECTED
3284  012044  001401                  BEQ     2$                      ;BRANCH IF GOOD
3285                                                                  ;BAD ROL ,CC SHOULD=1010
3286  012046  104001          1$:     ERROR   +1              ;CPU ERROR
3287  012050  012714  125252  2$:     MOV     #125252,(R4)            ;0=125252
3288  012054  005204                  INC     R4                      ;R4=1
3289  012056  000277                  SCC                             ;CC=1111
3290  012060  106114                  ROLB    (R4)                    ;**TEST INSTRUCTION
3291  012062  100406                  BMI     3$                      ;ERROR IF RESULT IS POSITIVE
3292  012064  103005                  BCC     3$                      ;ERROR IF NO CARRY
3293  012066  102004                  BVC     3$                      ;ERROR IF V CLEAR
3294  012070  005304                  DEC     R4                      ;R4=0
3295  012072  022714  052652          CMP     #52652,(R4)             ;ERROR IF 0 NE EXPECTED
3296  012076  001401                  BEQ     4$                      ;BRANCH IF GOOD
3297  012100  104001          3$:     ERROR   +1              ;CPU ERROR
3298                                                          ;BAD ROLB ODD BYTE,CC SHOULD=1011
3299  012102                  4$:
3300
3301                          ;
3302  012102                  MRL2:
3303
3304                          ;       TEST ROL, ROLB MODE 2
3305  012102  005004                  CLR     R4                      ;R4=0
3306  012104  012714  100000          MOV     #100000,(R4)            ;0=100000
3307  012110  000257                  CCC                             ;CC=0000
3308  012112  006124                  ROL     (R4)+                   ;*TEST INSTRUCTION
3309  012114  103002                  BCC     1$                      ;ERROR IF NO CARRY
3310  012116  102001                  BVC     1$                      ;ERROR IF NO OVERFLOW
3311  012120  001401                  BEQ     2$                      ;BRANCH IF GOOD
3312                                                                  ;ROL FAILED ,CCSHOULD= 0100
3313  012122  104001          1$:     ERROR   +1              ;CPU ERROR
3314  012124  005304          2$:     DEC     R4
3315  012126  005304                  DEC     R4
3316  012130  001012                  BNE     3$                      ;ERROR IN AUTO-DEC
3317  012132  012714  004040          MOV     #4040,(R4)              ;0=4040
3318  012136  000241                  CLC
3319  012140  106124                  ROLB    (R4)+                   ;**TEST INSTURCTION
```

```
3320  012142  103405                    BCS     3$                      ;ERROR IF CARRY SET
3321  012144  102404                    BVS     3$                      ;ERROR IF V
3322  012146  005304                    DEC     R4
3323  012150  022714  004100            CMP     #04100,(R4)             ;SEE IF 0= EXPECTED RESULT
3324  012154  001401                    BEQ     4$                      ;BRANCH IF GOOD
3325  012156  104001              3$:   ERROR   .1                ;CPU ERROR
3326                                                            ;BAD ROL
3327  012160                       4$:
3328
3329                               ;
3330  012160                       MRL3:
3331
3332                               ;       TEST ROL, ROLB MODE 3
3333  012160  005004                    CLR     R4                      ;R4=0
3334  012162  012714  052525            MOV     #052525,(R4)                    ;0=52525
3335  012166  000277                    SCC                             ;CC=1111
3336  012170  006137  000000            ROL     @#0                     ;**TEST INSRUCTION MODE 3 WITH PC
3337  012174  100005                    BPL     1$                      ;ERROR IF PLUS
3338  012176  102004                    BVC     1$                      ;ERROR IF NO OVERFLOW
3339  012200  103403                    BCS     1$                      ;ERROR IF CARRY
3340  012202  022714  125253            CMP     #125253,(R4)            ;COMPARE RESULT WITH EXPECTED
3341  012206  001401                    BEQ     2$                      ;BRANCH IF GOOD
3342                                                                    ;BAD ROL CC SHOULD=1010
3343  012210  104001              1$:   ERROR   .1                ;CPU ERROR
3344  012212  012714  125252      2$:   MOV     #125252,(R4)            ;0=125252
3345  012216  000261                    SEC                             ;CC=---1
3346  012220  106137  000000            ROLB    @#0                     ;**TEST INSTRUCTION
3347  012224  100402                    BMI     3$                      ;ERROR IF MINUS
3348  012226  103001                    BCC     3$                      ;ERROR IF NO CARRY
3349  012230  102401                    BVS     4$                      ;BRANCH IF OVERFLOW
3350  012232  104001              3$:   ERROR   .1                ;CPU ERROR
3351                                                            ;BAD ROL, CC SHOULD=1011
3352  012234                       4$:
3353
3354                               ;
3355  012234                       MRL4:
3356
3357                               ;       TEST ROL MODE 4
3358  012234  005001                    CLR     R1                      ;R1=0
3359  012236  012704  000002            MOV     #2,R4                   ;R4=2
3360  012242  012711  054321            MOV     #54321,(R1)             ;0=54321
3361  012246  000277                    SCC                             ;CC=1111
3362  012250  006144                    ROL     -(R4)                   ;**TEST INSTRUCTION
3363  012252  100007                    BPL     1$                      ;ERROR IF PLUS
3364  012254  102006                    BVC     1$                      ;ERROR IF NO OVERFLOW
3365  012256  103405                    BCS     1$                      ;ERROR IF CARRY
3366  012260  022711  130643            CMP     #130643,(R1)            ;SEE IF EXPECTED RESULT
3367  012264  001002                    BNE     1$                      ;BRANCH IF ROL FAILED
3368  012266  005704                    TST     R4                      ;SEE IF AUTO-DEC WORKED
3369  012270  001401                    BEQ     2$                      ;BRANCH IF GOOD
3370                                                                    ;ERROR! BAD ROL INST
3371  012272  104001              1$:   ERROR   .1                ;CPU ERROR
3372  012274                       2$:
3373
3374                               ;
3375  012274                       MMRL5:
```

```
3376
3377                                    ;       TEST ROL MODE 5
3378  012274  005004                            CLR     R4                      ;R4=0
3379  012276  012714  000400                    MOV     #400,(R4)               ;0=400
3380  012302  012734  123456                    MOV     #123456,@(R4)+          ;400=123465, R4=2
3381  012306  000277                            SCC                             ;CC=1111
3382  012310  006154                            ROL     @-(R4)                  ;**TEST INSTRUCTION
3383  012312  100410                            BMI     1$                      ;ERROR IF RESULT IS MINUS
3384  012314  103007                            BCC     1$                      ;ERROR IF NO CARRY
3385  012316  102006                            BVC     1$                      ;ERROR IF NO OVERFLOW
3386  012320  005704                            TST     R4                      ;SEE IF AUTO-DEC WORKED
3387  012322  001004                            BNE     1$                      ;ERROR OF AUTO-DEC
3388  012324  022737  047135  000400            CMP     #47135,@#400            ;SEE  IF CORRECT RESULT
3389  012332  001401                            BEQ     2$                      ;BRANCH IF GOOD
3390                                                                            ;BAD ROL MODE 5
3391  012334  104001                    1$:     ERROR   +1              ;CPU ERROR
3392  012336                            2$:
3393
3394                                    ;
3395  012336                            MRL6:
3396
3397                                    ;       TEST ROL MODE 6
3398  012336  012704  000400                    MOV     #400,R4                 ;R4=400
3399  012342  005001                            CLR     R1                      ;R1=0
3400  012344  012711  032525                    MOV     #32525,(R1)             ;0=32525
3401  012350  000277                            SCC
3402  012352  006164  177400                    ROL     -400(R4)                ;**TEST INSTRUCTION
3403  012356  100405                            BMI     1$                      ;ERROR IF MINUS
3404  012360  103404                            BCS     1$                      ;ERROR IF CARRY
3405  012362  102403                            BVS     1$                      ;ERROR IF OVERFLOW
3406  012364  022711  065253                    CMP     #65253,(R1)             ;SEE IF CORRECT RESULT
3407  012370  001401                            BEQ     2$                      ;BRANCH IF GOOD
3408                                                                            ;BAD ROL MODE 6
3409  012372  104001                    1$:     ERROR   +1              ;CPU ERROR
3410  012374                            2$:
3411
3412                                    ;
3413  012374                            MRL7:
3414
3415                                    ;       TEST ROL MODE 7
3416  012374  012704  000400                    MOV     #400,R4                 ;R4=400
3417  012400  005037  000402                    CLR     @#402                   ;402=0
3418  012404  012737  100000  000000            MOV     #100000,@#0             ;0=100000
3419  012412  006174  000002                    ROL     @2(R4)                  ;**TEST INSTRUCTION
3420  012416  100406                            BMI     1$                      ;ERROR IF MINUS
3421  012420  001005                            BNE     1$                      ;ERROR IF NOT ZERO
3422  012422  103004                            BCC     1$                      ;ERROR IF NO CARRY
3423  012424  102003                            BVC     1$                      ;ERROR IF NO OVERFLOW
3424  012426  005737  000000                    TST     @#0                     ;CHECK RESULT
3425  012432  001401                            BEQ     2$                      ;BRANCH IF GOOD
3426                                                                            ;BAD ROL MODE 7
3427  012434  104001                    1$:     ERROR   +1              ;CPU ERROR
3428  012436                            2$:
3429
3430                                    ;
3431  012436                            MSW37:
```

```
3432
3433                                    ;       TEST SWAB MODE 37
3434  012436  012704  000400                      MOV     #400,R4                 ;R4=400
3435  012442  012714  040700                      MOV     #40700,(R4)             ;400= 101 300
3436  012446  000337  000400                      SWAB    @#400                   ;400 SHOULD = 300 101
3437  012452  100406                              BMI     1$                      ;ERROR IF MINUS
3438  012454  022714  140101                      CMP     #140101,(R4)            ;SEE IF EXPECTED RESULT
3439  012460  001003                              BNE     1$                      ;BRANCH IF BAD
3440  012462  000337  000400                      SWAB    @#400                   ;400=101 300
3441  012466  100401                              BMI     2$                      ;BRANCH IF GOOD
3442                                                                            ;ERROR! BAD SWAB MODE 37
3443  012470  104001                  1$:         ERROR   +1              ;CPU ERROR
3444  012472                          2$:
3445
3446                                    ;
3447  012472                          MRR0:
3448
3449                                    ;       TEST ROR MODE 0
3450  012472  012704  052525                      MOV     #52525,R4               ;R4=52525
3451  012476  000257                              CCC                             ;CC=0000
3452  012500  006004                              ROR     R4                      ;R4 SHOULD = 25252 WITH CARRY
3453  012502  103003                              BCC     1$                      ;ERROR IF NO CARRY
3454  012504  022704  025252                      CMP     #25252,R4               ;SEE IF R4= EXPECTED
3455  012510  001401                              BEQ     2$                      ;BRANCH IF GOOD
3456                                                                            ;ROR MODE 0 FAILED
3457  012512  104001                  1$:         ERROR   +1              ;CPU ERROR
3458  012514                          2$:
3459
3460                                    ;
3461  012514                          MRRB1:
3462
3463                                    ;       TEST RORB MODE 1
3464  012514  005004                              CLR     R4                      ;R4=0
3465  012516  012714  000001                      MOV     #1,(R4)                 ;0=1
3466  012522  000277                              SCC                             ;CC=1111
3467  012524  106014                              RORB    (R4)                    ;0=000200, NO C
3468  012526  103004                              BCC     1$                      ;ERROR IF NO CARRY
3469  012530  100003                              BPL     1$                      ;ERROR IF PLUS
3470  012532  022714  000200                      CMP     #200,(R4)               ;CHECK RESULT
3471  012536  001401                              BEQ     2$                      ;BRANCH IF GOOD
3472  012540  104001                  1$:         ERROR   +1              ;CPU ERROR
3473                                                                            ;BAD RESULT
3474  012542                          2$:
3475
3476                                    ;
3477
3478  012542                          MJ:
3479
3480                                    ;       TEST JMP - ALL MODES
3481  012542  012737  000001  003032              MOV     #1,@#SEQ                ;SETUP TEST SEQUENCER
3482  012550  012701  012614                      MOV     #MJU1,R1                ;SET MODE 1 JUMP ADDRESS
3483  012554  000111                              JMP     (R1)                    ;*JMP MODE 1
3484  012556  023727  003032  000002  MJU2:       CMP     @#SEQ,#2                ;CHECK FOR CORRECT SEQUENCE
3485  012564  001401                              BEQ     MJU2A                   ;BRANCH IF GOOD
3486                                                                            ;MODE 2 JUMP FAILED
3487  012566  104001                  1$:         ERROR   +1              ;CPU ERROR
```

```
3488  012570  020127  012560          MJU2A:  CMP     R1,@MJU2+2              ;CHECK FOR AUTO-INCREMENT
3489  012574  001401                  BEQ     MJU2B                   ;BRANCH IF GOOD
3490  012576  104001                  2$:     ERROR   +1              ;CPU ERROR
3491                                                                  ;AUTO-INCR FAILED
3492  012600  005237  003032          MJU2B:  INC     @#SEQ                   ;UPDATE TEST SEQUENCER
3493  012604  012701  012612                  MOV     #MJ2,R1                 ;SETUP MODE 3 JUMP
3494  012610  000131                  JMP     @(R1)+                  ;JUMP MODE 3
3495  012612  012640                  MJ2:    .WORD   MJU3                    ;MODED 3 DETINATION
3496  012614  023727  003032  000001  MJU1:   CMP     @#SEQ,#1                ; TEST FOR CORRECT SEQUENCE
3497  012622  001401                  BEQ     MJU1A                   ;BRANCH IF GOOD
3498  012624  104001                  3$:     ERROR   +1              ;CPU ERROR
3499                                                                  ;JMP OUT OF SEQUENCE
3500  012626  005237  003032          MJU1A:  INC     @#SEQ                   ;UPDATE SEQUENCE
3501  012632  012701  012556                  MOV     #MJU2,R1                ;SETUP MODE 2 DESTINATION
3502  012636  000121                  JMP     (R1)+                   ;JUMP MODE 2
3503  012640  023727  003032  000003  MJU3:   CMP     @#SEQ,#3                ; TEST FOR CORRECT SEQUENCE
3504  012646  001401                  BEQ     MJU3A                   ;BRANCH IF GOOD
3505  012650  104001                  4$:     ERROR   +1              ;CPU ERROR
3506                                                                  ;JMP OUT OF SEQUENCE
3507  012652  022701  012614          MJU3A:  CMP     #MJ2+2,R1               ; TEST AUTO-INCREMENT
3508  012656  001401                  BEQ     MJU3B                   ;BRANCH IF GOOD
3509  012660  104001                  5$:     ERROR   +1              ;CPU ERROR
3510                                                                  ;AUTO-INCREMENT FAILED MODE 3
3511  012662  005237  003032          MJU3B:  INC     @#SEQ                   ;UPDATE SEQUENCER
3512  012666  012701  012736                  MOV     #MJU4+2,R1              ;SETUP DESTINATION MODE 4
3513  012672  000141                  JMP     -(R1)                   ;EXECUTE JUMP MODE 4
3514  012674  000000                  MJU5:   HALT
3515  012676  022701  012772                  CMP     #MJ5,R1                 ;CHECK AUTO-DECREMENT
3516  012702  001401                  BEQ     MJU5A                   ;BRANCH IF GOOD AUTO-DEC
3517  012704  104001                  6$:     ERROR   +1              ;CPU ERROR
3518                                                                  ;AUTO-DEC FAIELD MODE 5
3519  012706  023727  003032  000005  MJU5A:  CMP     @#SEQ,#5                ; TEST CORRECT SEQUENCE
3520  012714  001401                  BEQ     MJU5B                   ;BRANCH IF GOOD SEQUENCE
3521  012716  104001                  7$:     ERROR   +1              ;CPU ERROR
3522                                                                  ;JMP OUT OF SEQUENCE
3523  012720  005237  003032          MJU5B:  INC     @#SEQ                   ;UPDATE SEQUENCE COUNT
3524  012724  012701  012767                  MOV     #MJU6-5,R1              ;SETUP DESTINATION MODE6
3525  012730  000161  000005                  JMP     +5(R1)                  ;JUMP MODE 6
3526                                                  ;;
3527  012734  000240                  MJU4:   NOP
3528  012736  022701  012734                  CMP     #MJU4,R1                ; TEST AUTO-DECR
3529  012742  001401                  BEQ     MJU4A                   ;BRANCH IF GOOD
3530  012744  104001                  8$:     ERROR   +1              ;CPU ERROR
3531                                                                  ;MODE 4 AUTO-DEC FAILED
3532  012746  023727  003032  000004  MJU4A:  CMP     @#SEQ,#4                ; TEST FOR CORRECT SEQUENCE
3533  012754  001401                  BEQ     MJU4B                   ;BRANCH IF CORRECT SEQUENCE
3534  012756  104001                  9$:     ERROR   +1              ;CPU ERROR
3535                                                                  ;INCORRECT JMP SEQUENCE
3536  012760  005237  003032          MJU4B:  INC     @#SEQ                   ;UPDATE SEQUENCE
3537  012764  012701  012774                  MOV     #MJ5+2,R1               ;SETUP MODE 5 POINTER
3538  012770  000151                  JMP     @-(R1)                  ;EXECUTE MODE 5 JMP
3539                                                  ;
3540  012772  012676                  MJ5:    .WORD   MJU5+2                  ;POINTER MODE 5
3541                                                  ;
3542  012774  022737  000006  003032  MJU6:   CMP     #6,@#SEQ                ;CHECK FOR CORRECT SEQUENCE
3543  013002  001401                  BEQ     MJU6A                   ;BRANCH IF GOOD
```

```
3544  013004  104001                  10$:    ERROR   ·1              ;CPU ERROR
3545                                                                  ;INCORRECT SEQUENCE
3546  013006  005237  003032          MJU6A:  INC     @#SEQ                   ;UPDATE SEQUENCER
3547  013012  012701  013032                  MOV     #MJ7+10,R1              ;SETUP INDEX
3548  013016  000171  177770                  JMP     @-10(R1)                ;EXECUTE MODE 7 JUMP
3549  013022  013026                  MJ7:    .WORD   MJU7                    ;POINTER FOR MODE 7
3550  013024  000000                  MJU7:   HALT
3551  013026  022737  000007  003032          CMP     #7,@#SEQ                ; TEST FOR CORRECT SEQUENCE
3552  013034  001401                          BEQ     MJU7E                   ;BRANCH IF GOOD SEQUENCE
3553  013036  104001                  11$:    ERROR   ·1              ;CPU ERROR
3554                                                                  ; TESTING OUT OF SEQUENCE
3555  013040                          MJU7E:
3556                                  ;
3557
3558                                  ;       TEST THAT PRE-FETCH BUFFER CAN BE OVER WRITTEN
3559  013040  012701  013350                  MOV     #128$,R1                ;SET UP R1 WITH ADDRESS OF ERROR
3560                                                                          ;ROUTINE
3561  013044  012717                          MOV     (PC)+,(PC)              ;WRITE THE NOP OVER THE JMP INSTRUCTION
3562  013046  000240                          .WORD   NOP                     ;NOP INSTRUCTION
3563  013050  000111                  64$:    .WORD   111                     ;JMP    (R1)
3564  013052  012717                          MOV     (PC)+,(PC)              ;WRITE THE NOP OVER THE JMP INSTRUCTION
3565  013054  000240                          .WORD   NOP                     ;NOP INSTRUCTION
3566  013056  000111                  65$:    .WORD   111                     ;JMP    (R1)
3567  013060  012717                          MOV     (PC)+,(PC)              ;WRITE THE NOP OVER THE JMP INSTRUCTION
3568  013062  000240                          .WORD   NOP                     ;NOP INSTRUCTION
3569  013064  000111                  66$:    .WORD   111                     ;JMP    (R1)
3570  013066  012717                          MOV     (PC)+,(PC)              ;WRITE THE NOP OVER THE JMP INSTRUCTION
3571  013070  000240                          .WORD   NOP                     ;NOP INSTRUCTION
3572  013072  000111                  67$:    .WORD   111                     ;JMP    (R1)
3573  013074  012717                          MOV     (PC)+,(PC)              ;WRITE THE NOP OVER THE JMP INSTRUCTION
3574  013076  000240                          .WORD   NOP                     ;NOP INSTRUCTION
3575  013100  000111                  68$:    .WORD   111                     ;JMP    (R1)
3576  013102  012717                          MOV     (PC)+,(PC)              ;WRITE THE NOP OVER THE JMP INSTRUCTION
3577  013104  000240                          .WORD   NOP                     ;NOP INSTRUCTION
3578  013106  000111                  69$:    .WORD   111                     ;JMP    (R1)
3579  013110  012717                          MOV     (PC)+,(PC)              ;WRITE THE NOP OVER THE JMP INSTRUCTION
3580  013112  000240                          .WORD   NOP                     ;NOP INSTRUCTION
3581  013114  000111                  70$:    .WORD   111                     ;JMP    (R1)
3582  013116  012717                          MOV     (PC)+,(PC)              ;WRITE THE NOP OVER THE JMP INSTRUCTION
3583  013120  000240                          .WORD   NOP                     ;NOP INSTRUCTION
3584  013122  000111                  71$:    .WORD   111                     ;JMP    (R1)
3585  013124  012717                          MOV     (PC)+,(PC)              ;WRITE THE NOP OVER THE JMP INSTRUCTION
3586  013126  000240                          .WORD   NOP                     ;NOP INSTRUCTION
3587  013130  000111                  72$:    .WORD   111                     ;JMP    (R1)
3588  013132  012717                          MOV     (PC)+,(PC)              ;WRITE THE NOP OVER THE JMP INSTRUCTION
3589  013134  000240                          .WORD   NOP                     ;NOP INSTRUCTION
3590  013136  000111                  73$:    .WORD   111                     ;JMP    (R1)
3591  013140  012717                          MOV     (PC)+,(PC)              ;WRITE THE NOP OVER THE JMP INSTRUCTION
3592  013142  000240                          .WORD   NOP                     ;NOP INSTRUCTION
3593  013144  000111                  74$:    .WORD   111                     ;JMP    (R1)
3594  013146  012717                          MOV     (PC)+,(PC)              ;WRITE THE NOP OVER THE JMP INSTRUCTION
3595  013150  000240                          .WORD   NOP                     ;NOP INSTRUCTION
3596  013152  000111                  75$:    .WORD   111                     ;JMP    (R1)
3597  013154  012717                          MOV     (PC)+,(PC)              ;WRITE THE NOP OVER THE JMP INSTRUCTION
3598  013156  000240                          .WORD   NOP                     ;NOP INSTRUCTION
3599  013160  000111                  76$:    .WORD   111                     ;JMP    (R1)
```

```
3600  013162  012717                    MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
3601  013164  000240                    .WORD   NOP             ;NOP INSTRUCTION
3602  013166  000111            77$:    .WORD   111             ;JMP    (R1)
3603  013170  012717                    MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
3604  013172  000240                    .WORD   NOP             ;NOP INSTRUCTION
3605  013174  000111            78$:    .WORD   111             ;JMP    (R1)
3606  013176  012717                    MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
3607  013200  000240                    .WORD   NOP             ;NOP INSTRUCTION
3608  013202  000111            79$:    .WORD   111             ;JMP    (R1)
3609  013204  012717                    MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
3610  013206  000240                    .WORD   NOP             ;NOP INSTRUCTION
3611  013210  000111            80$:    .WORD   111             ;JMP    (R1)
3612  013212  012717                    MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
3613  013214  000240                    .WORD   NOP             ;NOP INSTRUCTION
3614  013216  000111            81$:    .WORD   111             ;JMP    (R1)
3615  013220  012717                    MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
3616  013222  000240                    .WORD   NOP             ;NOP INSTRUCTION
3617  013224  000111            82$:    .WORD   111             ;JMP    (R1)
3618  013226  012717                    MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
3619  013230  000240                    .WORD   NOP             ;NOP INSTRUCTION
3620  013232  000111            83$:    .WORD   111             ;JMP    (R1)
3621  013234  012717                    MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
3622  013236  000240                    .WORD   NOP             ;NOP INSTRUCTION
3623  013240  000111            84$:    .WORD   111             ;JMP    (R1)
3624  013242  012717                    MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
3625  013244  000240                    .WORD   NOP             ;NOP INSTRUCTION
3626  013246  000111            85$:    .WORD   111             ;JMP    (R1)
3627  013250  012717                    MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
3628  013252  000240                    .WORD   NOP             ;NOP INSTRUCTION
3629  013254  000111            86$:    .WORD   111             ;JMP    (R1)
3630  013256  012717                    MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
3631  013260  000240                    .WORD   NOP             ;NOP INSTRUCTION
3632  013262  000111            87$:    .WORD   111             ;JMP    (R1)
3633  013264  012717                    MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
3634  013266  000240                    .WORD   NOP             ;NOP INSTRUCTION
3635  013270  000111            88$:    .WORD   111             ;JMP    (R1)
3636  013272  012717                    MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
3637  013274  000240                    .WORD   NOP             ;NOP INSTRUCTION
3638  013276  000111            89$:    .WORD   111             ;JMP    (R1)
3639  013300  012717                    MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
3640  013302  000240                    .WORD   NOP             ;NOP INSTRUCTION
3641  013304  000111            90$:    .WORD   111             ;JMP    (R1)
3642  013306  012717                    MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
3643  013310  000240                    .WORD   NOP             ;NOP INSTRUCTION
3644  013312  000111            91$:    .WORD   111             ;JMP    (R1)
3645  013314  012717                    MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
3646  013316  000240                    .WORD   NOP             ;NOP INSTRUCTION
3647  013320  000111            92$:    .WORD   111             ;JMP    (R1)
3648  013322  012717                    MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
3649  013324  000240                    .WORD   NOP             ;NOP INSTRUCTION
3650  013326  000111            93$:    .WORD   111             ;JMP    (R1)
3651  013330  012717                    MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
3652  013332  000240                    .WORD   NOP             ;NOP INSTRUCTION
3653  013334  000111            94$:    .WORD   111             ;JMP    (R1)
3654  013336  012717                    MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
3655  013340  000240                    .WORD   NOP             ;NOP INSTRUCTION
```

```
3656  013342  000111                  95$:    .WORD   111                     ;JMP    (R1)
3657  013344  000137  013352                  JMP     @#129$                  ;JUMP OVER ERROR CALL
3658  013350                          128$:                                   ;ERROR! PRE-FETCH BUFFER WAS NOT
3659                                                                          ;OVER WRITTEN
3660  013350  104001                          ERROR   .1              ;CPU ERROR
3661                                  ;
3662                                  ;       NOW RESTORE THE OVER WRITTEN JMP INSTRUCTIONS FOR THE NEXT PASS.
3663                                  ;
3664  013352  012702  000040          129$:   MOV     #32.,R2                 ;SET UP R2 AS COUNTER
3665  013356  012703  013050                  MOV     #64$,R3                 ;SET UP R3 AS POINTER
3666  013362  012713  000111          130$:   MOV     #111,(R3)               ;RESTORE OVER WRITTEN JUMPS
3667  013366  062703  000006                  ADD     #6,R3                   ;POINT TO NEXT OVER WRITTEN ADDR.
3668  013372  077205                          SOB     R2,130$                 ;DO UNTIL R2=0
3669
3670                                  ;
3671  013374                          MJP:
3672
3673                                  ;       TEST JMP MODES 17,27,37,67,77
3674  013374  012737  000000  003032          MOV     #0,@#SEQ                ;SETUP TEST SEQUNCER
3675  013402  000117                          JMP     (R7)                    ;JUMP MODE 17(SHOULD BE IN-LINE)
3676                                  ;
3677  013404  005737  003032          MJP17:  TST     @#SEQ                   ;CHECK SEQUENCE
3678  013410  001401                          BEQ     2$                      ;BRANCH IF GOOD
3679                                                                          ;ERROR! BAD JUMP
3680  013412  104001                  1$:     ERROR   .1              ;CPU ERROR
3681  013414  005237  003032          2$:     INC     @#SEQ                   ;UPDATE SEQUENCE NUMBER
3682  013420  000127  000401                  JMP     #401                    ;JUMP MODE 27
3683                                                                          ;(THE #401=BR UPDATED PC+2)
3684                                  ;
3685  013424  000000                          HALT                            ;
3686                                  ;
3687  013426  023727  003032  000001  MJP27:  CMP     @#SEQ,#1                ;CHECK IF CORRECT SEQUENCE
3688  013434  001401                          BEQ     MJP27A                  ;BRANCH IF IN SEQUENCE
3689  013436  104001                  1$:     ERROR   .1              ;CPU ERROR
3690                                                                  ; TEST OUT OF SEQUENCE
3691  013440  005237  003032          MJP27A: INC     @#SEQ                   ;UPDATE SEQUENCER
3692  013444  000137  013512                  JMP     @#MJP37                 ;JUMP MODE 37
3693  013450  023727  003032  000003  MJP67:  CMP     @#SEQ,#3                ;CHECK FOR CORRECT SEQUENCE
3694  013456  001401                          BEQ     MJP67A                  ;BRANCH IF IN SEQUENCE
3695  013460  104001                  2$:     ERROR   .1              ;CPU ERROR
3696                                                                  ; TEST OUT OF SEQUENCE
3697  013462  005237  003032          MJP67A: INC     @#SEQ                   ;UPDATE SEQUENCER
3698  013466  000257                          CCC                             ;INSURE ZBIT=0
3699  013470  000177  000002                  JMP     @MJP67B                 ;JUMP MODE 7
3700                                  ;
3701  013474  000000                          HALT                            ;
3702                                  ;
3703  013476  013500                  MJP67B: .WORD   MJP77                   ;
3704                                  ;
3705  013500  023727  003032  000004  MJP77:  CMP     @#SEQ,#4                ; TEST FOR CORRECT SEQUENCE
3706  013506  001412                          BEQ     MJP77E                  ;BRANCH IF IN SEQUENCE
3707  013510  104001                  3$:     ERROR   .1              ;CPU ERROR
3708                                                                  ; TEST OUT OF SEQUENCE
3709  013512  023727  003032  000002  MJP37:  CMP     @#SEQ,#2                ; TEST FOR CORRECT SEQUENCE
3710  013520  001401                          BEQ     MJP37A                  ;BRANCH IF IN SEQUENCE
3711  013522  104001                  4$:     ERROR   .1              ;CPU ERROR
```

```
3712
3713  013524  005237  003032          MJP37A: INC     @#SEQ                   ; TEST OUT OF SEQUENCE
3714  013530  000167  177714                  JMP     MJP67                   ;UPDATE SEQUENCER
3715                                                                          ;JUMP MODE 6
3716                                  ;
3717  013534                          MJP77E:
3718                                  ;
3719                                  ;
3720  013534                          MJSR:
3721
3722                                  ;       TEST JSR ALL MODES
3723  013534  010637  003034                  MOV     R6,@#SPS                ;SAVE STACK POINTER LOCATION
3724  013540  010637  003036                  MOV     R6,@#SPSJ               ;
3725  013544  162737  000002  003036          SUB     #2,@#SPSJ               ;SPSJ = R6 AFTER DECRIMENT
3726  013552  012737  000001  003032          MOV     #1,@#SEQ                ;SETUP SEQUENCE COUNTER
3727  013560  012701  013654                  MOV     #MJSR1,R1               ;SETUP INITIAL JUMP IN MODE 1
3728  013564  005004                          CLR     R4                      ;
3729  013566  005104                          COM     R4                      ;R4=-1 TO BE SAVED ON STACK
3730  013570  004411                          JSR     R4,(R1)                 ;JSR MODE 1
3731                                  ;
3732  013572  022737  000002  003032  MJSR2:  CMP     #2,@#SEQ                ; TEST FOR CORRECT SEQUENCE
3733  013600  001401                          BEQ     MJSR2A                  ;BRANCH IF GOOD
3734  013602  104001                  5$:     ERROR   .1                      ;CPU ERROR
3735                                                                          ;MODE 2 JUMPED TO OUT OF SEQUENCE
3736  013604  023706  003036          MJSR2A: CMP     @#SPSJ,R6               ;VERIFY STACK DECRIMENT
3737  013610  001006                          BNE     6$                      ;BRANCH IF STACK INCORRECT
3738  013612  021627  125252                  CMP     (R6),#125252            ;VERIFY CONTENTS OF STACK
3739  013616  001003                          BNE     6$                      ;BRANCH IF DATA ON STACK INCORRECT
3740  013620  022704  013734                  CMP     #MJSR4,R4               ;SEE IF CORRECT RETURN ADDRESS
3741  013624  001401                          BEQ     MJSR2B                  ;BRANCH IF GOOD
3742  013626  104001                  6$:     ERROR   .1                      ;CPU ERROR
3743                                                                          ;JSR MODE 2 FAILED
3744  013630  005237  003032          MJSR2B: INC     @#SEQ                   ;UPDATE SEQUENCE COUNTER
3745  013634  013706  003034                  MOV     @#SPS,R6                ;RELOAD STACK POINTER
3746  013640  012701  013652                  MOV     #MJSRA,R1               ;SETUP JSR MODE 3
3747  013644  005004                          CLR     R4                      ;DIFFERENT DATA TO R4
3748  013646  004431                          JSR     R4,@(R1)+               ;JSR MODE 3
3749  013650  000000                          HALT
3750  013652  014020                  MJSRA:  .WORD   MJSR3                   ;LITERAL FOR JUMP MODE 3
3751                                  ;
3752  013654  022737  000001  003032  MJSR1:  CMP     #1,@#SEQ                ; TEST FOR CORRECT SEQUENCE
3753  013662  001401                          BEQ     MJSR1A                  ;BRANCH IF GOOD
3754  013664  104001                  7$:     ERROR   .1                      ;CPU ERROR
3755                                                                          ;MODE 1 JUMPED TO OUT OF SEQUENCE
3756  013666  023706  003036          MJSR1A: CMP     @#SPSJ,R6               ;VERIFY STACK DECRIMENT
3757  013672  001006                          BNE     8$                      ;BRANCH IF STACK INCORRECT
3758  013674  021627  177777                  CMP     (R6),#-1                ;VERIFY CONTENT OF STACK
3759  013700  001003                          BNE     8$                      ;BRANCH IF DATA ON STACK INCORRECT
3760  013702  022704  013572                  CMP     #MJSR2,R4               ;SEE IF CORRECT RETURN ADDRESS
3761  013706  001401                          BEQ     MJSR1B                  ;BRANCH IF GOOD
3762  013710  104001                  8$:     ERROR   .1                      ;CPU ERROR
3763                                                                          ;JSR MODE 2 FAILED
3764  013712  005237  003032          MJSR1B: INC     @#SEQ                   ;UPDATE SEQUENCE COUNTER
3765  013716  013706  003034                  MOV     @#SPS,R6                ;RELOAD STACK POINTER
3766  013722  012704  125252                  MOV     #125252,R4              ;SETUP R4 DATA
3767  013726  012701  013572                  MOV     #MJSR2,R1               ;SETUP MODE 2 JUMP ADDRESS
```

```
3768  013732  004421                    JSR     R4,(R1)+                ;*JUMP MODE 2
3769                                    ;
3770                                    ;
3771  013734  022737  000004  003032    MJSR4:  CMP     #4,@#SEQ                ; TEST FOR CORRECT SEQUENCE
3772  013742  001401                    BEQ     MJSR4A                  ;BRANCH IF GOOD
3773  013744  104001                    9$:     ERROR   +1              ;CPU ERROR
3774                                                                    ;MODE 4 JUMPED TO OUT OF SEQUENCE
3775  013746  023706  003036            MJSR4A: CMP     @#SPSJ,R6               ;VERIFY STACK DECRIMENT
3776  013752  001006                    BNE     10$                     ;BRANCH IF STACK INCORRECT
3777  013754  021627  052525            CMP     (R6),#052525            ;VERIFY CONTENTS OF STAACK
3778  013760  001003                    BNE     10$                     ;BRANCH IF DATA ON STACK INCORRECT
3779  013762  022704  014102            CMP     #MJSR6,R4               ;SEE IF CORRECT RETURN ADDRESS
3780  013766  001401                    BEQ     MJSR4B                  ;BRANCH IF GOOD
3781  013770  104001                    10$:    ERROR   +1              ;CPU ERROR
3782                                                                    ;JSR MODE 4 FAILED
3783  013772  005237  003032            MJSR4B: INC     @#SEQ                   ;UPDATE SEQUENCE COUNTER
3784  013776  013706  003034            MOV     @#SPS,R6                ;RELOAD STACK POINTER
3785  014002  012704  000377            MOV     #377,R4                 ;SETUP R4 DATA
3786  014006  012701  014020            MOV     #MJSRB+2,R1             ;SETUP JSR VECTOR
3787  014012  004451                    JSR     R4,@-(R1)               ;JSR MODE 5
3788  014014  000000                    HALT
3789  014016  014166                    MJSRB:  .WORD   MJSR5                   ;MODE 5 VECTOR
3790                                    ;
3791                                    ;
3792  014020  022737  000003  003032    MJSR3:  CMP     #3,@#SEQ                ; TEST FOR CORRECT SEQUENCE
3793  014026  001401                    BEQ     MJSR3A                  ;BRANCH IF GOOD
3794  014030  104001                    11$:    ERROR   +1              ;CPU ERROR
3795                                                                    ;MODE 3 JUMPED TO OUT OF SEQUENCE
3796  014032  023706  003036            MJSR3A: CMP     @#SPSJ,R6               ;VERIFY STACK DECRIMENT
3797  014036  001006                    BNE     12$                     ;BRANCH IF STACK INCORRECT
3798  014040  021627  000000            CMP     (R6),#0                 ;VERIFY CONTENTS OF STAACK
3799  014044  001003                    BNE     12$                     ;BRANCH IF DATA ON STACK INCORRECT
3800  014046  022704  013650            CMP     #MJSRA-2,R4             ;SEE IF CORRECT RETURN ADDRESS
3801  014052  001401                    BEQ     MJSR3B                  ;BRANCH IF GOOD
3802  014054  104001                    12$:    ERROR   +1              ;CPU ERROR
3803                                                                    ;JSR MODE 3 FAILED
3804  014056  005237  003032            MJSR3B: INC     @#SEQ                   ;UPDATE SEQUENCE COUNTER
3805  014062  013706  003034            MOV     @#SPS,R6                ;RELOAD STACK POINTER
3806  014066  012704  052525            MOV     #052525,R4              ;SETUP R4 DATA
3807  014072  012701  013736            MOV     #MJSR4+2,R1             ;SETUP JSR VECTOR
3808  014076  000257                    CCC                             ;CLEAR CONDITION CODES
3809  014100  004441                    JSR     R4,-(R1)                        ;JSR MODE 4
3810                                    ;
3811                                    ;
3812  014102  022737  000006  003032    MJSR6:  CMP     #6,@#SEQ                ; TEST FOR CORRECT SEQUENCE
3813  014110  001401                    BEQ     MJSR6A                  ;BRANCH IF GOOD
3814  014112  104001                    13$:    ERROR   +1              ;CPU ERROR
3815                                                                    ;MODE 6 JUMPED TO OUT OF SEQUENCE
3816  014114  023706  003036            MJSR6A: CMP     @#SPSJ,R6               ;VERIFY STACK DECRIMENT
3817  014120  001006                    BNE     14$                     ;BRANCH IF STACK INCORRECT
3818  014122  021627  123456            CMP     (R6),#123456            ;VERIFY CONTENTS OF STACK
3819  014126  001003                    BNE     14$                     ;BRANCH IF DATA ON STACK INCORRECT
3820  014130  022704  014250            CMP     #MJSR7,R4               ;SEE IF CORRECT RETURN ADDRESS
3821  014134  001401                    BEQ     MJSR6B                  ;BRANCH IF GOOD
3822  014136  104001                    14$:    ERROR   +1              ;CPU ERROR
3823                                                                    ;JSR MODE 6 FAILED
```

```
3824  014140  005237  003032          MJSR6B: INC     @#SEQ                   ;UPDATE SEQUENCE COUNTER
3825  014144  013706  003034                  MOV     @#SPS,R6                ;RELOAD STACK POINTER
3826  014150  012704  177773                  MOV     #-5,R4                  ;SETUP R4 DATA
3827  014154  012701  014174                  MOV     #MJSRC+10,R1            ;SETUP JSR VECTOR
3828  014160  004471  177770                  JSR     R4,@-10(R1)             ;JSR MODE 7
3829  014164  014250                  MJSRC:  .WORD   MJSR7                   ;JSR VECTOR
3830                                  ;
3831                                  ;
3832  014166  022737  000005  003032  MJSR5:  CMP     #5,@#SEQ                ; TEST FOR CORRECT SEQUENCE
3833  014174  001401                  MJSR5A  BEQ     MJSR5A                  ;BRANCH IF GOOD
3834  014176  104001                  15$:    ERROR   +1              ;CPU ERROR
3835                                                                  ;MODE 5 JUMPED TO OUT OF SEQUENCE
3836  014200  023706  003036          MJSR5A: CMP     @#SPSJ,R6               ;VERIFY STACK DECRIMENT
3837  014204  001006                  BNE     16$                     ;BRANCH IF STACK INCORRECT
3838  014206  021627  000377                  CMP     (R6),#377               ;VERIFY CONTENTS OF STACK
3839  014212  001003                  BNE     16$                     ;BRANCH IF DATA ON STACK INCORRECT
3840  014214  022704  014014                  CMP     #MJSRB-2,R4             ;SEE IF CORRECT RETURN ADDRESS
3841  014220  001401                  BEQ     MJSR5B                  ;BRANCH IF GOOD
3842  014222  104001                  16$:    ERROR   +1              ;CPU ERROR
3843                                                                  ;JSR MODE 5 FAILED
3844  014224  005237  003032          MJSR5B: INC     @#SEQ                   ;UPDATE SEQUENCE COUNTER
3845  014230  013706  003034                  MOV     @#SPS,R6                ;RELOAD STACK POINTER
3846  014234  012704  123456                  MOV     #123456,R4              ;SETUP DATA IN R4
3847  014240  012701  014112                  MOV     #MJSR6+10,R1            ;SETUP JSR VECTOR
3848  014244  004461  177770                  JSR     R4,-10(R1)              ;JUMP MODE 6
3849                                  ;
3850                                  ;
3851  014250  022737  000007  003032  MJSR7:  CMP     #7,@#SEQ                ; TEST FOR CORRECT SEQUENCE
3852  014256  001401                  MJSR7A  BEQ     MJSR7A                  ;BRANCH IF GOOD
3853  014260  104001                  17$:    ERROR   +1              ;CPU ERROR
3854                                                                  ;MODE 7 JUMPED TO OUT OF SEQUENCE
3855  014262  023706  003036          MJSR7A: CMP     @#SPSJ,R6               ;VERIFY STACK DECRIMENT
3856  014266  001006                  BNE     18$                     ;BRANCH IF STACK INCORRECT
3857  014270  021627  177773                  CMP     (R6),#-5                ;VERIFY CONTENTS OF STAACK
3858  014274  001003                  BNE     18$                     ;BRANCH IF DATA ON STACK INCORRECT
3859  014276  022704  014164                  CMP     #MJSR5-2,R4             ;SEE IF CORRECT RETURN ADDRESS
3860  014302  001401                  BEQ     MJSR7E                  ;BRANCH IF GOOD
3861  014304  104001                  18$:    ERROR   +1              ;CPU ERROR
3862                                                                  ;JSR MODE 7 FAILED
3863  014306                          MJSR7E:
3864  014306  013706  003034                  MOV     @#SPS,R6                ;REPLACE STACK
3865
3866                                  ;
3867  014312                          MJRA:
3868
3869                                  ;       TEST JSR MODES 27, 37, 67, 77
3870  014312  012737  000001  003032          MOV     #1,@#SEQ                ;SETUP SEQUENCER
3871  014320  010637  003034                  MOV     R6,@#SPS                ;SAVE STACK ADDRESS
3872  014324  010637  003036                  MOV     R6,@#SPSJ               ;SAVE STACK DECRIMENT ADDRESS
3873  014330  162737  000002  003036          SUB     #2,@#SPSJ               ;
3874  014336  012704  177777                  MOV     #-1,R4                  ;SETUP R4 DATA
3875  014342  004427  000240                  JSR     R4,#240                 ;EXECUTE A JSR MODE 27
3876                                  ;
3877  014346  022737  000001  003032  MJR27:  CMP     #1,@#SEQ                ;VERIFY COERRECT TEST SEQUENCE
3878  014354  001011                  BNE     1$                      ;INCORRECT TEST SEQUENCE
3879  014356  023706  003036                  CMP     @#SPSJ,R6               ;VERIFY STACK POINTER
```

```
3880  014362  001006                          BNE     1$
3881  014364  021627  177777                  CMP     (R6),#-1                ;VERIFY R4 GOT LOADED ON THE STACK
3882  014370  001003                          BNE     1$                      ;BRANCH IF INCORRECT STACK CONTENTS
3883  014372  020427  014346                  CMP     R4,#MJR27               ;VERIFY CORRECT RETURN ADDRESS
3884  014376  001401                          BEQ     MJR27A                  ;BRANCH IF GOOD RETURN ADDRESS ON STACK
3885  014400  104001                  1$:     ERROR   +1              ;CPU ERROR
3886                                                                  ;MODE 27 FAILED
3887  014402  005237  003032          MJR27A: INC     @#SEQ                   ;UPDATE SEQUENCER
3888  014406  012704  152525                  MOV     #152525,R4              ;SETUP R4 TEST DATA
3889  014412  013706  003034                  MOV     @#SPS,R6                ;RESET STACK POINTER
3890  014416  004437  014504                  JSR     R4,@#MJR37              ;JSR MODE 37
3891  014422  000000          MJR27B: HALT
3892                                  ;
3893  014424  023727  003032  000003  MJR67:  CMP     @#SEQ,#3                ;VERIFY TEST SEQUENCE
3894  014432  001011                          BNE     2$                      ;INCORRECT TEST SEQUENCE
3895  014434  023706  003036                  CMP     @#SPSJ,R6               ;VERIFY STACK DECRIMENT
3896  014440  001006                          BNE     2$                      ;INCORRECT STACK DECRIMENT
3897  014442  021627  000125                  CMP     (R6),#125               ;VERFIFY STACK WAS LOADED
3898  014446  001003                          BNE     2$
3899  014450  020427  014560                  CMP     R4,#MJR77               ;VERIFY RETURN ADDRESS
3900  014454  001401                          BEQ     MJR67A                  ;BRANCH IF GOOD
3901  014456  104001                  2$:     ERROR   +1              ;CPU ERROR
3902                                                                  ;MODE 67 FAILED
3903  014460  005237  003032          MJR67A: INC     @#SEQ                   ;UPDATE SEQUENCER
3904  014464  013706  003034                  MOV     @#SPS,R6                ;RESET STACK
3905  014470  012704  000001                  MOV     #1,R4                   ;SETUP R4 DATA
3906  014474  004477  000002                  JSR     R4,@MJR6B               ;JSR MODE 77
3907  014500  000000          MJR6A:  HALT
3908  014502  014560          MJR6B:  .WORD   MJR77                   ;DATA FOR MODE 77 JUMP
3909                                  ;
3910  014504  023727  003032  000002  MJR37:  CMP     @#SEQ,#2                ;VERIFY TEST SEQUENCE
3911  014512  001011                          BNE     2$                      ;INCORRECT TEST SEQUENCE
3912  014514  023706  003036                  CMP     @#SPSJ,R6               ;VERIFY STACK DECRIMENT
3913  014520  001006                          BNE     2$                      ;INCORRECT STACK DECRIMENT
3914  014522  021627  152525                  CMP     (R6),#152525            ;VERFIFY STACK WAS LOADED
3915  014526  001003                          BNE     2$
3916  014530  020427  014422                  CMP     R4,#MJR27B              ;VERIFY RETURN ADDRESS
3917  014534  001401                          BEQ     MJR37A                  ;BRANCH IF GOOD
3918  014536  104001                  2$:     ERROR   +1              ;CPU ERROR
3919                                                                  ;MODE 37 FAILED
3920  014540  005237  003032          MJR37A: INC     @#SEQ                   ;UPDATE SEQUENCER
3921  014544  013706  003034                  MOV     @#SPS,R6                ;RELOAD STACK
3922  014550  012704  000125                  MOV     #125,R4                 ;SETUP R4 TEST DATA
3923  014554  004467  177644                  JSR     R4,MJR67                ;JSR MODE 6
3924                                  ;
3925  014560  023727  003032  000004  MJR77:  CMP     @#SEQ,#4                ;VERIFY TEST SEQUENCE
3926  014566  001011                          BNE     2$                      ;INCORRECT TEST SEQUENCE
3927  014570  023706  003036                  CMP     @#SPSJ,R6               ;VERIFY STACK DECRIMENT
3928  014574  001006                          BNE     2$                      ;INCORRECT STACK DECRIMENT
3929  014576  021627  000001                  CMP     (R6),#1                 ;VERFIFY STACK WAS LOADED
3930  014602  001003                          BNE     2$
3931  014604  020427  014500                  CMP     R4,#MJR6A               ;VERIFY RETURN ADDRESS
3932  014610  001401                          BEQ     MJR77A                  ;BRANCH IF GOOD
3933  014612  104001                  2$:     ERROR   +1              ;CPU ERROR
3934                                                                  ;MODE 77 FAILED
3935  014614          MJR77A:
```

```
3936                                    ;
3937  014614  013706  003034                    MOV     @#SPS,R6                ;RESET STACK
3938
3939                                    ;
3940  014620                            MRTS:
3941
3942                                    ;       TEST RTS AND RTS R6
3943  014620  012706  001000                    MOV     #STBOT,R6               ;INSURE VALID STACK
3944  014624  012746  123456                    MOV     #123456,-(R6)           ;SETUP TEST REGISTER
3945  014630  012703  014640                    MOV     #RTS1,R3                ;SETUP TEST PC
3946  014634  000203                            RTS     R3                      ;**TEST INSTRUCTION
3947  014636  104001                            ERROR   .1              ;CPU ERROR
3948                                                                    ;INCORRECT PC ON RTS
3949  014640  022703  123456           RTS1:   CMP     #123456,R3
3950  014644  001401                            BEQ     RTS6                    ;BRANCH IF GOOD
3951  014646  104001                            ERROR   .1              ;CPU ERROR
3952                                                                    ;REGISTER CONTENTS INCORRECT
3953                                    ;
3954                                    ;THIS TEST CHECKS AN UN-TESTED PLA TERM
3955                                    ;
3956  014650  010601           RTS6:   MOV     R6,R1                   ;SAVE STACK IN R1
3957  014652  012705  014664            MOV     #1$,R5                  ;MOVE EXPECTED RETURN ADDR TO R5
3958  014656  010506                    MOV     R5,R6                   ;MOVE RETURN ADDR TO R6
3959  014660  000206                    RTS     R6                      ;
3960  014662  104001                    ERROR   .1              ;CPU ERROR
3961                                                                    ;ERROR! RTS NOT EXECUTED
3962  014664  021506           1$:     CMP     (R5),R6                 ;IS R6=31506?
3963  014666  001401                    BEQ     RTSE                    ;IF IT IS THEN GO TO END OF TEST
3964  014670  104001                    ERROR   .1              ;CPU ERROR
3965                                                                    ;ERROR! WRONG ADDR IN R6
3966  014672  010106           RTSE:   MOV     R1,R6                   ;RESTORE STACK
3967
3968  014674                            TSMMUO:
3969                                    ;       SETUP AND TEST KERNEL, SUPERVISOR AND USER STACKS
3970  014674  012737  040000  177776            MOV     #40000,@#177776         ;SET PS TO SUP MODE
3971  014702  012706  177777                    MOV     #177777,R6              ;INIT SUP STACK TO ALL ONES
3972  014706  022706  177777                    CMP     #177777,R6              ;ARE ALL BITS SET
3973  014712  001401                            BEQ     1$                      ;YES GO ON
3974                                                                            ;NO GO TO ERROR
3975  014714  104001                            ERROR   .1              ;CPU ERROR
3976  014716  005006           1$:     CLR     R6                      ;SET SUP STACK TO ALL ZEROES
3977  014720  022706  000000                    CMP     #0,R6                   ;ARE ALL BITS CLEARED
3978  014724  001401                            BEQ     2$                      ;YES GO ON
3979                                                                            ;NO GO TO ERROR
3980  014726  104001                            ERROR   .1              ;CPU ERROR
3981  014730  012706  125252           2$:     MOV     #125252,R6              ;SET SUP STACK TO ALTERNATING PATTERN
3982  014734  022706  125252                    CMP     #125252,R6              ;IS SUP SP CORRECT
3983  014740  001401                            BEQ     3$                      ;YES GO ON
3984                                                                            ;NO GO TO ERROR
3985  014742  104001                            ERROR   .1              ;CPU ERROR
3986  014744  012706  052525           3$:     MOV     #52525,R6               ;SET SUP STACK TO ALTERNATING PATTERN
3987  014750  022706  052525                    CMP     #52525,R6               ;IS SUP SP CORRECT
3988  014754  001401                            BEQ     4$                      ;YES GO ON
3989                                                                            ;NO GO TO ERROR
3990  014756  104001                            ERROR   .1              ;CPU ERROR
3991  014760  012706  000700           4$:     MOV     #700,R6                 ;SETUP SUP SP
```

```
3992  014764  012737  140000  177776          MOV     #140000,@#177776        ;SET PS TO USER MODE
3993  014772  012706  177777                  MOV     #177777,R6              ;INIT USER STACK TO ALL ONES
3994  014776  022706  177777                  CMP     #177777,R6              ;ARE ALL BITS SET
3995  015002  001401                          BEQ     5$                      ;YES GO ON
3996                                                                           ;NO GO TO ERROR
3997  015004  104001                          ERROR   .+1             ;CPU ERROR
3998  015006  005006                  5$:     CLR     R6                      ;SET USER STACK TO ALL ZEROES
3999  015010  022706  000000                  CMP     #0,R6                   ;ARE ALL BITS CLEARED
4000  015014  001401                          BEQ     6$                      ;YES GO ON
4001                                                                           ;NO GO TO ERROR
4002  015016  104001                          ERROR   .+1             ;CPU ERROR
4003  015020  012706  125252          6$:     MOV     #125252,R6              ;SET USER STACK TO ALTERNATING PATTERN
4004  015024  022706  125252                  CMP     #125252,R6              ;IS USER SP CORRECT
4005  015030  001401                          BEQ     7$                      ;YES GO ON
4006                                                                           ;NO GO TO ERROR
4007  015032  104001                          ERROR   .+1             ;CPU ERROR
4008  015034  012706  052525          7$:     MOV     #52525,R6               ;SET USER STACK TO ALTERNATING PATTERN
4009  015040  022706  052525                  CMP     #52525,R6               ;IS USER SP CORRECT
4010  015044  001401                          BEQ     8$                      ;YES GO ON
4011                                                                           ;NO GO TO ERROR
4012  015046  104001                          ERROR   .+1             ;CPU ERROR
4013  015050  012706  000600          8$:     MOV     #600,R6                 ;SETUP USER SP
4014  015054  005037  177776                  CLR     @#177776                ;SET PS TO KER MODE
4015  015060  012706  001000                  MOV     #STBOT,R6               ;SETUP KERNEL SP
4016  015064  005037  000700                  CLR     @#700                   ;CLEAR FIRST WORDS OF SUP, KER, AND USE STACKS
4017  015070  005037  000600                  CLR     @#600                   ;
4018  015074  005037  001000                  CLR     @#STBOT                 ;
4019  015100  004767  000054                  JSR     PC,CHECK                ; TEST KER, SUP, AND USE STACKS
4020  015104  012737  040000  177776  RET1:   MOV     #40000,@#177776         ;SET PSW TO SUP MODE
4021  015112  022706  000700                  CMP     #700,R6                 ;IS SUP SP CORRECT
4022  015116  001401                          BEQ     1$                      ;YES GO ON
4023                                                                           ;NO GO TO ERROR
4024  015120  104001                          ERROR   .+1             ;CPU ERROR
4025  015122  012737  140000  177776  1$:     MOV     #140000,@#177776        ;SET PSW TO USE MODE
4026  015130  022706  000600                  CMP     #600,R6                 ;IS USE SP CORRECT
4027  015134  001401                          BEQ     2$                      ;YES GO ON
4028                                                                           ;NO GO TO ERROR
4029  015136  104001                          ERROR   .+1             ;CPU ERROR
4030  015140  005037  177776          2$:     CLR     @#177776                ;SET PSW TO KER MODE
4031  015144  022706  001000                  CMP     #STBOT,R6               ;IS KER SP CORRECT
4032  015150  001401                          BEQ     3$                      ;YES GO ON
4033                                                                           ;NO GO TO ERROR
4034  015152  104001                          ERROR   .+1             ;CPU ERROR
4035  015154                          3$:
4036  015154  000167  000206                  JMP     MTS0
4037                                  ;
4038                                  ;ROUTINE TO CHECK STACKS AFTER TWO RTS STATEMENTS
4039                                  ;
4040  015160  012737  040000  177776  CHECK:  MOV     #40000,@#177776         ;SET PSW TO SUP MODE
4041  015166  004767  000044                  JSR     PC,CHECK1               ; TEST SUP, KER, AND USE STACKS
4042  015172  022716  000000          RET2:   CMP     #0,(SP)                 ;IS SUP STACK CLEARED
4043  015176  001401                          BEQ     1$                      ;YES GO ON
4044                                                                           ;NO GO TO ERROR
4045  015200  104001                          ERROR   .+1             ;CPU ERROR
4046  015202  012737  140000  177776  1$:     MOV     #140000,@#177776        ;SET PSW TO USE MODE
4047  015210  022716  000000                  CMP     #0,(SP)                 ;IS USE STACK CLEARED
```

```
4048  015214  001401                        BEQ     2$                      ;YES GO ON
4049                                                                          ;NO GO TO ERROR
4050  015216  104001                        ERROR   .1              ;CPU ERROR
4051  015220  005037  177776        2$:     CLR     @#177776                ;SET PSW TO KER MODE
4052  015224  022716  015104                CMP     #RET1,(SP)              ;DOES KER STACK HAVE CORRECT DATA
4053  015230  001401                        BEQ     3$                      ;YES GO ON
4054                                                                          ;NO GO TO ERROR
4055  015232  104001                        ERROR   .1              ;CPU ERROR
4056  015234  000207            3$:         RTS     PC                      ;RETURN
4057                                        ;
4058                                        ;ROUTINE TO CHECK STACKS AFTER ONE RTS
4059                                        ;
4060  015236  012737  140000  177776  CHECK1: MOV   #140000,@#177776       ;SET PSW TO USE MODE
4061  015244  004767  000044                JSR     PC,CHECK2               ; TEST KER, SUP, AND USE STACKS
4062  015250  022716  000000        RET3:   CMP     #0,(SP)                 ;IS USE STACK CLEARED
4063  015254  001401                        BEQ     1$                      ;YES GO ON
4064                                                                          ;NO GO TO ERROR
4065  015256  104001                        ERROR   .1              ;CPU ERROR
4066  015260  005037  177776        1$:     CLR     @#177776                ;SET PSW TO KER MODE
4067  015264  022716  015104                CMP     #RET1,(SP)              ;IS KER STACK CORRECT
4068  015270  001401                        BEQ     2$                      ;YES GO ON
4069                                                                          ;NO GO TO ERROR
4070  015272  104001                        ERROR   .1              ;CPU ERROR
4071  015274  012737  040000  177776  2$:     MOV   #40000,@#177776         ;SET PSW TO SUP MODE
4072  015302  022716  015172                CMP     #RET2,(SP)              ;IS SUP STACK CORRECT
4073  015306  001401                        BEQ     3$                      ;YES GO ON
4074                                                                          ;NO GO TO ERROR
4075  015310  104001                        ERROR   .1              ;CPU ERROR
4076  015312  000207            3$:         RTS     PC                      ;RETURN
4077                                        ;
4078                                        ;ROUTINE TO CHECK STACKS AFTER ZERO RTS
4079                                        ;
4080  015314  022716  015250        CHECK2: CMP     #RET3,(SP)              ;IS USE STACK CORRECT
4081  015320  001401                        BEQ     1$                      ;YES GO ON
4082                                                                          ;NO GO TO ERROR
4083  015322  104001                        ERROR   .1              ;CPU ERROR
4084  015324  012737  040000  177776  1$:     MOV   #40000,@#177776         ;SET PSW TO SUP MODE
4085  015332  022716  015172                CMP     #RET2,(SP)              ;IS SUP STACK CORRECT
4086  015336  001401                        BEQ     2$                      ;YES GO ON
4087                                                                          ;NO GO TO ERROR
4088  015340  104001                        ERROR   .1              ;CPU ERROR
4089  015342  005037  177776        2$:     CLR     @#177776                ;SET PSW TO KER MODE
4090  015346  022716  015104                CMP     #RET1,(SP)              ;IS KER STACK CORRECT
4091  015352  001401                        BEQ     3$                      ;YES GO ON
4092                                                                          ;NO GO TO ERROR
4093  015354  104001                        ERROR   .1              ;CPU ERROR
4094  015356  012737  140000  177776  3$:     MOV   #140000,@#177776        ;SET PSW TO USE MODE
4095  015364  000207                        RTS     PC                      ;RETURN
4096                                        ;
4097  015366                        MTSO:
4098  015366                        MMVCC:
4099
4100                                        ;       TEST MOV CONDITION CODES - **0
4101  015366  000277                        SCC                             ;
4102  015370  000244                        CLZ                             ;CC=1011
4103  015372  012704  000000                MOV     #0,R4                   ;CC=0101, R4=0
```

```
4104  015376  100403                  BMI     1$                      ;ERROR IF N FLAG
4105  015400  102402                  BVS     1$                      ;ERROR IF V FLAG SET
4106  015402  103001                  BCC     1$                      ;ERROR IF C FLAG CLEAR
4107  015404  001401                  BEQ     2$                      ;SKIP IF Z FLAG SET
4108                                                                  ;CC SHOULD=0101
4109  015406  104001          1$:     ERROR   .1              ;CPU ERROR
4110  015410  000277          2$:     SCC
4111  015412  000251                  .WORD   251                     ;CC=0110
4112  015414  012704  100000          MOV     #100000,R4              ;R4=100000, CC=1000
4113  015420  001403                  BEQ     3$                      ;ERROR IF Z SET
4114  015422  102402                  BVS     3$                      ;ERROR IF V SET
4115  015424  103401                  BCS     3$                      ;ERROR IF C SET
4116  015426  100401                  BMI     4$                      ;EXIT IF N SET
4117  015430  104001          3$:     ERROR   .1              ;CPU ERROR
4118                                                          ;CC SHOULD= 1000
4119  015432                  4$:
4120
4121
4122  015432                  MBTCC:
4123
4124                          ;       TEST BIT CONDITION CODES - **0-
4125  015432  005004                  CLR     R4
4126  015434  005104                  COM     R4                      ;R4=-1
4127  015436  000277                  SCC                             ;
4128  015440  000244                  CLZ                             ;CC=1011
4129  015442  032704  000000          BIT     #0,R4                   ;CC=0101
4130  015446  100403                  BMI     1$                      ;ERROR IF N FLAG
4131  015450  102402                  BVS     1$                      ;ERROR IF V FLAG SET
4132  015452  103001                  BCC     1$                      ;ERROR IF C FLAG CLEAR
4133  015454  001401                  BEQ     2$                      ;SKIP IF Z FLAG SET
4134  015456  104001          1$:     ERROR   .1              ;CPU ERROR
4135                                                          ;CC SHOULD=0101
4136  015460  000277          2$:     SCC
4137  015462  000251                  .WORD   251                     ;CC=0110
4138  015464  032704  100000          BIT     #100000,R4              ;CC=1000
4139  015470  001403                  BEQ     3$                      ;ERROR IF Z SET
4140  015472  102402                  BVS     3$                      ;ERROR IF V SET
4141  015474  103401                  BCS     3$                      ;ERROR IF C SET
4142  015476  100401                  BMI     4$                      ;EXIT IF N SET
4143  015500  104001          3$:     ERROR   .1              ;CPU ERROR
4144                                                          ;CC SHOULD= 1000
4145  015502                  4$:
4146
4147
4148  015502                  MBCCC:
4149
4150                          ;       TEST BIC CONDITION CODES - **0-
4151  015502  005004                  CLR     R4
4152  015504  005104                  COM     R4                      ;R4=-1
4153  015506  000277                  SCC                             ;
4154  015510  000244                  CLZ                             ;CC=1011
4155  015512  042704  177777          BIC     #177777,R4              ;CC=0101
4156  015516  100403                  BMI     1$                      ;ERROR IF N FLAG
4157  015520  102402                  BVS     1$                      ;ERROR IF V FLAG SET
4158  015522  103001                  BCC     1$                      ;ERROR IF C FLAG CLEAR
4159  015524  001401                  BEQ     2$                      ;SKIP IF Z FLAG SET
```

```
4160  015526  104001                  1$:     ERROR   ·1              ;CPU ERROR
4161                                                                  ;CC SHOULD=0101
4162  015530  005104                  2$:     COM     R4                      ;R4=-1
4163  015532  000277                          SCC
4164  015534  000251                          .WORD   251                     ;CC=0110
4165  015536  042704  077777                  BIC     #77777,R4               ;CC=1000
4166  015542  001403                          BEQ     3$                      ;ERROR IF Z SET
4167  015544  102402                          BVS     3$                      ;ERROR IF V SET
4168  015546  103401                          BCS     3$                      ;ERROR IF C SET
4169  015550  100401                          BMI     4$                      ;EXIT IF N SET
4170  015552  104001                  3$:     ERROR   ·1              ;CPU ERROR
4171                                                                  ;CC SHOULD= 1000
4172  015554                          4$:
4173
4174
4175  015554                          MBSCC:
4176
4177                                  ;       TEST BIS CONDITION CODES
4178  015554  005004                          CLR     R4                      ;R4=0
4179  015556  000277                          SCC
4180  015560  000246                          .WORD   246                     ;CC=1001
4181  015562  052704  000000                  BIS     #0,R4                   ;R4=0, CC=0101
4182  015566  100403                          BMI     1$                      ;ERROR IF MINUS
4183  015570  102402                          BVS     1$                      ;ERROR IF V SET
4184  015572  103001                          BCC     1$                      ;ERROR IF C CLEAR
4185  015574  001401                          BEQ     2$                      ;BRANCH IF GOOD
4186  015576  104001                  1$:     ERROR   ·1              ;CPU ERROR
4187                                                                  ;BIS CC FAILED
4188  015600  000277                  2$:     SCC
4189  015602  000241                          CLC                             ;CC=1110
4190  015604  052704  100076                  BIS     #100076,R4              ;R4=100076, CC=1000
4191  015610  001403                          BEQ     3$                      ;ERROR IF Z SET
4192  015612  102402                          BVS     3$                      ;ERROR IF V SET
4193  015614  103401                          BCS     3$                      ;ERROR IF C SET
4194  015616  100401                          BMI     4$                      ;BRANCH IF GOOD
4195  015620  104001                  3$:     ERROR   ·1              ;CPU ERROR
4196                                                                  ;BAD BIS CC
4197  015622                          4$:
4198
4199
4200  015622                          MDCCC:
4201
4202                                  ;       TEST DEC, INC CONDITION CODES
4203  015622  012704  077777                  MOV     #77777,R4               ;R4=77777
4204  015626  000257                          CCC
4205  015630  000261                          SEC                             ;CC=0001
4206  015632  005204                          INC     R4                      ;R4=100000, CC=0011
4207  015634  001403                          BEQ     1$                      ;ERROR IF ZERO
4208  015636  100002                          BPL     1$                      ;ERROR IF POSITIVE
4209  015640  102001                          BVC     1$                      ;ERROR IF V CLEAR
4210  015642  103401                          BCS     2$                      ;BRANCH IF GOOD
4211  015644  104001                  1$:     ERROR   ·1              ;CPU ERROR
4212                                                                  ;INC FAILED
4213  015646  000257                  2$:     CCC
4214  015650  005204                          INC     R4                      ;R4=100001, CC=1000
4215  015652  103413                          BCS     3$                      ;ERROR IF C SET
```

```
4216  015654  102412                  BVS     3$                      ;ERROR IF V SET
4217  015656  005304                  DEC     R4                      ;R4=100000, CC=1000
4218  015660  102410                  BVS     3$                      ;ERROR IF V SET
4219  015662  103407                  BCS     3$                      ;ERROR IF C SET
4220  015664  000277                  SCC                             ;
4221  015666  000252                  .WORD   252                     ;CC=0101
4222  015670  005304                  DEC     R4                      ;R4=77777, CC=1011
4223  015672  001403                  BEQ     3$                      ;ERROR IF Z SET
4224  015674  102002                  BVC     3$                      ;ERROR IF V CLEAR
4225  015676  103001                  BCC     3$                      ;ERROR IF C CLEAR
4226  015700  100001                  BPL     4$                      ;BRANCH IF GOOD
4227  015702  104001          3$:     ERROR   ;1              ;CPU ERROR
4228                                                          ;BAD CC
4229  015704                  4$:
4230
4231
4232  015704                  MCTSCC:
4233
4234                          ;       TEST CLR, TST, SWAB CONDITION CODES
4235                          ;;**********************************************************
4236                          ;0100 - **00 - **00
4237  015704  000277                  SCC
4238  015706  000244                  CLZ                             ;CC=1011
4239  015710  005004                  CLR     R4                      ;R4=0, CC=0100
4240  015712  100403                  BMI     1$                      ;ERROR IF MINUS
4241  015714  102402                  BVS     1$                              ;ERROR IF V SET
4242  015716  103401                  BCS     1$                      ;ERROR IF C SET
4243  015720  001401                  BEQ     2$                      ;BRANCH IF GOOD
4244  015722  104001          1$:     ERROR   ;1              ;CPU ERROR
4245                                                          ;BAD CC ON CLR
4246  015724  005104          2$:     COM     R4                      ;R4=-1
4247  015726  000277                  SCC                             ;CC=1111
4248  015730  005704                  TST     R4                      ;CC=1000
4249  015732  001403                  BEQ     3$                      ;ERROR IF Z SET
4250  015734  102402                  BVS     3$                      ;ERROR IF V SET
4251  015736  103401                  BCS     3$                      ;ERROR IF C SET
4252  015740  100401                  BMI     4$                      ;BRANCH IF GOOD
4253  015742  104001          3$:     ERROR   ;1              ;CPU ERROR
4254                                                          ;BAD TST CC
4255  015744  000277          4$:     SCC                             ;CC=1111
4256  015746  000304                  SWAB    R4                      ;CC=1000
4257  015750  102402                  BVS     5$                      ;ERROR IF V SET
4258  015752  103401                  BCS     5$                      ;ERROR IF C SET
4259  015754  100401                  BMI     6$                      ;BRANCH IF GOOD
4260  015756  104001          5$:     ERROR   ;1              ;CPU ERROR
4261                                                          ;BAD SWAB CC
4262  015760                  6$:
4263
4264
4265  015760                  MADCC:
4266
4267                          ;       TEST ADD CONDITION CODES
4268  015760  012704  077777          MOV     #77777,R4               ;R4=77777
4269  015764  012701  000001          MOV     #1,R1                   ;R1=1
4270  015770  000257                  CCC                             ;CC=0000
4271  015772  060401                  ADD     R4,R1                   ;77777 + 1 = 100000 IN R1
```

```
4272  015774  102003                  BVC     1$                      ;ERROR IF V CLEAR
4273  015776  103402                  BCS     1$                      ;ERROR IF CARRY
4274  016000  001401                  BEQ     1$                      ;ERROR IF Z SET
4275  016002  100401                  BMI     2$                      ;BRANCH IF GOOD
4276  016004  104001          1$:     ERROR   ·1              ;CPU ERROR
4277                                                        ;CC SHOULD ·1010
4278  016006  005204          2$:     INC     R4                      ;R4·100000
4279  016010  060401                  ADD     R4,R1                   ;100000 + 100000 = 0 IN R1
4280  016012  102002                  BVC     3$                      ;ERROR IF V CLEAR
4281  016014  103001                  BCC     3$                      ;ERROR IF CARRY  CLEAR
4282  016016  001401                  BEQ     4$                      ;BRANCH IF GOOD
4283  016020  104001          3$:     ERROR   ·1              ;CPU ERROR
4284                                                        ;CC SHOULD = 0111
4285  016022  060401          4$:     ADD     R4,R1                   ;0 + 100000 = 100000
4286  016024  102402                  BVS     5$                      ;ERROR IF V SET
4287  016026  103401                  BCS     5$                      ;ERROR IF  SET
4288  016030  100401                  BMI     6$                      ;BRANCH IF GOOD
4289  016032  104001          5$:     ERROR   ·1              ;CPU ERROR
4290                                                        ;CC SHOULD = 1000
4291  016034          6$:
4292
4293
4294  016034          MACCC:
4295
4296                  ;       TEST ADC CONDITION CODES
4297  016034  012704  177777          MOV     #177777,R4              ;R4·177777
4298  016040  000277                  SCC                             ;CC·1111
4299  016042  005504                  ADC     R4                      ·R4·0 CC·0101
4300  016044  100403                  BMI     1$                      ;ERROR IF MINUS
4301  016046  102402                  BVS     1$                      ;ERROR IF V SET
4302  016050  103001                  BCC     1$                      ;ERROR IF C SET
4303  016052  001401                  BEQ     2$                      ;BRANCH IF GOOD
4304  016054  104001          1$:     ERROR   ·1              ;CPU ERROR
4305                                                        ;BAD ADC
4306  016056  012704  077777  2$:     MOV     #077777,R4              ;R4·077777
4307  016062  000277                  SCC                             ;
4308  016064  000242                  CLV                             ;CC·1101
4309  016066  005504                  ADC     R4                      ;R4·100000 CC·1010
4310  016070  100003                  BPL     3$                      ;ERROR IF PLUS
4311  016072  103402                  BCS     3$                      ;ERROR IF C SET
4312  016074  001401                  BEQ     3$                      ;ERROR IF ZERO
4313  016076  102401                  BVS     4$                      ;BRANCH IF GOOD
4314  016100  104001          3$:     ERROR   ·1              ;CPU ERROR
4315                                                        ;BAD ADC
4316  016102  000277          4$:     SCC                             ;CC·1111
4317  016104  005504                  ADC     R4                      ;R4·100000 CC·1000
4318  016106  102402                  BVS     5$                      ;ERROR IF V SET
4319  016110  103401                  BCS     5$                      ;ERROR IF C SET
4320  016112  100401                  BMI     6$                      ;BRANCH IF GOOD
4321  016114  104001          5$:     ERROR   ·1              ;CPU ERROR
4322                                                        ;BAD ADC CC SHOULD· 1000
4323  016116          6$:
4324
4325
4326  016116          MNCCCC:
4327
```

```
4328                                    ;       TEST NEG, CMP, COM CONDITION CODES
4329  016116  012704  077777                    MOV     #077777,R4              ;R4=77777
4330  016122  000257                            CCC                             ;CC=0000
4331  016124  005404                            NEG     R4                      ;R4=100001      CC=1001
4332  016126  102403                            BVS     1$                      ;ERROR IF V SET
4333  016130  103002                            BCC     1$                      ;ERROR IF C CLEAR
4334  016132  001401                            BEQ     1$                      ;ERROR IF Z SET
4335  016134  100401                            BMI     2$                      ;BRANCH IF GOOD
4336  016136  104001                1$:         ERROR   .1              ;CPU ERROR
4337                                                                    ;BAD NEGATE
4338  016140  005004                2$:         CLR     R4                      ;R4=0
4339  016142  000257                            CCC                             ;CC=0000
4340  016144  005404                            NEG     R4                      ;CC=0101
4341  016146  100403                            BMI     3$                      ;ERROR IF N SET
4342  016150  103402                            BCS     3$                      ;ERROR IF C SET
4343  016152  102401                            BVS     3$                      ;ERROR IF V SET
4344  016154  001401                            BEQ     4$                      ;BRANCH IF GOOD
4345  016156  104001                3$:         ERROR   .1              ;CPU ERROR
4346                                                                    ;BAD NEG
4347  016160  012704  077777        4$:         MOV     #77777,R4               ;R4=77777
4348  016164  012701  170000                    MOV     #170000,R1              ;R1=170000
4349  016170  000257                            CCC                             ;CC=0000
4350  016172  020401                            CMP     R4,R1                   ;77777 - 170000 = 107777 CC= 1011
4351  016174  102003                            BVC     5$                      ;ERROR IF V CLEAR
4352  016176  103002                            BCC     5$                      ;ERROR IF C CLEAR
4353  016200  001401                            BEQ     5$                      ;ERROR IF ZERO
4354  016202  100401                            BMI     6$                      ;BRANCH IF GOOD
4355  016204  104001                5$:         ERROR   .1              ;CPU ERROR
4356                                                                    ;BAD CMP
4357  016206  000257                6$:         CCC
4358  016210  005101                            COM     R1                      ;R1=7777
4359  016212  100403                            BMI     7$                      ;ERROR IF MINUS
4360  016214  001402                            BEQ     7$                      ;ERROR IF ZERO
4361  016216  103001                            BCC     7$                      ;ERROR IF CARRY
4362  016220  102001                            BVC     8$                      ;BRANCH IF GOOD
4363  016222  104001                7$:         ERROR   .1              ;CPU ERROR
4364                                                                    ;BAD COM
4365  016224  000277                8$:         SCC
4366  016226  005101                            COM     R1
4367  016230  100401                            BMI     10$                     ;BRANCH IF GOOD
4368  016232  104001                9$:         ERROR   .1              ;CPU ERROR
4369                                                                    ;BAD COM
4370  016234                        10$:
4371
4372
4373  016234                        MSBCC:
4374
4375                                    ;       TEST SUB CONDITION CODES
4376  016234  012704  077775                    MOV     #77775,R4               ;R4=77775
4377  016240  000257                            CCC                             ;CC=0000
4378  016242  162704  137757                    SUB     #137757,R4              ;77775 - 137757
4379                                                                    ;TRY TO CAUSE AN ARITHMETIC OVERFLOW
4380  016246  102003                            BVC     1$                      ;ERROR IF V CLEAR
4381  016250  100002                            BPL     1$                      ;ERROR IF RESULT IS POSITIVE
4382  016252  001401                            BEQ     1$                      ;ERROR IF  Z SET
4383  016254  103401                            BCS     2$                      ;BRANCH IF GOOD
```

```
4384  016256  104001                  1$:     ERROR   ;1              ;CPU ERROR
4385                                                                  ;BAD SUBTRACT
4386  016260  012704  000005          2$:     MOV     #5,R4                   ;R4=5
4387  016264  000257                          CCC                             ;CC=0000
4388  016266  162704  000012                  SUB     #12,R4                  ;5-12=-5 AND SETS CARRY
4389  016272  103003                          BCC     3$                      ;ERROR IF CARRY CLEAR
4390  016274  102402                          BVS     3$                      ;ERROR IF OVERFLOW
4391  016276  001401                          BEQ     3$                      ;ERROR IF ZERO
4392  016300  100401                          BMI     4$                      ;BRANCH IF GOOD
4393  016302  104001                  3$:     ERROR   ;1              ;CPU ERROR
4394                                                                  ;SUBTRACT FAILED
4395  016304                          4$:
4396
4397
4398  016304                          MSBCCC:
4399
4400                                  ;       TEST SBC CONDITION CODES
4401  016304  012704  100000                  MOV     #100000,R4              ;R4=100000
4402  016310  000257                          CCC                             ;C=0000
4403  016312  005604                          SBC     R4                      ;TRY TO SET V
4404  016314  100006                          BPL     1$                      ;ERROR IF N CLEAR
4405  016316  102405                          BVS     1$                      ;ERROR IF V SET (HAVENT SET C YET)
4406  016320  000261                          SEC                             ;CC SHOULD = 1001
4407  016322  005604                          SBC     R4                      ;TRY AGAIN TO SET V
4408  016324  102002                          BVC     1$                      ;ERROR IF V CLEAR
4409  016326  103401                          BCS     1$                      ;ERROR IF C SET
4410  016330  100001                          BPL     2$                      ;BRANCH IF GOOD
4411  016332  104001                  1$:     ERROR   ;1              ;CPU ERROR
4412                                                                  ;SBC FAILED
4413  016334  005004                  2$:     CLR     R4                      ;R4=0
4414  016336  000277                          SCC
4415  016340  000241                          CLC                             ;CC=1110
4416  016342  005604                          SBC     R4                      ;TRY TO CAUSE C FLAG FAILURE
4417  016344  103410                          BCS     3$                      ;ERROR IF C SET
4418  016346  102407                          BVS     3$                      ;ERROR IF V SET
4419  016350  001006                          BNE     3$                      ;ERROR IF  NOT ZERO
4420  016352  000261                          SEC                             ;SET CARRY
4421  016354  005604                          SBC     R4                      ;NOW, 0 - CARRY = 177777
4422  016356  103003                          BCC     3$                      ;ERROR IF CARRY CLEAR
4423  016360  102402                          BVS     3$                      ;ERROR IF V SET
4424  016362  001401                          BEQ     3$                      ;ERROR IF ZERO
4425  016364  100401                          BMI     4$                      ;BRANCH IF GOOD
4426  016366  104001                  3$:     ERROR   ;1              ;CPU ERROR
4427                                                                  ;SBC FAILED
4428  016370                          4$:
4429
4430
4431  016370                          MRLCC:
4432
4433                                  ;       TEST ROL CONDITION CODES
4434  016370  012704  060000                  MOV     #60000,R4               ;R4= 0110000000000000
4435  016374  000257                          CCC                             ;CC=0000
4436  016376  006104                          ROL     R4                      ;R4= 1100000000000000
4437  016400  103402                          BCS     1$                      ;ERROR IF CARRY
4438  016402  102001                          BVC     1$                      ;ERROR IF V CLEAR
4439  016404  100401                          BMI     2$                      ;BRANCH IF GOOD
```

```
4440  016406  104001              1$:     ERROR   ;1              ;CPU ERROR
4441                                                              ;ROL  FAILED
4442  016410  006104              2$:     ROL     R4                      ;R4= 1000000000000000
4443  016412  103002                      BCC     3$                      ;ERROR IF CARRY CLEAR
4444  016414  102401                      BVS     3$                      ;ERROR IF V SET
4445  016416  100401                      BMI     4$                      ;BRANCH IF GOOD
4446  016420  104001              3$:     ERROR   ;1              ;CPU ERROR
4447                                                              ;BAD ROL
4448  016422  006104              4$:     ROL     R4                      ;R4 = 0000000000000001
4449  016424  102003                      BVC     5$                      ;ERROR IF V CLEAR
4450  016426  103002                      BCC     5$                      ;ERROR IF C CLEAR
4451  016430  100401                      BMI     5$                      ;ERROR IF MINUS
4452  016432  001001                      BNE     6$                      ;BRANCH IF GOOD
4453  016434  104001              5$:     ERROR   ;1              ;CPU ERROR
4454                                                              ;BAD ROL
4455  016436  006104              6$:     ROL     R4                      ;R4=000000000000011
4456  016440  102402                      BVS     7$                      ;ERROR IF V SET
4457  016442  103401                      BCS     7$                      ;ERROR IF C SET
4458  016444  100001                      BPL     8$                      ;BRANCH IF GOOD
4459  016446  104001              7$:     ERROR   ;1              ;CPU ERROR
4460                                                              ;BAD ROL
4461  016450                      8$:
4462
4463
4464  016450                      MRRCC:
4465
4466                              ;       TEST ROR CONDITION CODES
4467  016450  012704  000003              MOV     #3,R4                   ;R4= 0000000000000011
4468  016454  000257                      CCC                             ;CC= 0000
4469  016456  006004                      ROR     R4                      ;R4= 0000000000000001
4470  016460  103002                      BCC     1$                      ;ERROR IF NO CARRY
4471  016462  102001                      BVC     1$                      ;ERROR IF V CLEAR
4472  016464  100001                      BPL     2$                      ;BRANCH IF GOOD
4473  016466  104001              1$:     ERROR   ;1              ;CPU ERROR
4474                                                              ;ROR  FAILED
4475  016470  006004              2$:     ROR     R4                      ;R4= 1000000000000000
4476  016472  103002                      BCC     3$                      ;ERROR IF CARRY CLEAR
4477  016474  102401                      BVS     3$                      ;ERROR IF V SET
4478  016476  100401                      BMI     4$                      ;BRANCH IF GOOD
4479  016500  104001              3$:     ERROR   ;1              ;CPU ERROR
4480                                                              ;BAD ROR
4481  016502  006004              4$:     ROR     R4                      ;R4 = 1100000000000000
4482  016504  102002                      BVC     5$                      ;ERROR IF V
4483  016506  103401                      BCS     5$                      ;ERROR IF C SET
4484  016510  100401                      BMI     6$                      ;BRANCH IF GOOD
4485  016512  104001              5$:     ERROR   ;1              ;CPU ERROR
4486                                                              ;BAD ROR
4487  016514  006004              6$:     ROR     R4                      ;R4= 0110000000000000
4488  016516  102402                      BVS     7$                      ;ERROR IF V SET
4489  016520  103401                      BCS     7$                      ;ERROR IF C SET
4490  016522  100001                      BPL     8$                      ;BRANCH IF GOOD
4491  016524  104001              7$:     ERROR   ;1              ;CPU ERROR
4492                                                              ;BAD ROR
4493  016526                      8$:
4494
4495
```

```
4496  016526                          XCBIT:
4497
4498                                  ;       TEST C BIT WITH ROR/ROL
4499                                  ;;**************************************************************
4500                                  ;THIS TEST IS TO CHECK FOR A SLOW C BIT PATH INTERNAL TO THE J11 DATA CHIP
4501                                  ;PROBLEM IS ONLY EXHIBITED ON EARLY MASK SETS (1590 AND 1593)
4502  016526  012701  052525          MOV     #52525,R1               ;INIT R1 WITH DATA
4503  016532  000241                  CLC                             ;CLEAR THE C BIT
4504  016534  006001                  ROR     R1                      ;R1=025252, C BIT =1
4505  016536  006001                  ROR     R1                      ;R1=112525, C BIT =0
4506  016540  006001                  ROR     R1                      ;R1=045252, C BIT =1
4507  016542  103401                  BCS     1$                      ;BRANCH IF CARRY BIT SET
4508  016544  104001                  ERROR   +1              ;CPU ERROR
4509  016546  022701  045252  1$:     CMP     #45252,R1               ;IS DATA IN R1 = TO EXPECTED DATA?
4510  016552  001401                  BEQ     2$                      ;BRANCH IF YES
4511  016554  104001                  ERROR   +1              ;CPU ERROR
4512  016556  012701  125252  2$:     MOV     #125252,R1              ;SET UP R1
4513  016562  000241                  CLC                             ;CLEAR THE CARRY BIT
4514  016564  006101                  ROL     R1                      ;R1=052524, C BIT =1
4515  016566  006101                  ROL     R1                      ;R1=125251, C BIT =0
4516  016570  006101                  ROL     R1                      ;R1=052522, C BIT =1
4517  016572  103401                  BCS     3$                      ;BRANCH IF CARRY SET
4518  016574  104001                  ERROR   +1              ;CPU ERROR
4519  016576  022701  052522  3$:     CMP     #052522, R1             ;IS DATA IN R1 = TO EXPECTED DATA?
4520  016602  001401                  BEQ     4$
4521  016604  104001                  ERROR   +1              ;CPU ERROR
4522  016606                          4$:
4523
4524  016606                          MALCC:
4525
4526                                  ;       TEST ASL CONDITION CODES
4527  016606  012704  060000          MOV     #60000,R4               ;R4= 0110000000000000
4528  016612  000257                  CCC                             ;CC=0000
4529  016614  006304                  ASL     R4                      ;C=0 R4= 1100000000000000
4530  016616  103402                  BCS     1$                      ;ERROR IF CARRY
4531  016620  102001                  BVC     1$                      ;ERROR IF V CLEAR
4532  016622  100401                  BMI     2$                      ;BRANCH IF GOOD
4533  016624  104001          1$:     ERROR   +1              ;CPU ERROR
4534                                  ;ASL  FAILED
4535  016626  006304          2$:     ASL     R4                      ;C=1 R4= 1000000000000000
4536  016630  103002                  BCC     3$                      ;ERROR IF CARRY CLEAR
4537  016632  102401                  BVS     3$                      ;ERROR IF V SET
4538  016634  100401                  BMI     4$                      ;BRANCH IF GOOD
4539  016636  104001          3$:     ERROR   +1              ;CPU ERROR
4540                                  ;BAD ASL
4541  016640  006304          4$:     ASL     R4                      ;C=1 R4= 0000000000000000
4542  016642  102003                  BVC     5$                      ;ERROR IF V CLEAR
4543  016644  103002                  BCC     5$                      ;ERROR IF C CLEAR
4544  016646  100401                  BMI     5$                      ;ERROR IF MINUS
4545  016650  001401                  BEQ     6$                      ;BRANCH IF GOOD
4546  016652  104001          5$:     ERROR   +1              ;CPU ERROR
4547                                  ;BAD ASL
4548  016654  006304          6$:     ASL     R4                      ;C=0 R4= 0000000000000000
4549  016656  102402                  BVS     7$                      ;ERROR IF V SET
4550  016660  103401                  BCS     7$                      ;ERROR IF C SET
4551  016662  100001                  BPL     8$                      ;BRANCH IF GOOD
```

```
4552  016664  104001                  7$:     ERROR   ·1              ;CPU ERROR
4553                                                                    ;BAD ASL
4554  016666                          8$:
4555
4556  016666                          MARCC:
4557
4558                                  ;       TEST ASR CONDITION CODES
4559  016666  012704  000341                  MOV     #341,R4                 ;R4= 0000000011100001
4560  016672  000257                          CCC                             ;CC=0000
4561  016674  006204                          ASR     R4                      ;R4= 0000000001110000
4562  016676  103002                          BCC     1$                      ;ERROR IF NO CARRY
4563  016700  102001                          BVC     1$                      ;ERROR IF V CLEAR
4564  016702  100001                          BPL     2$                      ;BRANCH IF GOOD
4565  016704  104001                  1$:     ERROR   ·1              ;CPU ERROR
4566                                                                    ;ASR  FAILED
4567  016706  052704  100001          2$:     BIS     #100001,R4              ;R4= 1000000001110001
4568  016712  006204                          ASR     R4                      ;R4= 1100000000111000
4569  016714  103002                          BCC     3$                      ;ERROR IF CARRY CLEAR
4570  016716  102401                          BVS     3$                      ;ERROR IF V SET
4571  016720  100401                          BMI     4$                      ;BRANCH IF GOOD
4572  016722  104001                  3$:     ERROR   ·1              ;CPU ERROR
4573                                                                    ;BAD ASR
4574  016724  006204                  4$:     ASR     R4                      ;R4= 1110000000011100
4575  016726  102002                          BVC     5$                      ;ERROR IF V
4576  016730  103401                          BCS     5$                      ;ERROR IF C SET
4577  016732  100401                          BMI     6$                      ;BRANCH IF GOOD
4578  016734  104001                  5$:     ERROR   ·1              ;CPU ERROR
4579                                                                    ;BAD ASR
4580  016736  006204                  6$:     ASR     R4                      ;R4= 1111000000001110
4581  016740  102005                          BVC     7$                      ;ERROR IF V CLEAR
4582  016742  103404                          BCS     7$                      ;ERROR IF C SET
4583  016744  100003                          BPL     7$                      ;ERROR IF PLUS
4584  016746  022704  170016                  CMP     #170016,R4              ;SEE IF EXPECTED RESULT
4585  016752  001401                          BEQ     8$                      ;BRANCH IF GOOD
4586  016754  104001                  7$:     ERROR   ·1              ;CPU ERROR
4587                                                                    ;BAD ASR
4588  016756                          8$:
4589
4590
4591  016756                          MSXTCC:
4592
4593                                  ;       TEST SXT CONDITION CODES / -·0-
4594  016756  012704  123456                  MOV     #123456,R4                      ;R4=123456
4595  016762  010401                          MOV     R4,R1                   ;SAVE  CONTENTS
4596  016764  000257                          CCC                             ;CC=0000
4597  016766  006704                          SXT     R4                      ;R4=0 CC=0100
4598  016770  103403                          BCS     1$                      ;ERROR IF CARRY
4599  016772  100402                          BMI     1$                      ;ERROR IF MINUS
4600  016774  102401                          BVS     1$                      ;ERROR IF OVERFLOW
4601  016776  001401                          BEQ     2$                      ;BRANCH IF GOOD
4602  017000  104001                  1$:     ERROR   ·1              ;CPU ERROR
4603                                                                    ;BAD SXT
4604  017002  010104                  2$:     MOV     R1,R4                   ;RESTORE R4
4605  017004  000277                          SCC                             ;CC=1111
4606  017006  006704                          SXT     R4                      ;R4=-1 CC=1001
4607  017010  001405                          BEQ     3$                      ;ERROR IF ZERO
```

```
4608  017012  100004                  BPL     3$                      ;ERROR IF PLUS
4609  017014  103003                  BCC     3$                      ;ERROR IF NO CARRY
4610  017016  102402                  BVS     3$                      ;ERROR IF OVERFLOW
4611  017020  005104                  COM     R4                      ;R4=0
4612  017022  001401                  BEQ     4$                      ;BRANCH IF GOOD
4613  017024  104001          3$:     ERROR   +1              ;CPU ERROR
4614                                                        ;BAD SXT
4615  017026                  4$:
4616
4617
4618  017026                  MXRCC:
4619
4620                          ;       TEST XOR CONDITION CODES / **0-
4621  017026  012704  123456          MOV     #123456,R4              ;R4=123456
4622  017032  012701  052525          MOV     #52525,R1               ;R1=52525
4623  017036  000257                  CCC                             ;CC=0000
4624  017040  074104                  XOR     R1,R4                   ;*TI* R4=171173
4625  017042  102403                  BVS     1$                      ;;ERROR IF OVERFLOW
4626  017044  001402                  BEQ     1$                      ;ERROR IF ZERO
4627  017046  103401                  BCS     1$                      ;ERROR IF CARRY
4628  017050  100401                  BMI     2$                      ;BRANCH IF GOOD
4629  017052  104001          1$:     ERROR   +1              ;CPU ERROR
4630                                                        ;BAD XOR
4631  017054  012701  125252  2$:     MOV     #125252,R1              ;R1=125252
4632  017060  000277                  SCC                             ;CC=1111
4633  017062  074104                  XOR     R1,R4                   ;R4=054321
4634  017064  100403                  BMI     3$                      ;ERROR IF MINUS
4635  017066  001402                  BEQ     3$                      ;ERROR IF ZERO
4636  017070  103001                  BCC     3$                      ;ERROR IF CARRY CLEAR
4637  017072  102001                  BVC     4$                      ;BRANCH IF GOOD
4638  017074  104001          3$:     ERROR   +1              ;CPU ERROR
4639                                                        ;BAD XOR
4640  017076  074404          4$:     XOR     R4,R4                   ;R4=0
4641  017100  102406                  BVS     5$                      ;ERROR IF OVREFLOW
4642  017102  100405                  BMI     5$                      ;ERROR IF MINUS
4643  017104  103004                  BCC     5$                      ;ERROR IF NO CARRY
4644  017106  001003                  BNE     5$                      ;ERROR IF NOT ZERO
4645  017110  022704  000000          CMP     #0,R4                   ;SEE IF EXPECTED RESULT
4646  017114  001401                  BEQ     6$                      ;BRANCH IF GOOD
4647  017116  104001          5$:     ERROR   .1              ;CPU ERROR
4648                                                        ;BAD XOR
4649  017120                  6$:
4650
4651
4652                          ;
4653  017120                  MSXT:
4654
4655                          ;       TEST SXT (SIGN EXTEND INSTRUCTION)
4656                          ;;**********************************************************************
4657                          ;AN ADDITIONAL TEST IS INCLUDED TO CHECK FOR A SLOW N BIT PATH
4658                          ;ON A TRANSITION FROM ZERO TO ONE INTERNAL TO THE J11 DATA CHIP
4659                          ;THE PROBLEM IS ONLY EXHIBITED ON EARLY MASK SETS (1590 OR 1593)
4660  017120  005004                  CLR     R4                      ;TRASH R4
4661  017122  000257                  CCC                             ;CC=0000
4662  017124  000271                  .WORD   271                     ;CC=1001
4663  017126  006704                  SXT     R4                      ;*TEST INSTRUCTION
```

```
4664  017130  102405                          BVS     1$              ;BRANCH IF OVERFLOW IS NOT CLEARED
4665  017132  100004                          BPL     1$              ;BRANCH IF N BIT EFFECTED
4666  017134  001403                          BEQ     1$              ;BRANNCH IF Z BIT EFFECTED
4667  017136  103002                          BCC     1$              ;BRANCH IF C BIT EFFECTED
4668  017140  005204                          INC     R4
4669  017142  001401                          BEQ     2$              ;BRANCH IF R4 =0
4670  017144  104001              1$:         ERROR   •1              ;CPU ERROR
4671                                                  ;CC SHOULD HAVE = 1101
4672  017146  000277              2$:         SCC
4673  017150  000250                          CLN                     ;CC=0111
4674  017152  005004                          CLR     R4
4675  017154  012714  000055                  MOV     #55,(R4)        ;TRASH R4
4676  017160  006714                          SXT     (R4)            ;•TEST INSTRUCTION
4677  017162  001005                          BNE     3$              ;BRANCH IF BIT EFFECTED
4678  017164  102404                          BVS     3$              ;BRANCH IF OVERFLOW
4679  017166  103403                          BCS     3$              ;
4680  017170  100402                          BMI     3$              ;BRANCH IF N IS SET
4681  017172  005714                          TST     (R4)            ;VERIFY INSTRUCTION WORKED
4682  017174  001401                          BEQ     4$              ;BRANCH IF R4=0
4683  017176  104001              3$:         ERROR   •1              ;CPU ERROR
4684                                                  ;SXT FAILED
4685
4686                              ;           NOW TEST FOR SLOW N BIT IN J11 DATA CHIP
4687
4688  017200  012700  177777      4$:         MOV     #-1,R0          ;R0=177777, N BIT = 1
4689  017204  005004                          CLR     R4              ;CLEAT THE N BIT
4690  017206  006700                          SXT     R0              ;••••TEST INSTRUCTION•••
4691                                                                  ;TEST N BIT TRANSITION 1 TO 0
4692  017210  005700                          TST     R0              ;R0 SHOULD = 0
4693  017212  001401                          BEQ     5$              ;BRANCH IF OK
4694  017214  104001                          ERROR   •1              ;CPU ERROR
4695  017216  005000              5$:         CLR     R0              ;CLEAR R0, N BIT = 0
4696  017220  012704  177777                  MOV     #-1,R4          ;SET N BIT
4697  017224  006700                          SXT     R0              ;•••TEST INSTRUCTION•••
4698                                                                  ;TEST N BIT TRANSITION 0 TO 1
4699  017226  022700  177777                  CMP     #-1,R0          ;R0 SHOULD = 177777
4700  017232  001401                          BEQ     6$              ;BRANCH IF OK
4701  017234  104001                          ERROR   •1              ;CPU ERROR
4702  017236                          6$:
4703                                          ;
4704  017236                          MXOR:
4705
4706                              ;           TEST XOR
4707  017236  012701  007643                  MOV     #7643,R1                ;SETUP DATA
4708  017242  012704  133333                  MOV     #133333,R4              ;SETUP DATA
4709  017246  000277                          SCC
4710  017250  074401                          XOR     R4,R1                   ;•TEST INSTRUCTION
4711  017252  100006                          BPL     1$                      ;BRANCH IF PLUS TO ERROR
4712  017254  001405                          BEQ     1$                      ;ERROR IF ZERO
4713  017256  103004                          BCC     1$                      ;ERROR IF CARRY CLEAR
4714  017260  102403                          BVS     1$                      ;ERROR IF V SET
4715  017262  020127  134570                  CMP     R1,#134570              ;VERIFY CORRECT RESULT
4716  017266  001401                          BEQ     2$                      ;BRANCH IF GOOD
4717  017270  104001              1$:         ERROR   •1              ;CPU ERROR
4718                                                                  ;BAD XOR
4719  017272  010102              2$:         MOV     R1,R2
```

```
4720  017274  000257                          CCC
4721  017276  074402                          XOR     R4,R2                   ;*TEST INSTRUCTION
4722  017300  100405                          BMI     3$                      ;ERROR IF MINUS
4723  017302  102404                          BVS     3$                      ;ERROR IF OVERFLOW
4724  017304  103403                          BCS     3$                      ;ERROR IF CARRY
4725  017306  020227  007643                  CMP     R2,#7643
4726  017312  001401                          BEQ     4$                      ;BRANCH IF GOOD
4727  017314  104001                  3$:     ERROR   ·1              ;CPU ERROR
4728                                                                  ;BAD XOR
4729  017316                          4$:
4730
4731                                  ;
4732  017316                          MSOB:
4733
4734                                  ;       TEST SOB
4735  017316  012704  000555                  MOV     #555,R4         ;SETUP TEST COUNTER
4736  017322  000277                          SCC                     ;CC=17
4737  017324  103015                  1$:     BCC     2$              ;ERROR IF CARRY CLEAR
4738  017326  102014                          BVC     2$              ;ERROR IF NO OVERFLOW
4739  017330  100013                          BPL     2$              ;ERROR IF PLUS
4740  017332  001012                          BNE     2$              ;ERROR IF ZERO
4741  017334  077405                          SOB     R4,1$           ;*TEST INSTRUCTION
4742  017336  103005                          BCC     3$              ;ERROR IF CARRY CLEAR
4743  017340  102004                          BVC     3$              ;ERROR IF NO OVERFLOW
4744  017342  100003                          BPL     3$              ;ERROR IF PLUS
4745  017344  001002                          BNE     3$              ;ERROR IF ZERO
4746  017346  000167  000010                  JMP     4$
4747  017352  104001                  3$:     ERROR   ·1              ;CPU ERROR
4748                                                          ;CC EFFECTED DURING TEST
4749  017354  000167  000002                  JMP     4$
4750  017360  104001                  2$:     ERROR   ·1              ;CPU ERROR
4751                                                          ;CC EFFECTED AFTER TEST
4752  017362  020427  000000          4$:     CMP     R4,#0           ;IS R4 CORRECT
4753  017366  001401                          BEQ     5$              ;YES GO ON
4754  017370  104001                          ERROR   ·1              ;CPU ERROR
4755                                                          ;NO GO TO ERROR
4756  017372                          5$:
4757
4758
4759  017372                          MMARK:
4760
4761                                  ;       MARK INSTRUCTION TEST
4762  017372  012706  000700                  MOV     #STBOT-100,SP           ;SETUP TEST STACK = 700
4763  017376  012737  125252  000776          MOV     #125252,@#STBOT-2       ;SET UP NEW R5 VALUE ON STACK
4764  017404  012705  017422                  MOV     #1$,R5                  ;PUT NEW PC IN R5
4765  017410  012746  006437                  MOV     #MARK+37,-(SP)          ;INSERT MARK 37 INSTRUCTION ONTO STACK
4766  017414  000277                          SCC
4767  017416  000116                          JMP     (SP)                    ;* TEST INSTRUCTION
4768                                                                          ;MARK INSTRUCTION SHOULD HAVE GONE TO 1$
4769  017420  104001                          ERROR   ·1              ;CPU ERROR
4770
4771  017422  101002                  1$:     BHI     2$                      ;ERROR IF C OR Z BIT CLEAR
4772  017424  100001                          BPL     2$                      ;ERROR IF N BIT CLEAR
4773  017426  102401                          BVS     3$                      ;BRANCH IF V BIT SET
4774                                                                          ;BAD CONDITION CODES ON MARK
4775  017430  104001                  2$:     ERROR   ·1              ;CPU ERROR
```

```
4776  017432  022705  125252          3$:     CMP     #125252,R5              ;VERIFY R5
4777  017436  001401                          BEQ     4$                      ;BRANCH IF GOOD
4778  017440  104001                          ERROR   ·1              ;CPU ERROR
4779
4780  017442  020627  001000          4$:     CMP     SP,#STBOT               ;VERIFY THAT STACK IS CORRECT
4781  017446  001401                          BEQ     15$                     ;BRANCH IF OK
4782                                                                          ;ERROR! STACK WAS NOT CORRECT AFTER MARK
4783  017450  104001                          ERROR   ·1              ;CPU ERROR
4784
4785  017452  012746  052525          15$:    MOV     #52525,-(SP)            ;SETUP EXPECTED R5
4786  017456  012746  006400                  MOV     #6400,-(SP)             ;MOVE MARK 0 INSTRUCTION ON STACK
4787  017462  010605                          MOV     SP,R5                   ;R5=ADDRESS OF INSTRUCTION
4788  017464  004767  000004                  JSR     PC,5$                   ;LEAVE 6$ ON STACK
4789  017470  000167  000006          6$:     JMP     16$                     ;MARK RETURNED CORRECTLY
4790
4791  017474  000257                  5$:     CCC                             ;CLEAR THE CONDITION CODES
4792  017476  000205                          RTS     R5                      ;RETURN TO MARK INSTRUCTION
4793                                                                          ;NEXT INSTRUCTION ON STACK IS THE RETURN
4794                                                                          ;FROM THE JSR
4795                                                                          ;ERROR! BAD MARK SEQUENCE
4796  017500  104001                          ERROR   ·1              ;CPU ERROR
4797
4798  017502  101402          16$:    BLOS    7$                      ;IS C OR Z BIT SET?
4799  017504  100401                  BMI     7$                      ;IS N BIT SET?
4800  017506  102001                  BVC     8$                      ;IS V BIT SET?
4801                                                                  ;ERROR! CONDITIONS CODES INCORRECT
4802  017510  104001          7$:     ERROR   ·1              ;CPU ERROR
4803
4804  017512  020627  001000          8$:     CMP     R6,#STBOT               ;IS THE STACK CORRECT?
4805  017516  001401                          BEQ     9$                      ;BRANCH IF YES
4806                                                                          ;ERROR! BAD STACK CLEANUP
4807  017520  104001                          ERROR   ·1              ;CPU ERROR
4808  017522  022705  052525          9$:     CMP     #52525,R5               ;VERIFY THAT R5 WAS LOADED PROPERLY.
4809  017526  001401                          BEQ     10$                     ;IF OK, GO TO NEXT TEST.
4810                                                                          ;ERROR! R5 WAS NOT CORRECT AFTER MARK.
4811  017530  104001                          ERROR   ·1              ;CPU ERROR
4812  017532                          10$:
4813  017532                          XME100:
4814
4815                                  ;       TEST CLEAR CONDITION CODES INSTRUCTION
4816  017532  012737  030017  177776          MOV     #30017,@#177776         ;SETUP PSW
4817  017540  000257                          CCC                             ; TEST INSTRUCTION
4818  017542  022737  030000  177776          CMP     #30000,@#177776         ;DID IT CLEAR ALL CONDITION CODE BITS
4819  017550  001401                          BEQ     1$                      ;YES GO ON
4820  017552  104001                          ERROR   ·1              ;CPU ERROR
4821                                                                          ;NO GO TO ERROR
4822  017554                          1$:
4823                                  ;
4824  017554                          XME101:
4825
4826                                  ;       TEST CLEAR C BIT INSTRUCTION
4827  017554  012737  030017  177776          MOV     #30017,@#177776         ;SETUP PSW
4828  017562  000241                          CLC                             ; TEST INSTRUCTION
4829  017564  022737  030016  177776          CMP     #30016,@#177776         ;DID IT CLEAR CARRY BIT
4830  017572  001401                          BEQ     1$                      ;YES GO ON
4831                                                                          ;C BIT NOT CLEAR GO TO ERROR
```

```
4832  017574  104001                          ERROR   +1              ;CPU ERROR
4833  017576                          1$:
4834                                  ;
4835  017576                          TE102:
4836                                  ;       TEST CLEAR N BIT INST (CLN)
4837  017576  012737  030017  177776          MOV     #30017,@#177776         ;SETUP PSW
4838  017604  000250                          CLN                             ; TEST INSTRUCTION
4839  017606  022737  030007  177776          CMP     #30007,@#177776         ;DID IT CLEAR NEGATIVE BIT
4840  017614  001401                          BEQ     1$                      ;YES GO ON
4841  017616  104001                          ERROR   +1              ;CPU ERROR
4842                                                                  ;NO GO TO ERROR
4843  017620                          1$:
4844                                  ;
4845  017620                          TE103:
4846                                  ;       TEST CLEAR V BIT INST (CLV)
4847  017620  012737  030017  177776          MOV     #30017,@#177776         ;SETUP PSW
4848  017626  000242                          CLV                             ; TEST INSTRUCTION
4849  017630  022737  030015  177776          CMP     #30015,@#177776         ;DID IT CLEAR OVERFLOW BIT
4850  017636  001401                          BEQ     1$                      ;YES GO ON
4851  017640  104001                          ERROR   +1              ;CPU ERROR
4852                                                                  ;NO GO TO ERROR
4853  017642                          1$:
4854                                  ;
4855  017642                          TE104:
4856                                  ;       TEST CLEAR Z BIT INST (CLZ)
4857  017642  012737  030017  177776          MOV     #30017,@#177776         ;SETUP PSW
4858  017650  000244                          CLZ                             ; TEST INSTRUCTION
4859  017652  022737  030013  177776          CMP     #30013,@#177776         ;DID IT CLEAR ZERO BIT
4860  017660  001401                          BEQ     1$                      ;YES GO ON
4861  017662  104001                          ERROR   +1              ;CPU ERROR
4862                                                                  ;NO GO TO ERROR
4863  017664                          1$:
4864                                  ;
4865  017664                          TE105:
4866                                  ;       TEST SET CONDITION CODES INST (SCC)
4867  017664  012737  030000  177776          MOV     #30000,@#177776         ;SETUP PSW
4868  017672  000277                          SCC                             ; TEST INSTRUCTION
4869  017674  022737  030017  177776          CMP     #30017,@#177776         ;DID IT SET ALL CONDITION CODE BITS
4870  017702  001401                          BEQ     1$                      ;YES GO ON
4871  017704  104001                          ERROR   +1              ;CPU ERROR
4872                                                                  ;NO GO TO ERROR
4873  017706                          1$:
4874                                  ;
4875  017706                          TE106:
4876                                  ;       TEST SET C BIT INST (SEC)
4877  017706  012737  030000  177776          MOV     #30000,@#177776         ;SETUP PSW
4878  017714  000261                          SEC                             ; TEST INSTRUCTION
4879  017716  022737  030001  177776          CMP     #30001,@#177776         ;DID IT SET THE CARRY BIT
4880  017724  001401                          BEQ     1$                      ;YES GO ON
4881  017726  104001                          ERROR   +1              ;CPU ERROR
4882                                                                  ;NO GO TO ERROR
4883  017730                          1$:
4884                                  ;
4885  017730                          TE107:
4886                                  ;       TEST SET N BIT INST (SEN)
4887  017730  012737  030000  177776          MOV     #30000,@#177776         ;SETUP PSW
```

```
4888  017736  000270                          SEN                             ; TEST INSTRUCTION
4889  017740  022737  030010  177776          CMP     #30010,@#177776         ;DID IT SET THE NEGATIVE BIT
4890  017746  001401                          BEQ     1$                      ;YES GO ON
4891  017750  104001                          ERROR   +1              ;CPU ERROR
4892                                                                  ;NO GO TO ERROR
4893  017752                          1$:
4894                                  ;
4895  017752                          TE110:
4896                                  ;       TEST SET V BIT INST (SEV)
4897  017752  012737  030000  177776          MOV     #30000,@#177776         ;SETUP PSW
4898  017760  000262                          SEV                             ; TEST INSTRUCTION
4899  017762  022737  030002  177776          CMP     #30002,@#177776         ;DID IT SET THE OVERFLOW BIT
4900  017770  001401                          BEQ     1$                      ;YES GO ON
4901  017772  104001                          ERROR   +1              ;CPU ERROR
4902                                                                  ;NO GO TO ERROR
4903  017774                          1$:
4904                                  ;
4905  017774                          TE111:
4906                                  ;       TEST SET Z BIT INST (SEZ)
4907  017774  012737  030000  177776          MOV     #30000,@#177776         ;SETUP PSW
4908  020002  000264                          SEZ                             ; TEST INSTRUCTION
4909  020004  022737  030004  177776          CMP     #30004,@#177776         ;DID IT SET THE ZERO BIT
4910  020012  001401                          BEQ     1$                      ;YES GO ON
4911  020014  104001                          ERROR   +1              ;CPU ERROR
4912                                                                  ;NO GO TO ERROR
4913  020016                          1$:
4914                                  ;
4915  020016                          TE112:
4916                                  ;       TEST MULTIPLE CLEARS OF CC BITS
4917  020016  012737  030000  177776          MOV     #30000,@#177776         ;INIT PSW
4918  020024  000277                          SCC                             ;SETUP PSW
4919  020026  000243                          .WORD   243                     ; TEST CLC CLV
4920  020030  022737  030014  177776          CMP     #30014,@#177776         ;PSW CORRECT?
4921  020036  001401                          BEQ     1$                      ;YES GO ON
4922  020040  104001                          ERROR   +1              ;CPU ERROR
4923                                                                  ;NO GO TO ERROR
4924  020042  000277                  1$:     SCC                             ;SETUP PSW
4925  020044  000245                          .WORD   245                     ; TEST CLC CLZ
4926  020046  022737  030012  177776          CMP     #30012,@#177776         ;PSW CORRECT?
4927  020054  001401                          BEQ     2$                      ;YES GO ON
4928  020056  104001                          ERROR   +1              ;CPU ERROR
4929                                                                  ;NO GO TO ERROR
4930  020060  000277                  2$:     SCC                             ;SETUP PSW
4931  020062  000246                          .WORD   246                     ; TEST CLV CLZ
4932  020064  022737  030011  177776          CMP     #30011,@#177776         ;PSW CORRECT?
4933  020072  001401                          BEQ     3$                      ;YES GO ON
4934  020074  104001                          ERROR   +1              ;CPU ERROR
4935                                                                  ;NO GO TO ERROR
4936  020076  000277                  3$:     SCC                             ;SETUP PSW
4937  020100  000247                          .WORD   247                     ; TEST CLC CLV CLZ
4938  020102  022737  030010  177776          CMP     #30010,@#177776         ;PSW CORRECT?
4939  020110  001401                          BEQ     4$                      ;YES GO ON
4940  020112  104001                          ERROR   +1              ;CPU ERROR
4941                                                                  ;NO GO TO ERROR
4942  020114  000277                  4$:     SCC                             ;SETUP PSW
4943  020116  000251                          .WORD   251                     ; TEST CLN CLC
```

```
4944  020120  022737  030006  177776          CMP     #30006,@#177776         ;PSW CORRECT?
4945  020126  001401                          BEQ     5$                      ;YES GO ON
4946  020130  104001                          ERROR   .1              ;CPU ERROR
4947                                                                  ;NO GO TO ERROR
4948  020132  000277                  5$:     SCC                             ;SETUP PSW
4949  020134  000252                          .WORD   252                     ; TEST CLN CLV
4950  020136  022737  030005  177776          CMP     #30005,@#177776         ;PSW CORRECT?
4951  020144  001401                          BEQ     6$                      ;YES GO ON
4952  020146  104001                          ERROR   .1              ;CPU ERROR
4953                                                                  ;NO GO TO ERROR
4954  020150  000277                  6$:     SCC                             ;SETUP PSW
4955  020152  000253                          .WORD   253                     ; TEST CLN CLC CLV
4956  020154  022737  030004  177776          CMP     #30004,@#177776         ;PSW CORRECT?
4957  020162  001401                          BEQ     7$                      ;YES GO ON
4958  020164  104001                          ERROR   .1              ;CPU ERROR
4959                                                                  ;NO GO TO ERROR
4960  020166  000277                  7$:     SCC                             ;SETUP PSW
4961  020170  000254                          .WORD   254                     ; TEST CLN CLZ
4962  020172  022737  030003  177776          CMP     #30003,@#177776         ;PSW CORRECT?
4963  020200  001401                          BEQ     8$                      ;YES GO ON
4964  020202  104001                          ERROR   .1              ;CPU ERROR
4965                                                                  ;NO GO TO ERROR
4966  020204  000277                  8$:     SCC                             ;SETUP PSW
4967  020206  000255                          .WORD   255                     ; TEST CLN CLC CLZ
4968  020210  022737  030002  177776          CMP     #30002,@#177776         ;PSW CORRECT?
4969  020216  001401                          BEQ     9$                      ;YES GO ON
4970  020220  104001                          ERROR   .1              ;CPU ERROR
4971                                                                  ;NO GO TO ERROR
4972  020222  000277                  9$:     SCC                             ;SETUP PSW
4973  020224  000256                          .WORD   256                     ; TEST CLN CLV CLZ
4974  020226  022737  030001  177776          CMP     #30001,@#177776         ;SETUP PSW
4975  020234  001401                          BEQ     10$                     ;YES GO ON
4976  020236  104001                          ERROR   .1              ;CPU ERROR
4977                                                                  ;NO GO TO ERROR
4978  020240                          10$:
4979                                  ;
4980  020240                          TE113:
4981                                  ;       TEST MULTIPLE SETS OF CC BITS
4982  020240  012737  030000  177776          MOV     #30000,@#177776         ;INIT PSW
4983  020246  000263                          .WORD   263                     ; TEST SEC SEV
4984  020250  022737  030003  177776          CMP     #30003,@#177776         ;PSW CORRECT?
4985  020256  001401                          BEQ     1$                      ;YES GO ON
4986  020260  104001                          ERROR   .1              ;CPU ERROR
4987                                                                  ;NO GO TO ERROR
4988  020262  000257                  1$:     CCC                             ;SETUP PSW
4989  020264  000265                          .WORD   265                     ; TEST SEC SEZ
4990  020266  022737  030005  177776          CMP     #30005,@#177776         ;PSW CORRECT?
4991  020274  001401                          BEQ     2$                      ;YES GO ON
4992  020276  104001                          ERROR   .1              ;CPU ERROR
4993                                                                  ;NO GO TO ERROR
4994  020300  000257                  2$:     CCC                             ;SETUP PSW
4995  020302  000266                          .WORD   266                     ; TEST SEV SEZ
4996  020304  022737  030006  177776          CMP     #30006,@#177776         ;PSW CORRECT?
4997  020312  001401                          BEQ     3$                      ;YES GO ON
4998  020314  104001                          ERROR   .1              ;CPU ERROR
4999                                                                  ;NO GO TO ERROR
```

```
5000  020316  000257                  3$:     CCC                             ;SETUP PSW
5001  020320  000267                          .WORD   267                     ; TEST SEC SEV SEZ
5002  020322  022737  030007  177776          CMP     #30007,@#177776         ;PSW CORRECT?
5003  020330  001401                          BEQ     4$                      ;YES GO ON
5004  020332  104001                          ERROR   .1              ;CPU ERROR
5005                                                                  ;NO GO TO ERROR
5006  020334  000257                  4$:     CCC                             ;SETUP PSW
5007  020336  000271                          .WORD   271                     ; TEST SEN SEC
5008  020340  022737  030011  177776          CMP     #30011,@#177776         ;PSW CORRECT?
5009  020346  001401                          BEQ     5$                      ;YES GO ON
5010  020350  104001                          ERROR   .1              ;CPU ERROR
5011                                                                  ;NO GO TO ERROR
5012  020352  000257                  5$:     CCC                             ;SETUP PSW
5013  020354  000272                          .WORD   272                     ; TEST SEN SEV
5014  020356  022737  030012  177776          CMP     #30012,@#177776         ;PSW CORRECT?
5015  020364  001401                          BEQ     6$                      ;YES GO ON
5016  020366  104001                          ERROR   .1              ;CPU ERROR
5017                                                                  ;NO GO TO ERROR
5018  020370  000257                  6$:     CCC                             ;SETUP PSW
5019  020372  000273                          .WORD   273                     ; TEST SEN SEC SEV
5020  020374  022737  030013  177776          CMP     #30013,@#177776         ;PSW CORRECT?
5021  020402  001401                          BEQ     7$                      ;YES GO ON
5022  020404  104001                          ERROR   .1              ;CPU ERROR
5023                                                                  ;NO GO TO ERROR
5024  020406  000257                  7$:     CCC                             ;SETUP PSW
5025  020410  000274                          .WORD   274                     ; TEST SEN SEZ
5026  020412  022737  030014  177776          CMP     #30014,@#177776         ;PSW CORRECT?
5027  020420  001401                          BEQ     8$                      ;YES GO ON
5028  020422  104001                          ERROR   .1              ;CPU ERROR
5029                                                                  ;NO GO TO ERROR
5030  020424  000257                  8$:     CCC                             ;SETUP PSW
5031  020426  000275                          .WORD   275                     ; TEST SEN SEC SEZ
5032  020430  022737  030015  177776          CMP     #30015,@#177776         ;PSW CORRECT?
5033  020436  001401                          BEQ     9$                      ;YES GO ON
5034  020440  104001                          ERROR   .1              ;CPU ERROR
5035                                                                  ;NO GO TO ERROR
5036  020442  000257                  9$:     CCC                             ;SETUP PSW
5037  020444  000276                          .WORD   276                     ; TEST SEN SEV SEZ
5038  020446  022737  030016  177776          CMP     #30016,@#177776         ;PSW CORRECT?
5039  020454  001401                          BEQ     10$                     ;YES GO ON
5040  020456  104001                          ERROR   .1              ;CPU ERROR
5041                                                                  ;NO GO TO ERROR
5042  020460                          10$:
5043                                  ;
5044  020460                          TE113A:
5045                                  ;       TEST SIGNED AND CONDITIONAL BRANCHES
5046  020460  000257                          CCC                             ;CLEAR ALL CC BITS IN PSW
5047  020462  002001                          BGE     1$                      ;BGE SHOULD BRANCH
5048  020464  104001                          ERROR   .1              ;CPU ERROR
5049                                                                  ;ERROR; DIDN'T BRANCH
5050  020466  003001                  1$:     BGT     2$                      ;BGT SHOULD BRANCH
5051  020470  104001                          ERROR   .1              ;CPU ERROR
5052                                                                  ;ERROR;  DIDN'T BRANCH
5053  020472  003401                  2$:     BLE     3$                      ;BLE SHOULDN'T BRANCH
5054  020474  000401                          BR      4$                      ;BRANCH TO NEXT TEST
5055  020476  104001                  3$:     ERROR   .1              ;CPU ERROR
```

```
5056
5057  020500  002401          4$:     BLT     5$              ;ERROR; BLE SHOULD NOT HAVE BRANCHED
5058  020502  000401                  BR      6$                      ;BLT SHOULD NOT BRANCH
5059  020504  104001          5$:     ERROR   ;1                      ;BRANCH TO NEXT TEST
5060                                                          ;CPU ERROR
5061  020506  000264          6$:     SEZ                     ;ERROR; BLT SHOULD NOT HAVE BRANCHED
5062  020510  003401                  BLE     7$                      ;SET THE Z BIT IN PSW
5063  020512  104001                  ERROR   ;1                      ;BLE SHOULD BRANCH
5064                                                          ;CPU ERROR
5065  020514  003001          7$:     BGT     8$              ;ERROR; BLE DIDN'T BRANCH
5066  020516  000401                  BR      9$                      ;BGT SHOULD NOT BRANCH
5067  020520  104001          8$:     ERROR   ;1                      ;BRANCH TO NEXT TEST
5068                                                          ;CPU ERROR
5069  020522  000257          9$:     CCC                     ;ERROR; BGT SHOULD NOT HAVE BRANCHED
5070  020524  000270                  SEN                             ;CLEAR ALL CC BITS IN PSW
5071  020526  002401                  BLT     10$                     ;SET N BIT IN PSW
5072  020530  104001                  ERROR   ;1                      ;SHOULD BRANCH TO NEXT TEST
5073                                                          ;CPU ERROR
5074  020532  003401          10$:    BLE     11$             ;ERROR; BLT SHOULD HAVE BRANCHED
5075  020534  104001                  ERROR   ;1                      ;SHOULD BRANCH TO NEXT TEST
5076                                                          ;CPU ERROR
5077  020536  002001          11$:    BGE     12$             ;ERROR; BLE SHOULD HAVE BRANCHED
5078  020540  000401                  BR      13$                     ;BGE SHOULD NOT BRANCH
5079  020542  104001          12$:    ERROR   ;1                      ;BRANCH TO NEXT TEST
5080                                                          ;CPU ERROR
5081  020544  003001          13$:    BGT     14$             ;ERROR; BGE SHOULD NOT HAVE BRANCHED
5082  020546  000401                  BR      15$                     ;BGT SHOULD NOT BRANCH
5083  020550  104001          14$:    ERROR   ;1                      ;BRANCH TO NEXT TEST
5084                                                          ;CPU ERROR
5085  020552  000257          15$:    CCC                     ;ERROR; BGT SHOULD NOT HAVE BRANCHED
5086  020554  000262                  SEV                             ;CLEAR ALL CC BITS
5087  020556  003401                  BLE     16$                     ;SET V BIT IN PSW
5088  020560  104001                  ERROR   ;1                      ;BLE SHOULD BRANCH
5089                                                          ;CPU ERROR
5090  020562  002401          16$:    BLT     17$             ;ERROR; BLE DIDN'T BRANCH
5091  020564  104001                  ERROR   ;1                      ;BLT SHOULD BRANCH
5092                                                          ;CPU ERROR
5093  020566  002001          17$:    BGE     18$             ;ERROR; BLT DIDN'T BRANCH
5094  020570  000401                  BR      19$                     ;BGE SHOULDN'T BRANCH
5095  020572  104001          18$:    ERROR   ;1                      ;BRANCH TO NEXT TEST
5096                                                          ;CPU ERROR
5097  020574  003001          19$:    BGT     20$             ;ERROR; BGE SHOULD NOT HAVE BRANCHED
5098  020576  000401                  BR      21$                     ;BGT SHOULD NOT BRANCH
5099  020600  104001          20$:    ERROR   ;1                      ;BRANCH TO NEXT TEST
5100                                                          ;CPU ERROR
5101  020602  000257          21$:    CCC                     ;ERROR; BGT SHOULD NOT HAVE BRANCHED
5102  020604  000272                  .WORD   272                     ;CLEAR ALL CC BITS
5103  020606  002001                  BGE     22$                     ;SET N AND V BITS IN PSW
5104  020610  104001                  ERROR   ;1                      ;BGE SHOULD BRANCH
5105                                                          ;CPU ERROR
5106  020612  003001          22$:    BGT     23$             ;ERROR; BGE DIDN'T BRANCH
5107  020614  104001                  ERROR   ;1                      ;BGT SHOULD BRANCH
5108                                                          ;CPU ERROR
5109  020616  003401          23$:    BLE     24$             ;ERROR; BGT DIDN'T BRANCH
5110  020620  000401                  BR      25$                     ;BLE SHOULDN'T BRANCH
5111  020622  104001          24$:    ERROR   ;1                      ;BRANCH TO NEXT TEST
                                                            ;CPU ERROR
```

```
                                                        ;ERROR; BLE SHOULD NOT HAVE BRANCHED
020624  002401                  25$:     BLT     26$            ;BLT SHOULD NOT BRANCH
020626  000401                           BR      27$            ;BRANCH TO NEXT TEST
020630  104001                  26$:     ERROR   .1          ;CPU ERROR
                                                        ;ERROR; BLT SHOULD NOT HAVE BRANCHED
020632                          27$:
020632                          TE114:
                                ;        TEST NOP INST
020632  012737  030000  177776           MOV     #30000,@#177776        ;INIT PSW
020640  012700  000001                   MOV     #1,R0          ;INIT R0
020644  012701  000002                   MOV     #2,R1          ;INIT R1
020650  012702  000003                   MOV     #3,R2          ;INIT R2
020654  012703  000004                   MOV     #4,R3          ;INIT R3
020660  012704  000005                   MOV     #5,R4          ;INIT R4
020664  012705  000006                   MOV     #6,R5          ;INIT R5
020670  010637  002762                   MOV     R6,@#SLOC00    ;SAVE SP
020674  012737  030017  110502           MOV     #30017,@#EXPDAT        ;SETUP PSW
020702  000277                           SCC                    ;SET ALL CONDITION CODE BITS
020704  000240                           NOP                    ; TEST INSTRUCTION
020706  000240                           NOP                    ; TEST INSTRUCTION
020710  000240                           NOP                    ; TEST INSTRUCTION
020712  004767  000046                   JSR     PC,T114        ;CHECK PSW, AND GPR'S
020716  020637  002762                   CMP     R6,@#SLOC00    ;CHECK SP
020722  001401                           BEQ     1$             ;OK GO ON
020724  104001                           ERROR   .1          ;CPU ERROR
                                                        ;NO GO TO ERROR
020726  012737  030000  110502  1$:      MOV     #30000,@#EXPDAT        ;SETUP PSW
020734  000257                           CCC                    ;CLEAR ALL CONDITION CODE BITS
020736  000260                           .WORD   260            ; TEST FOR NOP OPERATION
020740  000260                           .WORD   260            ; TEST FOR NOP OPERATION
020742  000260                           .WORD   260            ; TEST FOR NOP OPERATION
020744  004767  000014                   JSR     PC,T114        ;CHECK PSW, AND GPR'S
020750  020637  002762                   CMP     R6,@#SLOC00    ;CHECK SP
020754  001401                           BEQ     2$             ;OK GO ON
020756  104001                           ERROR   .1          ;CPU ERROR
                                                        ;NO GO TO ERROR
020760                          2$:
020760  000167  000104                   JMP     FINNOP
                                ;
020764  023737  110502  177776  T114:    CMP     @#EXPDAT,@#177776      ;CHECK PSW
020772  001405                           BEQ     TA114          ;OK GO ON
020774  010067  157400                   MOV     R0,400         ;SAVE R0
021000  104001                           ERROR   .1          ;CPU ERROR
                                                        ;NO GO TO ERROR
021002  016700  157372                   MOV     400,R0         ;RESTORE R0
021006  022700  000001          TA114:   CMP     #1,R0          ;CHECK R0
021012  001401                           BEQ     TB114          ;OK GO ON
021014  104001                           ERROR   .1          ;CPU ERROR
                                                        ;NO GO TO ERROR
021016  022701  000002          TB114:   CMP     #2,R1          ;CHECK R1
021022  001401                           BEQ     TC114          ;OK GO ON
021024  104001                           ERROR   .1          ;CPU ERROR
                                                        ;NO GO TO ERROR
021026  022702  000003          TC114:   CMP     #3,R2          ;CHECK R2
021032  001401                           BEQ     TD114          ;OK GO ON
021034  104001                           ERROR   .1          ;CPU ERROR
```

```
                                                                ;NO GO TO ERROR
021036  022703  000004          TD114:  CMP     #4,R3                   ;CHECK R3
021042  001401                          BEQ     TF114                   ;OK GO ON
021044  104001                          ERROR   .1              ;CPU ERROR
                                                                ;NO GO TO ERROR
021046  022704  000005          TF114:  CMP     #5,R4                   ;CHECK R4
021052  001401                          BEQ     TG114                   ;OK GO ON
021054  104001                          ERROR   .1              ;CPU ERROR
                                                                ;NO GO TO ERROR
021056  022705  000006          TG114:  CMP     #6,R5                   ;CHECK R5
021062  001401                          BEQ     TH114                   ;OK GO ON
021064  104001                          ERROR   .1              ;CPU ERROR
                                                                ;NO GO TO ERROR
021066  000207                  TH114:  RTS     PC                      ;RETURN
                                ;
021070  000240                  FINNOP: NOP
                                ;
                                ;
                                ;-----------------------------------
                                ;TEST ATERNATE REGISTER SET
                                ;
021072  005000                          CLR     R0                      ;------CLEAR------
021074  005001                          CLR     R1                      ;-----PRIMARY----
021076  005002                          CLR     R2                      ;-----GENERAL-----
021100  005003                          CLR     R3                      ;-----PURPOSE-----
021102  005004                          CLR     R4                      ;-----REGISTER----
021104  005005                          CLR     R5                      ;-------SET-------
021106  012767  004000  156662          MOV     #4000,PS                ;SELECT ALTERNATE REGISTER SET
021114  032767  004000  156654          BIT     #BIT11,PS               ;TEST TO SEE THAT BIT IS SET IN PS
021122  001001                          BNE     1$                      ;IF IT'S SET GO TESTS REGISTERS
021124  104001                          ERROR   .1              ;CPU ERROR
                                                                ;ERROR! ALTERNATE REGISTER SELECT
                                                                        ;BIT IN PS IS NOT SET
021126  012700  177777          1$:     MOV     #177777,R0              ;WRITE ALL ONES TO ALTERNATE REG SET
021132  010001                          MOV     R0,R1                   ;
021134  010102                          MOV     R1,R2                   ;
021136  010203                          MOV     R2,R3                   ;
021140  010304                          MOV     R3,R4                   ;
021142  010405                          MOV     R4,R5                   ;
021144  042767  004000  156624          BIC     #BIT11,PS               ;CLEAR BIT 11 IN PS
021152  032767  004000  156616          BIT     #BIT11,PS               ;IS BIT 11 = 0?
021160  001401                          BEQ     2$                      ;IF IT'S NOT ZERO
021162  104001                          ERROR   .1              ;CPU ERROR
                                                                ;ERROR! BIT 11 DID NOT CLEAR
021164  005700                  2$:     TST     R0                      ;MAKE SURE THAT WE REALLY
021166  001401                          BEQ     3$                      ;WERE TALKING TO ALTERNATE REGISTERS.
021170  104001                          ERROR   .1              ;CPU ERROR
                                                                ;ERROR!
021172  005701                  3$:     TST     R1                      ;
021174  001401                          BEQ     4$
021176  104001                          ERROR   .1              ;CPU ERROR
                                                                ;ERROR!
021200  005702                  4$:     TST     R2                      ;
021202  001401                          BEQ     5$                      ;
021204  104001                          ERROR   .1              ;CPU ERROR
                                                                ;ERROR!
```

```
5224  021206  005703                  5$:     TST     R3                      ;
5225  021210  001401                          BEQ     6$                      ;
5226  021212  104001                          ERROR   ·1              ;CPU ERROR
5227                                                                  ;
5228  021214  005704                  6$:     TST     R4                      ;
5229  021216  001401                          BEQ     7$                      ;
5230  021220  104001                          ERROR   ·1              ;CPU ERROR
5231                                                                  ;
5232  021222  005705                  7$:     TST     R5                      ;
5233  021224  001401                          BEQ     8$                      ;
5234  021226  104001                          ERROR   ·1              ;CPU ERROR
5235                                                                  ;ERROR!
5236  021230  012767  004000  156540  8$:     MOV     #BIT11,PS               ;ENABLE ALTERNATE REGISTER SET
5237  021236  005000                          CLR     R0                      ;-----CLEAR-----
5238  021240  005001                          CLR     R1                      ;---ALTERNATE---
5239  021242  005002                          CLR     R2                      ;----REGISTER---
5240  021244  005003                          CLR     R3                      ;------SET------
5241  021246  005004                          CLR     R4                      ;
5242  021250  005005                          CLR     R5                      ;
5243  021252  042767  004000  156516          BIC     #BIT11,PS               ;BACK TO PRIMARY GPRS
5244  021260  012700  177777                  MOV     #177777,R0              ;ENSURE THAT WRITES TO PRIMARY GPRS
5245  021264  010001                          MOV     R0,R1                   ;DO NOT EFFECT ALTERNATE GPRS
5246  021266  010102                          MOV     R1,R2                   ;
5247  021270  010203                          MOV     R2,R3                   ;
5248  021272  010304                          MOV     R3,R4                   ;
5249  021274  010405                          MOV     R4,R5                   ;
5250  021276  052767  004000  156472          BIS     #BIT11,PS               ;RETURN TO ALTERNATE REGISTERS
5251  021304  005700                          TST     R0                      ;SHOULD BE ALL 0'S
5252  021306  001401                          BEQ     9$                      ;
5253  021310  104001                          ERROR   ·1              ;CPU ERROR
5254                                                                  ;ERROR!
5255  021312  005701                  9$:     TST     R1                      ;
5256  021314  001401                          BEQ     10$                     ;
5257  021316  104001                          ERROR   ·1              ;CPU ERROR
5258                                                                  ;
5259  021320  005702                  10$:    TST     R2                      ;
5260  021322  001401                          BEQ     11$                     ;
5261  021324  104001                          ERROR   ·1              ;CPU ERROR
5262                                                                  ;
5263  021326  005703                  11$:    TST     R3                      ;
5264  021330  001401                          BEQ     12$                     ;
5265  021332  104001                          ERROR   ·1              ;CPU ERROR
5266                                                                  ;
5267  021334  005704                  12$:    TST     R4                      ;
5268  021336  001401                          BEQ     13$                     ;
5269  021340  104001                          ERROR   ·1              ;CPU ERROR
5270                                                                  ;
5271  021342  005705                  13$:    TST     R5                      ;
5272  021344  001401                          BEQ     14$                     ;
5273  021346  104001                          ERROR   ·1              ;CPU ERROR
5274  021350                          14$:
5275
5276  021350                          ALROTS:
5277                                  ;       ALTERNATE REGISTER SET R0 BIT TESTS
5278  021350  012700  177777                  MOV     #177777,R0              ;R0=177777
5279  021354  020027  177777                  CMP     R0,#177777              ;DOES R0=177777
```

```
5280  021360  001401                    BEQ     1$                      ;YES GO ON
5281  021362  104001                    ERROR   +1              ;CPU ERROR
5282                                                              ;NO GO TO ERROR
5283  021364  005000              1$:   CLR     R0                      ;R0=0
5284  021366  020027  000000            CMP     R0,#0                   ;DOES R0=0
5285  021372  001401                    BEQ     2$                      ;YES GO ON
5286  021374  104001                    ERROR   +1              ;CPU ERROR
5287                                                              ;NO GO TO ERROR
5288  021376  012700  125252        2$: MOV     #125252,R0              ;R0=125252
5289  021402  020027  125252            CMP     R0,#125252              ;DOES R0=125252
5290  021406  001401                    BEQ     3$                      ;YES GO ON
5291  021410  104001                    ERROR   +1              ;CPU ERROR
5292                                                              ;NO GO TO ERROR
5293  021412  012700  052525        3$: MOV     #52525,R0               ;R0=52525
5294  021416  020027  052525            CMP     R0,#52525               ;DOES R0=52525
5295  021422  001401                    BEQ     4$                      ;YES GO ON
5296  021424  104001                    ERROR   +1              ;CPU ERROR
5297                                                              ;NO GO TO ERROR
5298  021426                        4$:
5299                                  ;
5300  021426                        ALR1TS:
5301                                  ;       ALTERNATE REGISTER SET R1 BIT TESTS
5302  021426  012701  177777            MOV     #177777,R1              ;R1=177777
5303  021432  020127  177777            CMP     R1,#177777              ;DOES R1=177777
5304  021436  001401                    BEQ     1$                      ;YES GO ON
5305  021440  104001                    ERROR   +1              ;CPU ERROR
5306                                                              ;NO GO TO ERROR
5307  021442  005001              1$:   CLR     R1                      ;R1=0
5308  021444  020127  000000            CMP     R1,#0                   ;DOES R1=0
5309  021450  001401                    BEQ     2$                      ;YES GO ON
5310  021452  104001                    ERROR   +1              ;CPU ERROR
5311                                                              ;NO GO TO ERROR
5312  021454  012701  125252        2$: MOV     #125252,R1              ;R1=125252
5313  021460  020127  125252            CMP     R1,#125252              ;DOES R1=125252
5314  021464  001401                    BEQ     3$                      ;YES GO ON
5315  021466  104001                    ERROR   +1              ;CPU ERROR
5316                                                              ;NO GO TO ERROR
5317  021470  012701  052525        3$: MOV     #52525,R1               ;R1=52525
5318  021474  020127  052525            CMP     R1,#52525               ;DOES R1=52525
5319  021500  001401                    BEQ     4$                      ;YES GO ON
5320  021502  104001                    ERROR   +1              ;CPU ERROR
5321                                                              ;NO GO TO ERROR
5322  021504                        4$:
5323                                  ;
5324  021504                        ALR2TS:
5325                                  ;       ALTERNATE REGISTER SET R2 BIT TESTS
5326  021504  012702  177777            MOV     #177777,R2              ;R2=177777
5327  021510  020227  177777            CMP     R2,#177777              ;DOES R2=177777
5328  021514  001401                    BEQ     1$                      ;YES GO ON
5329  021516  104001                    ERROR   +1              ;CPU ERROR
5330                                                              ;NO GO TO ERROR
5331  021520  005002              1$:   CLR     R2                      ;R2=0
5332  021522  020227  000000            CMP     R2,#0                   ;DOES R2=0
5333  021526  001401                    BEQ     2$                      ;YES GO ON
5334  021530  104001                    ERROR   +1              ;CPU ERROR
5335                                                              ;NO GO TO ERROR
```

```
021532  012702  125252          2$:     MOV     #125252,R2              ;R2=125252
021536  020227  125252                  CMP     R2,#125252              ;DOES R2=125252
021542  001401                          BEQ     3$                      ;YES GO ON
021544  104001                          ERROR   +1              ;CPU ERROR
                                                                        ;NO GO TO ERROR
021546  012702  052525          3$:     MOV     #52525,R2               ;R2=52525
021552  020227  052525                  CMP     R2,#52525               ;DOES R2=52525
021556  001401                          BEQ     4$                      ;YES GO ON
021560  104001                          ERROR   +1              ;CPU ERROR
                                                                        ;NO GO TO ERROR
021562                          4$:
                                ;
021562                          ALR3TS:
                                ;       ALTERNATE REGISTER SET R3 BIT TESTS
021562  012703  177777                  MOV     #177777,R3              ;R3=177777
021566  020327  177777                  CMP     R3,#177777              ;DOES R3=177777
021572  001401                          BEQ     1$                      ;YES GO ON
021574  104001                          ERROR   +1              ;CPU ERROR
                                                                        ;NO GO TO ERROR
021576  005003                  1$:     CLR     R3                      ;R3=0
021600  020327  000000                  CMP     R3,#0                   ;DOES R3=0
021604  001401                          BEQ     2$                      ;YES GO ON
021606  104001                          ERROR   +1              ;CPU ERROR
                                                                        ;NO GO TO ERROR
021610  012703  125252          2$:     MOV     #125252,R3              ;R3=125252
021614  020327  125252                  CMP     R3,#125252              ;DOES R3=125252
021620  001401                          BEQ     3$                      ;YES GO ON
021622  104001                          ERROR   +1              ;CPU ERROR
                                                                        ;NO GO TO ERROR
021624  012703  052525          3$:     MOV     #52525,R3               ;R3=52525
021630  020327  052525                  CMP     R3,#52525               ;DOES R3=52525
021634  001401                          BEQ     4$                      ;YES GO ON
021636  104001                          ERROR   +1              ;CPU ERROR
                                                                        ;NO GO TO ERROR
021640                          4$:
                                ;
021640                          ALR4TS:
                                ;       ALTERNATE REGISTER SET R4 BIT TESTS
021640  012704  177777                  MOV     #177777,R4              ;R4=177777
021644  020427  177777                  CMP     R4,#177777              ;DOES R4=177777
021650  001401                          BEQ     1$                      ;YES GO ON
021652  104001                          ERROR   +1              ;CPU ERROR
                                                                        ;NO GO TO ERROR
021654  005004                  1$:     CLR     R4                      ;R4=0
021656  020427  000000                  CMP     R4,#0                   ;DOES R4=0
021662  001401                          BEQ     2$                      ;YES GO ON
021664  104001                          ERROR   +1              ;CPU ERROR
                                                                        ;NO GO TO ERROR
021666  012704  125252          2$:     MOV     #125252,R4              ;R4=125252
021672  020427  125252                  CMP     R4,#125252              ;DOES R4=125252
021676  001401                          BEQ     3$                      ;YES GO ON
021700  104001                          ERROR   +1              ;CPU ERROR
                                                                                ;NO GO TO ERROR
021702  012704  052525          3$:     MOV     #52525,R4               ;R4=52525
021706  020427  052525                  CMP     R4,#52525               ;DOES R4=52525
021712  001401                          BEQ     4$                      ;YES GO ON
```

```
5392  021714  104001                          ERROR   ·1              ;CPU ERROR
5393                                                                  ;NO GO TO ERROR
5394  021716                          4$:
5395                                  ;
5396  021716                          ALR5TS:
5397                                  ;       ALTERNATE REGISTER SET R5 BIT TESTS
5398  021716  012705  177777                  MOV     #177777,R5              ;R5=177777
5399  021722  020527  177777                  CMP     R5,#177777              ;DOES R5=177777
5400  021726  001401                          BEQ     1$                      ;YES GO ON
5401  021730  104001                          ERROR   ·1              ;CPU ERROR
5402                                                                  ;NO GO TO ERROR
5403  021732  005005                  1$:     CLR     R5                      ;R5=0
5404  021734  020527  000000                  CMP     R5,#0                   ;DOES R5=0
5405  021740  001401                          BEQ     2$                      ;YES GO ON
5406  021742  104001                          ERROR   ·1              ;CPU ERROR
5407                                                                  ;NO GO TO ERROR
5408  021744  012705  125252          2$:     MOV     #125252,R5              ;R5=125252
5409  021750  020527  125252                  CMP     R5,#125252              ;DOES R5=125252
5410  021754  001401                          BEQ     3$                      ;YES GO ON
5411  021756  104001                          ERROR   ·1              ;CPU ERROR
5412                                                                  ;NO GO TO ERROR
5413  021760  012705  052525          3$:     MOV     #52525,R5               ;R5=52525
5414  021764  020527  052525                  CMP     R5,#52525               ;DOES R5=52525
5415  021770  001401                          BEQ     4$                      ;YES GO ON
5416  021772  104001                          ERROR   ·1              ;CPU ERROR
5417                                                                  ;NO GO TO ERROR
5418  021774  042767  004000  155774  4$:     BIC     #BIT11,PS               ;RETURN TO PRIMARY GEN PURPOSE REGS
5419
5420                                  ;
5421  022002                          TE115:
5422                                  ;       TEST MOVE FROM PROCCESSOR STATUS INST (MFPS)
5423  022002  005004                          CLR     R4                      ;SETUP DESTINATION R4
5424
5425  022004  012701  022212                  MOV     #TE115A,R1              ;SETUP POINTERS TO TABLES
5426  022010  012702  022222                  MOV     #TE115B,R2              ;
5427  022014  012703  022230                  MOV     #TE115C,R3              ;
5428  022020  012137  177776          1$:     MOV     (R1)·,@#177776          ;SETUP PSW
5429  022024  106704                          MFPS    R4                      ; TEST INSTRUCTION
5430  022026  023722  177776                  CMP     @#177776,(R2)·          ;CHECK PSW
5431  022032  001401                          BEQ     2$                      ;OK GO ON
5432  022034  104001                          ERROR   ·1              ;CPU ERROR
5433                                                                  ;NO GO TO ERROR
5434  022036  020423                  2$:     CMP     R4,(R3)·                ;CHECK R4
5435  022040  001401                          BEQ     3$                      ;OK GO ON
5436  022042  104001                          ERROR   ·1              ;CPU ERROR
5437                                                                  ;NO GO TO ERROR
5438  022044  021127  177777          3$:     CMP     (R1),#177777            ;ARE WE DONE
5439  022050  001363                          BNE     1$                      ;NO GO TO 1$
5440
5441
5442  022052  012701  022212                  MOV     #TE115A,R1              ;SETUP POINTERS TO TABLES
5443  022056  012702  022222                  MOV     #TE115B,R2              ;
5444  022062  012703  022230                  MOV     #TE115C,R3              ;
5445  022066  010605                          MOV     R6,R5                   ;SAVE STACK IN R5
5446  022070  011137  177776          101$:   MOV     (R1),@#177776           ;SETUP PSW
5447  022074  106706                          MFPS    R6                      ; TEST INSTRUCTION
```

```
5448  022076  023712  177776          CMP     @#177776,(R2)           ;CHECK PSW
5449  022102  001401                  BEQ     102$                    ;OK GO ON
5450  022104  104001                  ERROR   .1                      ;CPU ERROR
5451                                                                  ;NO GO TO ERROR
5452  022106  020613          102$:   CMP     R6,(R3)                 ;CHECK R6
5453  022110  001401                  BEQ     103$                    ;OK GO ON
5454  022112  104001                  ERROR   .1                      ;CPU ERROR
5455                                                                  ;NO GO TO ERROR
5456  022114  010506          103$:   MOV     R5,R6                   ;RESTORE STACK
5457
5458
5459  022116  012701  022212          MOV     #TE115A,R1              ;SETUP POINTERS TO TABLES
5460  022122  012702  022222          MOV     #TE115B,R2              ;
5461  022126  012703  022236          MOV     #TE115D,R3              ;
5462  022132  005037  110502          CLR     @#EXPDAT                ;INIT EXPECTED DATA HOLDER.
5463  022136  012704  110502  4$:     MOV     #EXPDAT,R4              ;SETUP POINTER TO TEST LOCATION
5464  022142  012137  177776          MOV     (R1)+,@#177776          ;SETUP PSW
5465  022146  106724                  MFPS    (R4)+                   ; TEST INSTRUCTION
5466  022150  023722  177776          CMP     @#177776,(R2)+          ;CHECK PSW
5467  022154  001401                  BEQ     5$                      ;OK GO ON
5468  022156  104001                  ERROR   .1                      ;CPU ERROR
5469                                                                  ;NO GO TO ERROR
5470  022160  020427  110503  5$:     CMP     R4,#EXPDAT+1            ;CHECK R4
5471  022164  001401                  BEQ     6$                      ;OK GO ON
5472  022166  104001                  ERROR   .1                      ;CPU ERROR
5473                                                                  ;NO GO TO ERROR
5474  022170  023723  110502  6$:     CMP     @#EXPDAT,(R3)+          ;CHECK TEST LOCATION
5475  022174  001401                  BEQ     7$                      ;OK GO ON
5476  022176  104001                  ERROR   .1                      ;CPU ERROR
5477                                                                  ;NO GO TO ERROR
5478  022200  021127  177777  7$:     CMP     (R1),#177777            ;ARE WE DONE
5479  022204  001354                  BNE     4$                      ;NO GO TO 4$
5480
5481  022206  000167  000032          JMP     TE115F
5482
5483                                  ;
5484                                  ;
5485  022212  030207          TE115A: .WORD   30207
5486  022214  030000                  .WORD   30000
5487  022216  030057                  .WORD   30057
5488  022220  177777                  .WORD   177777
5489  022222  030211          TE115B: .WORD   30211
5490  022224  030004                  .WORD   30004
5491  022226  030041                  .WORD   30041
5492  022230  177607          TE115C: .WORD   177607
5493  022232  000000                  .WORD   0
5494  022234  000057                  .WORD   57
5495  022236  000207          TE115D: .WORD   207
5496  022240  000000                  .WORD   0
5497  022242  000057                  .WORD   57
5498  022244  000240          TE115F: NOP
5499                                  ;
5500  022246                  TE116:
5501                          ;       TEST MOVE TO PROCESSOR STATUS INST (MTPS)
5502
5503  022246  012737  030000  177776        MOV     #30000,@#177776         ;SET PSW TO KERNEL MODE
```

```
5504  022254  012701  022550           MOV     #TE116D,R1              ;SETUP POINTERS TO TABLES
5505  022260  012702  022526           MOV     #TE116B,R2              ;
5506  022264  010103                   MOV     R1,R3                   ;
5507  022266  004767  000136           JSR     PC,T116                 ; TEST INSTRUCTION AND CHECK PSW
5508                                                                    ;AND SOURCE OPERAND
5509
5510
5511  022272  012737  140000  177776   MOV     #140000,@#177776        ;SET PSW TO USER MODE
5512  022300  012706  000600           MOV     #600,R6                 ;SETUP USER STACK
5513  022304  012701  022550           MOV     #TE116D,R1              ;SETUP POINTERS TO TABLES
5514  022310  012702  022540           MOV     #TE116C,R2              ;
5515  022314  010103                   MOV     R1,R3                   ;
5516  022316  004767  000106           JSR     PC,T116                 ; TEST INSTRUCTION AND CHECK PSW
5517                                                                    ;AND SOURCE OPERAND
5518
5519
5520  022322  012737  140000  177776   MOV     #140000,@#177776        ;SET PSW TO USER MODE AND CLEAR CC BITS
5521  022330  012701  022522           MOV     #TE116A,R1              ;SETUP POINTERS TO TABLES
5522  022334  012702  022540           MOV     #TE116C,R2              ;
5523  022340  012703  022550           MOV     #TE116D,R3              ;
5524  022344  010104                   MOV     R1,R4                   ;SAVE A COPY OF R1 INTO R4
5525  022346  004767  000110           JSR     PC,TA116                ; TEST INSTRUCTION AND CHECK PSW
5526                                                                    ;AND SOURCE OPERAND
5527
5528
5529  022352  012737  030000  177776   MOV     #30000,@#177776         ;SET PSW TO KERNEL MODE
5530  022360  012701  022522           MOV     #TE116A,R1              ;SETUP POINTERS TO TABLES
5531  022364  012702  022526           MOV     #TE116B,R2              ;
5532  022370  012703  022550           MOV     #TE116D,R3              ;
5533  022374  010104                   MOV     R1,R4                   ;SAVE A COPY OF R1 INTO R4
5534  022376  004767  000060           JSR     PC,TA116                ; TEST INSTRUCTION AND CHECK PSW
5535                                                                    ;AND SOURCE OPERAND
5536
5537  022402  005037  177776           CLR     @#177776                ;SET PSW TO KERNEL MODE
5538  022406  106427  177412           MTPS    #177412                 ;TEST INSTRUCTION
5539  022412  022737  000012  177776   CMP     #12,@#177776            ;IS PSW CORRECT
5540  022420  001401                   BEQ     100$                    ;YES GO ON
5541  022422  104001                   ERROR   .1                      ;CPU ERROR
5542                                                                    ;NO GO TO ERROR
5543  022424                   100$:
5544  022424  000167  000130           JMP     FIN116
5545                           ;
5546  022430  012105           T116:   MOV     (R1)+,R5                ;MOVE TEST DATA TO R5
5547  022432  106405                   MTPS    R5                      ; TEST INSTRUCTION
5548  022434  023722  177776           CMP     @#177776,(R2)+          ;IS PSW CORRECT
5549  022440  001401                   BEQ     1$                      ;YES GO ON
5550  022442  104001                   ERROR   .1                      ;CPU ERROR
5551                                                                    ;NO GO TO ERROR
5552  022444  022305           1$:     CMP     (R3)+,R5                ;IS R5 CORRECT
5553  022446  001401                   BEQ     2$                      ;YES GO ON
5554  022450  104001                   ERROR   .1                      ;CPU ERROR
5555                                                                    ;NO GO TO ERROR
5556  022452  021227  177777   2$:     CMP     (R2),#177777            ;ARE WE DONE
5557  022456  001364                   BNE     T116                    ;NO GO TO T116
5558  022460  000207                   RTS     PC                      ;RETURN
5559                           ;
```

```
5560  022462  106421                  TA116:  MTPS    (R1)+                   ; TEST INSTRUCTION
5561  022464  023722  177776                  CMP     @#177776,(R2)+          ;IS PSW CORRECT
5562  022470  001401                          BEQ     1$                      ;YES GO ON
5563  022472  104001                          ERROR   +1              ;CPU ERROR
5564                                                                  ;NO GO TO ERROR
5565  022474  122423                  1$:     CMPB    (R4)+,(R3)+             ;CHECK TEST LOCATION
5566  022476  001401                          BEQ     2$                      ;OK GO ON
5567  022500  104001                          ERROR   +1              ;CPU ERROR
5568                                                                  ;NO GO TO ERROR
5569  022502  020104                  2$:     CMP     R1,R4                   ;IS SOURCE OPERAND CORRECT
5570  022504  001401                          BEQ     3$                      ;YES GO ON
5571  022506  104001                          ERROR   +1              ;CPU ERROR
5572                                                                  ;NO GO TO ERROR
5573  022510  005203                  3$:     INC     R3                      ;POINT TO NEXT WORD
5574  022512  021227  177777                  CMP     (R2),#177777            ;ARE WE DONE
5575  022516  001361                          BNE     TA116                   ;NO GO TO TA116
5576  022520  000207                          RTS     PC                      ;RETURN
5577                                          ;
5578  022522     377                  TE116A: .BYTE   377
5579  022523     000                          .BYTE   0
5580  022524     252                          .BYTE   252
5581  022525     125                          .BYTE   125
5582  022526  030357                  TE116B: .WORD   30357
5583  022530  030000                          .WORD   30000
5584  022532  030252                          .WORD   30252
5585  022534  030105                          .WORD   30105
5586  022536  177777                          .WORD   177777
5587  022540  140017                  TE116C: .WORD   140017
5588  022542  140000                          .WORD   140000
5589  022544  140012                          .WORD   140012
5590  022546  140005                          .WORD   140005
5591  022550  177777                  TE116D: .WORD   177777
5592  022552  177400                          .WORD   177400
5593  022554  177652                          .WORD   177652
5594  022556  177525                          .WORD   177525
5595                                          ;
5596  022560                          FIN116:
5597                                          ;
5598  022560                          TE117:
5599                                  ;       TEST MFPT (MOVE FROM PROCESSOR TYPE)
5600  022560  013746  000010                  MOV     @#10,-(SP)              ;SAVE VECTOR
5601  022564  012737  022672  000010          MOV     #TE117A,@#10            ;SETUP VECTOR TO HANDLE POSSIBLE ILLEGAL INST TR
5602  022572  012700  177777                  MOV     #177777,R0              ;INIT R0
5603  022576  012737  030000  177776          MOV     #30000,@#177776         ;SETUP PSW
5604  022604  000007                          .WORD   7                       ; TEST INSTRUCTION
5605  022606  022737  030000  177776          CMP     #30000,@#177776         ;IS PSW CORRECT
5606  022614  001401                          BEQ     1$                      ;YES GO ON
5607  022616  104001                          ERROR   +1              ;CPU ERROR
5608                                                                  ;NO GO TO ERROR
5609  022620  020027  000005          1$:     CMP     R0,#5                   ;IS R0 CORRECT
5610  022624  001401                          BEQ     2$                      ;YES GO ON
5611  022626  104001                          ERROR   +1              ;CPU ERROR
5612                                                                  ;NO GO TO ERROR
5613  022630  012700  177777          2$:     MOV     #177777,R0              ;INIT R0
5614  022634  000277                          SCC                             ;SET ALL CC BITS
5615  022636  000007                          .WORD   7                       ; TEST INSTRUCTION
```

```
5616  022640  022737  030017  177776          CMP     #30017,@#177776         ;IS PSW CORRECT
5617  022646  001401                          BEQ     3$                      ;YES GO ON
5618  022650  104001                          ERROR   .1              ;CPU ERROR
5619                                                                  ;NO GO TO ERROR
5620  022652  020027  000005          3$:     CMP     R0,#5                   ;IS R0 CORRECT
5621  022656  001401                          BEQ     4$                      ;YES GO ON
5622  022660  104001                          ERROR   .1              ;CPU ERROR
5623                                                                  ;NO GO TO ERROR
5624  022662  012637  000010          4$:     MOV     (SP)+,@#10              ;RESTORE VECTOR
5625
5626  022666  000167  000002                  JMP     FIN117
5627                                  ;
5628  022672  104001                  TE117A: ERROR   .1              ;CPU ERROR
5629                                                                  ;GO TO ERROR IF TRAP TAKES PLACE
5630                                  ;
5631  022674  000240                  FIN117: NOP
5632                                  ;
5633  022676                          TE120:
5634                                  ;       TEST HALT (NOT KERNEL MODE)
5635  022676  005037  177766                  CLR     @#177766                ;INIT CPU ERROR REG
5636  022702  005037  177776                  CLR     @#177776                ;INIT PSW-SET KERNEL MODE
5637  022706  013746  000004                  MOV     @#4,-(SP)               ;SAVE VECTOR
5638  022712  013746  000006                  MOV     @#6,-(SP)               ;SAVE VECTOR
5639  022716  012737  022752  000004          MOV     #TE120A,@#4             ;SET UP VECTOR TO HANDLE ILLEGAL HALT
5640  022724  005037  000006                  CLR     @#6                     ;SET UP VECTOR TO COME BACK IN KERNEL MODE
5641  022730  012767  140000  155040          MOV     #140000,PS              ;SET IN USER MODE
5642  022736  012706  000600                  MOV     #600,R6                 ;INITIALIZE THE USER STACK POINTER
5643  022742  000000                          HALT                            ; TEST INSTRUCTION
5644  022744  104001                  PROCNT: ERROR   .1              ;CPU ERROR
5645                                                                  ;IF NOTHING HAPPENED GO TO ERROR
5646  022746  000167  000044                  JMP     FIN120                  ;
5647                                  ;
5648                                  ;
5649  022752  022737  030000  177776  TE120A: CMP     #30000,@#177776         ;IS PSW CORRECT/PREVIOUS MODE = USER?
5650  022760  001401                          BEQ     1$                      ;YES GO ON
5651  022762  104001                          ERROR   .1              ;CPU ERROR
5652                                                                  ;NO GO TO ERROR
5653  022764  022737  000200  177766  1$:     CMP     #200,@#177766           ; TEST CPU ERROR REGISTER
5654  022772  001401                          BEQ     2$                      ;YES GO ON
5655  022774  104001                          ERROR   .1              ;CPU ERROR
5656                                                                  ;NO GO TO ERROR
5657  022776  022627  022744          2$:     CMP     (SP)+,#PROCNT           ;DOES STACK CONTAIN CORRECT PC
5658  023002  001401                          BEQ     3$                      ;YES GO ON
5659  023004  104001                          ERROR   .1              ;CPU ERROR
5660                                                                  ;NO GO TO ERROR
5661  023006  022627  140000          3$:     CMP     (SP)+,#140000           ;DOES STACK CONTAIN CORRECT PSW
5662  023012  001401                          BEQ     FIN120                  ;YES GO ON
5663  023014  104001                          ERROR   .1              ;CPU ERROR
5664                                                                  ;NO GO TO ERROR
5665  023016  012637  000006          FIN120: MOV     (SP)+,@#6               ;RESTORE VECTOR
5666  023022  012637  000004                  MOV     (SP)+,@#4               ;RESTORE VECTOR
5667
5668                                  ;
5669  023026                          TE121:
5670                                  ;       TEST RESET
5671  023026  122767  000001  156164          CMPB    #APTENV,$ENV            ;ARE WE IN APT MODE?
```

```
5672  023034  001002                          BNE     1$                      ;IF NOT: DO THIS TEST
5673  023036  000167  000622                  JMP     FIN122                  ;ELSE SKIP THIS TEST BECAUSE RESETS
5674                                                                          ;SCREW UP THE APT MONITOR.
5675  023042  012737  030340  177776  1$:     MOV     #030340,@#177776        ;SETUP PSW TO KERNEL MODE
5676  023050  012737  160000  177572          MOV     #160000,@#177572        ;SETUP MMR0
5677  023056  012737  000077  172516          MOV     #77,@#172516            ;SETUP MMR3
5678  023064  005037  177772                  CLR     @#177772                ;CLEAR PIRQ
5679  023070  023727  177772  000000          CMP     @#177772,#0             ;IS PIRQ CORRECT
5680  023076  001401                          BEQ     C121A                   ;YES GO ON
5681  023100  104001                          ERROR   .1              ;CPU ERROR
5682                                                                  ;NO GO TO ERROR
5683  023102  012737  025000  177772  C121A:  MOV     #25000,@#177772         ;MOVE AN ALTERNATING PATTERN TO PIRQ
5684  023110  022737  025252  177772          CMP     #25252,@#177772         ;IS PIRQ CORRECT
5685  023116  001401                          BEQ     C121B                   ;YES GO ON
5686  023120  104001                          ERROR   .1              ;CPU ERROR
5687                                                                  ;NO GO TO ERROR
5688  023122  012737  077000  177772  C121B:  MOV     #77000,@#177772         ;SETUP PIRQ
5689  023130  022737  077314  177772          CMP     #77314,@#177772         ;IS PIRQ CORRECT
5690  023136  001401                          BEQ     C121C                   ;YES GO ON
5691  023140  104001                          ERROR   .1              ;CPU ERROR
5692                                                                  ;NO GO TO ERROR
5693  023142  000277                  C121C:  SCC                             ;SET ALL CC BITS
5694  023144  000005                          RESET                           ; TEST INSTRUCTION
5695  023146  022737  030357  177776          CMP     #030357,@#177776        ;IS PSW CORRECT
5696  023154  001401                          BEQ     1$                      ;YES GO ON
5697  023156  104001                          ERROR   .1              ;CPU ERROR
5698                                                                  ;NO GO TO ERROR
5699  023160  013701  177572          1$:     MOV     @#SR0,R1                ;SAVE SR0 IN R1.
5700  023164  042701  000176                  BIC     #176,R1                 ;STRIP OFF UNDEFINED BITS 1-6 FROM MMR0
5701  023170  022701  000000                  CMP     #0,R1                   ;IS MMR0 CORRECT
5702  023174  001401                          BEQ     2$                      ;YES GO ON
5703  023176  104001                          ERROR   .1              ;CPU ERROR
5704                                                                  ;NO GO TO ERROR
5705  023200  022737  000000  172516  2$:     CMP     #0,@#172516             ;IS MMR3 CORRECT
5706  023206  001401                          BEQ     3$                      ;YES GO ON
5707  023210  104001                          ERROR   .1              ;CPU ERROR
5708                                                                  ;NO GO TO ERROR
5709  023212  022737  000000  177772  3$:     CMP     #0,@#177772             ;IS PIRQ CORRECT
5710  023220  001401                          BEQ     4$                      ;YES GO ON
5711  023222  104001                          ERROR   .1              ;CPU ERROR
5712                                                                  ;NO GO TO ERROR
5713
5714  023224  013702  000004          4$:     MOV     @#4,R2                  ;SAVE LOC 4 IN R2
5715  023230  013703  000006                  MOV     @#6,R3                  ;SAVE LOC 6 IN R3
5716  023234  012737  023372  000004          MOV     #8$,@#4                 ;SETUP VECTORS IN CASE AN ERROR CAUSES
5717                                                                          ;A HALT TO OCCUR IN USER MODE.
5718  023242  012737  000340  000006          MOV     #340,@#6                ;THIS SETS KERNEL MODE, AND PREVENTS PRIQ
5719                                                                          ;INTERRUPTS FROM TRASHING THE STACK IF A
5720                                                                          ;USER MODE HALT OCCURS.
5721  023250  012737  140340  177776          MOV     #140340,@#177776        ;SETUP PSW TO USER MODE
5722  023256  012737  160000  177572          MOV     #160000,@#177572        ;SETUP MMR0
5723  023264  012737  000077  172516          MOV     #77,@#172516            ;SETUP MMR3
5724  023272  012737  077000  177772          MOV     #77000,@#177772         ;SETUP PIRQ
5725  023300  000277                          SCC                             ;SET ALL CC BITS
5726  023302  000005                          RESET                           ; TEST INSTRUCTION
5727  023304  022737  140357  177776          CMP     #140357,@#177776        ;IS PSW CORRECT
```

```
5728  023312  001402                        BEQ     5$                      ;YES GO ON
5729  023314  000000                        HALT                            ;USER MODE HALT; WILL TRAP TO LOC 4
5730  023316  104001                        ERROR   .1              ;CPU ERROR
5731                                                                ;NO GO TO ERROR
5732  023320  013701  177572          5$:   MOV     @#177572,R1             ;SAVE MMR0 IN R1
5733  023324  042701  000176              BIC     #176,R1                 ;CLEAR BITS 1-6 FROM MMR0
5734  023330  022701  160000              CMP     #160000,R1              ;IS MMR0 CORRECT
5735  023334  001402                        BEQ     6$                      ;YES GO ON
5736  023336  000000                        HALT                            ;USER MODE HALT; WILL TRAP TO LOC 4
5737  023340  104001                        ERROR   .1              ;CPU ERROR
5738                                                                ;NO GO TO ERROR
5739  023342  022737  000077  172516  6$:   CMP     #77,@#172516            ;IS MMR3 CORRECT
5740  023350  001402                        BEQ     7$                      ;YES GO ON
5741  023352  000000                        HALT                            ;USER MODE HALT; WILL TRAP TO LOC 4
5742  023354  104001                        ERROR   .1              ;CPU ERROR
5743                                                                ;NO GO TO ERROR
5744  023356  022737  077314  177772  7$:   CMP     #77314,@#177772         ;IS PIRQ CORRECT
5745  023364  001403                        BEQ     9$                      ;YES GO ON
5746  023366  000000                        HALT                            ;USER MODE HALT; WILL TRAP TO LOC 4
5747  023370  104001                        ERROR   .1              ;CPU ERROR
5748                                                                ;NO GO TO ERROR
5749  023372  000002                  8$:   RTI                             ;USER MODE HALT OCCURRED; GO TO ERROR.
5750
5751  023374  005037  177772          9$:   CLR     @#177772                ;CLEAR PIRQ
5752  023400  005037  177776              CLR     @#177776                ;CLEAR PSW
5753  023404  010237  000004              MOV     R2,@#4                  ;RESTORE VECTORS TO PREVIOUS STATE
5754  023410  010337  000006              MOV     R3,@#6                  ;
5755  023414                        FIN121:
5756                                  ;
5757  023414                        TE122:
5758                                  ;       TEST SPL (SET PRIORITY LEVEL)
5759
5760  023414  012705  000010              MOV     #8.,R5                  ;INIT COUNTER
5761  023420  012701  023644              MOV     #T122B,R1               ;SETUP POINTER TO DATA
5762  023424  012737  030000  177776  1$:   MOV     #30000,@#177776         ;INIT PSW
5763  023432  004767  000054              JSR     PC,T122A                ; TEST INSTRUCTION
5764  023436  022137  177776              CMP     (R1)+,@#177776          ;IS PSW CORRECT
5765  023442  001401                        BEQ     2$                      ;YES GO ON
5766  023444  104001                        ERROR   .1              ;CPU ERROR
5767                                                                ;NO GO TO ERROR
5768  023446  077512                  2$:   SOB     R5,1$                   ;REPEAT UNTIL ALL CASES ARE TESTED
5769
5770
5771  023450  012705  000010              MOV     #8.,R5                  ;INIT COUNTER
5772  023454  012737  140000  177776  3$:   MOV     #140000,@#177776        ;SETUP PSW TO USER MODE
5773  023462  012706  000600              MOV     #600,R6                 ;SETUP USER STACK
5774  023466  004767  000020              JSR     PC,T122A                ; TEST INSTRUCTION
5775  023472  022737  140017  177776      CMP     #140017,@#177776        ;IS PSW CORRECT
5776  023500  001401                        BEQ     4$                      ;YES GO ON
5777  023502  104001                        ERROR   .1              ;CPU ERROR
5778                                                                ;NO GO TO ERROR
5779  023504  077515                  4$:   SOB     R5,3$                   ;REPEAT UNTIL ALL CASES ARE TESTED
5780
5781
5782  023506  000167  000152              JMP     FIN122
5783                                  ;
```

```
5784  023512  020527  000010          T122A:  CMP     R5,#8.                  ;FIND OUT WHAT COUNTER IS
5785  023516  001003                          BNE     1$                      ;IF NOT PRIORITY 0 GO TO 1$
5786  023520  000277                          SCC                             ;SET ALL CC BITS
5787  023522  000230                          SPL     0                       ;SET PRIORITY TO 0
5788  023524  000446                          BR      8$                      ;RETURN
5789  023526  020527  000007          1$:     CMP     R5,#7                   ;FIND OUT WHAT COUNTER IS
5790  023532  001003                          BNE     2$                      ;IF NOT PRIORITY 1 GO TO 2$
5791  023534  000277                          SCC                             ;SET ALL CC BITS
5792  023536  000231                          SPL     1                       ;SET PRIORITY TO 1
5793  023540  000440                          BR      8$                      ;RETURN
5794  023542  020527  000006          2$:     CMP     R5,#6                   ;FIND OUT WHAT COUNTER IS
5795  023546  001003                          BNE     3$                      ;IF NOT PRIORITY 2 GO TO 3$
5796  023550  000277                          SCC                             ;SET ALL CC BITS
5797  023552  000232                          SPL     2                       ;SET PRIORITY TO 2
5798  023554  000432                          BR      8$                      ;RETURN
5799  023556  020527  000005          3$:     CMP     R5,#5                   ;FIND OUT WHAT COUNTER IS
5800  023562  001003                          BNE     4$                      ;IF NOT PRIORITY 3 GO TO 4$
5801  023564  000277                          SCC                             ;SET ALL CC BITS
5802  023566  000233                          SPL     3                       ;SET PRIORITY TO 3
5803  023570  000424                          BR      8$                      ;RETURN
5804  023572  020527  000004          4$:     CMP     R5,#4                   ;FIND OUT WHAT COUNTER IS
5805  023576  001003                          BNE     5$                      ;IF NOT PRIORITY 4 GO TO 5$
5806  023600  000277                          SCC                             ;SET ALL CC BITS
5807  023602  000234                          SPL     4                       ;SET PRIORITY TO 4
5808  023604  000416                          BR      8$                      ;RETURN
5809  023606  020527  000003          5$:     CMP     R5,#3                   ;FIND OUT WHAT COUNTER IS
5810  023612  001003                          BNE     6$                      ;IF NOT PRIORITY 5 GO TO 6$
5811  023614  000277                          SCC                             ;SET ALL CC BITS
5812  023616  000235                          SPL     5                       ;SET PRIORITY TO 5
5813  023620  000410                          BR      8$                      ;RETURN
5814  023622  020527  000002          6$:     CMP     R5,#2                   ;FIND OUT WHAT COUNTER IS
5815  023626  001003                          BNE     7$                      ;IF NOT PRIORITY 6 GO TO 7$
5816  023630  000277                          SCC                             ;SET ALL CC BITS
5817  023632  000236                          SPL     6                       ;SET PRIORITY TO 6
5818  023634  000402                          BR      8$                      ;RETURN
5819  023636  000277                  7$:     SCC                             ;SET ALL CC BITS
5820  023640  000237                          SPL     7                       ;SET PRIORITY TO 7
5821  023642  000207                  8$:     RTS     PC                      ;RETURN
5822                                  ;
5823  023644  030017                  T122B:  .WORD   30017
5824  023646  030057                          .WORD   30057
5825  023650  030117                          .WORD   30117
5826  023652  030157                          .WORD   30157
5827  023654  030217                          .WORD   30217
5828  023656  030257                          .WORD   30257
5829  023660  030317                          .WORD   30317
5830  023662  030357                          .WORD   30357
5831  023664  000240                  FIN122: NOP
5832                                  ;
5833  023666                          TE123:
5834                                  ;       TEST TSTSET INSTRUCTION (MULTI PROCESSING INST)
5835  023666  005037  177776                  CLR     @#177776                ;INIT PSW
5836  023672  012703  000012                  MOV     #10.,R3                 ;INIT COUNTER
5837  023676  012701  000400                  MOV     #400,R1                 ;SETUP DESTINATION
5838  023702  012700  024050                  MOV     #T123A,R0               ;SETUP SOURCE
5839  023706  012021                  100$:   MOV     (R0)+,(R1)+             ;RELOCATE TABLES
```

```
5840  023710  077302                        SOB     R3,100$                 ;ARE WE DONE
5841  023712  013746  000010                MOV     @#10,-(SP)              ;SAVE VECTOR
5842  023716  012737  024074  000010        MOV     #T123D,@#10             ;SETUP NEW VECTOR
5843  023724  005000                        CLR     R0                      ;INIT R0
5844  023726  012701  000400                MOV     #400,R1                 ;SETUP POINTERS TO TABLES
5845  023732  012702  000410                MOV     #410,R2                 ;
5846  023736  012703  000416                MOV     #416,R3                 ;
5847  023742  010104                        MOV     R1,R4                   ;
5848  023744  012737  030000  177776  1$:   MOV     #30000,@#177776         ;SETUP PSW
5849  023752  000262                        SEV                             ;SET V BIT
5850  023754  007221                        .WORD   7221                    ; TEST INSTRUCTION
5851  023756  022237  177776                CMP     (R2)+,@#177776          ;IS PSW CORRECT
5852  023762  001401                        BEQ     2$                      ;YES GO ON
5853  023764  104001                        ERROR   .1                      ;CPU ERROR
5854                                                                        ;NO GO TO ERROR
5855  023766  020013                2$:     CMP     R0,(R3)                 ;IS R0 CORRECT
5856  023770  001401                        BEQ     3$                      ;YES GO ON
5857  023772  104001                        ERROR   .1                      ;CPU ERROR
5858                                                                        ;NO GO TO ERROR
5859  023774  005204                3$:     INC     R4                      ;SETUP EXPECTED DATA
5860  023776  005204                        INC     R4                      ;
5861  024000  020401                        CMP     R4,R1                   ;IS R1 CORRECT
5862  024002  001401                        BEQ     4$                      ;YES GO ON
5863  024004  104001                        ERROR   .1                      ;CPU ERROR
5864                                                                        ;NO GO TO ERROR
5865  024006  052713  000001          4$:   BIS     #1,(R3)                 ;SETUP EXPECTED DATA
5866  024012  022341                        CMP     (R3)+,-(R1)             ;IS TEST LOCATION CORRECT
5867  024014  001401                        BEQ     5$                      ;YES GO ON
5868  024016  104001                        ERROR   .1                      ;CPU ERROR
5869                                                                        ;NO GO TO ERROR
5870  024020  005201                5$:     INC     R1                      ;POINT TO NEXT TEST LOCATION
5871  024022  005201                        INC     R1                      ;
5872  024024  021127  177777                CMP     (R1),#177777            ;ARE WE DONE
5873  024030  001345                        BNE     1$                      ;NO GO TO 1$
5874  024032  012737  024076  000010        MOV     #T123E,@#10             ;SETUP NEW VECTOR
5875  024040  007201                        .WORD   7201                    ; TEST INSTRUCTION ILLEGAL MODE
5876  024042  104001                        ERROR   .1                      ;CPU ERROR
5877                                                                        ;GO TO ERROR IF DIDN'T TRAP
5878  024044  000167  000032                JMP     T123F
5879                                ;
5880                                ;
5881  024050  167604                T123A:  .WORD   167604
5882  024052  000000                        .WORD   0
5883  024054  000001                        .WORD   1
5884  024056  177777                        .WORD   177777
5885  024060  030010                T123B:  .WORD   30010
5886  024062  030004                        .WORD   30004
5887  024064  030001                        .WORD   30001
5888  024066  167604                T123C:  .WORD   167604
5889  024070  000000                        .WORD   0
5890  024072  000001                        .WORD   1
5891  024074  104001                T123D:  ERROR   .1                      ;CPU ERROR
5892                                                                        ;GO TO ERROR IF TRAPPED
5893  024076  005726                T123E:  TST     (SP)+                   ;CLEAN UP STACK
5894  024100  005726                        TST     (SP)+                   ;
5895  024102  012637  000010        T123F:  MOV     (SP)+,@#10              ;RESTORE VECTOR
```

```
5896
5897                                        ;
5898  024106                                TE124:
5899                                        ;       TEST WRTLCK (WRITE LOCK MULTI PROCESSING INST)
5900  024106  005037  177776                       CLR     @#177776                ;INIT PSW
5901  024112  012703  000012                       MOV     #10.,R3                 ;INIT COUNTER
5902  024116  012701  000400                       MOV     #400,R1                 ;SETUP DESTINATION
5903  024122  012700  024302                       MOV     #T124A,R0               ;SETUP SOURCE
5904  024126  012021                        100$:  MOV     (R0)+,(R1)+             ;RELOCATE TABLES
5905  024130  077302                               SOB     R3,100$                 ;ARE WE DONE
5906  024132  013746  000010                       MOV     @#10,-(SP)              ;SAVE VECTOR
5907  024136  012737  024326  000010               MOV     #T124D,@#10             ;SETUP NEW VECTOR
5908  024144  012701  000400                       MOV     #400,R1                 ;SETUP POINTERS TO TABLES
5909  024150  012702  000410                       MOV     #410,R2                 ;
5910  024154  012703  000416                       MOV     #416,R3                 ;
5911  024160  010204                               MOV     R2,R4                   ;
5912  024162  012737  030000  177776  1$:    MOV     #30000,@#177776         ;SETUP PSW
5913  024170  011100                               MOV     (R1),R0                 ;SETUP R0
5914  024172  020327  000416                       CMP     R3,#416                 ;IS THIS THE FIRST TEST CASE
5915  024176  001401                               BEQ     2$                      ;YES GO TO 2$
5916  024200  000402                               BR      3$                      ;NO GO TO 3$
5917  024202  000261                        2$:    SEC                             ;SET C BIT
5918  024204  000401                               BR      4$                      ;
5919  024206  000241                        3$:    CLC                             ;CLEAR C BIT
5920  024210  000262                        4$:    SEV                             ;SET V BIT
5921  024212  007322                               .WORD   7322                    ; TEST INSTRUCTION
5922  024214  022337  177776                       CMP     (R3)+,@#177776          ;IS PSW CORRECT
5923  024220  001401                               BEQ     5$                      ;YES GO ON
5924  024222  104001                               ERROR   +1              ;CPU ERROR
5925                                                                    ;NO GO TO ERROR
5926  024224  021100                        5$:    CMP     (R1),R0                 ;IS R0 CORRECT
5927  024226  001401                               BEQ     6$                      ;YES GO ON
5928  024230  104001                               ERROR   +1              ;CPU ERROR
5929                                                                    ;NO GO TO ERROR
5930  024232  005204                        6$:    INC     R4                      ;SETUP EXPECTED DATA
5931  024234  005204                               INC     R4                      ;
5932  024236  020204                               CMP     R2,R4                   ;IS R2 CORRECT
5933  024240  001401                               BEQ     7$                      ;YES GO ON
5934  024242  104001                               ERROR   +1              ;CPU ERROR
5935                                                                    ;NO GO TO ERROR
5936  024244  022142                        7$:    CMP     (R1)+,-(R2)             ;IS TEST LOCATION CORRECT
5937  024246  001401                               BEQ     8$                      ;YES GO ON
5938  024250  104001                               ERROR   +1              ;CPU ERROR
5939                                                                    ;NO GO TO ERROR
5940  024252  005202                        8$:    INC     R2                      ;POINT TO NEXT TEST LOCATION
5941  024254  005202                               INC     R2                      ;
5942  024256  021127  177777                       CMP     (R1),#177777            ;ARE WE DONE
5943  024262  001337                               BNE     1$                      ;NO GO TO 1$
5944  024264  012737  024330  000010               MOV     #T124E,@#10             ;SETUP NEW VECTOR
5945  024272  007302                               .WORD   7302                    ; TEST INSTRUCTION ILLEGAL MODE
5946  024274  104001                               ERROR   +1              ;CPU ERROR
5947                                                                    ;GO TO ERROR IF DIDN'T TRAP
5948  024276  000167  000032                       JMP     T124F
5949                                        ;
5950                                        ;
5951  024302  167604                        T124A: .WORD   167604
```

```
5952  024304  000000                  .WORD   0
5953  024306  000001                  .WORD   1
5954  024310  177777                  .WORD   177777
5955  024312  177777          T124B:  .WORD   177777
5956  024314  177777                  .WORD   177777
5957  024316  177777                  .WORD   177777
5958  024320  030011          T124C:  .WORD   30011
5959  024322  030004                  .WORD   30004
5960  024324  030000                  .WORD   30000
5961  024326  104001          T124D:  ERROR   .1              ;CPU ERROR
5962                                                          ;GO TO ERROR IF TRAPPED
5963  024330  005726          T124E:  TST     (SP)+                   ;CLEAN UP STACK
5964  024332  005726                  TST     (SP)+                   ;
5965  024334  012637  000010  T124F:  MOV     (SP)+,@#10              ;RESTORE VECTOR
5966
5967                          ;
5968  024340                  TE125:
5969                          ;       TEST MUL (MULTIPLY INST)
5970  024340  005037  177776          CLR     @#177776                ;INIT PS
5971  024344  012701  024600          MOV     #TE125A,R1              ;SETUP POINTERS TO TABLES
5972
5973  024350  010137  110502  1$:     MOV     R1,@#EXPDAT             ;
5974  024354  062737  000002  110502  ADD     #2,@#EXPDAT             ;POINT TO SOURCE
5975  024362  012703  122222          MOV     #122222,R3              ;INIT R3 TO A KNOWN STATE
5976  024366  011102                  MOV     (R1),R2         ;INIT DESTINATION REG
5977  024370  000277                  SCC                             ;SET ALL CC BITS
5978  024372  070261  000002          MUL     2(R1),R2                ; TEST INSTRUCTION
5979  024376  026137  000004  177776  CMP     4(R1),@#177776          ;IS PS CORRECT
5980  024404  001401                  BEQ     2$                      ;YES GO ON
5981  024406  104001                  ERROR   .1              ;CPU ERROR
5982                                                          ;NO GO TO ERROR
5983  024410  026103  000006  2$:     CMP     6(R1),R3                ;IS R3 CORRECT
5984  024414  001401                  BEQ     3$                      ;YES GO ON
5985  024416  104001                  ERROR   .1              ;CPU ERROR
5986                                                          ;NO GO TO ERROR
5987  024420  026102  000010  3$:     CMP     10(R1),R2               ;IS R2 CORRECT
5988  024424  001401                  BEQ     4$                      ;YES GO ON
5989  024426  104001                  ERROR   .1              ;CPU ERROR
5990                                                          ;NO GO TO ERROR
5991  024430  026177  000002  064044  4$:     CMP     2(R1),@EXPDAT           ;IS SOURCE LOCATION OK
5992  024436  001401                  BEQ     5$                      ;YES GO ON
5993  024440  104001                  ERROR   .1              ;CPU ERROR
5994                                                          ;NO GO TO ERROR
5995  024442  062701  000012  5$:     ADD     #12,R1                  ;GO TO NEXT TEST
5996  024446  020127  025026          CMP     R1,#FIN125              ;ARE WE FINISHED
5997  024452  001336                  BNE     1$                      ;NO GO TO 1$
5998
5999                          ;
6000                          ;SECOND PART
6001                          ;USING ODD REGISTER
6002                          ;
6003  024454  012701  024600  6$:     MOV     #TE125A,R1              ;SETUP POINTERS TO TABLES
6004  024460                  7$:
6005  024460  010102                  MOV     R1,R2                   ;
6006  024462  012706  001000          MOV     #STBOT,R6               ;INIT R6 TO A KNOWN STATE
6007  024466  012704  000004          MOV     #4,R4                   ;SETUP R4 VALUE
```

```
6008  024472  011105                        MOV     (R1),R5         ;INIT DESTINATION REG
6009  024474  000277                        SCC                             ;SET ALL CC BITS
6010  024476  070561  000002                MUL     2(R1),R5                ; TEST INSTRUCTION
6011  024502  026137  000004  177776        CMP     4(R1),#177776           ;IS PS CORRECT
6012  024510  001401                        BEQ     8$                      ;YES GO ON
6013  024512  104001                        ERROR   .1              ;CPU ERROR
6014                                                                  ;NO GO TO ERROR
6015  024514  026105  000006          8$:   CMP     6(R1),R5                ;IS R5 CORRECT
6016  024520  001401                        BEQ     9$                      ;YES GO ON
6017  024522  104001                        ERROR   .1              ;CPU ERROR
6018                                                                  ;NO GO TO ERROR
6019  024524  020627  001000          9$:   CMP     R6,#STBOT               ;IS R6 CORRECT
6020  024530  001403                        BEQ     10$                     ;YES GO ON
6021  024532  012706  001000                MOV     #STBOT,R6               ;RESTORE SP
6022  024536  104001                        ERROR   .1              ;CPU ERROR
6023                                                                  ;NO GO TO ERROR
6024  024540  005722                  10$:  TST     (R2)+                   ;POINT TO SOURCE OPERAND
6025  024542  021261  000002                CMP     (R2),2(R1)              ;IS SOURCE LOCATION OK
6026  024546  001401                        BEQ     11$                     ;YES GO ON
6027  024550  104001                        ERROR   .1              ;CPU ERROR
6028                                                                  ;NO GO TO ERROR
6029  024552  020427  000004          11$:  CMP     R4,#4                   ;IS R4 CORRECT
6030  024556  001401                        BEQ     12$                     ;YES GO ON
6031  024560  104001                        ERROR   .1              ;CPU ERROR
6032                                                                  ;NO GO TO ERROR
6033  024562  062701  000012          12$:  ADD     #12,R1                  ;
6034  024566  020127  025026                CMP     R1,#FIN125              ;ARE WE FINISHED
6035  024572  001332                        BNE     7$                      ;NO GO TO 7$
6036
6037
6038  024574  000167  000226                JMP     FIN125
6039                                  ;
6040                                  ;
6041  024600  177777                  TE125A: .WORD   177777                  ;MULTIPLICAND
6042  024602  177777                          .WORD   177777                  ;MULTIPLIER
6043  024604  000000                          .WORD   0
6044  024606  000001                          .WORD   1
6045  024610  000000                          .WORD   0
6046
6047  024612  006772                          .WORD   6772                    ;MULTIPLICAND
6048  024614  100000                          .WORD   100000                  ;MULTIPLIER
6049  024616  000011                          .WORD   11
6050  024620  000000                          .WORD   0
6051  024622  174403                          .WORD   174403
6052
6053  024624  177777                          .WORD   177777                  ;MULTIPLICAND
6054  024626  077777                          .WORD   77777                   ;MULTIPLIER
6055  024630  000010                          .WORD   10
6056  024632  100001                          .WORD   100001
6057  024634  177777                          .WORD   177777
6058
6059  024636  077777                          .WORD   77777                   ;MULTIPLICAND
6060  024640  000456                          .WORD   456                     ;MULTIPLIER
6061  024642  000001                          .WORD   1
6062  024644  177322                          .WORD   177322
6063  024646  000226                          .WORD   226
```

```
6064
6065  024650  173210                  .WORD   173210          ;MULTIPLICAND
6066  024652  000000                  .WORD   0               ;MULTIPLIER
6067  024654  000004                  .WORD   4
6068  024656  000000                  .WORD   0
6069  024660  000000                  .WORD   0
6070
6071  024662  000000                  .WORD   0               ;MULTIPLICAND
6072  024664  003251                  .WORD   3251            ;MULTIPLIER
6073  024666  000004                  .WORD   4
6074  024670  000000                  .WORD   0
6075  024672  000000                  .WORD   0
6076
6077  024674  000000                  .WORD   0               ;MULTIPLICAND
6078  024676  000000                  .WORD   0               ;MULTIPLIER
6079  024700  000004                  .WORD   4
6080  024702  000000                  .WORD   0
6081  024704  000000                  .WORD   0
6082
6083  024706  100000                  .WORD   100000          ;MULTIPLICAND
6084  024710  000001                  .WORD   1               ;MULTIPLIER
6085  024712  000010                  .WORD   10
6086  024714  100000                  .WORD   100000
6087  024716  177777                  .WORD   177777
6088
6089  024720  077777                  .WORD   77777           ;MULTIPLICAND
6090  024722  000001                  .WORD   1               ;MULTIPLIER
6091  024724  000000                  .WORD   0
6092  024726  077777                  .WORD   77777
6093  024730  000000                  .WORD   0
6094
6095  024732  000010                  .WORD   10              ;MULTIPLICAND
6096  024734  010000                  .WORD   10000           ;MULTIPLIER
6097  024736  000001                  .WORD   1
6098  024740  100000                  .WORD   100000
6099  024742  000000                  .WORD   0
6100
6101  024744  001452                  .WORD   1452            ;MULTIPLICAND
6102  024746  034527                  .WORD   34527           ;MULTIPLIER
6103  024750  000001                  .WORD   1
6104  024752  066506                  .WORD   66506
6105  024754  000265                  .WORD   265
6106
6107  024756  000007                  .WORD   7               ;MULTIPLICAND
6108  024760  000400                  .WORD   400             ;MULTIPLIER
6109  024762  000000                  .WORD   0
6110  024764  003400                  .WORD   3400
6111  024766  000000                  .WORD   0
6112
6113  024770  000002                  .WORD   2               ;MULTIPLICAND
6114  024772  100000                  .WORD   100000          ;MULTIPLIER
6115  024774  000011                  .WORD   11
6116  024776  000000                  .WORD   0
6117  025000  177777                  .WORD   177777
6118
6119  025002  100000                  .WORD   100000          ;MULTIPLICAND
```

```
6120  025004  077777                          .WORD   77777                   ;MULTIPLIER
6121  025006  000011                          .WORD   11
6122  025010  100000                          .WORD   100000
6123  025012  140000                          .WORD   140000
6124
6125  025014  000001                          .WORD   1                       ;MULTIPLICAND
6126  025016  177777                          .WORD   177777                  ;MULTIPLIER
6127  025020  000010                          .WORD   10
6128  025022  177777                          .WORD   177777
6129  025024  177777                          .WORD   177777
6130  025026                          FIN125:
6131                                  ;
6132  025026                          TE126:
6133                                  ;       TEST DIV (DIVIDE INST)
6134  025026  005037  177776                  CLR     @#177776                ;INIT PSW
6135  025032  005006                          CLR     R6                      ;INIT SP
6136  025034  013705  000000                  MOV     @#0,R5                  ;SAVE VECTORS
6137  025040  013701  000002                  MOV     @#2,R1                  ;
6138  025044  012737  000137  000000          MOV     #137,@#0                ;SETUP NEW VECTORS
6139  025052  012737  025074  000002          MOV     #TE126A,@#2             ;
6140  025060  000277                          SCC                             ;SET ALL CC BITS
6141  025062  071627  000002                  DIV     #2,R6                   ; TEST INSTRUCTION
6142  025066  012706  001000          A126:   MOV     #STBOT,R6               ;RESTORE SP BEFORE GOING TO ERROR
6143  025072  104001                          ERROR   ·1                      ;CPU ERROR
6144                                                                          ;IF R7 ISN'T CORRECT GO TO ERROR
6145  025074  022737  000000  177776  TE126A: CMP     #0,@#177776             ;IS PS CORRECT
6146  025102  001403                          BEQ     1$                      ;YES GO ON
6147  025104  012706  001000                  MOV     #STBOT,R6               ;RESTORE SP BEFORE GOING TO ERROR
6148  025110  104001                          ERROR   ·1                      ;CPU ERROR
6149                                                                          ;NO GO TO ERROR
6150  025112  012704  025066          1$:     MOV     #A126,R4                ;SETUP EXPECTED DATA
6151  025116  006204                          ASR     R4                      ;
6152  025120  020406                          CMP     R4,R6                   ;IS R6 CORRECT
6153  025122  001403                          BEQ     2$                      ;YES GO ON
6154  025124  012706  001000                  MOV     #STBOT,R6               ;RESTORE SP BEFORE GOING TO ERROR
6155  025130  104001                          ERROR   ·1                      ;CPU ERROR
6156                                                                          ;NO GO TO ERROR
6157  025132  010537  000000          2$:     MOV     R5,@#0                  ;RESTORE VECTORS
6158  025136  010137  000002                  MOV     R1,@#2                  ;
6159  025142  012706  001000                  MOV     #STBOT,R6               ;INIT SP
6160  025146  012702  000006                  MOV     #6,R2                   ;INIT GPR 2
6161  025152  012703  000047                  MOV     #47,R3                  ;INIT GPR 3
6162  025156  000277                          SCC                             ;SET ALL CC BITS
6163  025160  071302                          DIV     R2,R3                   ; TEST INSTRUCTION
6164  025162  022737  000002  177776          CMP     #2,@#177776             ;IS PS CORRECT
6165  025170  001401                          BEQ     3$                      ;YES GO ON
6166  025172  104001                          ERROR   ·1                      ;CPU ERROR
6167                                                                          ;NO GO TO ERROR
6168  025174  022702  000006          3$:     CMP     #6,R2                   ;IS R2 CORRECT
6169  025200  001401                          BEQ     4$                      ;YES GO ON
6170  025202  104001                          ERROR   ·1                      ;CPU ERROR
6171                                                                          ;NO GO TO ERROR
6172  025204  022703  000047          4$:     CMP     #47,R3                  ;IS R3 CORRECT
6173  025210  001401                          BEQ     5$                      ;YES GO ON
6174  025212  104001                          ERROR   ·1                      ;CPU ERROR
6175                                                                          ;NO GO TO ERROR
```

```
6176  025214  005004                  5$:     CLR     R4                      ;INIT R4
6177  025216  012705  000004                  MOV     #4,R5                   ;INIT R5
6178  025222  000277                          SCC                             ;SET ALL CC BITS
6179  025224  071427  000000                  DIV     #0,R4                   ; TEST INSTRUCTION
6180  025230  022737  000007  177776          CMP     #7,@#177776             ;IS PS CORRECT
6181  025236  001401                          BEQ     6$                      ;YES GO ON
6182  025240  104001                          ERROR   .+1             ;CPU ERROR
6183                                                                  ;NO GO TO ERROR
6184  025242  022704  000000          6$:     CMP     #0,R4                   ;IS R4 CORRECT
6185  025246  001401                          BEQ     7$                      ;YES GO ON
6186  025250  104001                          ERROR   .+1             ;CPU ERROR
6187                                                                  ;NO GO TO ERROR
6188  025252  022705  000004          7$:     CMP     #4,R5                   ;IS R5 CORRECT
6189  025256  001401                          BEQ     8$                      ;YES GO ON
6190  025260  104001                          ERROR   .+1             ;CPU ERROR
6191                                                                  ;NO GO TO ERROR
6192  025262  012700  000004          8$:     MOV     #4,R0                   ;INIT R0
6193  025266  012705  000010                  MOV     #10,R5                  ;INIT R5
6194  025272  005004                          CLR     R4                      ;INIT R4
6195  025274  000277                          SCC                             ;SET ALL CC BITS
6196  025276  071400                          DIV     R0,R4                   ; TEST INSTRUCTION
6197  025300  022737  000000  177776          CMP     #0,@#177776             ;IS PS CORRECT
6198  025306  001405                          BEQ     9$                      ;YES GO ON
6199  025310  010067  153064                  MOV     R0,400                  ;SAVE R0
6200  025314  104001                          ERROR   .+1             ;CPU ERROR
6201                                                                  ;NO GO TO ERROR
6202  025316  016700  153056                  MOV     400,R0                  ;RESTORE R0
6203  025322  022700  000004          9$:     CMP     #4,R0                   ;IS R0 CORRECT
6204  025326  001401                          BEQ     10$                     ;YES GO ON
6205  025330  104001                          ERROR   .+1             ;CPU ERROR
6206                                                                  ;NO GO TO ERROR
6207  025332  022704  000002          10$:    CMP     #2,R4                   ;IS R4 CORRECT
6208  025336  001401                          BEQ     11$                     ;YES GO ON
6209  025340  104001                          ERROR   .+1             ;CPU ERROR
6210                                                                  ;NO GO TO ERROR
6211  025342  022705  000000          11$:    CMP     #0,R5                   ;IS R5 CORRECT
6212  025346  001401                          BEQ     12$                     ;YES GO ON
6213  025350  104001                          ERROR   .+1             ;CPU ERROR
6214                                                                  ;NO GO TO ERROR
6215  025352  012705  000010          12$:    MOV     #10,R5                  ;INIT R5
6216  025356  005004                          CLR     R4                      ;INIT R4
6217  025360  000277                          SCC                             ;SET ALL CC BITS
6218  025362  071427  000003                  DIV     #3,R4                   ; TEST INSTRUCTION
6219  025366  022737  000000  177776          CMP     #0,@#177776             ;IS PS CORRECT
6220  025374  001401                          BEQ     13$                     ;YES GO ON
6221  025376  104001                          ERROR   .+1             ;CPU ERROR
6222                                                                  ;NO GO TO ERROR
6223  025400  022704  000002          13$:    CMP     #2,R4                   ;IS R4 CORRECT
6224  025404  001401                          BEQ     14$                     ;YES GO ON
6225  025406  104001                          ERROR   .+1             ;CPU ERROR
6226                                                                  ;NO GO TO ERROR
6227  025410  022705  000002          14$:    CMP     #2,R5                   ;IS R5 CORRECT
6228  025414  001401                          BEQ     15$                     ;YES GO ON
6229  025416  104001                          ERROR   .+1             ;CPU ERROR
6230                                                                  ;NO GO TO ERROR
6231                                  ;
```

```
6232                                    ;
6233                                    ;
6234  025420  012701  025550            15$:    MOV     #TE126B,R1              ;SETUP POINTERS TO TABLES
6235
6236  025424  010137  110502            16$:    MOV     R1,@#EXPDAT             ;SAVE A COPY OF R1
6237  025430  011104                            MOV     (R1),R4         ;INIT R4
6238  025432  016103  000004                    MOV     4(R1),R3                ;SAVE SOURCE
6239                                    ;
6240  025436  016105  000002                    MOV     2(R1),R5                ;INIT R5
6241  025442  000277                            SCC                             ;SET ALL CC BITS
6242  025444  071461  000004                    DIV     4(R1),R4                ; TEST INSTRUCTION
6243  025450  026137  000006  177776            CMP     6(R1),@#177776          ;IS PS CORRECT
6244  025456  001401                            BEQ     17$                     ;YES GO ON
6245  025460  104001                            ERROR   +1              ;CPU ERROR
6246                                                                    ;NO GO TO ERROR
6247  025462  026105  000010            17$:    CMP     10(R1),R5               ;IS R5 CORRECT
6248  025466  001401                            BEQ     18$                     ;YES GO ON
6249  025470  104001                            ERROR   +1              ;CPU ERROR
6250                                                                    ;NO GO TO ERROR
6251  025472  026104  000012            18$:    CMP     12(R1),R4               ;IS R4 CORRECT
6252  025476  001401                            BEQ     19$                     ;YES GO ON
6253  025500  104001                            ERROR   +1              ;CPU ERROR
6254                                                                    ;NO GO TO ERROR
6255  025502  023701  110502            19$:    CMP     @#EXPDAT,R1             ;IS R1 CORRECT
6256  025506  001403                            BEQ     20$                     ;YES GO ON
6257  025510  104001                            ERROR   +1              ;CPU ERROR
6258                                                                    ;NO GO TO ERROR
6259  025512  013701  110502                    MOV     @#EXPDAT,R1             ;RESTORE CORRECT VALUE
6260  025516  026103  000004            20$:    CMP     4(R1),R3                ;IS SOURCE CORRECT
6261  025522  001403                            BEQ     21$                     ;YES GO ON
6262  025524  104001                            ERROR   +1              ;CPU ERROR
6263                                                                    ;NO GO TO ERROR
6264  025526  010361  000004                    MOV     R3,4(R1)                ;TRY TO RESTORE CODE
6265  025532  062701  000014            21$:    ADD     #14,R1          ;POINT TO NEXT LOCATION
6266  025536  021127  000333                    CMP     (R1),#333               ;ARE WE DONE
6267  025542  001330                            BNE     16$                     ;NO GO TO 16$
6268
6269
6270  025544  000167  000316                    JMP     FIN126
6271                                    ;
6272                                    ;
6273  025550  177777            TE126B: .WORD   177777  ;DIVIDEND
6274  025552  177777                    .WORD   177777  ;INIT R5
6275  025554  177777                    .WORD   177777  ;DIVISOR
6276  025556  000000                    .WORD   0       ;PSW
6277  025560  000000                    .WORD   0       ;R5 RESULT
6278  025562  000001                    .WORD   1       ;R4 RESULT
6279
6280  025564  000000                    .WORD   0       ;DIVIDEND
6281  025566  177777                    .WORD   177777  ;INIT R5
6282  025570  177777                    .WORD   177777  ;DIVISOR
6283  025572  000012                    .WORD   12      ;PSW
6284  025574  177777                    .WORD   177777  ;R5 RESULT
6285  025576  000000                    .WORD   0       ;R4 RESULT
6286
6287  025600  177777                    .WORD   177777  ;DIVIDEND
```

```
6288  025602  000000                  .WORD   0       ;INIT R5
6289  025604  177777                  .WORD   177777  ;DIVISOR
6290  025606  000002                  .WORD   2       ;PSW
6291  025610  000000                  .WORD   0       ;R5 RESULT
6292  025612  177777                  .WORD   177777  ;R4 RESULT
6293
6294  025614  000000                  .WORD   0       ;DIVIDEND
6295  025616  007642                  .WORD   7642    ;INIT R5
6296  025620  007643                  .WORD   7643    ;DIVISOR
6297  025622  000004                  .WORD   4       ;PSW
6298  025624  007642                  .WORD   7642    ;R5 RESULT
6299  025626  000000                  .WORD   0       ;R4 RESULT
6300
6301  025630  000000                  .WORD   0       ;DIVIDEND
6302  025632  000137                  .WORD   137     ;INIT R5
6303  025634  177543                  .WORD   177543  ;DIVISOR
6304  025636  000004                  .WORD   4       ;PSW
6305  025640  000137                  .WORD   137     ;R5 RESULT
6306  025642  000000                  .WORD   0       ;R4 RESULT
6307
6308  025644  000000                  .WORD   0       ;DIVIDEND
6309  025646  007643                  .WORD   7643    ;INIT R5
6310  025650  007643                  .WORD   7643    ;DIVISOR
6311  025652  000000                  .WORD   0       ;PSW
6312  025654  000000                  .WORD   0       ;R5 RESULT
6313  025656  000001                  .WORD   1       ;R4 RESULT
6314
6315  025660  100000                  .WORD   100000  ;DIVIDEND
6316  025662  004376                  .WORD   4376    ;INIT R5
6317  025664  010021                  .WORD   10021   ;DIVISOR
6318  025666  000012                  .WORD   12      ;PSW
6319  025670  004376                  .WORD   4376    ;R5 RESULT
6320  025672  100000                  .WORD   100000  ;R4 RESULT
6321
6322  025674  177700                  .WORD   177700  ;DIVIDEND
6323  025676  170033                  .WORD   170033  ;INIT R5
6324  025700  010021                  .WORD   10021   ;DIVISOR
6325  025702  000010                  .WORD   10      ;PSW
6326  025704  171307                  .WORD   171307  ;R5 RESULT
6327  025706  176024                  .WORD   176024  ;R4 RESULT
6328
6329  025710  177700                  .WORD   177700  ;DIVIDEND
6330  025712  170033                  .WORD   170033  ;INIT R5
6331  025714  167757                  .WORD   167757  ;DIVISOR
6332  025716  000000                  .WORD   0       ;PSW
6333  025720  171307                  .WORD   171307  ;R5 RESULT
6334  025722  001754                  .WORD   1754    ;R4 RESULT
6335
6336  025724  000000                  .WORD   0       ;DIVIDEND
6337  025726  177777                  .WORD   177777  ;INIT R5
6338  025730  000001                  .WORD   1       ;DIVISOR
6339  025732  000002                  .WORD   2       ;PSW
6340  025734  177777                  .WORD   177777  ;R5 RESULT
6341  025736  000000                  .WORD   0       ;R4 RESULT
6342
6343  025740  177777                  .WORD   177777  ;DIVIDEND
```

```
6344  025742  045716                          .WORD   45716   ;INIT R5
6345  025744  000001                          .WORD   1       ;DIVISOR
6346  025746  000012                          .WORD   12      ;PSW
6347  025750  045716                          .WORD   45716   ;R5 RESULT
6348  025752  177777                          .WORD   177777  ;R4 RESULT
6349
6350  025754  000000                          .WORD   0       ;DIVIDEND
6351  025756  000002                          .WORD   2       ;INIT R5
6352  025760  177770                          .WORD   177770  ;DIVISOR
6353  025762  000004                          .WORD   4       ;PSW
6354  025764  000002                          .WORD   2       ;R5 RESULT
6355  025766  000000                          .WORD   0       ;R4 RESULT
6356
6357  025770  177777                          .WORD   177777  ;DIVIDEND
6358  025772  177776                          .WORD   177776  ;INIT R5
6359  025774  000010                          .WORD   10      ;DIVISOR
6360  025776  000004                          .WORD   4       ;PSW
6361  026000  177776                          .WORD   177776  ;R5 RESULT
6362  026002  000000                          .WORD   0       ;R4 RESULT
6363
6364  026004  000001                          .WORD   1       ;DIVIDEND
6365  026006  177777                          .WORD   177777  ;INIT R5
6366  026010  000001                          .WORD   1       ;DIVISOR
6367  026012  000002                          .WORD   2       ;PSW
6368  026014  177777                          .WORD   177777  ;R5 RESULT
6369  026016  000001                          .WORD   1       ;R4 RESULT
6370
6371  026020  000001                          .WORD   1       ;DIVIDEND
6372  026022  000000                          .WORD   0       ;INIT R5
6373  026024  000002                          .WORD   2       ;DIVISOR
6374  026026  000002                          .WORD   2       ;PSW
6375  026030  000000                          .WORD   0       ;R5 RESULT
6376  026032  000001                          .WORD   1       ;R4 RESULT
6377
6378  026034  000001                          .WORD   1       ;DIVIDEND
6379  026036  000000                          .WORD   0       ;INIT R5
6380  026040  000003                          .WORD   3       ;DIVISOR
6381  026042  000000                          .WORD   0       ;PSW
6382  026044  000001                          .WORD   1       ;R5 RESULT
6383  026046  052525                          .WORD   52525   ;R4 RESULT
6384
6385  026050  000023                          .WORD   23      ;DIVIDEND
6386  026052  016054                          .WORD   16054   ;INIT R5
6387  026054  016537                          .WORD   16537   ;DIVISOR
6388  026056  000000                          .WORD   0       ;PSW
6389  026060  010222                          .WORD   10222   ;R5 RESULT
6390  026062  000246                          .WORD   246     ;R4 RESULT
6391
6392  026064  000333                          .WORD   333
6393  026066                          FIN126:
6394                                  ;
6395  026066                          TE127:
6396                                  ;       TEST ASH (ARITHMETIC SHIFT)
6397  026066  005037  177776                  CLR     @#177776                ;INIT PSW
6398  026072  012702  000001                  MOV     #1,R2                   ;SETUP OPERAND
6399  026076  000277                          SCC                             ;SET ALL CC BITS
```

```
6400  026100  072202                        ASH     R2,R2           ; TEST INSTRUCTION
6401  026102  022737  000000  177776        CMP     #0,@#177776     ;IS PS CORRECT
6402  026110  001401                        BEQ     1$              ;YES GO ON
6403  026112  104001                        ERROR   .1              ;CPU ERROR
6404                                                                ;NO GO TO ERROR
6405  026114  020227  000002          1$:   CMP     R2,#2           ;IS R2 CORRECT
6406  026120  001401                        BEQ     2$              ;YES GO ON
6407  026122  104001                        ERROR   .1              ;CPU ERROR
6408                                                                ;NO GO TO ERROR
6409  026124  012702  100000          2$:   MOV     #100000,R2      ;SETUP R2
6410  026130  012703  000001                MOV     #1,R3           ;SETUP R3
6411  026134  000257                        CCC                     ;CLEAR ALL CC BITS
6412  026136  072203                        ASH     R3,R2           ; TEST INSTRUCTION
6413  026140  022737  000007  177776        CMP     #7,@#177776     ;IS PS CORRECT
6414  026146  001401                        BEQ     3$              ;YES GO ON
6415  026150  104001                        ERROR   .1              ;CPU ERROR
6416                                                                ;NO GO TO ERROR
6417  026152  020327  000001          3$:   CMP     R3,#1           ;IS R3 CORRECT
6418  026156  001401                        BEQ     4$              ;YES GO ON
6419  026160  104001                        ERROR   .1              ;CPU ERROR
6420                                                                ;NO GO TO ERROR
6421  026162  020227  000000          4$:   CMP     R2,#0           ;IS R2 CORRECT
6422  026166  001401                        BEQ     5$              ;YES GO ON
6423  026170  104001                        ERROR   .1              ;CPU ERROR
6424                                                                ;NO GO TO ERROR
6425  026172  012701  026266          5$:   MOV     #TE127A,R1      ;SETUP POINTERS TO TABLES
6426
6427  026176  010103                  6$:   MOV     R1,R3           ;
6428  026200  016102  000002                MOV     2(R1),R2        ;SETUP R2
6429  026204  000277                        SCC                     ;SET ALL CC BITS
6430  026206  072211                        ASH     (R1),R2         ; TEST INSTRUCTION
6431  026210  026137  000004  177776        CMP     4(R1),@#177776  ;IS PS CORRECT
6432  026216  001401                        BEQ     7$              ;YES GO ON
6433  026220  104001                        ERROR   .1              ;CPU ERROR
6434                                                                ;NO GO TO ERROR
6435  026222  026102  000006          7$:   CMP     6(R1),R2        ;IS R2 CORRECT
6436  026226  001401                        BEQ     8$              ;YES GO ON
6437  026230  104001                        ERROR   .1              ;CPU ERROR
6438                                                                ;NO GO TO ERROR
6439  026232  020301                  8$:   CMP     R3,R1           ;IS R1 CORRECT
6440  026234  001402                        BEQ     9$              ;YES GO ON
6441  026236  104001                        ERROR   .1              ;CPU ERROR
6442                                                                ;NO GO TO ERROR
6443  026240  010301                        MOV     R3,R1           ;RESTORE R1
6444  026242  021311                  9$:   CMP     (R3),(R1)       ;IS SOURCE CORRECT
6445  026244  001401                        BEQ     10$             ;YES GO ON
6446  026246  104001                        ERROR   .1              ;CPU ERROR
6447                                                                ;NO GO TO ERROR
6448                                                                ;SOURCE LOOKS INCORRECT
6449  026250  062701  000010          10$:  ADD     #10,R1          ;INCREMENT POINTER
6450  026254  020127  026526                CMP     R1,#FIN127      ;ARE WE DONE
6451  026260  001346                        BNE     6$              ;NO GO TO 6$
6452
6453
6454  026262  000167  000240                JMP     FIN127
6455                                  ;
```

```
6456
6457  026266  177761          ;TE127A: .WORD   177761          ;SOURCE
6458  026270  077777                   .WORD   77777           ;DEST
6459  026272  000005                   .WORD   5
6460  026274  000000                   .WORD   0
6461
6462  026276  177700                   .WORD   177700          ;SOURCE
6463  026300  017777                   .WORD   17777           ;DEST
6464  026302  000000                   .WORD   0
6465  026304  017777                   .WORD   17777
6466
6467  026306  177700                   .WORD   177700          ;SOURCE
6468  026310  100000                   .WORD   100000          ;DEST
6469  026312  000010                   .WORD   10
6470  026314  100000                   .WORD   100000
6471
6472  026316  177777                   .WORD   177777          ;SOURCE
6473  026320  100000                   .WORD   100000          ;DEST
6474  026322  000010                   .WORD   10
6475  026324  140000                   .WORD   140000
6476
6477  026326  177737                   .WORD   177737          ;SOURCE
6478  026330  177777                   .WORD   177777          ;DEST
6479  026332  000011                   .WORD   11
6480  026334  177777                   .WORD   177777
6481
6482  026336  177706                   .WORD   177706          ;SOURCE
6483  026340  102000                   .WORD   102000          ;DEST
6484  026342  000007                   .WORD   7
6485  026344  000000                   .WORD   0
6486
6487  026346  177710                   .WORD   177710          ;SOURCE
6488  026350  017777                   .WORD   17777           ;DEST
6489  026352  000013                   .WORD   13
6490  026354  177400                   .WORD   177400
6491
6492  026356  177713                   .WORD   177713          ;SOURCE
6493  026360  000012                   .WORD   12              ;DEST
6494  026362  000000                   .WORD   0
6495  026364  050000                   .WORD   50000
6496
6497  026366  177707                   .WORD   177707          ;SOURCE
6498  026370  170001                   .WORD   170001          ;DEST
6499  026372  000002                   .WORD   2
6500  026374  000200                   .WORD   200
6501
6502  026376  177717                   .WORD   177717          ;SOURCE
6503  026400  000001                   .WORD   1               ;DEST
6504  026402  000012                   .WORD   12
6505  026404  100000                   .WORD   100000
6506
6507  026406  177740                   .WORD   177740          ;SOURCE
6508  026410  017777                   .WORD   17777           ;DEST
6509  026412  000004                   .WORD   4
6510  026414  000000                   .WORD   0
6511
```

```
6512  026416  177771                          .WORD   177771          ;SOURCE
6513  026420  150000                          .WORD   150000          ;DEST
6514  026422  000010                          .WORD   10
6515  026424  177640                          .WORD   177640
6516
6517  026426  177742                          .WORD   177742          ;SOURCE
6518  026430  100000                          .WORD   100000          ;DEST
6519  026432  000011                          .WORD   11
6520  026434  177777                          .WORD   177777
6521
6522  026436  177764                          .WORD   177764          ;SOURCE
6523  026440  100000                          .WORD   100000          ;DEST
6524  026442  000010                          .WORD   10
6525  026444  177770                          .WORD   177770
6526
6527  026446  177750                          .WORD   177750          ;SOURCE
6528  026450  052525                          .WORD   52525           ;DEST
6529  026452  000004                          .WORD   4
6530  026454  000000                          .WORD   0
6531
6532  026456  177760                          .WORD   177760          ;SOURCE
6533  026460  100000                          .WORD   100000          ;DEST
6534  026462  000011                          .WORD   11
6535  026464  177777                          .WORD   177777
6536
6537  026466  177770                          .WORD   177770          ;SOURCE
6538  026470  100000                          .WORD   100000          ;DEST
6539  026472  000010                          .WORD   10
6540  026474  177600                          .WORD   177600
6541
6542  026476  177712                          .WORD   177712          ;SOURCE
6543  026500  004367                          .WORD   4367            ;DEST
6544  026502  000013                          .WORD   13
6545  026504  156000                          .WORD   156000
6546
6547  026506  177764                          .WORD   177764          ;SOURCE
6548  026510  017777                          .WORD   17777           ;DEST
6549  026512  000001                          .WORD   1
6550  026514  000001                          .WORD   1
6551
6552  026516  177701                          .WORD   177701          ;SOURCE
6553  026520  110000                          .WORD   110000          ;DEST
6554  026522  000003                          .WORD   3
6555  026524  020000                          .WORD   20000
6556
6557  026526  000240                  FIN127: NOP
6558                                  ;
6559  026530                          TE130:
6560                                  ;       TEST ASHC (ARITHMETIC SHIFT COMBINED)
6561  026530  005037  177776                  CLR     @#177776                ;INIT PSW
6562  026534  012701  000023                  MOV     #23,R1                  ;SETUP R1
6563  026540  012705  052525                  MOV     #052525,R5              ;SETUP R5
6564  026544  005004                          CLR     R4                      ;SETUP R4
6565  026546  000277                          SCC                             ;SET ALL CC BITS
6566  026550  073401                          ASHC    R1,R4                   ; TEST INSTRUCTION
6567  026552  023727  177776  000012          CMP     @#177776,#12            ;IS PS CORRECT
```

```
6568  026560  001401                    BEQ     1$                      ;YES GO ON
6569  026562  104001                    ERROR   .1              ;CPU ERROR
6570                                                            ;NO GO TO ERROR
6571  026564  020127  000023    1$:     CMP     R1,#23                  ;IS R1 CORRECT
6572  026570  001401                    BEQ     2$                      ;YES GO ON
6573  026572  104001                    ERROR   .1              ;CPU ERROR
6574                                                            ;NO GO TO ERROR
6575  026574  020427  125250    2$:     CMP     R4,#125250              ;IS R4 CORRECT
6576  026600  001401                    BEQ     3$                      ;YES GO ON
6577  026602  104001                    ERROR   .1              ;CPU ERROR
6578                                                            ;NO GO TO ERROR
6579  026604  020527  000000    3$:     CMP     R5,#0                   ;IS R5 CORRECT
6580  026610  001401                    BEQ     4$                      ;YES GO ON
6581  026612  104001                    ERROR   .1              ;CPU ERROR
6582                                                            ;NO GO TO ERROR
6583  026614  012703  052525    4$:     MOV     #52525,R3               ;SETUP R3
6584  026620  005002                    CLR     R2                      ;SETUP R2
6585  026622  012704  164731            MOV     #164731,R4              ;SETUP R4
6586  026626  000277                    SCC                             ;SET ALL CC BITS
6587  026630  073327  000027            ASHC    #23,R3                  ; TEST INSTRUCTION
6588  026634  023727  177776  000012    CMP     @#177776,#12            ;IS PS CORRECT
6589  026642  001401                    BEQ     5$                      ;YES GO ON
6590  026644  104001                    ERROR   .1              ;CPU ERROR
6591                                                            ;NO GO TO ERROR
6592  026646  020227  000000    5$:     CMP     R2,#0                   ;IS R2 CORRECT
6593  026652  001401                    BEQ     6$                      ;YES GO ON
6594  026654  104001                    ERROR   .1              ;CPU ERROR
6595                                                            ;NO GO TO ERROR
6596  026656  020327  000000    6$:     CMP     R3,#0                   ;IS R3 CORRECT
6597  026662  001401                    BEQ     7$                      ;YES GO ON
6598  026664  104001                    ERROR   .1              ;CPU ERROR
6599                                                            ;NO GO TO ERROR
6600  026666  020427  164731    7$:     CMP     R4,#164731              ;IS R4 CORRECT
6601  026672  001401                    BEQ     8$                      ;YES GO ON
6602  026674  104001                    ERROR   .1              ;CPU ERROR
6603                                                            ;NO GO TO ERROR
6604                                                                    ;
6605                                                                    ;
6606  026676  012701  027006    8$:     MOV     #TE130A,R1              ;SETUP POINTERS TO TABLES
6607
6608  026702  010104                9$:     MOV     R1,R4                   ;S   E A COPY OF R1
6609  026704  016102  000002                MOV     2(R1),R2                ;P R2
6610  026710  016103  000004                MOV     4(R1),R3                ;  R3
6611  026714  000277                        SCC                             ;S   ALL CC BITS
6612  026716  073211                        ASHC    (R1),R2         ; TEST INSTRUCTION
6613  026720  023761  177776  000006        CMP     @#177776,6(R1)          ;IS PS CORRECT
6614  026726  001401                        BEQ     10$                     ;YES GO ON
6615  026730  104001                        ERROR   .1              ;CPU ERROR
6616                                                            ;NO GO TO ERROR
6617  026732  026102  000010    10$:    CMP     10(R1),R2               ;IS R2 CORRECT
6618  026736  001401                    BEQ     11$                     ;YES GO ON
6619  026740  104001                    ERROR   .1              ;CPU ERROR
6620                                                            ;NO GO TO ERROR
6621  026742  026103  000012    11$:    CMP     12(R1),R3               ;IS R3 CORRECT
6622  026746  001401                    BEQ     12$                     ;YES GO ON
6623  026750  104001                    ERROR   .1              ;CPU ERROR
```

```
6624
6625  026752  020401                  12$:    CMP     R4,R1           ;NO GO TO ERROR
6626  026754  001402                          BEQ     13$             ;IS R1 CORRECT
6627  026756  104001                          ERROR   +1              ;YES GO ON
6628                                                                  ;CPU ERROR
6629  026760  010401                          MOV     R4,R1           ;NO GO TO ERROR
6630  026762  021114                  13$:    CMP     (R1),(R4)               ;IS SOURCE CORRECT
6631  026764  001401                          BEQ     14$             ;YES GO ON
6632  026766  104001                          ERROR   +1              ;CPU ERROR
6633                                                                  ;NO GO TO ERROR
6634                                                                  ;POSSIBLE SOURCE CODE CORRUPTION
6635  026770  062701  000014          14$:    ADD     #14,R1          ;GO TO NEXT TEST
6636  026774  020127  027416                  CMP     R1,#FIN130      ;ARE WE DONE
6637  027000  001340                          BNE     9$              ;NO GO TO 9$
6638
6639
6640  027002  000167  000410                  JMP     FIN130
6641                                  ;
6642                                  ;
6643  027006  177700                  TE130A: .WORD   177700          ;SOURCE
6644  027010  100125                          .WORD   100125          ;DESTINATION WORD 1
6645  027012  177777                          .WORD   177777          ;DESTINATION WORD 2
6646  027014  000010                          .WORD   10              ;TEST PSW
6647  027016  100125                          .WORD   100125          ;RESULT WORD 1
6648  027020  177777                          .WORD   177777          ;RESULT WORD 2
6649
6650  027022  177777                          .WORD   177777          ;SOURCE
6651  027024  000001                          .WORD   1               ;DESTINATION WORD 1
6652  027026  000000                          .WORD   0               ;DESTINATION WORD 2
6653  027030  000000                          .WORD   0               ;TEST PSW
6654  027032  000000                          .WORD   0               ;RESULT WORD 1
6655  027034  100000                          .WORD   100000          ;RESULT WORD 2
6656
6657  027036  177701                          .WORD   177701          ;SOURCE
6658  027040  047777                          .WORD   47777           ;DESTINATION WORD 1
6659  027042  100000                          .WORD   100000          ;DESTINATION WORD 2
6660  027044  000012                          .WORD   12              ;TEST PSW
6661  027046  117777                          .WORD   117777          ;RESULT WORD 1
6662  027050  000000                          .WORD   0               ;RESULT WORD 2
6663
6664  027052  177706                          .WORD   177706          ;SOURCE
6665  027054  004256                          .WORD   4256            ;DESTINATION WORD 1
6666  027056  177700                          .WORD   177700          ;DESTINATION WORD 2
6667  027060  000002                          .WORD   2               ;TEST PSW
6668  027062  025677                          .WORD   25677           ;RESULT WORD 1
6669  027064  170000                          .WORD   170000          ;RESULT WORD 2
6670
6671  027066  177711                          .WORD   177711          ;SOURCE
6672  027070  065700                          .WORD   65700           ;DESTINATION WORD 1
6673  027072  000012                          .WORD   12              ;DESTINATION WORD 2
6674  027074  000013                          .WORD   13              ;TEST PSW
6675  027076  100000                          .WORD   100000          ;RESULT WORD 1
6676  027100  012000                          .WORD   12000           ;RESULT WORD 2
6677
6678  027102  177737                          .WORD   177737          ;SOURCE
6679  027104  000000                          .WORD   0               ;DESTINATION WORD 1
```

```
6680  027106  000001                  .WORD   1               ;DESTINATION WORD 2
6681  027110  000004                  .WORD   4               ;TEST PSW
6682  027112  000000                  .WORD   0               ;RESULT WORD 1
6683  027114  000000                  .WORD   0               ;RESULT WORD 2
6684
6685  027116  177736                  .WORD   177736          ;SOURCE
6686  027120  000000                  .WORD   0               ;DESTINATION WORD 1
6687  027122  000001                  .WORD   1               ;DESTINATION WORD 2
6688  027124  000000                  .WORD   0               ;TEST PSW
6689  027126  040000                  .WORD   40000           ;RESULT WORD 1
6690  027130  000000                  .WORD   0               ;RESULT WORD 2
6691
6692  027132  177740                  .WORD   177740          ;SOURCE
6693  027134  100000                  .WORD   100000          ;DESTINATION WORD 1
6694  027136  000000                  .WORD   0               ;DESTINATION WORD 2
6695  027140  000011                  .WORD   11              ;TEST PSW
6696  027142  177777                  .WORD   177777          ;RESULT WORD 1
6697  027144  177777                  .WORD   177777          ;RESULT WORD 2
6698
6699  027146  177725                  .WORD   177725          ;SOURCE
6700  027150  177777                  .WORD   177777          ;DESTINATION WORD 1
6701  027152  174000                  .WORD   174000          ;DESTINATION WORD 2
6702  027154  000007                  .WORD   7               ;TEST PSW
6703  027156  000000                  .WORD   0               ;RESULT WORD 1
6704  027160  000000                  .WORD   0               ;RESULT WORD 2
6705
6706  027162  177724                  .WORD   177724          ;SOURCE
6707  027164  177777                  .WORD   177777          ;DESTINATION WORD 1
6708  027166  174000                  .WORD   174000          ;DESTINATION WORD 2
6709  027170  000011                  .WORD   11              ;TEST PSW
6710  027172  100000                  .WORD   100000          ;RESULT WORD 1
6711  027174  000000                  .WORD   0               ;RESULT WORD 2
6712
6713  027176  177733                  .WORD   177733          ;SOURCE
6714  027200  177777                  .WORD   177777          ;DESTINATION WORD 1
6715  027202  157023                  .WORD   157023          ;DESTINATION WORD 2
6716  027204  000012                  .WORD   12              ;TEST PSW
6717  027206  114000                  .WORD   114000          ;RESULT WORD 1
6718  027210  000000                  .WORD   0               ;RESULT WORD 2
6719
6720  027212  177727                  .WORD   177727          ;SOURCE
6721  027214  000000                  .WORD   0               ;DESTINATION WORD 1
6722  027216  177777                  .WORD   177777          ;DESTINATION WORD 2
6723  027220  000013                  .WORD   13              ;TEST PSW
6724  027222  177600                  .WORD   177600          ;RESULT WORD 1
6725  027224  000000                  .WORD   0               ;RESULT WORD 2
6726
6727  027226  177717                  .WORD   177717          ;SOURCE
6728  027230  177777                  .WORD   177777          ;DESTINATION WORD 1
6729  027232  000001                  .WORD   1               ;DESTINATION WORD 2
6730  027234  000011                  .WORD   11              ;TEST PSW
6731  027236  100000                  .WORD   100000          ;RESULT WORD 1
6732  027240  100000                  .WORD   100000          ;RESULT WORD 2
6733
6734  027242  177741                  .WORD   177741          ;SOURCE
6735  027244  100000                  .WORD   100000          ;DESTINATION WORD 1
```

```
6736  027246  000000                  .WORD   0               ;DESTINATION WORD 2
6737  027250  000010                  .WORD   10              ;TEST PSW
6738  027252  177777                  .WORD   177777          ;RESULT WORD 1
6739  027254  177777                  .WORD   177777          ;RESULT WORD 2
6740
6741  027256  177742                  .WORD   177742          ;SOURCE
6742  027260  037777                  .WORD   37777           ;DESTINATION WORD 1
6743  027262  177777                  .WORD   177777          ;DESTINATION WORD 2
6744  027264  000005                  .WORD   5               ;TEST PSW
6745  027266  000000                  .WORD   0               ;RESULT WORD 1
6746  027270  000000                  .WORD   0               ;RESULT WORD 2
6747
6748  027272  177742                  .WORD   177742          ;SOURCE
6749  027274  077777                  .WORD   77777           ;DESTINATION WORD 1
6750  027276  177777                  .WORD   177777          ;DESTINATION WORD 2
6751  027300  000001                  .WORD   1               ;TEST PSW
6752  027302  000000                  .WORD   0               ;RESULT WORD 1
6753  027304  000001                  .WORD   1               ;RESULT WORD 2
6754
6755  027306  177711                  .WORD   177711          ;SOURCE
6756  027310  065600                  .WORD   65600           ;DESTINATION WORD 1
6757  027312  000012                  .WORD   12              ;DESTINATION WORD 2
6758  027314  000003                  .WORD   3               ;TEST PSW
6759  027316  000000                  .WORD   0               ;RESULT WORD 1
6760  027320  012000                  .WORD   12000           ;RESULT WORD 2
6761
6762  027322  177740                  .WORD   177740          ;SOURCE
6763  027324  077777                  .WORD   77777           ;DESTINATION WORD 1
6764  027326  177777                  .WORD   177777          ;DESTINATION WORD 2
6765  027330  000004                  .WORD   4               ;TEST PSW
6766  027332  000000                  .WORD   0               ;RESULT WORD 1
6767  027334  000000                  .WORD   0               ;RESULT WORD 2
6768
6769  027336  177737                  .WORD   177737          ;SOURCE
6770  027340  177777                  .WORD   177777          ;DESTINATION WORD 1
6771  027342  177774                  .WORD   177774          ;DESTINATION WORD 2
6772  027344  000011                  .WORD   11              ;TEST PSW
6773  027346  177777                  .WORD   177777          ;RESULT WORD 1
6774  027350  177777                  .WORD   177777          ;RESULT WORD 2
6775
6776  027352  177747                  .WORD   177747          ;SOURCE
6777  027354  100000                  .WORD   100000          ;DESTINATION WORD 1
6778  027356  174000                  .WORD   174000          ;DESTINATION WORD 2
6779  027360  000010                  .WORD   10              ;TEST PSW
6780  027362  177777                  .WORD   177777          ;RESULT WORD 1
6781  027364  177700                  .WORD   177700          ;RESULT WORD 2
6782
6783  027366  177753                  .WORD   177753          ;SOURCE
6784  027370  006324                  .WORD   6324            ;DESTINATION WORD 1
6785  027372  071002                  .WORD   71002           ;DESTINATION WORD 2
6786  027374  000001                  .WORD   1               ;TEST PSW
6787  027376  000000                  .WORD   0               ;RESULT WORD 1
6788  027400  000146                  .WORD   146             ;RESULT WORD 2
6789
6790  027402  177765                  .WORD   177765          ;SOURCE
6791  027404  102351                  .WORD   102351          ;DESTINATION WORD 1
```

```
                                                .WORD   177231          ;DESTINATION WORD 2
                                                .WORD   11              ;TEST PSW
                                                .WORD   177760          ;RESULT WORD 1
                                                .WORD   116477          ;RESULT WORD 2
                                        FIN130:
                                        MSPAU:
                                        ;       TEST THAT AUTO DEC/INC OPERATIONS USING SP ARE ON WORD BOUNDRYS
027416  005006                                  CLR     R6                      ;CLEAR SP
027420  112667  153434                        MOVB    (R6)+,COUNT             ;TRY AUTOINC ON R6
027424  022706  000002                        CMP     #2,R6                   ;VERIFY AUTO INC BY 2
027430  001401                                BEQ     SPAU1                   ;BRANCH IF GOOD
                                                                                ;BAD AUTO-INC
027432  104001                                ERROR   +1              ;CPU ERROR
027434  005006                        SPAU1:  CLR     R6                      ;CLEAR R6
027436  112667  153416                        MOVB    (R6)+,COUNT
027442  112667  153412                        MOVB    (R6)+,COUNT             ;DOUBLE BYTE AUTO-INC
027446  022706  000004                        CMP     #4,R6           ;VERIFY RESULT
027452  001401                                BEQ     SPAU2           ;BRANCH IF GOOD
027454  104001                                ERROR   +1              ;CPU ERROR
                                                                ;BAD DOUBLE AUTO-INC
027456  012706  001000                SPAU2:  MOV     #STBOT,R6               ;LOAD R6
027462  114667  153372                        MOVB    -(R6),COUNT             ; TEST AUTO-DEC
027466  022706  000776                        CMP     #776,R6         ;VERIFY RESULT
027472  001401                                BEQ     SPAU3           ;BRANCH IF GOOD
027474  104001                                ERROR   +1              ;CPU ERROR

027476  012706  001000                SPAU3:  MOV     #STBOT,R6               ;LOAD R6
027502  114667  153352                        MOVB    -(R6),COUNT             ; TEST AUTO-DEC
027506  114667  153346                        MOVB    -(R6),COUNT             ; TEST AUTO-DEC
027512  022706  000774                        CMP     #774,R6         ;VERIFY RESULT
027516  001401                                BEQ     SPAU4           ;BRANCH IF GOOD
027520  104001                                ERROR   +1              ;CPU ERROR

027522  005006                        SPAU4:  CLR     R6              ; TEST AUTO-INC ON SOP
027524  105726                                TSTB    (R6)+           ; TEST AUTO-INC
027526  020627  000002                        CMP     R6,#2
027532  001401                                BEQ     SPAU5           ;BRANCH IF GOOD
027534  104001                                ERROR   +1              ;CPU ERROR

027536  012706  001000                SPAU5:  MOV     #STBOT,R6               ;LOAD R6
027542  105746                                TSTB    -(R6)           ; TEST AUTO-DEC
027544  022706  000776                        CMP     #776,R6         ;VERIFY RESULT
027550  001401                                BEQ     SPAU6           ;BRANCH IF GOOD
027552  104001                                ERROR   +1              ;CPU ERROR

027554  012706  001000                SPAU6:  MOV     #STBOT,R6

                                        ;
027560                                  MTRY:

                                        ;       VERIFY YELLOW ZONE TRAP ON AUTO DEC OF R6
027560  005067  150202                        CLR     CPEREG                  ;INIT CPU ERROR REGISTER
027564  012706  000150                        MOV     #150,R6                 ;LOAD R6 WITH A VALUE THAT WILL
                                                                                ;CAUSE A YELLOW STACK TRAP(IE. <400)
027570  016767  150210  153164                MOV     4,SLOC00                ;SAVE VECTOR
```

```
6848  027576  012767  027634  150200          MOV     #MTRYA,4        ;SETUP THE STACK OVERFLOW TRAP POINTER
6849  027604  016701  150336                  MOV     146,R1          ;SAVE VECTOR
6850  027610  016702  150330                  MOV     144,R2          ;SAVE VECTOR
6851  027614  016703  150322                  MOV     142,R3          ;SAVE VECTOR
6852  027620  005067  150322                  CLR     146             ;JUST AS A PRECAUTION
6853  027624  005046                          CLR     -(R6)           ;CAUSE A STACK OVERFLOW TRAP
6854  027626  012706  001000                  MOV     #STBOT,R6       ;RESTORE R6 FOR ERROR CALL
6855  027632  104001                          ERROR   +1              ;CPU ERROR
6856                                                          ;OVERFLOW TRAP FAILED
6857  027634                          MTRYA:
6858  027634  022767  000010  150124          CMP     #BIT03,CPEREG   ;WAS CPU ERROR REG SET PROPERLY?
6859  027642  001003                          BNE     1$              ;GO TO ERROR IF NOT
6860  027644  020627  000142                  CMP     R6,#142         ;VERIFY CORRECT DECRIMENT OF R6
6861  027650  001401                          BEQ     MTRYB           ;BRANCH IF GOOD
6862  027652  104001                  1$:     ERROR   +1              ;CPU ERROR
6863                                                          ;R6 IMPROPERLY DECRIMENTED
6864                                                          ;OR CPU ERROR REGISTER NOT CORRECT
6865  027654                          MTRYB:
6866  027654  005067  150106                  CLR     CPEREG          ;CLEAR THE CPU ERROR REGISTER
6867  027660  016767  153076  150116          MOV     SLOC00,4        ;RESTORE VECTOR
6868  027666  010167  150254                  MOV     R1,146          ;RESTORE VECTORS
6869  027672  010267  150246                  MOV     R2,144          ;       "
6870  027676  010367  150240                  MOV     R3,142          ;       "
6871  027702  012706  001000                  MOV     #STBOT,R6
6872
6873
6874                                  ;
6875  027706                          MTRYM:
6876
6877                                  ;       TEST STACK OVERFLOW TRAPS IN VARIOUS MODES
6878  027706  005067  150054                  CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
6879  027712  012706  000400                  MOV     #400,R6                 ;SETUP OVERFLOW R6 DATA
6880  027716  016767  150062  153036          MOV     4,SLOC00                ;SAVE VECTOR
6881  027724  012767  027746  150052          MOV     #TRYMA,4
6882  027732  005067  150440                  CLR     376                     ;JUST AS A PRECAUTION
6883  027736  005046                          CLR     -(R6)                   ;CAUSE OVERFLOW TRAP
6884  027740  012706  001000                  MOV     #STBOT,R6               ;RESTORE R6 FOR ERROR CALL
6885  027744  104001                          ERROR   +1              ;CPU ERROR
6886                                                                  ;NO OVERFLOW TRAP
6887  027746                          TRYMA:
6888  027746  005067  150014                  CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
6889  027752  012705  001000                  MOV     #1000,R5                ;SETUP R5 DATA
6890  027756  012706  000400                  MOV     #400,R6                 ;SETUP OVERFLOW R6 DATA
6891  027762  012767  030000  150014          MOV     #TRYMB,4
6892  027770  064645                          ADD     -(R6),-(R5)             ;CAUSE OVERFLOW TRAP
6893  027772  012706  001000                  MOV     #STBOT,R6               ;RESTORE R6 FOR ERROR CALL
6894  027776  104001                          ERROR   +1              ;CPU ERROR
6895                                                                  ;NO OVERFLOW TRAP
6896  030000                          TRYMB:
6897  030000  005067  147762                  CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
6898  030004  012706  000150                  MOV     #150,R6                 ;SETUP OVERFLOW R6 DATA
6899  030010  012767  030026  147766          MOV     #TRYMC,4
6900  030016  044546                          BIC     -(R5),-(R6)             ;CAUSE OVERFLOW TRAP
6901  030020  012706  001000                  MOV     #STBOT,R6               ;RESTORE R6 FOR ERROR CALL
6902  030024  104001                          ERROR   +1              ;CPU ERROR
6903                                                                  ;NO OVERFLOW TRAP
```

```
 6904  030026  005067  147734          TRYMC:  CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
 6905  030032  016767  152724  147744          MOV     SLOC00,4                ;RESTORE VECTOR
 6906  030040  012706  001000                  MOV     #STBOT,R6
 6907
 6908
 6909                                  ;
 6910  030044                          MILLO:
 6911
 6912                                  ;       TEST STACK OVERFLOW ON ILLEGAL INST TRAP
 6913  030044  005067  147716                  CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
 6914  030050  012706  000400                  MOV     #400,R6                 ;SETUP FOR OVERFLOW TRAP
 6915  030054  016767  147730  152700          MOV     10,SLOC00               ;SAVE VECTOR
 6916  030062  012767  030110  147720          MOV     #MILLOA,10              ;SETUP ILLEGAL TRAP VECTOR
 6917  030070  016767  147710  152666          MOV     4,SLOC01                ;SAVE VECTOR
 6918  030076  012767  030116  147700          MOV     #MILLOB,4               ;SETUP OVERFLOW TRAP VECTOR
 6919  030104  000077                          77                              ;UNUSED INSTRUCTION TRAP
 6920  030106  000240                          NOP
 6921  030110  012706  001000          MILLOA: MOV     #STBOT,R6               ;RESTORE R6 FOR ERROR CALL
 6922  030114  104001                          ERROR   +1              ;CPU ERROR
 6923                                                                  ;UNUSED INSTRUCTION TRAP
 6924  030116                          MILLOB:
 6925  030116  016767  152642  147660          MOV     SLOC01,4                ;RESTORE VECTOR
 6926  030124  016767  152632  147656          MOV     SLOC00,10               ;RESTORE VECTOR
 6927  030132  005067  147630                  CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
 6928  030136  012706  001000                  MOV     #STBOT,R6               ;RESTORE R6
 6929
 6930
 6931
 6932                                  ;
 6933  030142                          MIOTO:
 6934
 6935                                  ;       TEST STACK OVERFLOW ON IOT TRAP
 6936  030142  005067  147620                  CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
 6937  030146  012706  000400                  MOV     #400,R6                 ;SETUP STACK FOR OVERFLOW
 6938  030152  016767  147642  152602          MOV     20,SLOC00               ;SAVE OLD IOT VECTOR
 6939  030160  012767  030206  147632          MOV     #IOTOA,20               ;SETUP ERROR ACTION ON IOT
 6940  030166  016767  147612  152570          MOV     4,SLOC01                ;SAVE VECTOR
 6941  030174  012767  030214  147602          MOV     #IOTOB,4                ;SETUP CORRECT TRAP VECTOR FOR
 6942                                                                          ;OVERFLOW
 6943  030202  000004                          IOT                             ; TEST INSTRUCTION
 6944  030204  000240                          NOP
 6945  030206  012706  001000          IOTOA:  MOV     #STBOT,R6               ;RESTORE R6 FOR ERROR CALL
 6946  030212  104001                          ERROR   +1              ;CPU ERROR
 6947                                                                  ;FAILURE OF STACK OVERFLOW
 6948  030214                          IOTOB:
 6949  030214  005067  147546                  CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
 6950  030220  012706  001000                  MOV     #STBOT,R6
 6951  030224  016767  152534  147552          MOV     SLOC01,4                ;RESTORE VECTOR
 6952  030232  016767  152524  147560          MOV     SLOC00,20               ;RESTORE TRAP VECTOR
 6953
 6954                                  ;
 6955                                  ;
 6956  030240                          MEMTO:
 6957
 6958                                  ;       TEST STACK OVERFLOW ON EMT TRAP
 6959  030240  005067  147522                  CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
```

```
6960  030244  012706  000400              MOV     #400,R6                         ;SETUP STACK FOR OVERFLOW
6961  030250  016767  147554  152504      MOV     30,SLOC00                       ;SAVE OLD EMT VECTOR
6962  030256  012767  030304  147544      MOV     #EMT0A,30                       ;SETUP ERROR ACTION ON EMT
6963  030264  016767  147514  152472      MOV     4,SLOC01                        ;SAVE VECTOR
6964  030272  012767  030312  147504      MOV     #EMT0B,4                        ;SETUP CORRECT TRAP VECTOR FOR
6965                                                                              ;OVERFLOW
6966  030300  104000                      EMT                                     ; TEST INSTRUCTION
6967  030302  000240                      NOP
6968  030304  012706  001000      EMT0A:  MOV     #STBOT,R6                       ;RESTORE R6 FOR ERROR CALL
6969  030310  104001                      ERROR   +1              ;CPU ERROR
6970                                                                  ;FAILURE OF STACK OVERFLOW
6971  030312                      EMT0B:
6972  030312  016767  152444  147510      MOV     SLOC00,30                       ;RESTORE TRAP VECTOR
6973  030320  016767  152440  147456      MOV     SLOC01,4                        ;RESTORE VECTOR
6974  030326  005067  147434              CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
6975  030332  012706  001000              MOV     #STBOT,R6
6976
6977  030336                      MTRP0:
6978
6979                              ;       TEST STACK OVERFLOW ON TRAP
6980  030336  005067  147424              CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
6981  030342  012706  000400              MOV     #400,R6                         ;SETUP STACK FOR OVERFLOW
6982  030346  016767  147462  152406      MOV     34,SLOC00                       ;SAVE OLD TRP VECTOR
6983  030354  012767  030402  147452      MOV     #TRP0A,34                       ;SETUP ERROR ACTION ON TRP
6984  030362  016767  147416  152374      MOV     4,SLOC01                        ;SAVE VECTOR
6985  030370  012767  030410  147406      MOV     #TRP0B,4                        ;SETUP CORRECT TRAP VECTOR FOR
6986                                                                              ;OVERFLOW
6987  030376  104400                      TRAP                                    ; TEST INSTRUCTION
6988  030400  000240                      NOP
6989  030402  012706  001000      TRP0A:  MOV     #STBOT,R6                       ;RESTORE R6 FOR ERROR CALL
6990  030406  104001                      ERROR   +1              ;CPU ERROR
6991                                                                  ;FAILURE OF STACK OVERFLOW
6992  030410                      TRP0B:
6993  030410  016767  152346  147416      MOV     SLOC00,34                       ;RESTORE TRAP VECTOR
6994  030416  016767  152342  147360      MOV     SLOC01,4                        ;RESTORE VECTOR
6995  030424  005067  147336              CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
6996  030430  012706  001000              MOV     #STBOT,R6
6997
6998                              ;
6999                              ;
7000  030434                      MBPT0:
7001
7002                              ;       TEST STACK OVERFLOW ON BPT
7003  030434  005067  147326              CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
7004  030440  012706  000400              MOV     #400,R6                         ;SETUP STACK FOR OVERFLOW
7005  030444  016767  147344  152310      MOV     14,SLOC00                       ;SAVE OLD BPT VECTOR
7006  030452  012767  030500  147334      MOV     #BPT0A,14                       ;SETUP ERROR ACTION ON BPT
7007  030460  016767  147320  152276      MOV     4,SLOC01                        ;SAVE VECTOR
7008  030466  012767  030506  147310      MOV     #BPT0B,4                        ;SETUP CORRECT TRAP VECTOR FOR
7009                                                                              ;OVERFLOW
7010  030474  000003                      BPT                                     ; TEST INSTRUCTION
7011  030476  000240                      NOP
7012  030500  012706  001000      BPT0A:  MOV     #STBOT,R6                       ;RESTORE R6 FOR ERROR CALL
7013  030504  104001                      ERROR   +1              ;CPU ERROR
7014                                                                  ;FAILURE OF STACK OVERFLOW
7015  030506                      BPT0B:
```

```
7016  030506  005067  147254                  CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
7017  030512  016767  152244  147274          MOV     SLOC00,14                       ;RESTORE TRAP VECTOR
7018  030520  016767  152240  147256          MOV     SLOC01,4                        ;RESTORE VECTOR
7019  030526  012706  001000                  MOV     #STBOT,R6
7020
7021                                  ;
7022                                  ;
7023  030532                          MILAO:
7024
7025                                  ;       TEST STACK OVERFLOW AND ILLEGAL JMP INSTRUCTION
7026  030532  005067  147230                  CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
7027  030536  012706  000400                  MOV     #400,R6                         ;SETUP STACK FOR OVERFLOW
7028  030542  016767  147242  152212          MOV     10,SLOC00                       ;SAVE OLD ILLEGAL INST. VECTOR
7029  030550  012767  030600  147232          MOV     #ILAOA,10                       ;SETUP ERROR ACTION ILLEGAL OPCODE
7030  030556  016767  147222  152200          MOV     4,SLOC01                        ;SAVE VECTOR
7031  030564  012767  030606  147212          MOV     #ILBOB,4                        ;SETUP CORRECT TRAP VECTOR FOR
7032                                                                                  ;OVERFLOW
7033  030572  005001                          CLR     R1
7034  030574  000101                          JMP     R1                              ; TEST INSTRUCTION
7035  030576  000240                          NOP
7036  030600  012706  001000          ILAOA:  MOV     #STBOT,R6                       ;RESTORE R6 FOR ERROR CALL
7037  030604  104001                          ERROR   +1                      ;CPU ERROR
7038                                                                          ;FAILURE OF STACK OVERFLOW
7039  030606                          ILBOB:
7040  030606  016767  152152  147170          MOV     SLOC01,4                        ;RESTORE VECTOR
7041  030614  016767  152142  147166          MOV     SLOC00,10                       ;RESTORE TRAP VECTOR
7042  030622  005067  147140                  CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
7043  030626  012706  001000                  MOV     #STBOT,R6
7044
7045                                  ;
7046                                  ;
7047  030632                          MILLBO:
7048
7049                                  ;       TEST STACK OVERFLOW ON ILLEGAL JSR INST
7050  030632  012706  000400                  MOV     #400,R6                         ;SETUP STACK FOR OVERFLOW
7051  030636  016767  147146  152116          MOV     10,SLOC00                       ;SAVE OLD VECTOR
7052  030644  012767  030674  147136          MOV     #ILLBOA,10                      ;SETUP ERROR ACTION ON ILL. OPCODE
7053  030652  016767  147126  152104          MOV     4,SLOC01                        ;SAVE VECTOR
7054  030660  012767  030702  147116          MOV     #ILLBOB,4                       ;SETUP CORRECT TRAP VECTOR FOR
7055                                                                                  ;OVERFLOW
7056  030666  005001                          CLR     R1
7057  030670  004501                          JSR     R5,R1                           ; TEST INSTRUCTION
7058  030672  000240                          NOP
7059  030674  012706  001000          ILLBOA: MOV     #STBOT,R6                       ;RESTORE R6 FOR ERROR CALL
7060  030700  104001                          ERROR   +1                      ;CPU ERROR
7061                                                                          ;FAILURE OF STACK OVERFLOW
7062  030702                          ILLBOB:
7063  030702  016767  152056  147074          MOV     SLOC01,4                        ;RESTORE VECTOR
7064  030710  016767  152046  147072          MOV     SLOC00,10                       ;RESTORE TRAP VECTOR
7065  030716  012706  001000                  MOV     #STBOT,R6
7066
7067                                  ;
7068                                  ;
7069  030722                          MSTO:
7070
7071                                  ;       TEST FOR FALSE STACK OVERFLOW
```

```
7072  030722  016767  147056  152032          MOV     4,SLOC00                ;SAVE VECTOR
7073  030730  012767  030776  147046          MOV     #MSTOE,4                ;ANTICIPATE OVERFLOW ERROR
7074  030736  012706  001002                  MOV     #1002,R6                ;SETUP LEGAL R6
7075  030742  005746                          TST     -(R6)                   ;TRY TO CAUSE STACK OVERFLOW
7076  030744  012706  002002                  MOV     #2002,R6                ;SETUP LEGAL R6
7077  030750  005746                          TST     -(R6)                   ;TRY TO CAUSE STACK OVERFLOW
7078  030752  012706  004002                  MOV     #4002,R6                ;SETUP LEGAL R6
7079  030756  005746                          TST     -(R6)                   ;TRY TO CAUSE STACK OVERFLOW
7080  030760  012706  010002                  MOV     #10002,R6               ;SETUP LEGAL R6
7081  030764  005746                          TST     -(R6)                   ;TRY TO CAUSE STACK OVERFLOW
7082  030766  012706  100402                  MOV     #100402,R6              ;SETUP  LEGAL R6
7083  030772  005746                          TST     -(R6)                   ;TRY TO CAUSE STACK OVERFLOW
7084  030774  000403                          BR      MSTOEE                  ;EXIT MODULE
7085  030776  012706  001000          MSTOE:  MOV     #STBOT,R6               ;RESTORE R6 FOR ERROR CALL
7086  031002  104001                          ERROR   +1              ;CPU ERROR
7087                                          ;STACK OVERFLOW ERROR
7088  031004  016767  151752  146772  MSTOEE: MOV     SLOC00,4                ;RESTORE VECTOR
7089  031012  012706  001000                  MOV     #STBOT,R6
7090
7091                                  ;
7092                                  ;
7093  031016                          MTT:
7094
7095                                  ;       TEST T-BIT TRAPS
7096  031016  012706  001000                  MOV     #STBOT,R6               ;SETUP STACK
7097  031022  016767  146766  151732          MOV     14,SLOC00               ;SAVE OLD T-BIT VECTOR
7098  031030  012746  000020                  MOV     #20,-(R6)               ;PUSH T-BIT
7099  031034  012746  031052                  MOV     #MTTA,-(R6)             ;SETUP ERROR TRAP VECTOR
7100  031040  012767  031054  146746          MOV     #MTTB,14                ;SETUP NEW T-BIT VECTOR
7101  031046  000002                          RTI                             ;CAUSE A T BIT SET IN PSW
7102  031050  104001                          ERROR   +1              ;CPU ERROR
7103                                          ;SHOULD NEVER BE EXECUTED
7104  031052  104001                  MTTA:   ERROR   +1              ;CPU ERROR
7105                                          ;DIDNT TAKE CORRECT TRAP
7106  031054  022706  000[illegible]  MTTB:   CMP     #STBOT-4,R6             ;VERIFY SP DECIRMENT
7107  031060  001401                          BEQ     MTTD                    ;BRANCH IF GOOD
7108  031062  104001                          ERROR   +1              ;CPU ERROR
7109                                          ;BAD SP
7110  031064  021627  031052          MTTD:   CMP     (R6),#MTTA              ;VERIFY PC SAVED ON STACK
7111  031070  001401                          BEQ     MTTE                    ;BRANCH IF GOOD
7112  031072  104001                          ERROR   +1              ;CPU ERROR
7113                                          ;INCORRECT PC ON STACK
7114  031074                          MTTE:
7115  031074  016767  151662  146712          MOV     SLOC00,14               ;RESTORE VECTOR 14
7116
7117  031102  012706  001000                  MOV     #STBOT,R6
7118
7119                                  ;
7120  031106                          MTTS:
7121
7122                                  ;       TEST T-BIT TRAPS WITH RTT
7123  031106  012706  001000                  MOV     #STBOT,R6               ;SETUP STACK
7124  031112  016767  146676  151642          MOV     14,SLOC00               ;SAVE OLD T-BIT VECTOR
7125  031120  012746  000020                  MOV     #20,-(R6)               ;PUSH T-BIT
7126  031124  012746  031142                  MOV     #MTTSA,-(R6)            ;SETUP ERROR TRAP VECTOR
7127  031130  012767  031146  146656          MOV     #MTTSB,14               ;SETUP NEW T-BIT VECTOR
```

```
 7128  031136  000006                          RTT                             ;CAUSE A T BIT SET IN PSW
 7129  031140  104001                          ERROR   .1                      ;CPU ERROR
 7130                                                                          ;SHOULD NEVER BE EXECUTED
 7131  031142  000240                  MTTSA:  NOP                             ;RTT WILL EXECUTE THIS INSTRUCTION
 7132                                                                          ;WITH A T-BIT TRAP
 7133  031144  104001                  MTTSQ:  ERROR   .1                      ;CPU ERROR
 7134                                                                          ;DIDNT TAKE CORRECT TRAP
 7135  031146  022706  000774          MTTSB:  CMP     @STBOT-4,R6             ;VERIFY SP DECIRMENT
 7136  031152  001401                          BEQ     MTTSD                   ;BRANCH IF GOOD
 7137  031154  104001                          ERROR   .1                      ;CPU ERROR
 7138                                                                          ;BAD SP
 7139  031156  021627  031144          MTTSD:  CMP     (R6),@MTTSQ             ;VERIFY PC SAVED ON STACK
 7140  031162  001401                          BEQ     MTTSE                   ;BRANCH IF GOOD
 7141  031164  104001                          ERROR   .1                      ;CPU ERROR
 7142                                                                          ;INCORRECT PC ON STACK
 7143  031166                          MTTSE:
 7144  031166  016767  151570  146620          MOV     SLOC00,14               ;RESTORE VECTOR 14
 7145  031174  012706  001000                  MOV     @STBOT,R6
 7146
 7147  031200                          MTTR:
 7148
 7149                                  ;       TEST OLD STATUS ON T-BIT TRAP
 7150  031200  012706  001000                  MOV     @STBOT,R6               ;SETUP STACK
 7151  031204  016767  146604  151550          MOV     14,SLOC00               ;SAVE OLD  VECTOR
 7152  031212  012746  000020                  MOV     @20,-(R6)               ;PUSH T-BIT
 7153  031216  012746  031244                  MOV     @MTTRA,-(R6)            ;SETUP ERROR TRAP VECTOR
 7154  031222  012767  031246  146564          MOV     @MTTRB,14               ;SETUP NEW T-BIT VECTOR
 7155  031230  012767  000357  146540          MOV     @357,PS                 ;SET PRIORITY AND COND C
 7156  031236  000277                          SCC
 7157  031240  000002                          RTI
 7158  031242  104001                          ERROR   .1                      ;CPU ERROR
 7159                                                                          ;SHOULD NEVER EXECUTE
 7160  031244  104001                  MTTRA:  ERROR   .1                      ;CPU ERROR
 7161                                                                          ;DIDNT TAKE CORRECT TRAP
 7162  031246  026727  147524  000020  MTTRB:  CMP     STBOT-2,@20             ;VERIFY PSW ON STACK
 7163  031254  001401                          BEQ     MTTRC                   ;BRANCH IF CORRECT STATUS
 7164  031256  104001                          ERROR   .1                      ;CPU ERROR
 7165                                                                          ;BAD STATUS ON STACK
 7166  031260  012706  001000          MTTRC:  MOV     @STBOT,R6               ;SETUP STACK
 7167  031264  012746  000377                  MOV     @377,-(R6)              ;PUSH T-BIT
 7168  031270  012746  031316                  MOV     @MTTRD,-(R6)            ;SETUP ERROR TRAP VECTOR
 7169  031274  012767  031320  146512          MOV     @MTTRE,14               ;SETUP NEW T-BIT VECTOR
 7170  031302  012767  000000  146466          MOV     @0,PS                   ;CLEAR PRIORITY
 7171  031310  000257                          CCC                             ;CLEAR CONDITION CODES
 7172  031312  000002                          RTI
 7173  031314  104001                          ERROR   .1                      ;CPU ERROR
 7174                                                                          ;SHOULD NEVER EXECUTE
 7175  031316  104001                  MTTRD:  ERROR   .1                      ;CPU ERROR
 7176                                                                          ;DIDNT TAKE CORRECT TRAP
 7177  031320  026727  147452  000377  MTTRE:  CMP     STBOT-2,@377            ;VERIFY OLD PSW ON STACK
 7178  031326  001401                          BEQ     MTTRF                   ;BRANCH IF GOOD
 7179  031330  104001                          ERROR   .1                      ;CPU ERROR
 7180                                                                          ;OLD PSW INCORRECT
 7181  031332                          MTTRF:
 7182  031332  016767  151424  146454          MOV     SLOC00,14               ;RESTORE VECTOR
 7183  031340  012706  001000                  MOV     @STBOT,R6
```

```
7184
7185                                    ;
7186  031344                            MRT:
7187
7188                                    ;       TEST RESERVED INST TRAP
7189  031344  012706  001000                      MOV     #STBOT,R6               ;SETUP STACK
7190  031350  016767  146434  151404            MOV     10,SLOC00               ;SAVE OLD  VECTOR
7191  031356  012767  031370  146424            MOV     #MRTB,10                ;SETUP NEW RESERVED VECTOR
7192  031364  000077                            77
7193  031366  104001                    MRTA:   ERROR   .1              ;CPU ERROR
7194                                            ;DIDNT TAKE CORRECT TRAP
7195  031370  022706  000774            MRTB:   CMP     #STBOT-4,R6             ;VERIFY SP DECRIMENT
7196  031374  001401                            BEQ     MRTE                    ;BRANCH IF GOOD
7197  031376  104001                            ERROR   .1              ;CPU ERROR
7198                                            ;BAD PC ON STACK
7199  031400  021627  031366            MRTE:   CMP     (R6),#MRTA              ;VERFY PROPER PC ON STACK
7200  031404  001401                            BEQ     MRTF                    ;BRANCH IF GOOD
7201  031406  104001                            ERROR   .1              ;CPU ERROR
7202                                            ;INCORRECT PC ON STACK
7203  031410                            MRTF:
7204  031410  016767  151346  146372            MOV     SLOC00,10               ;RESTORE TRAP VECTOR
7205  031416  012706  001000                    MOV     #STBOT,R6
7206
7207                                    ;
7208                                    ;
7209  031422                            MRTO:
7210
7211                                    ;       TEST OLD STATUS ON RESERVED INST TRAP
7212  031422  012706  001000                    MOV     #STBOT,R6               ;SETUP STACK
7213  031426  016767  146356  151326            MOV     10,SLOC00               ;SAVE OLD  VECTOR
7214  031434  012767  031454  146346            MOV     #MRTOB,10               ;SETUP NEW VECTOR
7215  031442  005067  146330                    CLR     PS                      ;CLEAR PRIORITY AND COND C
7216  031446  000257                            CCC
7217  031450  000077                            77
7218  031452  104001                    MRTOA:  ERROR   .1              ;CPU ERROR
7219                                            ;DIDNT TAKE CORRECT TRAP
7220  031454  026727  147316  000000    MRTOB:  CMP     STBOT-2,#0              ;VERIFY PSW ON STACK
7221  031462  001401                            BEQ     MRTOC                   ;BRANCH IF CORRECT STATUS
7222  031464  104001                            ERROR   .1              ;CPU ERROR
7223                                            ;BAD STATUS ON STACK
7224  031466  012706  001000            MRTOC:  MOV     #STBOT,R6               ;SETUP STACK
7225  031472  012767  031514  146310            MOV     #MRTOE,10               ;SET UP TRAP VECTOR
7226  031500  012767  000357  146270            MOV     #357,PS                 ;SET PRIORITY
7227  031506  000277                            SCC                             ;SET CONDITION CODES
7228  031510  000077                            77                              ;RESERVED INSTRUCTION
7229
7230  031512  104001                    MRTOD:  ERROR   .1              ;CPU ERROR
7231                                            ;DIDNT TAKE CORRECT TRAP
7232  031514  026727  147256  000357    MRTOE:  CMP     STBOT-2,#357            ;VERIFY OLD PSW ON STACK
7233  031522  001401                            BEQ     MRTOF                   ;BRANCH IF GOOD
7234  031524  104001                            ERROR   .1              ;CPU ERROR
7235                                            ;OLD PSW INCORRECT
7236  031526                            MRTOF:
7237  031526  016767  151230  146254            MOV     SLOC00,10               ;RESOTRE TRAP VECTOR
7238  031534  012706  001000                    MOV     #STBOT,R6
7239
```

```
7240  031540                          MTP:
7241
7242                                  ;       TEST TRAP INST
7243  031540  012706  001000                  MOV     #STBOT,R6               ;SETUP STACK
7244  031544  016767  146264  151210          MOV     34,SLOC00               ;SAVE OLD  VECTOR
7245  031552  012767  031572  146254          MOV     #MTPB,34                ;SETUP NEW TRAP VECTOR
7246  031560  005067  146212                  CLR     PS                      ;CLEAR PRIORITY ABND COND C
7247  031564  000257                          CCC
7248  031566  104400                          TRAP
7249  031570  104001                  MTPR:   ERROR   .1              ;CPU ERROR
7250                                                                  ;DIDNT TAKE CORRECT TRAP
7251  031572  022706  000774          MTPB:   CMP     #STBOT-4,R6             ;VERIFY SP DECRIMENT
7252  031576  001401                          BEQ     MTPQ                    ;BRANCH IF GOOD
7253  031600  104001                          ERROR   .1              ;CPU ERROR
7254                                                                  ;BAD PC ON STACK
7255  031602  021627  031570          MTPQ:   CMP     (R6),#MTPR              ;VERFY PROPER PC ON STACK
7256  031606  001401                          BEQ     MTPF                    ;BRANCH IF GOOD
7257  031610  104001                          ERROR   .1              ;CPU ERROR
7258                                                                  ;INCORRECT PC ON STACK
7259  031612                          MTPF:
7260  031612  016767  151144  146214          MOV     SLOC00,34               ;RESTORE VECTOR
7261  031620  012706  001000                  MOV     #STBOT,R6
7262
7263                                  ;
7264                                  ;
7265  031624                          MTPO:
7266
7267                                  ;       TEST OLD STATUS SAVED ON TRAP
7268  031624  012706  001000                  MOV     #STBOT,R6               ;SETUP STACK
7269  031630  016767  146200  151124          MOV     34,SLOC00               ;SAVE OLD  VECTOR
7270  031636  012767  031656  146170          MOV     #MTPOB,34               ;SETUP NEW TRAP VECTOR
7271  031644  005067  146126                  CLR     PS                      ;CLEAR PRIORITY AND COND C
7272  031650  000257                          CCC
7273  031652  104400                          TRAP
7274  031654  104001                  MTPOA:  ERROR   .1              ;CPU ERROR
7275                                                                  ;DIDNT TAKE CORRECT TRAP
7276  031656  026727  147114  000000  MTPOB:  CMP     STBOT-2,#0              ;VERIFY PSW ON STACK
7277  031664  001401                          BEQ     MTPOC                   ;BRANCH IF CORRECT STATUS
7278  031666  104001                          ERROR   .1              ;CPU ERROR
7279                                                                  ;BAD STATUS ON STACK
7280  031670  012706  001000          MTPOC:  MOV     #STBOT,R6               ;SETUP STACK
7281  031674  012767  031716  146132          MOV     #MTPOE,34               ;SET UP TRAP VECTOR
7282  031702  012767  000357  146066          MOV     #357,PS                 ;SET PRIORITY
7283  031710  000277                          SCC                             ;SET CONDITION CODES
7284  031712  104400                          TRAP                            ;ISSUE TRAP
7285  031714  104001                  MTPOD:  ERROR   .1              ;CPU ERROR
7286                                                                  ;DIDNT TAKE CORRECT TRAP
7287  031716  026727  147054  000357  MTPOE:  CMP     STBOT-2,#357            ;VERIFY OLD PSW ON STACK
7288  031724  001401                          BEQ     MTPOF                   ;BRANCH IF GOOD
7289  031726  104001                          ERROR   .1              ;CPU ERROR
7290                                                                  ;OLD PSW INCORRECT
7291  031730                          MTPOF:
7292  031730  016767  151026  146076          MOV     SLOC00,34               ;RESTORE TRAP VECTOR
7293  031736  012706  001000                  MOV     #STBOT,R6
7294
7295                                  ;
```

```
7296
7297  031742                                ;
                                            MTPA:
7298
7299                                        ;       TEST ALL TRAP OPCODES - SELF MODIFYING
7300  031742  005003                                CLR     R3                      ;SETUP REGISTER TO INDICATE OPCODE
7301  031744  012706  001000                        MOV     #STBOT,R6               ;SETUP STACK
7302  031750  016767  146060  151004              MOV     34,SLOC00               ;SAVE OLD  VECTOR
7303  031756  016767  146022  151000              MOV     4,SLOC01                ;SAVE IN CASE OF HALT
7304  031764  012767  032014  146012              MOV     #MTPAH,4                ;SETUP HALT TRAP
7305  031772  012767  032016  146034              MOV     #MTPAA,34               ;SETUP NEW TRAP VECTOR
7306  032000  000167  000012                      JMP     MTPAA                   ;GO INTO LOOPING CODE
7307                                        ;
7308  032004  000000                        MTPAL:  HALT                            ;SET TO A ZERO
7309  032006  104001                                ERROR   .1                      ;CPU ERROR
7310                                                                                ;TRAP INSTRUCTION FAILED TO TRAP
7311                                                                                ;EXAMINE OPCCODE AT LOCATION MTPAL:
7312  032010  000167  000002                      JMP     MTPAA                   ;ATTEMPT TO GO ON
7313                                        ;
7314  032014  104001                        MTPAH:  ERROR   .1                      ;CPU ERROR
7315                                                                                ;ERROR, EITHER CANT MODIFY LOCATION MTPAL
7316                                                                                ;OR TRAP INSTRUCTION FAILED
7317  032016                                MTPAA:
7318  032016  005203                                INC     R3                      ;GET NEXT OPCODE
7319
7320  032020  012706  001000                        MOV     #STBOT,R6               ;RESTORE STACK
7321  032024  020327  000400                        CMP     R3,#400                 ;SEE IF LAST OPCODE
7322  032030  001406                                BEQ     MTPAE                   ;BRANCH IF DONE
7323  032032  012767  104400  177744              MOV     #104400,MTPAL           ;TRAP OPCODE INTO LOCATION
7324  032040  060367  177740                      ADD     R3,MTPAL                ;FORM TEST OPCODE
7325  032044  000757                                BR      MTPAL                   ;EXECUTE TEST
7326  032046                                MTPAE:
7327
7328  032046  016767  150710  145760              MOV     SLOC00,34
7329  032054  016767  150704  145722              MOV     SLOC01,4                ;RESTORE VECTORS
7330
7331  032062  012706  001000                        MOV     #STBOT,R6
7332
7333                                            ;
7334                                            ;
7335  032066                                MIOT:
7336
7337                                        ;       TEST IOT TRAP
7338  032066  012706  001000                        MOV     #STBOT,R6               ;SETUP STACK
7339  032072  016767  145722  150662              MOV     20,SLOC00               ;SAVE OLD  VECTOR
7340  032100  012767  032112  145712              MOV     #MIOTB,20               ;SETUP NEW IOT VECTOR
7341  032106  000004                                IOT
7342  032110  104001                        MIOTA:  ERROR   .1                      ;CPU ERROR
7343                                                                                ;DIDNT TAKE CORRECT TRAP
7344  032112  022706  000774                MIOTB:  CMP     #STBOT-4,R6             ;VERIFY SP DECRIMENT
7345  032116  001401                                BEQ     MIOTD                   ;BRANCH IF GOOD
7346  032120  104001                                ERROR   .1                      ;CPU ERROR
7347                                                                                ;BAD PC ON STACK
7348  032122  021627  032110                MIOTD:  CMP     (R6),#MIOTA             ;VERFY PROPER PC ON STACK
7349  032126  001401                                BEQ     MIOTF                   ;BRANCH IF GOOD
7350  032130  104001                                ERROR   .1                      ;CPU ERROR
7351                                                                                ;INCORRECT PC ON STACK
```

```
7352  032132  016767  150624  145660  MIOTF:  MOV     SLOC00,20               ;RESTORE VECTOR
7353  032140  012706  001000          MOV     #STBOT,R6
7354
7355  032144                          MITO:
7356
7357                                  ;       TEST OLD STATUS ON IOT TRAP
7358  032144  012706  001000          MOV     #STBOT,R6                       ;SETUP STACK
7359  032150  016767  145644  150604  MOV     20,SLOC00                       ;SAVE OLD  VECTOR
7360  032156  012767  032176  145634  MOV     #MITOB,20                       ;SETUP NEW IOT VECTOR
7361  032164  005067  145606          CLR     PS                              ;CLEAR PRIORITY AND COND C
7362  032170  000257                  CCC
7363  032172  000004                  IOT
7364  032174  104001          MITOA:  ERROR   .1              ;CPU ERROR
7365                                                          ;DIDNT TAKE CORRECT TRAP
7366  032176  026727  146574  000000  MITOB:  CMP     STBOT-2,#0                      ;VERIFY PSW ON STACK
7367  032204  001401                  BEQ     MITOC                           ;BRANCH IF CORRECT STATUS
7368  032206  104001                  ERROR   .1              ;CPU ERROR
7369                                                          ;BAD STATUS ON STACK
7370  032210  012706  001000  MITOC:  MOV     #STBOT,R6                       ;SETUP STACK
7371  032214  012767  032236  145576  MOV     #MITOE,20                       ;SET UP TRAP VECTOR
7372  032222  012767  000357  145546  MOV     #357,PS                         ;SET PRIORITY
7373  032230  000277                  SCC                                     ;SET CONDITION CODES
7374  032232  000004                  IOT
7375  032234  104001          MITOD:  ERROR   .1              ;CPU ERROR
7376                                                          ;DIDNT TAKE CORRECT TRAP
7377  032236  026727  146534  000357  MITOE:  CMP     STBOT-2,#357                    ;VERIFY OLD PSW ON STACK
7378  032244  001401                  BEQ     MITOF                           ;BRANCH IF GOOD
7379  032246  104001                  ERROR   .1              ;CPU ERROR
7380                                                          ;OLD PSW INCORRECT
7381  032250                          MITOF:
7382  032250  016767  150506  145542  MOV     SLOC00,20                       ;RESTORE VECTOR
7383  032256  012706  001000          MOV     #STBOT,R6
7384
7385                                  ;
7386  032262                          MET:
7387
7388                                  ;       TEST EMULATOR TRAP INSTRUCTION  (EMT)
7389  032262  012706  001000          MOV     #STBOT,R6                       ;SETUP STACK
7390  032266  016767  145536  150466  MOV     30,SLOC00                       ;SAVE OLD  VECTOR
7391  032274  012767  032330  145526  MOV     #METB,30                        ;SETUP NEW EMT VECTOR
7392  032302  016767  145526  150454  MOV     34,SLOC01                       ;SAVE TRAP VECTOR
7393  032310  012767  134132  145516  MOV     #$ERROR,34                      ;SET UP TO HANDLE EMT ERROR
7394  032316  104000                  EMT
7395  032320  104400          META:   TRAP                                    ;TRAP ON ERROR
7396  032322  001057                  .WORD   559.
7397  032324  000001                  .WORD   1                               ;CPUERR
7398  032326  000001                  .WORD   1                               ;ERRTN
7399                                                          ;DIDNT TAKE CORRECT TRAP
7400  032330  022706  000774  METB:   CMP     #STBOT-4,R6                     ;VERIFY SP DECRIMENT
7401  032334  001401                  BEQ     METD                            ;BRANCH IF GOOD
7402  032336  104001                  ERROR   .1              ;CPU ERROR
7403                                                          ;BAD PC ON STACK
7404  032340  021627  032320  METD:   CMP     (R6),#META                      ;VERFY PROPER PC ON STACK
7405  032344  001401                  BEQ     METF                            ;BRANCH IF GOOD
7406  032346  104001                  ERROR   .1              ;CPU ERROR
7407                                                          ;INCORRECT PC ON STACK
```

```
 7408  032350  016767  150410  145456  METF:   MOV     SLOC01,34               ;RESTORE VECTOR
 7409  032356  016767  150400  145444          MOV     SLOC00,30               ;RESTORE VECTOR
 7410  032364  012706  001000                  MOV     #STBOT,R6
 7411
 7412                                  ;
 7413  032370                          METO:
 7414
 7415                                  ;       TEST OLD STATUS ON EMT TRAP
 7416  032370  012706  001000                  MOV     #STBOT,R6               ;SETUP STACK
 7417  032374  016767  145430  150360          MOV     30,SLOC00               ;SAVE OLD  VECTOR
 7418  032402  012767  032444  145420          MOV     #METOB,30               ;SETUP NEW EMT VECTOR
 7419  032410  016767  145420  150346          MOV     34,SLOC01               ;SAVE TRAP VECTOR
 7420  032416  012767  134132  145410          MOV     #$ERROR,34              ;SET UP TRAP VECTOR
 7421  032424  005067  145346                  CLR     PS                      ;CLEAR PRIORITY AND COND C
 7422  032430  000257                          CCC
 7423  032432  104000                          EMT
 7424  032434  104400                  METOA:  TRAP
 7425  032436  001062                          .WORD   562.
 7426  032440  000001                          .WORD   1                       ;CPUERR
 7427  032442  000001                          .WORD   1                       ;ERRTN
 7428                                                          ;DIDNT TAKE CORRECT TRAP
 7429  032444  026727  146326  000000  METOB:  CMP     STBOT-2,#0              ;VERIFY PSW ON STACK
 7430  032452  001401                          BEQ     METOC                   ;BRANCH IF CORRECT STATUS
 7431  032454  104001                          ERROR   .1                      ;CPU ERROR
 7432                                                          ;BAD STATUS ON STACK
 7433  032456  012706  001000          METOC:  MOV     #STBOT,R6               ;SETUP STACK
 7434  032462  012767  032512  145340          MOV     #METOE,30               ;SET UP TRAP VECTOR
 7435  032470  012767  000357  145300          MOV     #357,PS                 ;SET PRIORITY
 7436  032476  000277                          SCC                             ;SET CONDITION CODES
 7437  032500  104000                          EMT
 7438  032502  104400                  METOD:  TRAP
 7439  032504  001064                          .WORD   564.
 7440  032506  000001                          .WORD   1                       ;CPUERR
 7441  032510  000001                          .WORD   1                       ;ERRTN
 7442                                                          ;DIDNT TAKE CORRECT TRAP
 7443  032512  026727  146260  000357  METOE:  CMP     STBOT-2,#357            ;VERIFY OLD PSW ON STACK
 7444  032520  001401                          BEQ     METOF                   ;BRANCH IF GOOD
 7445  032522  104001                          ERROR   .1                      ;CPU ERROR
 7446                                                          ;OLD PSW INCORRECT
 7447  032524                          METOF:
 7448  032524  016767  150234  145302          MOV     SLOC01,34               ;RESTORE VECTOR
 7449  032532  016767  150224  145270          MOV     SLOC00,30               ;RESTORE VECTOR
 7450  032540  012706  001000                  MOV     #STBOT,R6
 7451
 7452                                  ;
 7453                                  ;
 7454  032544                          MBT:
 7455
 7456                                  ;       TEST BPT TRAP
 7457  032544  012706  001000                  MOV     #STBOT,R6               ;SETUP STACK
 7458  032550  016767  145240  150204          MOV     14,SLOC00               ;SAVE OLD  VECTOR
 7459  032556  012767  032570  145230          MOV     #MBTB,14                ;SETUP NEW BPT VECTOR
 7460  032564  000003                          BPT
 7461  032566  104001                  MBTA:   ERROR   .1                      ;CPU ERROR
 7462                                                          ;DIDNT TAKE CORRECT TRAP
 7463  032570  022706  000774          MBTB:   CMP     #STBOT-4,R6             ;VERIFY SP DECRIMENT
```

```
7464  032574  001401                        BEQ     MBTD                    ;BRANCH IF GOOD
7465  032576  104001                        ERROR   .1              ;CPU ERROR
7466                                                                ;BAD PC ON STACK
7467  032600  021627  032566        MBTD:   CMP     (R6),#MBTA              ;VERFY PROPER PC ON STACK
7468  032604  001401                        BEQ     MBTF                    ;BRANCH IF GOOD
7469  032606  104001                        ERROR   .1              ;CPU ERROR
7470                                                                ;INCORRECT PC ON STACK
7471  032610  016767  150146  145176  MBTF: MOV     SLOC00,14               ;RESTORE VECTOR
7472  032616  012706  001000                MOV     #STBOT,R6
7473
7474                                  ;
7475                                  ;
7476  032622                          MBTO:
7477
7478                                  ;       TEST OLD STATUS ON BPT TRAP
7479  032622  012706  001000                MOV     #STBOT,R6               ;SETUP STACK
7480  032626  016767  145162  150126        MOV     14,SLOC00               ;SAVE OLD  VECTOR
7481  032634  012767  032654  145152        MOV     #MBTOB,14               ;SETUP NEW BPT VECTOR
7482  032642  005067  145130                CLR     PS                      ;CLEAR PRIORITY AND COND C
7483  032646  000257                        CCC
7484  032650  000003                        BPT
7485  032652  104001                MBTOA:  ERROR   .1              ;CPU ERROR
7486                                                                ;DIDNT TAKE CORRECT TRAP
7487  032654  026727  146116  000000  MBTOB: CMP     STBOT-2,#0              ;VERIFY PSW ON STACK
7488  032662  001401                        BEQ     MBTOC           ;BRANCH IF CORRECT STATUS
7489  032664  104001                        ERROR   .1              ;CPU ERROR
7490                                                                ;BAD STATUS ON STACK
7491  032666  012706  001000        MBTOC:  MOV     #STBOT,R6               ;SETUP STACK
7492  032672  012767  032714  145114        MOV     #MBTOE,14               ;SET UP TRAP VECTOR
7493  032700  012767  000357  145070        MOV     #357,PS                 ;SET PRIORITY
7494  032706  000277                        SCC                             ;SET CONDITION CODES
7495  032710  000003                        BPT
7496  032712  104001                MBTOD:  ERROR   .1              ;CPU ERROR
7497                                                                ;DIDNT TAKE CORRECT TRAP
7498  032714  026727  146056  000357  MBTOE: CMP     STBOT-2,#357            ;VERIFY OLD PSW ON STACK
7499  032722  001401                        BEQ     MBTOF                   ;BRANCH IF GOOD
7500  032724  104001                        ERROR   .1              ;CPU ERROR
7501                                                                ;OLD PSW INCORRECT
7502  032726                          MBTOF:
7503  032726  016767  150030  145060        MOV     SLOC00,14               ;RESTORE VECTOR
7504  032734  012706  001000                MOV     #STBOT,R6
7505
7506                                  ;
7507                                  ;
7508  032740                          MIL:
7509
7510                                  ;       TEST ILLEGAL JUMP INSTRUCTION TRAP
7511  032740  012706  001000                MOV     #STBOT,R6               ;SETUP STACK
7512  032744  016767  145040  150010        MOV     10,SLOC00               ;SAVE OLD  VECTOR
7513  032752  012767  032766  145030        MOV     #MILB,10                ;SETUP NEW ILLEGAL VECTOR
7514  032760  005001                        CLR     R1
7515  032762  000101                        JMP     R1                      ;**TEST INSTRUCTIO
7516  032764  104001                MILA:   ERROR   .1              ;CPU ERROR
7517                                                                ;DIDNT TAKE CORRECT TRAP
7518  032766  022706  000774        MILB:   CMP     #STBOT-4,R6             ;VERIFY SP DECRIMENT
7519  032772  001401                        BEQ     MILD                    ;BRANCH IF GOOD
```

```
7520  032774  104001                        ERROR   ;1              ;CPU ERROR
7521                                                                  ;BAD PC ON STACK
7522  032776  021627  032764        MILD:   CMP     (R6),#MILA      ;VERFY PROPER PC ON STACK
7523  033002  001401                        BEQ     MILF            ;BRANCH IF GOOD
7524  033004  104001                        ERROR   ;1              ;CPU ERROR
7525                                                                  ;INCORRECT PC ON STACK
7526  033006  016767  147750  144774  MILF: MOV     SLOC00,10       ;RESTORE VECTOR
7527  033014  012706  001000                MOV     #STBOT,R6
7528
7529  033020                        MILO:
7530
7531                                ;       TEST OLD STATUS ON ILLEGAL JUMP TRAP
7532  033020  012706  001000                MOV     #STBOT,R6       ;SETUP STACK
7533  033024  016767  144760  147730        MOV     10,SLOC00       ;SAVE OLD  VECTOR
7534  033032  012767  033054  144750        MOV     #MILOB,10       ;SETUP NEW ILLEGAL VECTOR
7535  033040  005067  144732                CLR     PS              ;CLEAR PRIORITY AND COND C
7536  033044  000257                        CCC
7537  033046  005001                        CLR     R1
7538  033050  000101                        JMP     R1
7539  033052  104001                MILOA:  ERROR   ;1              ;CPU ERROR
7540                                                                  ;DIDNT TAKE CORRECT TRAP
7541  033054  026727  145716  000004  MILOB: CMP     STBOT-2,#4      ;VERIFY PSW ON STACK
7542  033062  001401                        BEQ     MILOC           ;BRANCH IF CORRECT STATUS
7543  033064  104001                        ERROR   ;1              ;CPU ERROR
7544                                                                  ;BAD STATUS ON STACK
7545  033066  012706  001000        MILOC:  MOV     #STBOT,R6       ;SETUP STACK
7546  033072  012767  033114  144710        MOV     #MILOE,10       ;SET UP TRAP VECTOR
7547  033100  012767  000357  144670        MOV     #357,PS         ;SET PRIORITY
7548  033106  000277                        SCC                     ;SET CONDITION CODES
7549  033110  000101                        JMP     R1
7550  033112  104001                MILOD:  ERROR   ;1              ;CPU ERROR
7551                                                                  ;DIDNT TAKE CORRECT TRAP
7552  033114  026727  145656  000357  MILOE: CMP     STBOT-2,#357    ;VERIFY OLD PSW ON STACK
7553  033122  001401                        BEQ     MILOF           ;BRANCH IF GOOD
7554  033124  104001                        ERROR   ;1              ;CPU ERROR
7555                                                                  ;OLD PSW INCORRECT
7556  033126                        MILOF:
7557  033126  016767  147630  144654        MOV     SLOC00,10       ;RESTORE VECTOR
7558  033134  012706  001000                MOV     #STBOT,R6
7559
7560                                ;
7561                                ;
7562  033140                        MIALL:
7563
7564                                ;       TEST ILLEGAL JSR INSTRUCTION TRAP
7565  033140  012706  001000                MOV     #STBOT,R6       ;SETUP STACK
7566  033144  016767  144640  147610        MOV     10,SLOC00       ;SAVE OLD  VECTOR
7567  033152  012767  033166  144630        MOV     #MIALLB,10      ;SETUP NEW ILLEGAL VECTOR
7568  033160  005003                        CLR     R3
7569  033162  004303                        JSR     R3,R3
7570  033164  104001                MIALLA: ERROR   ;1              ;CPU ERROR
7571                                                                  ;DIDNT TAKE CORRECT TRAP
7572  033166  022706  000774        MIALLB: CMP     #STBOT-4,R6     ;VERIFY SP DECRIMENT
7573  033172  001401                        BEQ     MIALLD          ;BRANCH IF GOOD
7574  033174  104001                        ERROR   ;1              ;CPU ERROR
7575                                                                  ;BAD PC ON STACK
```

```
7576  033176  021627  033164          MIALLD: CMP     (R6),#MIALLA            ;VERFY PROPER PC ON STACK
7577  033202  001401                          BEQ     MIALLF                  ;BRANCH IF GOOD
7578  033204  104001                          ERROR   +1              ;CPU ERROR
7579                                                                  ;INCORRECT PC ON STACK
7580  033206  016767  147550  144574  MIALLF: MOV     SLOC00,10               ;RESTORE VECTOR
7581  033214  012706  001000                  MOV     #STBOT,R6
7582
7583                                  ;
7584                                  ;
7585  033220                          MJSI:
7586
7587                                  ;       TEST OLD STATUS ON ILLEGAL JSR TRAP
7588  033220  012706  001000                  MOV     #STBOT,R6               ;SETUP STACK
7589  033224  016767  144560  147530          MOV     10,SLOC00               ;SAVE OLD  VECTOR
7590  033232  012767  033254  144550          MOV     #MJSIB,10               ;SETUP NEW VECTOR
7591  033240  005067  144532                  CLR     PS                      ;CLEAR PRIORITY AND COND C
7592  033244  000257                          CCC
7593  033246  005003                          CLR     R3
7594  033250  004303                          JSR     R3,R3
7595  033252  104001                  MJSIA:  ERROR   +1              ;CPU ERROR
7596                                                                  ;DIDNT TAKE CORRECT TRAP
7597  033254  026727  145516  000004  MJSIB:  CMP     STBOT-2,#4              ;VERIFY PSW ON STACK
7598  033262  001401                          BEQ     MJSIC                   ;BRANCH IF CORRECT STATUS
7599  033264  104001                          ERROR   +1              ;CPU ERROR
7600                                                                  ;BAD STATUS ON STACK
7601  033266  012706  001000          MJSIC:  MOV     #STBOT,R6               ;SETUP STACK
7602  033272  012767  033314  144510          MOV     #MJSIE,10               ;SET UP TRAP VECTOR
7603  033300  012767  000357  144470          MOV     #357,PS                 ;SET PRIORITY
7604  033306  000277                          SCC                             ;SET CONDITION CODES
7605  033310  004303                          JSR     R3,R3
7606  033312  104001                  MJSID:  ERROR   +1              ;CPU ERROR
7607                                                                  ;DIDNT TAKE CORRECT TRAP
7608  033314  026727  145456  000357  MJSIE:  CMP     STBOT-2,#357            ;VERIFY OLD PSW ON STACK
7609  033322  001401                          BEQ     MJSIF                   ;BRANCH IF GOOD
7610  033324  104001                          ERROR   +1              ;CPU ERROR
7611                                                                  ;OLD PSW INCORRECT
7612  033326                          MJSIF:
7613  033326  016767  147430  144454          MOV     SLOC00,10               ;RESTORE VECTOR
7614  033334  012706  001000                  MOV     #STBOT,R6
7615
7616                                  ;
7617                                  ;
7618  033340                          IOXXX:
7619                                  ;       I/O TIME OUT TEST
7620  033340  005067  144422                  CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
7621  033344  016767  144434  147410          MOV     4,SLOC00                ;SAVE VECTOR
7622  033352  012767  033374  144424          MOV     #2$,4                   ;SET UP VECTOR TO HANDLE NXM
7623  033360  012767  030000  144410          MOV     #30000,PS               ;INIT THE PSW TO A KNOWN STATE
7624  033366  005737  177700                  TST     @#177700                ;TRY TO ACCESS HARDWARE ADDRESS
7625                                                                          ;FOR GENERAL PURPOSE REG 0. THIS
7626                                                                          ;IS NOT IMPLEMENTED ON KDJ11
7627                                                                          ;SHOULD CAUSE TIME OUT.
7628  033372  104001                  1$:   ERROR   +1              ;CPU ERROR
7629  033374  022767  000020  144364  2$:     CMP     #BIT04,CPEREG           ;IS CPU ERROR REGISTER CORRECT?
7630  033402  001401                          BEQ     3$
7631  033404  104001                          ERROR   +1              ;CPU ERROR
```

```
7632  033406  022627  033372          3$:     CMP     (SP)+,#1$               ;CHECK THAT STACK CONTAINS CORRECT ADDR.
7633  033412  001401                          BEQ     4$                      ;
7634  033414  104001                          ERROR   +1              ;CPU ERROR
7635  033416  022627  030000          4$:     CMP     (SP)+,#30000            ;IS THE PSW OK?
7636  033422  001401                          BEQ     5$                      ;
7637  033424  104001                          ERROR   +1              ;CPU ERROR
7638  033426  005067  144334          5$:     CLR     CPEREG                  ;CLEAR THE CPU ERROR REGISTER
7639  033432  016767  147324  144344          MOV     SLOC00,4                ;RESTORE VECTOR
7640
7641  033440                          ODDXX:
7642
7643                                  ;       ODD ADDRESS/ILLEGAL INST FETCH TRAP TEST
7644                                  ;;**************************************************************
7645                                  ;THIS PROGRAM GENERATES AN ODD ADDRESS IN THE PC. THE KDJ11 SHOULD
7646                                  ;TRAP THROUGH ADDR 4
7647  033440  005067  144322                  CLR     CPEREG                  ;INIT THE CPU ERROR REG
7648  033444  016767  144334  147310          MOV     4,SLOC00                ;SAVE VECTOR
7649  033452  012767  033506  144324          MOV     #2$,4                   ;SET UP VECTOR TO HANDLE ODD ADDR TRAP
7650  033460  016767  144322  147276          MOV     6,SLOC01                ;SAVE VECTOR
7651  033466  005067  144314                  CLR     6                       ;INIT VECTOR
7652  033472  012746  030000                  MOV     #30000,-(SP)            ;PUSH A KNOWN PSW ON THE STACK
7653  033476  012746  033505                  MOV     #1$+1,-(SP)             ;PUSH AN ODD NUMBER ON THE STACK
7654  033502  000002                          RTI                             ;POP ODD ADDRESS OFF STACK INTO PC
7655                                                                          ;SHOULD TRAP HERE
7656  033504  104001                  1$:     ERROR   +1              ;CPU ERROR
7657  033506  022767  000100  144252  2$:     CMP     #BIT06,CPEREG           ;IS CPU ERROR REGISTER CORRECT?
7658  033514  001401                          BEQ     3$
7659  033516  104001                          ERROR   +1              ;CPU ERROR
7660  033520  022726  033505          3$:     CMP     #1$+1,(SP)+             ;IS STACK CONTENTS CORRECT?
7661  033524  001401                          BEQ     4$                      ;
7662  033526  104001                          ERROR   +1              ;CPU ERROR
7663  033530  022726  030000          4$:     CMP     #30000,(SP)+            ;IS PSW CORRECT ON STACK?
7664  033534  001401                          BEQ     5$                      ;
7665  033536  104001                          ERROR   +1              ;CPU ERROR
7666  033540  005067  144222          5$:     CLR     CPEREG                  ;CLEAR CPU ERROR REG
7667                                  ;
7668                                  ;NOW WE'LL TRY TO FETCH AN INSTRUCTION FROM AN INTERNAL REGISTER.
7669                                  ;THIS SHOULD CAUSE A TRAP TO ADDR 4 AND SET BIT 6 IN THE CPU
7670                                  ;ERROR REGISTER.
7671                                  ;
7672  033544  012767  033572  144232          MOV     #7$,4                   ;LOAD VECTOR WITH TRAP HANDLER ADDR
7673  033552  012767  030340  144226          MOV     #30340,6                ;LOAD VEC WITH PSW VALUE ON TRAP
7674  033560  005067  144212                  CLR     PS                      ;CLEAR THE PSW
7675  033564  000167  144206                  JMP     PS                      ;********TEST INSTRUCTION********
7676                                                                          ;TRY INSTRUCTION FETCH FROM INTERNAL
7677                                                                          ;REGISTER-- SHOULD TRAP VIA ADDR 4
7678  033570  104001                  6$:     ERROR   +1              ;CPU ERROR
7679  033572  016701  144200          7$:     MOV     PS,R1                   ;SAVE CONTENTS OF PSW IN R1
7680  033576  022767  000100  144162          CMP     #BIT06,CPEREG           ;IS CPU ERROR REGISTER CORRECT?
7681  033604  001401                          BEQ     8$                      ;
7682  033606  104001                          ERROR   +1              ;CPU ERROR
7683  033610  022726  177776          8$:     CMP     #PS,(SP)+               ;IS STACK CONTENTS CORRECT?
7684  033614  001401                          BEQ     9$                      ;
7685  033616  104001                          ERROR   +1              ;CPU ERROR
7686  033620  022726  000000          9$:     CMP     #0,(SP)+                ;IS STACK CONTENTS CORRECT?
7687  033624  001401                          BEQ     10$                     ;
```

```
7688  033626  104001                        ERROR   +1              ;CPU ERROR
7689  033630  022701  000340        10$:    CMP     #340,R1                 ;WAS PSW LOADED PROPERLY ON TRAP?
7690  033634  001401                        BEQ     11$                     ;
7691  033636  104001                        ERROR   +1              ;CPU ERROR
7692  033640  005067  144122        11$:    CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
7693  033644  016767  147112  144132        MOV     SLOC00,4                ;RESTORE VECTOR
7694  033652  016767  147106  144126        MOV     SLOC01,6                ;       "
7695
7696  033660                        RXXX:
7697
7698                                ;       RED ZONE TRAP TEST
7699  033660  013767  000004  147074        MOV     @#4,SLOC00              ;SAVE VECTOR
7700  033666  012737  033716  000004        MOV     #2$,@#4                 ;SET UP VECTOR
7701  033674  012706  000777                MOV     #777,R6                 ;SET UP THE STACK WITH ODD ADDRESS.
7702  033700  005067  144062                CLR     CPEREG                  ;CLEAR THE CPU ERROR REGISTER
7703  033704  005067  144066                CLR     PSW                     ;CLEAR THE PSW
7704  033710  005737  177700                TST     @#177700                ;ACCESS NON-EXISTANT I/O ADDRESS
7705  033714  104001                1$:     ERROR   +1              ;CPU ERROR
7706  033716  022706  000000        2$:     CMP     #0,SP                   ;IS R6 CORRECT?
7707  033722  001401                        BEQ     3$                      ;BRANCH IF YES
7708  033724  104001                        ERROR   +1              ;CPU ERROR
7709  033726  022726  033714        3$:     CMP     #1$,(SP)+               ;IS DATA AT ADDR 0 CORRECT?
7710  033732  001401                        BEQ     4$                      ;BRANCH IF YES
7711  033734  104001                        ERROR   +1              ;CPU ERROR
7712  033736  022716  000000        4$:     CMP     #0,(SP)                 ;IS PSW DATA IN ADDR 2 CORRECT?
7713  033742  001401                        BEQ     5$                      ;BRANCH IF YES
7714  033744  104001                        ERROR   +1              ;CPU ERROR
7715  033746  022767  000124  144012  5$:   CMP     #124,CPEREG             ;IS CPU ERROR REGISTER CORRECT?
7716  033754  001401                        BEQ     6$                      ;BRANCH IF YES
7717  033756  104001                        ERROR   +1              ;CPU ERROR
7718  033760  005067  144002        6$:     CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
7719  033764  012706  001000                MOV     #STBOT,SP               ;RESTORE STACK
7720  033770  016737  146766  000004        MOV     SLOC00,@#4              ;RESTORE VECTOR
7721  033776  005037  000000                CLR     @#0                     ;RESTORE ADDR 0
7722  034002  005037  000002                CLR     @#2                     ;RESTORE ADDR 2
7723
7724
7725
7726  034006                        PIRXXX:
7727                                ;       TEST PIRQ REGISTER RESPONSE
7728  034006  016767  143772  146746        MOV     4,SLOC00                ;SAVE CONTENTS OF VECTORS
7729  034014  012767  034042  143762        MOV     #PIRQNX,4               ;SETUP INTERRUPT VECTOR
7730  034022  005067  143740                CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
7731  034026  005737  177772                TST     @#PIRQ                  ;LOOK FOR REPLY FROM PIRQ
7732  034032  016767  146724  143744        MOV     SLOC00,4                ;RESTORE VECTORS
7733  034040  000401                        BR      PIRTEX                  ;IF IT RESPONDS, CONTINUE TESTING.
7734                                                                        ;ERROR! NO RESPONSE FROM PIRQ REG
7735  034042  104001                PIRQNX: ERROR   +1              ;CPU ERROR
7736  034044                        PIRTEX:
7737
7738  034044                        PIR1:
7739                                ;       TEST PIRQ REGISTER DATA
7740  034044  013767  000240  146710        MOV     @#PIRQVEC,SLOC00        ;SAVE PIRQ VECTORS
7741  034052  013767  000242  146704        MOV     @#PIRQVEC+2,SLOC01      ;       "
7742  034060  012737  034270  000240        MOV     #UNXPIR,@#PIRQVEC       ;SET UP PIRQ VECTOR FOR UNEXPECTED INTERRUPT
7743  034066  012737  000340  000242        MOV     #PR7,@#PIRQVEC+2        ;
```

```
7744  034074  012703  001000              MOV     #1000,R3                ;PUT 1000 IN R3: START TESTING
7745                                                                        ;BITS IN PIRQ REG BY FLOATING
7746                                                                        ;A BIT THROUGH BITS 9-15.
7747  034100  012704  034144              MOV     #PIRTBL,R4              ;SET UP R4 AS A POINTER TO EXPECTED
7748                                                                        ;ENCODED PRIORITY LEVELS IN PIRTBL
7749  034104  000237          1$:         SPL     7                       ;DON'T ALLOW INTERRUPTS.
7750  034106  005037  177772              CLR     @#PIRQ                  ;CLEAR OUT THE PIRQ
7751  034112  023724  177772              CMP     @#PIRQ,(R4)+            ;IS PIRQ OK??
7752  034116  001401                      BEQ     2$                      ;BRANCH IF OK
7753                                                                        ;ERROR; PIRQ REG WAS NOT CLEAR
7754  034120  104001                      ERROR   +1              ;CPU ERROR
7755  034122  010337  177772      2$:     MOV     R3,@#PIRQ               ;SET A BIT IN PIRQ REGISTER
7756  034126  023724  177772              CMP     @#PIRQ,(R4)+            ;COMPARE THE ENCODED PRIORITY BITS
7757                                                                        ;WITH DATA IN THE TABLE (PIRTBL)
7758  034132  001401                      BEQ     3$                      ;BRANCH IF ITS OK
7759                                                                        ;ERROR; ENCODED PRIORITY LEVELS ARE
7760                                                                        ;NOT CORRECT
7761  034134  104001                      ERROR   +1              ;CPU ERROR
7762  034136  006303          3$:         ASL     R3                      ;FLOAT A "1" THRU BITS 9-15
7763  034140  103370                      BCC     2$                      ;IF CARRY BIT ISN'T SET, DO AGAIN.
7764  034142  000410                      BR      EXPIR1                  ;GO TO EXIT TEST.
7765
7766  034144  000000          PIRTBL:     .WORD   0
7767  034146  001042                      .WORD   1042
7768  034150  002104                      .WORD   2104
7769  034152  004146                      .WORD   4146
7770  034154  010210                      .WORD   10210
7771  034156  020252                      .WORD   20252
7772  034160  040314                      .WORD   40314
7773  034162  100356                      .WORD   100356
7774
7775  034164                      EXPIR1:
7776
7777
7778  034164                      PIR2:
7779                                      ;       TEST PIRQ REGISTER LEVEL ENCODING
7780                                      ;;**************************************************************************
7781                                      ;THIS TEST IS TO CHECK THAT THE HIGHEST PRIORITY LEVEL SET IN THE
7782                                      ;PROGRAM INTERRUPT REQUEST BITS IS REFLECTED IN THE ENCODED PROGRAM
7783                                      ;INTERRUPT ACTIVE BITS.
7784  034164  000237                      SPL     7                       ;SHUT OFF INTERRUPTS
7785  034166  005037  177772              CLR     @#PIRQ                  ;CLEAR PIRQ REGISTER
7786  034172  012767  000001  146576      MOV     #1,FLAG                 ;SETUP END OF LOOP SIGNAL
7787  034200  012703  177000              MOV     #177000,R3              ;SET UP DESIRED PATTERN IN R3
7788  034204  012704  034250              MOV     #PITBL1,R4              ;SETUP R4 AS A TABLE POINTER
7789  034210  010337  177772      1$:     MOV     R3,@#PIRQ               ;SET PATERN IN PIRQ REGISTER
7790
7791  034214  023724  177772              CMP     @#PIRQ,(R4)+            ;COMPARE PATTERN IN PIRQ WITH PATTERN IN
7792                                                                        ;EXPECTED PATTERN TABLE.
7793  034220  001012                      BNE     2$                      ;GO TO ERROR IF NOT THE SAME
7794  034222  005737  002776              TST     @#FLAG                  ;IS THIS THE LAST TIME THROUGH??
7795  034226  001421                      BEQ     PIR2EX                  ;YES, IF FLAG IS ZERO
7796  034230  006003                      ROR     R3                      ;SHIFT PATTERN TO RIGHT FOR NEXT TEST
7797  034232  042703  000777              BIC     #777,R3                 ;STRIP OFF BITS  8-0
7798  034236  001364                      BNE     1$                      ;IF R3 IS NOT = 0 GO DO THE NEXT PATTERN
7799  034240  005037  002776              CLR     @#FLAG                  ;CLR THE FLAG TO INDICATE LAST
```

```
7800                                                                    ;TIME THROUGH THE LOOP
7801  034244  000761                    BR      1$                      ;GO THROUGH LOOP ONCE MORE
7802                                                                    ;ERROR; PATTERNS WERE NOT THE SAME
7803  034246  104001            2$:     ERROR   +1              ;CPU ERROR
7804
7805  034250  177356  077314  037252  PITBL1: .WORD   177356,77314,37252,17210,7146,3104,1042,0
7806  034256  017210  007146  003104
7807  034264  001042  000000
7808
7809  034270                    UNXPIR:                                 ;UNEXPECTED PIRQ INTERRUPT
7810  034270  104001                    ERROR   +1              ;CPU ERROR
7811
7812  034272                    PIR2EX:
7813
7814  034272                    PIR3:
7815                            ;       TEST PIRQ INTERRUPTS
7816                            ;;**************************************************************
7817                            ;THIS TEST CHECKS THAT EACH PROGRAM INTERRUPT OCCURS PROPERLY
7818  034272  012703  001000            MOV     #1000,R3                ;SETUP R3 AS A WORKING REGISTER
7819  034276  012737  034342  000240      MOV     #PIRRTN,@#PIRQVEC       ;SET UP INTERRUPT ROUTINE AT PIR VECTOR
7820  034304  012737  000340  000242      MOV     #340,@#PIRQVEC+2        ;       "       "       "       "
7821  034312  005004                    CLR     R4                      ;INITIALIZE R4 AS EXPECTED DATA HOLDER.
7822  034314  005724            PI1:    TST     (R4)+                   ;INCREMENT R4 BY 2
7823  034316  050337  177772            BIS     R3,@#PIRQ               ;SET PIRQ TO INTERRUPT
7824  034322  000230                    SPL     0                       ;ENABLE INTERRUPT TO OCCUR
7825                                                                    ;ERROR! INTERRUPT DID NOT OCCUR
7826  034324  104001                    ERROR   +1              ;CPU ERROR
7827                                                                    ;ERROR! INTERRUPT OCCURRED IN WRONG SEQUENCE
7828  034326  104001            PI2:    ERROR   +1              ;CPU ERROR
7829  034330  062706  000004    PI3:    ADD     #4,SP                   ;CLEAN UP THE STACK
7830  034334  006303                    ASL     R3                      ;SHIFT R3 TO LEFT FOR NEXT INTERRUPT
7831  034336  103366                    BCC     PI1                     ;END IF CARRY BIT IS SET
7832  034340  000412                    BR      PIR3EX                  ;ON TO THE NEXT TEST
7833
7834  034342  013705  177772    PIRRTN: MOV     @#PIRQ,R5               ;MOVE THE CONTENTS OF PIRQ REG TO R5
7835  034346  005037  177772            CLR     @#PIRQ                  ;KILL PIRQ INTERRUPT.
7836  034352  042705  177761            BIC     #177761,R5              ;MASK OFF ALL BITS EXCEPT 1-4.
7837  034356  020405                    CMP     R4,R5                   ;DID THE CORRECT LEVEL INTERRUPT OCCUR?
7838  034360  001362                    BNE     PI2                     ;IF NOT; GO TO ERROR.
7839  034362  000167  177742            JMP     PI3                     ;RETURN FROM SUCCESSFUL INTERRUPT,GET
7840                                                                    ;READY FOR THE NEXT ONE.
7841
7842  034366                    PIR3EX:
7843
7844
7845  034366                    PIR4:
7846                            ;       TEST PIRQ VS PSW INTERRUPT LEVEL
7847                            ;;**************************************************************
7848                            ;THIS TEST IS TO ENSURE THAT PIRQ CANNOT INTERRUPT WHEN PSW INTERRUPT
7849                            ;LEVEL IS SET TO BLOCK THEM OUT.
7850  034366  005037  177772            CLR     @#PIRQ                  ;CLEAR THE PIRQ
7851  034372  005037  177776            CLR     @#PSW                   ;CLEAR THE PSW.
7852  034376  000237                    SPL     7                       ;BLOCK INTERRUPTS FROM BEING SERVICED.
7853  034400  012703  001000            MOV     #1000,R3                ;USE R3 AS A WORKING REGISTER.
7854  034404  012737  034436  000240      MOV     #2$,@#PIRQVEC           ;SETUP INTERRUPT VECTORS
7855  034412  012737  000340  000242      MOV     #340,@#PIRQVEC+2                ;
```

```
7856  034420  010337  177772          1$:     MOV     R3,@#PIRQ               ;SET INTERRUPT LEVEL IN PIRQ.
7857  034424  006303                          ASL     R3                      ;SHIFT A "1" LEFT TO INCREASE INTERRUPT
7858                                                                          ;PRIORITY LEVEL.
7859  034426  103374                          BCC     1$                      ;IF C BIT NOT SET THEN DO IT AGAIN.
7860  034430  005037  177772                  CLR     @#PIRQ                  ;ELSE CLEAR THE PIRQ
7861  034434  000403                          BR      3$                      ;EXIT TEST.
7862
7863  034436  005037  177772          2$:     CLR     @#PIRQ                  ;STOP PIRQ FROM INTERRUPTING.
7864                                                                          ;ERROR! NO INTERRUPTS SHOULD OCCUR.
7865  034442  104001                          ERROR   .+1             ;CPU ERROR
7866  034444                          3$:
7867
7868
7869  034444                          PIR5:
7870                                  ;       TEST PIRQ INTERRUPTS OCCUR AT PROPER LEVEL
7871                                  ;;**********************************************************************
7872                                  ;THIS TEST ENSURES THAT INTERRUPTS OCCUR AT THE PROPER LEVEL
7873                                  ;THE PRIORITY LEVEL IS INITIALLY SET TO PRI6. THE PIRQ THEN STARTS INTERRUPTING
7874                                  ;AT PRI1. NO INTERRUPTS SHOULD OCCUR UNTIL THE INTERRUPTING LEVEL EXCEEDS THE
7875                                  ;PRIORITY LEVEL IN THE PSW. AFTER EACH LEVEL OF INTERRUPT HAS BEEN TRIED; THE
7876                                  ;PSW PRIORITY LEVEL IS LOWERED ONE NOTCH, AND THE CYCLE REPEATS UNTIL PSW
7877                                  ;PRIORITY LEVEL 0 IS REACHED. INTERRUPTS AT PSW PRIORITY LEVEL 0 HAVE BEEN DONE
7878                                  ;PREVIOUSLY.
7879  034444  005037  177772                  CLR     @#PIRQ                  ;CLR THE PIRQ
7880  034450  012737  034562  000240          MOV     #10$,@#PIRQVEC          ;SET UP INTERRUPT VECTORS
7881  034456  012737  000340  000242          MOV     #PR7,@#PIRQVEC+2        ;
7882  034464  012705  034640                  MOV     #PITBL2,R5              ;SETUP R5 AS A TABLE POINTER.
7883  034470  012737  000006  110500          MOV     #6,@#DCOUNT             ;INIT LOCATION DCOUNT:
7884  034476  005004                  1$:     CLR     R4                      ;INITIALIZE R4 AS A COUNTER.
7885  034500  012703  001000                  MOV     #1000,R3                ;USE R3 AS A WORKING REGISTER.
7886  034504  012567  143266                  MOV     (R5)+,PS                ;MOVE PRIORITY LEVEL TO PSW
7887  034510  005724                  3$:     TST     (R4)+                   ;INCREMENT R4 BY 2
7888  034512  010337  177772                  MOV     R3,@#PIRQ               ;SET INTERRUPT LEVEL IN PIRQ.
7889
7890  034516  013701  177776          ;*******INTERRUPTS WILL HAPPEN HERE*********
7891  034522  013702  177772                  MOV     @#PS,R1                 ;SAVE PS IN R1          !ONLY EXECUTED
7892  034526  042701  177437                  MOV     @#PIRQ,R2               ;SAVE PIRQ IN R2        !   IF NO
7893  034532  042702  177437                  BIC     #177437,R1              ;CLEAR EXTRANEOUS BITS  ! INTERRUPT
7894  034536  020102                          BIC     #177437,R2              ;                       !   HAS
7895  034540  002001                          CMP     R1,R2                   ;R1 SHOULD BE >= R2     !  OCCURRED
7896                                          BGE     4$                      ;IF IT IS GO ON         !-------
7897  034542  104001                          ERROR   .+1                     ;ERROR! SHOULD HAVE INTERRUPTED.
7898  034544  006303                  4$:     ASL     R3              ;CPU ERROR
7899                                                                          ;SHIFT A "1" LEFT UNTIL INTERRUPT LEVEL
7900  034546  103360                          BCC     3$                      ;IS REACHED.
7901  034550  005337  110500                  DEC     @#DCOUNT                ;BRANCH BACK IF CARRY IS NOT SET.
7902  034554  001350                          BNE     1$                      ;LOWER INTERRUPT PRIORITY LEVEL
7903  034556  000167  000072                  JMP     PIR5EX                  ;IF DCOUNT= 0 , WE'RE DONE.
7904                                                                          ;JUMP TO END OF THIS TEST.
7905                                  ;PIRQ INTERRUPT SERVICE ROUTINE
7906                                  ;
7907
7908  034562  013746  177772          10$:    MOV     @#PIRQ,-(SP)            ;SAVE PIRQ DATA ON STACK
7909  034566  005037  177772                  CLR     @#PIRQ                  ;SHUT OFF PIRQ INTERRUPTS
7910  034572  016601  000004                  MOV     4(SP),R1                ;GET OLD PSW. SAVE IN R1.
7911  034576  011602                          MOV     (SP),R2                 ;GET PIRQ FROM STACK.
```

```
7912  034600  042702  177437          BIC     #177437,R2              ;CLEAR UNWANTED BITS FROM R2
7913  034604  042701  177437          BIC     #177437,R1              ;CLEAR UNWANTED BITS FROM R1
7914  034610  020102                  CMP     R1,R2                   ;R2 SHOULD BE > R1.
7915  034612  100401                  BMI     20$                     ;GO CHECK SEQUENCE OF INTERRUPT.
7916                                                                  ;ERROR! PRIORITY OF INTERRUPT WAS NOT
7917                                                                  ;HIGH ENOUGH. SHOULDN'T HAVE OCCURRED.
7918  034614  104001                  ERROR   .1              ;CPU ERROR
7919  034616  012602          20$:    MOV     (SP)+,R2                ;POP OLD PIRQ OFF THE STACK.
7920  034620  042702  177761          BIC     #177761,R2              ;CLEAR OFF EXTRANEOUS BITS.
7921  034624  020402                  CMP     R4,R2                   ;SHOULD BE EQUAL.
7922  034626  001401                  BEQ     21$                     ;IF THEY ARE EQUAL, CLEAN UP STACK AND
7923                                                                  ;GET READY FOR THE NEXT INTERRUPT
7924                                                                  ;ELSE
7925                                                                  ;ERROR! INTERRUPT OCCURRED OUT OF SEQUENCE.
7926  034630  104001                  ERROR   .1              ;CPU ERROR
7927  034632  012716  034544  21$:    MOV     #4$,(SP)                ;PUT RETURN ADDRESS ON THE STACK.
7928  034636  000002                  RTI                             ;RETURN FROM INTERRUPT. RESTORE PSW.
7929
7930  034640  000300          PITBL2: .WORD   300             ;PRIORITY LEVEL 6
7931  034642  000240                  .WORD   240             ;PRIORITY LEVEL 5
7932  034644  000200                  .WORD   200             ;PRIORITY LEVEL 4
7933  034646  000140                  .WORD   140             ;PRIORITY LEVEL 3
7934  034650  000100                  .WORD   100             ;PRIORITY LEVEL 2
7935  034652  000040                  .WORD   40              ;PRIORITY LEVEL 1
7936
7937  034654          PIR5EX:
7938
7939
7940  034654          PIR6:
7941                  ;       TEST THAT PIRQS ARE SERVICED IN CORRECT ORDER
7942                  ;;**************************************************************
7943                  ;THIS TEST CHECKS THAT ALL PIRQ INTERRUPTS ARE SERVICED
7944                  ;IN THE CORRECT DECENDING ORDER. I.E. IRQ6 IS NOT SERVICED
7945                  ;BEFORE IRQ7 ETC. THE PIRQ IS LOADED WITH ALL INTERRUPTS SIMULTANEOUSLY
7946                  ;TURNED ON.  THE PSW PRIORITY LEVEL IS THEN LOWERED TO 0. EACH INTERRUPT
7947                  ;SHOULD BE SERVICED IN DECENDING ORDER. EACH TIME AN INTERRUPT OCCURRS, THE PSW
7948                  ;IS LOADED WITH PRIORITY LEVEL 7 WHICH STOPS THE PIRQ FROM FURTHER INTERRUPTS.
7949                  ;AFTER A CORRECT INTERRUPT, AN RTI IS EXECUTED, AND THE PSW PRIORITY LEVEL
7950                  ;RETRUNS TO ZERO ALLOWING THE NEXT LOWER INTERRUPT TO OCCUR.
7951  034654  005037  177772                  CLR     @#PIRQ                  ;CLEAR OUT THE PIRQ
7952  034660  000237                          SPL     7                       ;NO INTERRUPTS WILL BE SERVICED.
7953  034662  012737  034722  000240          MOV     #10$,@#PIRQVEC          ;SET UP VECTORS
7954  034670  012737  000340  000242          MOV     #PR7,@#PIRQVEC+2        ;
7955  034676  012703  177000                  MOV     #177000,R3              ;SETUP DESIRED PATTERN IN R3
7956  034702  010337  177772                  MOV     R3,@#PIRQ               ;MOVE PATTERN IN R3 TO PIRQ.
7957  034706  012704  000016                  MOV     #16,R4                  ;PRELOAD R4 AS A DECREMENTING COUNTER.
7958  034712  000230                          SPL     0                       ;LET THE PIRQ INTERRUPT
7959                  ;*********INTERRUPTS OCCUR HERE!!!*********************
7960  034714  005704                          TST     R4                      ;IS R4 ZERO? IT SHOULD BE. THIS
7961                                                                          ;INSTRUCTION SHOULDN'T BE EXECUTED
7962                                                                          ;UNTIL ALL INTERRUPTS HAVE OCCURRED.
7963  034716  001417                          BEQ     PIR6EX                  ;GO TO NEXT TEST.
7964                                                                          ;ERROR! ALL INTERRUPTS DID NOT OCCUR.
7965  034720  104001                          ERROR   .1              ;CPU ERROR
7966  034722  013702  177772          10$:    MOV     @#PIRQ,R2               ;SAVE THE PIRQ IN R2
7967  034726  042702  177761                  BIC     #177761,R2              ;STRIP OFF EXTRANEOUS BITS.
```

```
7968  034732  020402                        CMP     R4,R2                   ;SHOULD BE EQUAL.
7969  034734  001401                        BEQ     15$                     ;SETUP FOR NEXT INTERRUPT.
7970                                                                          ;ERROR! INTERRUPT WAS SERVICED OUT OF ORDER
7971  034736  104001                        ERROR   .1              ;CPU ERROR
7972  034740  005744                15$:    TST     -(R4)                   ;DECREMENT R4 BY 2
7973  034742  006003                        ROR     R3                      ;CHANGE PATTERN IN R3; LOWER INTERRUPT PRIORITY
7974  034744  042703  000777                BIC     #777,R3                 ;STRIP OFF UNNEEDED BITS
7975  034750  010337  177772                MOV     R3,@#PIRQ               ;LOWER INTERRUPT PRIORITY LEVEL.
7976  034754  000002                        RTI                             ;
7977
7978  034756  016737  146000  000240  PIR6EX: MOV    SLOC00,@#PIRQVEC        ;RESTORE VECTORS
7979  034764  016737  145774  000242        MOV     SLOC01,@#PIRQVEC+2              ;       "
7980                                                      ;
7981
7982                                .SBTTL  MEMORY MANAGEMENT TESTS
7983                                ;*******************************************************************************
7984                                ;*******************************************************************************
7985                                ;                       BEGIN MMU TESTING
7986                                ;*******************************************************************************
7987                                ;*******************************************************************************
7988  034772                        TSMMU1:
7989                                ;       STATUS REGISTER TEST
7990  034772  005067  142770                CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
7991  034776  005037  177572                CLR     @#177572                ;TURN MMU OFF
7992  035002  005037  002776                CLR     @#FLAG                  ;CLEAR MMU TRAP FLAG
7993  035006  013746  000004                MOV     @#4,-(SP)               ;SAVE OLD VECTOR
7994  035012  012737  133302  000004        MOV     #ADDTRP,@#4             ;SETUP NEW VECTOR
7995  035020  005005                        CLR     R5                      ;CLEAR FLAG
7996  035022  013701  177572                MOV     @#177572,R1             ; TEST MMR0
7997  035026  013701  177574                MOV     @#177574,R1             ; TEST MMR1
7998  035032  013701  177576                MOV     @#177576,R1             ; TEST MMR2
7999  035036  013701  172516                MOV     @#172516,R1             ; TEST MMR3
8000  035042  012637  000004                MOV     (SP)+,@#4               ;RESTORE VECTOR
8001  035046  020527  000000                CMP     R5,#0                   ;DID WE TRAP
8002  035052  001401                        BEQ     1$                      ;NO, THEN BRANCH
8003  035054  104002                        ERROR   .2              ;MMU ERROR
8004                                                                  ;YES, GO TO ERROR
8005  035056                        1$:
8006  035056                        TSMMU2:
8007                                ;       ADDRESS TEST OF PARS,PDRS, AND FP REGS
8008  035056  005067  142704                CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
8009  035062  005037  177572                CLR     @#177572                ;MMU OFF
8010  035066  005037  002776                CLR     @#FLAG                  ;CLEAR MMU TRAP FLAG
8011  035072  013746  000244                MOV     @#244,-(SP)             ;SAVE FP VECTOR
8012  035076  013746  000246                MOV     @#246,-(SP)
8013  035102  013746  000004                MOV     @#4,-(SP)               ;SAVE TIME OUT VECTOR
8014  035106  012737  000246  000244        MOV     #246,@#244              ;SETUP NEW FP VECTOR
8015  035114  012737  000002  000246        MOV     #2,@#246                ;
8016  035122  012737  133302  000004        MOV     #ADDTRP,@#4             ;SETUP NEW TIME OUT VECTOR
8017  035130  005005                        CLR     R5                      ;CLEAR TIMEOUT FLAG
8018  035132  012700  172200                MOV     #172200,R0              ;LOAD ALL PARS AND PDRS WITH ZERO
8019  035136  005020                1$:     CLR     (R0)+                   ;
8020  035140  020027  172400                CMP     R0,#172400              ;
8021  035144  001374                        BNE     1$                      ;
8022  035146  012700  177600                MOV     #177600,R0              ;
8023  035152  005020                2$:     CLR     (R0)+                   ;
```

```
8024  035154  020027  177700          CMP     R0,#177700              ;
8025  035160  001374                  BNE     2$                      ;
8026  035162  170127  000200          LDFPS   #200                    ;
8027  035166  012700  003012          MOV     #FLOAT,R0               ;LOAD AC0-AC5 WITH 0
8028  035172  005020                  CLR     (R0)+                   ;
8029  035174  005020                  CLR     (R0)+                   ;
8030  035176  005020                  CLR     (R0)+                   ;
8031  035200  005020                  CLR     (R0)+                   ;
8032  035202  012700  003012          MOV     #FLOAT,R0               ;
8033  035206  172410                  LDD     (R0),AC0                ;
8034  035210  172510                  LDD     (R0),AC1                ;
8035  035212  172610                  LDD     (R0),AC2                ;
8036  035214  172710                  LDD     (R0),AC3                ;
8037  035216  174004                  STD     AC0,AC4                 ;
8038  035220  174005                  STD     AC0,AC5                 ;
8039  035222  174500          3$:     DIVD    AC0,AC1                 ;LOAD FEC WITH 4 AND FEA WITH #3$
8040  035224  170337  003022          STST    @#FLO                   ;CHECK FEC FOR 4 AND FEA FOR #3$
8041  035230  012704  003022          MOV     #FLO,R4                 ;
8042  035234  022427  000004          CMP     (R4)+,#4                ;
8043  035240  001401                  BEQ     21$                     ;
8044  035242  104002                  ERROR   +2              ;MMU ERROR
8045                                                          ;
8046  035244  021427  035222  21$:     CMP     (R4),#3$                ;
8047  035250  001401                  BEQ     22$                     ;
8048  035252  104002                  ERROR   +2              ;MMU ERROR
8049                                                          ;
8050  035254  012704  172200  22$:     MOV     #172200,R4              ;CHECK EACH PAR, PDR FOR 0 THEN
8051  035260  012701  000001          MOV     #1,R1                   ;WRITE A UNIQUE NUMBER TO IT
8052  035264  010102          4$:     MOV     R1,R2                   ;
8053  035266  072227  000010          ASH     #10,R2                  ;
8054  035272  021427  000000          CMP     (R4),#0                 ;
8055  035276  001401                  BEQ     5$                      ;
8056  035300  104002                  ERROR   +2              ;MMU ERROR
8057                                                          ;
8058  035302  010224          5$:     MOV     R2,(R4)+                ;
8059  035304  005201                  INC     R1                      ;
8060  035306  020427  172400          CMP     R4,#172400              ;
8061  035312  001364                  BNE     4$                      ;
8062  035314  012704  177600          MOV     #177600,R4              ;
8063  035320  010102          6$:     MOV     R1,R2                   ;
8064  035322  072227  000010          ASH     #10,R2                  ;
8065  035326  021427  000000          CMP     (R4),#0                 ;
8066  035332  001401                  BEQ     7$                      ;
8067  035334  104002                  ERROR   +2              ;MMU ERROR
8068                                                          ;
8069  035336  010224          7$:     MOV     R2,(R4)+                ;
8070  035340  005201                  INC     R1                      ;
8071  035342  020427  177700          CMP     R4,#177700              ;
8072  035346  001364                  BNE     6$                      ;
8073  035350  012704  003022          MOV     #FLO,R4                 ;CHECK AC5 FOR ALL ZEROES THEN LOAD A &
8074  035354  012703  003012          MOV     #FLOAT,R3               ;
8075  035360  174014                  STD     AC0,(R4)                ;
8076  035362  172405                  LDD     AC5,AC0                 ;
8077  035364  174013                  STD     AC0,(R3)                ;
8078  035366  012702  000004          MOV     #4,R2                   ;
8079  035372  022327  000000  8$:     CMP     (R3)+,#0                ;
```

```
8080  035376  001401                    BEQ     9$                  ;
8081  035400  104002                    ERROR   .2                 ;MMU ERROR
8082                                                                ;
8083  035402  005302              9$:   DEC     R2                  ;
8084  035404  001372                    BNE     8$                  ;
8085  035406  012703  003012            MOV     #FLOAT,R3           ;
8086  035412  012713  000006            MOV     #6,(R3)             ;
8087  035416  172413                    LDD     (R3),AC0            ;
8088  035420  174005                    STD     AC0,AC5             ;
8089  035422  172404                    LDD     AC4,AC0             ;CHECK AC4 FOR ALL ZEROES THEN LOAD A 5
8090  035424  174013                    STD     AC0,(R3)            ;
8091  035426  012702  000004            MOV     #4,R2               ;
8092  035432  022327  000000      10$:  CMP     (R3)+,#0            ;
8093  035436  001401                    BEQ     11$                 ;
8094  035440  104002                    ERROR   .2                  ;MMU ERROR
8095                                                                ;
8096  035442  005302              11$:  DEC     R2                  ;
8097  035444  001372                    BNE     10$                 ;
8098  035446  012703  003012            MOV     #FLOAT,R3           ;
8099  035452  012713  000005            MOV     #5,(R3)             ;
8100  035456  172413                    LDD     (R3),AC0            ;
8101  035460  174004                    STD     AC0,AC4             ;
8102  035462  012702  000004            MOV     #4,R2               ;CHECK AC0 FOR ALL ZEROES THEN LOAD A 1
8103  035466  022427  000000      12$:  CMP     (R4)+,#0            ;
8104  035472  001401                    BEQ     13$                 ;
8105  035474  104002                    ERROR   .2                  ;MMU ERROR
8106                                                                ;
8107  035476  005302              13$:  DEC     R2                  ;
8108  035500  001372                    BNE     12$                 ;
8109  035502  012713  000001            MOV     #1,(R3)             ;
8110  035506  172413                    LDD     (R3),AC0            ;
8111  035510  012704  003022            MOV     #FLO,R4             ;CHECK AC1 FOR ALL ZEROES THEN LOAD A 2
8112  035514  012702  000004            MOV     #4,R2               ;
8113  035520  174114                    STD     AC1,(R4)            ;
8114  035522  022427  000000      14$:  CMP     (R4)+,#0            ;
8115  035526  001401                    BEQ     15$                 ;
8116  035530  104002                    ERROR   .2                  ;MMU ERROR
8117                                                                ;
8118  035532  005302              15$:  DEC     R2                  ;
8119  035534  001372                    BNE     14$                 ;
8120  035536  012713  000002            MOV     #2,(R3)             ;
8121  035542  172513                    LDD     (R3),AC1            ;
8122  035544  012704  003022            MOV     #FLO,R4             ;CHECK AC2 FOR ALL ZEROES THEN LOAD A 3
8123  035550  012702  000004            MOV     #4,R2               ;
8124  035554  174214                    STD     AC2,(R4)            ;
8125  035556  022427  000000      16$:  CMP     (R4)+,#0            ;
8126  035562  001401                    BEQ     17$                 ;
8127  035564  104002                    ERROR   .2                  ;MMU ERROR
8128                                                                ;
8129  035566  005302              17$:  DEC     R2                  ;
8130  035570  001372                    BNE     16$
8131  035572  012713  000003            MOV     #3,(R3)             ;
8132  035576  172613                    LDD     (R3),AC2            ;
8133  035600  012704  003022            MOV     #FLO,R4             ;CHECK AC3 FOR ALL ZEROES THEN LOAD A 4
8134  035604  012702  000004            MOV     #4,R2               ;
8135  035610  174314                    STD     AC3,(R4)            ;
```

```
8136  035612  022427  000000          18$:    CMP     (R4)+,#0                ;
8137  035616  001401                          BEQ     19$                     ;
8138  035620  104002                          ERROR   .2              ;MMU ERROR
8139                                                                  ;
8140  035622  005302                    19$:  DEC     R2                      ;
8141  035624  001372                          BNE     18$                     ;
8142  035626  012713  000004                  MOV     #4,(R3)                 ;
8143  035632  172713                          LDD     (R3),AC3                ;
8144  035634  012704  003022                  MOV     #FLO,R4                 ;CHECK FPS FOR 100204 THEN LOAD IT WITH 200
8145  035640  170214                          STFPS   (R4)                    ;
8146  035642  022714  100204                  CMP     #100204,(R4)            ;
8147  035646  001401                          BEQ     20$                     ;
8148  035650  104002                          ERROR   .2              ;MMU ERROR
8149                                                                  ;
8150  035652  170127  000200            20$:  LDFPS   #200                    ;
8151  035656  012704  172200                  MOV     #172200,R4              ;CHECK PDR, PAR FOR UNIQUE NUMBERS
8152  035662  012701  000001                  MOV     #1,R1                   ;
8153  035666  010102                    23$:  MOV     R1,R2                   ;
8154  035670  072227  000010                  ASH     #10,R2                  ;
8155  035674  022402                          CMP     (R4)+,R2                ;
8156  035676  001401                          BEQ     24$                     ;
8157  035700  104002                          ERROR   .2              ;MMU ERROR
8158                                                                  ;
8159  035702  005201                    24$:  INC     R1                      ;
8160  035704  020427  172400                  CMP     R4,#172400              ;
8161  035710  001366                          BNE     23$                     ;
8162  035712  012704  177600                  MOV     #177600,R4              ;
8163  035716  010102                    25$:  MOV     R1,R2                   ;
8164  035720  072227  000010                  ASH     #10,R2                  ;
8165  035724  022402                          CMP     (R4)+,R2                ;
8166  035726  001401                          BEQ     26$                     ;
8167  035730  104002                          ERROR   .2              ;MMU ERROR
8168                                                                  ;
8169  035732  005201                    26$:  INC     R1                      ;
8170  035734  020427  177700                  CMP     R4,#177700              ;
8171  035740  001366                          BNE     25$                     ;
8172  035742  012701  003012                  MOV     #FLOAT,R1               ;CHECK AC5 FOR #6
8173  035746  012704  003022                  MOV     #FLO,R4                 ;
8174  035752  174014                          STD     AC0,(R4)                ;
8175  035754  172405                          LDD     AC5,AC0                 ;
8176  035756  174011                          STD     AC0,(R1)                ;
8177  035760  022127  000006                  CMP     (R1)+,#6                ;
8178  035764  001401                          BEQ     27$                     ;
8179  035766  104002                          ERROR   .2              ;MMU ERROR
8180                                                                  ;
8181  035770  012703  000003            27$:  MOV     #3,R3                   ;
8182  035774  022127  000000            28$:  CMP     (R1)+,#0                ;
8183  036000  001401                          BEQ     29$                     ;
8184  036002  104002                          ERROR   .2              ;MMU ERROR
8185                                                                  ;
8186  036004  005303                    29$:  DEC     R3                      ;
8187  036006  001372                          BNE     28$                     ;
8188  036010  012701  003012                  MOV     #FLOAT,R1               ;CHECK AC4 FOR #5
8189  036014  172404                          LDD     AC4,AC0                 ;
8190  036016  174011                          STD     AC0,(R1)                ;
8191  036020  022127  000005                  CMP     (R1)+,#5                ;
```

```
8192  036024  001401                        BEQ     30$                     ;
8193  036026  104002                        ERROR   .2                    ;MMU ERROR
8194                                                                        ;
8195  036030  012703  000003        30$:    MOV     #3,R3                   ;
8196  036034  022127  000000        31$:    CMP     (R1)+,#0                ;
8197  036040  001401                        BEQ     32$                     ;
8198  036042  104002                        ERROR   .2                    ;MMU ERROR
8199                                                                        ;
8200  036044  005303                32$:    DEC     R3                      ;
8201  036046  001372                        BNE     31$                     ;
8202  036050  022427  000001                CMP     (R4)+,#1                ;CHECK AC0 FOR #1
8203  036054  001401                        BEQ     33$                     ;
8204  036056  104002                        ERROR   .2                    ;MMU ERROR
8205                                                                        ;
8206  036060  012703  000003        33$:    MOV     #3,R3                   ;
8207  036064  022427  000000        34$:    CMP     (R4)+,#0                ;
8208  036070  001401                        BEQ     35$                     ;
8209  036072  104002                        ERROR   .2                    ;MMU ERROR
8210                                                                        ;
8211  036074  005303                35$:    DEC     R3                      ;
8212  036076  001372                        BNE     34$                     ;
8213  036100  012701  003012                MOV     #FLOAT,R1               ;CHECK AC1 FOR #2
8214  036104  174111                        STD     AC1,(R1)                ;
8215  036106  022127  000002                CMP     (R1)+,#2                ;
8216  036112  001401                        BEQ     36$                     ;
8217  036114  104002                        ERROR   .2                    ;MMU ERROR
8218                                                                        ;
8219  036116  012703  000003        36$:    MOV     #3,R3                   ;
8220  036122  022127  000000        37$:    CMP     (R1)+,#0                ;
8221  036126  001401                        BEQ     38$                     ;
8222  036130  104002                        ERROR   .2                    ;MMU ERROR
8223                                                                        ;
8224  036132  005303                38$:    DEC     R3                      ;
8225  036134  001372                        BNE     37$                     ;
8226  036136  012701  003012                MOV     #FLOAT,R1               ;CHECK AC2 FOR #3
8227  036142  174211                        STD     AC2,(R1)                ;
8228  036144  022127  000003                CMP     (R1)+,#3                ;
8229  036150  001401                        BEQ     39$                     ;
8230  036152  104002                        ERROR   .2                    ;MMU ERROR
8231                                                                        ;
8232  036154  012703  000003        39$:    MOV     #3,R3                   ;
8233  036160  022127  000000        40$:    CMP     (R1)+,#0                ;
8234  036164  001401                        BEQ     41$                     ;
8235  036166  104002                        ERROR   .2                    ;MMU ERROR
8236                                                                        ;
8237  036170  005303                41$:    DEC     R3                      ;
8238  036172  001372                        BNE     40$                     ;
8239  036174  012701  003012                MOV     #FLOAT,R1               ;CHECK AC3 FOR #4
8240  036200  174311                        STD     AC3,(R1)                ;
8241  036202  022127  000004                CMP     (R1)+,#4                ;
8242  036206  001401                        BEQ     42$                     ;
8243  036210  104002                        ERROR   .2                    ;MMU ERROR
8244                                                                        ;
8245  036212  012703  000003        42$:    MOV     #3,R3                   ;
8246  036216  022127  000000        43$:    CMP     (R1)+,#0                ;
8247  036222  001401                        BEQ     44$                     ;
```

```
8248  036224  104002                    ERROR   .2              ;MMU ERROR
8249                                    ;
8250  036226  005303            44$:    DEC     R3                      ;
8251  036230  001372                    BNE     43$                     ;
8252  036232  020527  000000            CMP     R5,#0                   ;IS TIME OUT FLAG 0
8253  036236  001401                    BEQ     45$                     ;YES GO ON
8254  036240  104002                    ERROR   .2              ;MMU ERROR
8255                                    ;NO GO TO ERROR
8256  036242  012637  000004    45$:    MOV     (SP)+,@#4               ;RESTORE TIME OUT VECTOR
8257  036246  012637  000246            MOV     (SP)+,@#246             ;RESTORE FP VECTOR
8258  036252  012637  000244            MOV     (SP)+,@#244             ;
8259
8260  036256                    TSMMU3:
8261                            ;       WRITE ALL PARS/PDRS WITH ONES THEN ZEROS
8262  036256  005037  177572            CLR     @#177572                ;MMU OFF
8263  036262  005037  002776            CLR     @#FLAG                  ;CLEAR MMU ABORT FLAG
8264  036266  012703  172200            MOV     #172200,R3              ;LOAD ALL PARS AND PDRS WITH ONES
8265  036272  012723  177777    1$:     MOV     #177777,(R3)+           ;
8266  036276  020327  172400            CMP     R3,#172400              ;
8267  036302  001373                    BNE     1$                      ;
8268  036304  012703  177600            MOV     #177600,R3              ;
8269  036310  012723  177777    2$:     MOV     #177777,(R3)+           ;
8270  036314  020327  177700            CMP     R3,#177700              ;
8271  036320  001373                    BNE     2$                      ;
8272  036322  012703  172200            MOV     #172200,R3              ;CHECK SPDRS FOR ONES
8273  036326  022327  177416    3$:     CMP     (R3)+,#177416           ;
8274  036332  001401                    BEQ     4$                      ;
8275  036334  104002                    ERROR   .2              ;MMU ERROR
8276                                    ;
8277  036336  020327  172240    4$:     CMP     R3,#172240              ;
8278  036342  001371                    BNE     3$                      ;
8279  036344  022327  177777    5$:     CMP     (R3)+,#177777           ;CHECK SPARS FOR ONES
8280  036350  001401                    BEQ     6$                      ;
8281  036352  104002                    ERROR   .2              ;MMU ERROR
8282                                    ;
8283  036354  020327  172300    6$:     CMP     R3,#172300              ;
8284  036360  001371                    BNE     5$                      ;
8285  036362  022327  177416    7$:     CMP     (R3)+,#177416           ;CHECK KPDRS FOR ONES
8286  036366  001401                    BEQ     8$                      ;
8287  036370  104002                    ERROR   .2              ;MMU ERROR
8288                                    ;
8289  036372  020327  172340    8$:     CMP     R3,#172340              ;
8290  036376  001371                    BNE     7$                      ;
8291  036400  022327  177777    9$:     CMP     (R3)+,#177777           ;CHECK KPARS FOR ONES
8292  036404  001401                    BEQ     10$                     ;
8293  036406  104002                    ERROR   .2              ;MMU ERROR
8294                                    ;
8295  036410  020327  172400    10$:    CMP     R3,#172400              ;
8296  036414  001371                    BNE     9$                      ;
8297  036416  012703  177600            MOV     #177600,R3              ;CHECK UPDRS FOR ONES
8298  036422  022327  177416    11$:    CMP     (R3)+,#177416           ;
8299  036426  001401                    BEQ     12$                     ;
8300  036430  104002                    ERROR   .2              ;MMU ERROR
8301                                    ;
8302  036432  020327  177640    12$:    CMP     R3,#177640              ;
8303  036436  001371                    BNE     11$                     ;
```

```
8304  036440  022327  177777          13$:    CMP     (R3)+,#177777           ;CHECK UPARS FOR ONES
8305  036444  001401                          BEQ     14$                     ;
8306  036446  104002                          ERROR   +2              ;MMU ERROR
8307                                          ;
8308  036450  020327  177700          14$:    CMP     R3,#177700              ;
8309  036454  001371                          BNE     13$                     ;
8310  036456  012703  172200                  MOV     #172200,R3              ;LOAD ALL PARS AND PDRS WITH ZEROES
8311  036462  012723  000000          15$:    MOV     #0,(R3)+                ;
8312  036466  020327  172400                  CMP     R3,#172400              ;
8313  036472  001373                          BNE     15$                     ;
8314  036474  012703  177600                  MOV     #177600,R3              ;
8315  036500  012723  000000          16$:    MOV     #0,(R3)+                ;
8316  036504  020327  177700                  CMP     R3,#177700              ;
8317  036510  001373                          BNE     16$                     ;
8318  036512  012703  172200                  MOV     #172200,R3              ;CHECK ALL PARS AND PDRS FOR ZEROES
8319  036516  022327  000000          17$:    CMP     (R3)+,#0                ;
8320  036522  001401                          BEQ     18$                     ;
8321  036524  104002                          ERROR   +2              ;MMU ERROR
8322                                          ;
8323  036526  020327  172400          18$:    CMP     R3,#172400              ;
8324  036532  001371                          BNE     17$                     ;
8325  036534  012703  177600                  MOV     #177600,R3              ;
8326  036540  022327  000000          19$:    CMP     (R3)+,#0                ;
8327  036544  001401                          BEQ     20$                     ;
8328  036546  104002                          ERROR   +2              ;MMU ERROR
8329                                          ;
8330  036550  020327  177700          20$:    CMP     R3,#177700              ;
8331  036554  001371                          BNE     19$                     ;
8332
8333  036556                          TSMMU4:
8334                                  ;       TEST FOR ADJACENT SHORTS IN PARS/PDRS
8335  036556  005037  177572                  CLR     @#177572                ;MMU OFF
8336  036562  005067  144210                  CLR     FLAG                    ;CLEAR MMU ABORT FLAG
8337  036566  012700  172200                  MOV     #172200,R0              ;LOAD SPDRS WITH ALTERNATING PATTERN
8338  036572  012720  052404          1$:     MOV     #52404,(R0)+            ;
8339  036576  012720  125012                  MOV     #125012,(R0)+           ;
8340  036602  020027  172240                  CMP     R0,#172240              ;
8341  036606  001371                          BNE     1$                      ;
8342  036610  012720  125252          2$:     MOV     #125252,(R0)+           ;LOAD SPARS WITH ALTERNATING PATTERN
8343  036614  012720  052525                  MOV     #52525,(R0)+            ;
8344  036620  020027  172300                  CMP     R0,#172300              ;
8345  036624  001371                          BNE     2$                      ;
8346  036626  012720  052404          3$:     MOV     #52404,(R0)+            ;LOAD KPDRS WITH ALTERNATING PATTERN
8347  036632  012720  125012                  MOV     #125012,(R0)+           ;
8348  036636  020027  172340                  CMP     R0,#172340              ;
8349  036642  001371                          BNE     3$                      ;
8350  036644  012720  125252          4$:     MOV     #125252,(R0)+           ;LOAD KPARS WITH ALTERNATING PATTERN
8351  036650  012720  052525                  MOV     #52525,(R0)+            ;
8352  036654  020027  172400                  CMP     R0,#172400              ;
8353  036660  001371                          BNE     4$                      ;
8354  036662  012700  177600                  MOV     #177600,R0              ;LOAD UPDRS WITH ALTERNATING PATTERN
8355  036666  012720  052404          5$:     MOV     #52404,(R0)+            ;
8356  036672  012720  125012                  MOV     #125012,(R0)+           ;
8357  036676  020027  177640                  CMP     R0,#177640              ;
8358  036702  001371                          BNE     5$                      ;
8359  036704  012720  125252          6$:     MOV     #125252,(R0)+           ;LOAD UPARS WITH ALTERNATING PATTERN
```

```
8360  036710  012720  052525              MOV     #52525,(R0)+            ;
8361  036714  020027  177700              CMP     R0,#177700              ;
8362  036720  001371                      BNE     6$                      ;
8363                                  ;
8364  036722  012703  172200              MOV     #172200,R3              ;CHECK SPDRS
8365  036726  022327  052404        7$:   CMP     (R3)+,#52404            ;
8366  036732  001401                      BEQ     8$                      ;
8367  036734  104002                      ERROR   .+2             ;MMU ERROR
8368                                                                  ;
8369  036736  022327  125012        8$:   CMP     (R3)+,#125012           ;
8370  036742  001401                      BEQ     9$                      ;
8371  036744  104002                      ERROR   .+2             ;MMU ERROR
8372                                                                  ;
8373  036746  020327  172240        9$:   CMP     R3,#172240              ;
8374  036752  001365                      BNE     7$                      ;
8375  036754  022327  125252        10$:  CMP     (R3)+,#125252           ;CHECK SPARS
8376  036760  001401                      BEQ     11$                     ;
8377  036762  104002                      ERROR   .+2             ;MMU ERROR
8378                                                                  ;
8379  036764  022327  052525        11$:  CMP     (R3)+,#52525            ;
8380  036770  001401                      BEQ     12$                     ;
8381  036772  104002                      ERROR   .+2             ;MMU ERROR
8382                                                                  ;
8383  036774  020327  172300        12$:  CMP     R3,#172300              ;
8384  037000  001365                      BNE     10$                     ;
8385  037002  022327  052404        13$:  CMP     (R3)+,#52404            ;CHECK KPDRS
8386  037006  001401                      BEQ     14$                     ;
8387  037010  104002                      ERROR   .+2             ;MMU ERROR
8388                                                                  ;
8389  037012  022327  125012        14$:  CMP     (R3)+,#125012           ;
8390  037016  001401                      BEQ     15$                     ;
8391  037020  104002                      ERROR   .+2             ;MMU ERROR
8392                                                                  ;
8393  037022  020327  172340        15$:  CMP     R3,#172340              ;
8394  037026  001365                      BNE     13$                     ;
8395  037030  022327  125252        16$:  CMP     (R3)+,#125252           ;CHECK KPARS
8396  037034  001401                      BEQ     17$                     ;
8397  037036  104002                      ERROR   .+2             ;MMU ERROR
8398                                                                  ;
8399  037040  022327  052525        17$:  CMP     (R3)+,#52525            ;
8400  037044  001401                      BEQ     18$                     ;
8401  037046  104002                      ERROR   .+2             ;MMU ERROR
8402                                                                  ;
8403  037050  020327  172400        18$:  CMP     R3,#172400              ;
8404  037054  001365                      BNE     16$                     ;
8405  037056  012703  177600              MOV     #177600,R3              ;CHECK UPDRS
8406  037062  022327  052404        19$:  CMP     (R3)+,#52404            ;
8407  037066  001401                      BEQ     20$                     ;
8408  037070  104002                      ERROR   .+2             ;MMU ERROR
8409                                                                  ;
8410  037072  022327  125012        20$:  CMP     (R3)+,#125012           ;
8411  037076  001401                      BEQ     21$                     ;
8412  037100  104002                      ERROR   .+2             ;MMU ERROR
8413                                                                  ;
8414  037102  020327  177640        21$:  CMP     R3,#177640              ;
8415  037106  001365                      BNE     19$                     ;
```

```
8416  037110  022327  125252          22$:    CMP     (R3)+,#125252           ;CHECK UPARS
8417  037114  001401                          BEQ     23$                     ;
8418  037116  104002                          ERROR   +2              ;MMU ERROR
8419                                                                  ;
8420  037120  022327  052525          23$:    CMP     (R3)+,#52525            ;
8421  037124  001401                          BEQ     24$                     ;
8422  037126  104002                          ERROR   +2              ;MMU ERROR
8423                                                                  ;
8424  037130  020327  177700          24$:    CMP     R3,#177700              ;
8425  037134  001365                          BNE     22$                     ;
8426                                  ;
8427                                  ;REVERSE ALTERNATING PATTERN
8428                                  ;
8429  037136  012700  172200                  MOV     #172200,R0              ;LOAD SPDRS WITH REVERSE PATTERN
8430  037142  012720  125012          25$:    MOV     #125012,(R0)+           ;
8431  037146  012720  052404                  MOV     #52404,(R0)+            ;
8432  037152  020027  172240                  CMP     R0,#172240              ;
8433  037156  001371                          BNE     25$                     ;
8434  037160  012720  052525          26$:    MOV     #52525,(R0)+            ;LOAD SPARS WITH REVERSE PATTERN
8435  037164  012720  125252                  MOV     #125252,(R0)+           ;
8436  037170  020027  172300                  CMP     R0,#172300              ;
8437  037174  001371                          BNE     26$                     ;
8438  037176  012720  125012          27$:    MOV     #125012,(R0)+           ;LOAD KPDRS WITH REVERSE PATTERN
8439  037202  012720  052404                  MOV     #52404,(R0)+            ;
8440  037206  020027  172340                  CMP     R0,#172340              ;
8441  037212  001371                          BNE     27$                     ;
8442  037214  012720  052525          28$:    MOV     #52525,(R0)+            ;LOAD KPARS WITH REVERSE PATTERN
8443  037220  012720  125252                  MOV     #125252,(R0)+           ;
8444  037224  020027  172400                  CMP     R0,#172400              ;
8445  037230  001371                          BNE     28$                     ;
8446  037232  012700  177600                  MOV     #177600,R0              ;LOAD UPDRS WITH REVERSE PATTERN
8447  037236  012720  125012          29$:    MOV     #125012,(R0)+           ;
8448  037242  012720  052404                  MOV     #52404,(R0)+            ;
8449  037246  020027  177640                  CMP     R0,#177640              ;
8450  037252  001371                          BNE     29$                     ;
8451  037254  012720  052525          30$:    MOV     #52525,(R0)+            ;LOAD UPARS WITH REVERSE PATTERN
8452  037260  012720  125252                  MOV     #125252,(R0)+           ;
8453  037264  020027  177700                  CMP     R0,#177700              ;
8454  037270  001371                          BNE     30$                     ;
8455                                  ;
8456  037272  012703  172200                  MOV     #172200,R3              ;CHECK SPDRS
8457  037276  022327  125012          31$:    CMP     (R3)+,#125012           ;
8458  037302  001401                          BEQ     32$                     ;
8459  037304  104002                          ERROR   +2              ;MMU ERROR
8460                                                                  ;
8461  037306  022327  052404          32$:    CMP     (R3)+,#52404            ;
8462  037312  001401                          BEQ     33$                     ;
8463  037314  104002                          ERROR   +2              ;MMU ERROR
8464                                                                  ;
8465  037316  020327  172240          33$:    CMP     R3,#172240              ;
8466  037322  001365                          BNE     31$                     ;
8467  037324  022327  052525          34$:    CMP     (R3)+,#52525            ;CHECK SPARS
8468  037330  001401                          BEQ     35$                     ;
8469  037332  104002                          ERROR   +2              ;MMU ERROR
8470                                                                  ;
8471  037334  022327  125252          35$:    CMP     (R3)+,#125252           ;
```

```
037340  001401                          BEQ     36$                     ;
037342  104002                          ERROR   +2                      ;MMU ERROR
                                                                        ;
037344  020327  172300          36$:    CMP     R3,#172300              ;
037350  001365                          BNE     34$                     ;
037352  022327  125012          37$:    CMP     (R3)+,#125012           ;CHECK KPDRS
037356  001401                          BEQ     38$                     ;
037360  104002                          ERROR   +2                      ;MMU ERROR
                                                                        ;
037362  022327  052404          38$:    CMP     (R3)+,#52404            ;
037366  001401                          BEQ     39$                     ;
037370  104002                          ERROR   +2                      ;MMU ERROR
                                                                        ;
037372  020327  172340          39$:    CMP     R3,#172340              ;
037376  001365                          BNE     37$                     ;
037400  022327  052525          40$:    CMP     (R3)+,#52525            ;CHECK KPARS
037404  001401                          BEQ     41$                     ;
037406  104002                          ERROR   +2                      ;MMU ERROR
                                                                        ;
037410  022327  125252          41$:    CMP     (R3)+,#125252           ;
037414  001401                          BEQ     42$                     ;
037416  104002                          ERROR   +2                      ;MMU ERROR
                                                                        ;
037420  020327  172400          42$:    CMP     R3,#172400              ;
037424  001365                          BNE     40$                     ;
037426  012703  177600                  MOV     #177600,R3              ;CHECK UPDRS
037432  022327  125012          43$:    CMP     (R3)+,#125012           ;
037436  001401                          BEQ     44$                     ;
037440  104002                          ERROR   +2                      ;MMU ERROR
                                                                        ;
037442  022327  052404          44$:    CMP     (R3)+,#52404            ;
037446  001401                          BEQ     45$                     ;
037450  104002                          ERROR   +2                      ;MMU ERROR
                                                                        ;
037452  020327  177640          45$:    CMP     R3,#177640              ;
037456  001365                          BNE     43$                     ;
037460  022327  052525          46$:    CMP     (R3)+,#52525            ;CHECK UPARS
037464  001401                          BEQ     47$                     ;
037466  104002                          ERROR   +2                      ;MMU ERROR
                                                                        ;
037470  022327  125252          47$:    CMP     (R3)+,#125252           ;
037474  001401                          BEQ     48$                     ;
037476  104002                          ERROR   +2                      ;MMU ERROR
                                                                        ;
037500  020327  177700          48$:    CMP     R3,#177700              ;
037504  001365                          BNE     46$                     ;

037506                          TSMMU5:
                                ;       TEST MMR0 ABORT BITS
037506  012737  160000  177572          MOV     #160000,@#177572        ;LOAD MMR0<15:13>=111
037514  005067  143256                  CLR     FLAG                    ;CLEAR MMU ABORT FLAG
037520  013700  177572                  MOV     @#SR0,R0                ;SAVE SR0 IN R0
037524  042700  000176                  BIC     #176,R0                 ;CLEAR UNDEFINED BITS FROM SR0
037530  020027  160000                  CMP     R0,#160000              ;CHECK MMR0
037534  001401                          BEQ     1$                      ;
037536  104002                          ERROR   +2                      ;MMU ERROR
```

```
8528                                                    ;
8529  037540  005037  177572          1$:     CLR     @#177572                ;LOAD MMR0=0
8530  037544  013700  177572                  MOV     @#SR0,R0                ;SAVE SR0 IN R0
8531  037550  042700  000176                  BIC     #176,R0                 ;CLEAR UNDEFINED BITS FROM SR0
8532  037554  020027  000000                  CMP     R0,#0                   ;CHECK MMR0
8533  037560  001401                          BEQ     2$                      ;
8534  037562  104002                          ERROR   .+2             ;MMU ERROR
8535                                                    ;
8536  037564  012737  120000  177572  2$:     MOV     #120000,@#177572        ;LOAD MMR0<15:13>=101
8537  037572  013700  177572                  MOV     @#SR0,R0                ;SAVE SR0 IN R0
8538  037576  042700  000176                  BIC     #176,R0                 ;CLEAR UNDEFINED BITS FROM SR0.
8539  037602  020027  120000                  CMP     R0,#120000              ;CHECK MMR0
8540  037606  001401                          BEQ     3$                      ;
8541  037610  104002                          ERROR   .+2             ;MMU ERROR
8542                                                    ;
8543  037612  012737  040000  177572  3$:     MOV     #40000,@#177572         ;LOAD MMR0<15:13>=010
8544  037620  013700  177572                  MOV     @#SR0,R0                ;SAVE SR0 IN R0
8545  037624  042700  000176                  BIC     #176,R0                 ;CLEAR UNDEFINED BITS FROM SR0.
8546  037630  020027  040000                  CMP     R0,#40000               ;CHECK MMR0
8547  037634  001401                          BEQ     4$                      ;
8548  037636  104002                          ERROR   .+2             ;MMU ERROR
8549  037640                          4$:
8550  037640                          TSMMU6:
8551                                  ;       TEST MMR3 BITS 5-0
8552  037640  005037  177572                  CLR     @#177572                ;MMU OFF
8553  037644  005067  143126                  CLR     FLAG                    ;CLEAR MMU ABORT FLAG
8554  037650  012737  000077  172516          MOV     #77,@#172516            ;LOAD MMR3<5:0>=77
8555  037656  023727  172516  000077          CMP     @#172516,#77            ;CHECK MMR3
8556  037664  001401                          BEQ     1$                      ;
8557  037666  104002                          ERROR   .+2             ;MMU ERROR
8558  037670  005037  172516          1$:     CLR     @#172516                ;LOAD MMR3<5:0>=0
8559  037674  023727  172516  000000          CMP     @#172516,#0             ;CHECK MMR3
8560  037702  001401                          BEQ     2$                      ;
8561  037704  104002                          ERROR   .+2             ;MMU ERROR
8562  037706  012737  000052  172516  2$:     MOV     #52,@#172516            ;LOAD MMR3<5:0>=52
8563  037714  023727  172516  000052          CMP     @#172516,#52            ;CHECK MMR3
8564  037722  001401                          BEQ     3$                      ;
8565  037724  104002                          ERROR   .+2             ;MMU ERROR
8566  037726  012737  000025  172516  3$:     MOV     #25,@#172516            ;LOAD MMR3<5:0>=25
8567  037734  023727  172516  000025          CMP     @#172516,#25            ;CHECK MMR3
8568  037742  001401                          BEQ     4$                      ;
8569  037744  104002                          ERROR   .+2             ;MMU ERROR
8570  037746                          4$:
8571  037746                          TSMM6A:
8572                                  ;       TEST MFPI (MOVE FROM PREVIOUS INST SPACE)
8573  037746  005037  177572                  CLR     @#177572                ;MMU OFF
8574  037752  005037  002776                  CLR     @#FLAG                  ;CLEAR MMU ABORT FLAG
8575  037756  012737  140000  177776          MOV     #140000,@#177776        ;POINT TO USER SPACE
8576  037764  012706  001000                  MOV     #STBOT,SP               ;INIT THE USER STACK POINTER
8577  037770  010637  003002                  MOV     R6,@#SAVUSE             ;SAVE USER SP
8578  037774  012737  040000  177776          MOV     #40000,@#177776         ;POINT TO SUPERVISOR SPACE
8579  040002  012706  001000                  MOV     #STBOT,SP               ;INIT THE SUPERVISOR STACK POINTER
8580  040006  010637  003000                  MOV     R6,@#SAVSUP             ;SAVE SUPERVISOR SP
8581  040012  012737  030000  177776          MOV     #30000,@#177776         ;SETUP PSW
8582  040020  004767  073110                  JSR     PC,MMU                  ;INIT MMU
8583  040024  012737  000027  172516          MOV     #27,@#172516            ;SETUP MMR3
```

```
8584  040032  013746  000244              MOV     @#244,-(SP)             ;SAVE DATA AT TEST LOCATION
8585  040036  012746  177777              MOV     #177777,-(SP)           ;PUT KNOWN DATA ON TOP OF STACK
8586  040042  012737  135072  000244      MOV     #135072,@#244           ;SETUP DATA AT TEST LOCATION
8587  040050  012767  077400  137542      MOV     #77400,UDPDR0           ;SETUP UDPDR0 TO ABORT
8588  040056  012703  000244              MOV     #244,R3                 ;SETUP POINTER TO TEST LOCATION
8589  040062  005237  177572              INC     @#177572                ;TURN MMU ON
8590  040066  006523                       MFPI    (R3)+                   ; TEST INSTRUCTION
8591  040070  022737  030010  177776      CMP     #30010,@#177776         ;IS PSW CORRECT
8592  040076  001401                       BEQ     1$                      ;YES GO ON
8593  040100  104002                       ERROR   .2              ;MMU ERROR
8594                                                                ;NO GO TO ERROR
8595  040102  005037  177572      1$:     CLR     @#177572                ;TURN MMU OFF
8596  040106  012737  140000  177776      MOV     #140000,@#177776        ;POINT TO USER SPACE
8597  040114  020637  003002              CMP     R6,@#SAVUSE             ;IS USER SP CORRECT
8598  040120  001401                       BEQ     100$                    ;YES GO ON
8599  040122  104002                       ERROR   .2              ;MMU ERROR
8600                                                                ;NO GO TO ERROR
8601  040124  012737  040000  177776  100$: MOV     #40000,@#177776         ;POINT TO SUPERVISOR SPACE
8602  040132  020637  003000              CMP     R6,@#SAVSUP             ;IS SUPERVISOR SP CORRECT
8603  040136  001401                       BEQ     200$                    ;YES GO ON
8604  040140  104002                       ERROR   .2              ;MMU ERROR
8605                                                                ;NO GO TO ERROR
8606  040142  023727  000244  135072  200$: CMP     @#244,#135072           ;IS TEST DATA OK
8607  040150  001401                       BEQ     2$                      ;YES GO ON
8608  040152  104002                       ERROR   .2              ;MMU ERROR
8609                                                                ;NO GO TO ERROR
8610  040154  020327  000246          2$:  CMP     R3,#246                 ;IS R3 CORRECT
8611  040160  001401                       BEQ     3$                      ;YES GO ON
8612  040162  104002                       ERROR   .2              ;MMU ERROR
8613                                                                ;NO GO TO ERROR
8614  040164  005037  177776          3$:  CLR     @#177776                ;SET PSW TO KERNEL MODE
8615  040170  022627  135072              CMP     (SP)+,#135072           ;IS KERNEL STACK CORRECT
8616  040174  001401                       BEQ     4$                      ;YES GO ON
8617  040176  104002                       ERROR   .2              ;MMU ERROR
8618                                                                ;NO GO TO ERROR
8619  040200  021627  177777          4$:  CMP     (SP),#177777            ;IS STACK CORRECT
8620  040204  001401                       BEQ     5$                      ;YES GO ON
8621  040206  104002                       ERROR   .2              ;MMU ERROR
8622                                                                ;NO GO TO ERROR
8623  040210  012737  030017  177776  5$:  MOV     #30017,@#177776         ;SETUP PSW
8624  040216  012737  173621  000244      MOV     #173621,@#244           ;SETUP TEST LOCATION
8625  040224  012701  000244              MOV     #244,R1                 ;SETUP R1
8626  040230  005237  177572              INC     @#177572                ;TURN MMU ON
8627  040234  006511                       MFPI    (R1)                    ;TEST INSTRUCTION
8628  040236  022737  030011  177776      CMP     #30011,@#177776         ;IS PSW CORRECT
8629  040244  001401                       BEQ     300$                    ;YES GO ON
8630  040246  104002                       ERROR   .2              ;MMU ERROR
8631                                                                ;NO GO TO ERROR
8632  040250  005037  177572      300$:   CLR     @#177572                ;TURN MMU OFF
8633  040254  023727  000244  173621      CMP     @#244,#173621           ;IS TEST LOCATION CORRECT
8634  040262  001401                       BEQ     301$                    ;YES GO ON
8635  040264  104002                       ERROR   .2              ;MMU ERROR
8636                                                                ;NO GO TO ERROR
8637  040266  020127  000244      301$:   CMP     R1,#244                 ;IS R1 CORRECT
8638  040272  001401                       BEQ     302$                    ;YES GO ON
8639  040274  104002                       ERROR   .2              ;MMU ERROR
```

```
8640
8641  040276  005037  177776          302$:   CLR     @#177776                ;NO GO TO ERROR
8642  040302  022627  173621                  CMP     (SP)+,#173621           ;SET PSW TO KERNEL MODE
8643  040306  001401                         BEQ     303$                    ;IS STACK CORRECT
8644  040310  104002                         ERROR   .2                      ;YES GO ON
8645                                                                          ;MMU ERROR
8646  040312  021627  177777          303$:   CMP     (SP),#177777            ;NO GO TO ERROR
8647  040316  001401                         BEQ     304$                    ;IS STACK CORRECT
8648  040320  104002                         ERROR   .2                      ;YES GO ON
8649                                                                          ;MMU ERROR
8650  040322  005003                 304$:   CLR     R3                      ;NO GO TO ERROR
8651  040324  005237  177572                  INC     @#177572                ;SETUP SOURCE FOR NEXT TEST
8652  040330  006503                         MFPI    R3                      ;TURN MMU ON
8653  040332  022737  000004  177776        CMP     #4,@#177776             ; TEST INSTRUCTION
8654  040340  001401                         BEQ     6$                      ;IS PSW CORRECT
8655  040342  104002                         ERROR   .2                      ;YES GO ON
8656                                                                          ;MMU ERROR
8657  040344  005037  177572          6$:     CLR     @#177572                ;NO GO TO ERROR
8658  040350  020327  000000                  CMP     R3,#0                   ;TURN MMU OFF
8659  040354  001401                         BEQ     7$                      ;IS R3 CORRECT
8660  040356  104002                         ERROR   .2                      ;YES GO ON
8661                                                                          ;MMU ERROR
8662  040360  022627  000000          7$:     CMP     (SP)+,#0                ;NO GO TO ERROR
8663  040364  001401                         BEQ     8$                      ;IS STACK CORRECT
8664  040366  104002                         ERROR   .2                      ;YES GO ON
8665                                                                          ;MMU ERROR
8666  040370  022627  177777          8$:     CMP     (SP)+,#177777           ;NO GO TO ERROR
8667  040374  001401                         BEQ     9$                      ;IS STACK CORRECT
8668  040376  104002                         ERROR   .2                      ;YES GO ON
8669                                                                          ;MMU ERROR
8670  040400  012637  000244          9$:     MOV     (SP)+,@#244             ;NO GO TO ERROR
8671                                                                          ;RESTORE TEST LOCATION
8672                                  ;
8673  040404                          TSMM6B:
8674                                  ;       TEST MFPD (MOVE FROM PREVIOUS DATA SPACE)
8675  040404  005037  177572                  CLR     @#177572                ;MMU OFF
8676  040410  005037  002776                  CLR     @#FLAG                  ;CLEAR MMU ABORT FLAG
8677  040414  012737  140000  177776        MOV     #140000,@#177776        ;POINT TO USER SPACE
8678  040422  010637  003002                  MOV     R6,@#SAVUSE             ;SAVE USER SP
8679  040426  012737  040000  177776        MOV     #40000,@#177776         ;POINT TO SUPERVISOR SPACE
8680  040434  010637  003000                  MOV     R6,@#SAVSUP             ;SAVE SUPERVISOR SP
8681  040440  012737  030000  177776        MOV     #30000,@#177776         ;SETUP PSW
8682  040446  004767  072462                  JSR     PC,MMU                  ;INIT MMU
8683  040452  012737  000027  172516        MOV     #27,@#172516            ;SETUP MMR3
8684  040460  013746  000244                  MOV     @#244,-(SP)             ;SAVE DATA AT TEST LOCATION
8685  040464  012746  177777                  MOV     #177777,-(SP)           ;PUT KNOWN DATA ON TOP OF STACK
8686  040470  012737  157002  000244        MOV     #157002,@#244           ;SETUP DATA AT TEST LOCATION
8687  040476  012767  077400  137074        MOV     #77400,UIPDR0           ;SETUP UIPDR0 TO ABORT
8688  040504  012703  000244                  MOV     #244,R3                 ;SETUP POINTER TO TEST LOCATION
8689  040510  005237  177572                  INC     @#177572                ;TURN MMU ON
8690  040514  106523                         MFPD    (R3)+                   ; TEST INSTRUCTION
8691  040516  022737  030010  177776        CMP     #30010,@#177776         ;IS PSW CORRECT
8692  040524  001401                         BEQ     1$                      ;YES GO ON
8693  040526  104002                         ERROR   .2              ;MMU ERROR
8694                                                                  ;NO GO TO ERROR
8695  040530  005037  177572          1$:     CLR     @#177572                ;TURN MMU OFF
```

```
8696  040534  012737  140000  177776        MOV     #140000,@#177776        ;POINT TO USER SPACE
8697  040542  020637  003002                CMP     R6,@#SAVUSE             ;IS USER SP CORRECT
8698  040546  001401                        BEQ     100$                    ;YES GO ON
8699  040550  104002                        ERROR   .2              ;MMU ERROR
8700                                                                  ;NO GO TO ERROR
8701  040552  012737  040000  177776  100$: MOV     #40000,@#177776         ;POINT TO SUPERVISOR SPACE
8702  040560  020637  003000                CMP     R6,@#SAVSUP             ;IS SUPERVISOR SP CORRECT
8703  040564  001401                        BEQ     200$                    ;YES GO ON
8704  040566  104002                        ERROR   .2              ;MMU ERROR
8705                                                                  ;NO GO TO ERROR
8706  040570  023727  000244  157002  200$: CMP     @#244,#157002           ;IS TEST DATA OK
8707  040576  001401                        BEQ     2$                      ;YES GO ON
8708  040600  104002                        ERROR   .2              ;MMU ERROR
8709                                                                  ;NO GO TO ERROR
8710  040602  020327  000246          2$:   CMP     R3,#246                 ;IS R3 CORRECT
8711  040606  001401                        BEQ     3$                      ;YES GO ON
8712  040610  104002                        ERROR   .2              ;MMU ERROR
8713                                                                  ;NO GO TO ERROR
8714  040612  005037  177776          3$:   CLR     @#177776                ;SET PSW TO KERNEL MODE
8715  040616  022627  157002                CMP     (SP)+,#157002           ;IS KERNEL STACK CORRECT
8716  040622  001401                        BEQ     4$                      ;YES GO ON
8717  040624  104002                        ERROR   .2              ;MMU ERROR
8718                                                                  ;NO GO TO ERROR
8719  040626  021627  177777          4$:   CMP     (SP),#177777            ;IS STACK CORRECT
8720  040632  001401                        BEQ     5$                      ;YES GO ON
8721  040634  104002                        ERROR   .2              ;MMU ERROR
8722                                                                  ;NO GO TO ERROR
8723  040636  012737  030017  177776  5$:   MOV     #30017,@#177776         ;SETUP PSW
8724  040644  012737  103456  000244        MOV     #103456,@#244           ;SETUP TEST LOCATION
8725  040652  012701  000244                MOV     #244,R1                 ;SETUP R1
8726  040656  005237  177572                INC     @#177572                ;TURN MMU ON
8727  040662  106511                        MFPD    (R1)                    ;TEST INSTRUCTION
8728  040664  022737  030011  177776        CMP     #30011,@#177776         ;IS PSW CORRECT
8729  040672  001401                        BEQ     300$                    ;YES GO ON
8730  040674  104002                        ERROR   .2              ;MMU ERROR
8731                                                                  ;NO GO TO ERROR
8732  040676  005037  177572        300$:   CLR     @#177572                ;TURN MMU OFF
8733  040702  023727  000244  103456        CMP     @#244,#103456           ;IS TEST LOCATION CORRECT
8734  040710  001401                        BEQ     301$                    ;YES GO ON
8735  040712  104002                        ERROR   .2              ;MMU ERROR
8736                                                                  ;NO GO TO ERROR
8737  040714  020127  000244        301$:   CMP     R1,#244                 ;IS R1 CORRECT
8738  040720  001401                        BEQ     302$                    ;YES GO ON
8739  040722  104002                        ERROR   .2              ;MMU ERROR
8740                                                                  ;NO GO TO ERROR
8741  040724  005037  177776        302$:   CLR     @#177776                ;SET PSW TO KERNEL MODE
8742  040730  022627  103456                CMP     (SP)+,#103456           ;IS STACK CORRECT
8743  040734  001401                        BEQ     303$                    ;YES GO ON
8744  040736  104002                        ERROR   .2              ;MMU ERROR
8745                                                                  ;NO GO TO ERROR
8746  040740  021627  177777        303$:   CMP     (SP),#177777            ;IS STACK CORRECT
8747  040744  001401                        BEQ     304$                    ;YES GO ON
8748  040746  104002                        ERROR   .2              ;MMU ERROR
8749                                                                  ;NO GO TO ERROR
8750  040750  012737  030017  177776  304$:   MOV     #30017,@#177776         ;SETUP PSW
8751  040756  012737  113672  000244        MOV     #113672,@#244           ;SETUP TEST LOCATION
```

```
8752  040764  012701  000246              MOV     #246,R1                 ;SETUP R1
8753  040770  005237  177572              INC     @#177572                ;TURN MMU ON
8754  040774  106541                      MFPD    -(R1)                   ;TEST INSTRUCTION
8755  040776  022737  030011  177776      CMP     #30011,@#177776         ;IS PSW CORRECT
8756  041004  001401                      BEQ     400$                    ;YES GO ON
8757  041006  104002                      ERROR   .2                  ;MMU ERROR
8758                                                                  ;NO GO TO ERROR
8759  041010  005037  177572      400$:   CLR     @#177572                ;TURN MMU OFF
8760  041014  023727  000244  113672      CMP     @#244,#113672           ;IS TEST LOCATION CORRECT
8761  041022  001401                      BEQ     401$                    ;YES GO ON
8762  041024  104002                      ERROR   .2                  ;MMU ERROR
8763                                                                  ;NO GO TO ERROR
8764  041026  020127  000244      401$:   CMP     R1,#244                 ;IS R1 CORRECT
8765  041032  001401                      BEQ     402$                    ;YES GO ON
8766  041034  104002                      ERROR   .2                  ;MMU ERROR
8767                                                                  ;NO GO TO ERROR
8768  041036  005037  177776      402$:   CLR     @#177776                ;SET PSW TO KERNEL MODE
8769  041042  022627  113672              CMP     (SP)+,#113672           ;IS STACK CORRECT
8770  041046  001401                      BEQ     403$                    ;YES GO ON
8771  041050  104002                      ERROR   .2                  ;MMU ERROR
8772                                                                  ;NO GO TO ERROR
8773  041052  021627  177777      403$:   CMP     (SP),#177777            ;IS STACK CORRECT
8774  041056  001401                      BEQ     404$                    ;YES GO ON
8775  041060  104002                      ERROR   .2                  ;MMU ERROR
8776                                                                  ;NO GO TO ERROR
8777  041062  005003              404$:   CLR     R3                      ;SETUP SOURCE FOR NEXT TEST
8778  041064  005237  177572              INC     @#177572                ;TURN MMU ON
8779  041070  106503                      MFPD    R3                      ; TEST INSTRUCTION
8780  041072  022737  000004  177776      CMP     #4,@#177776             ;IS PSW CORRECT
8781  041100  001401                      BEQ     6$                      ;YES GO ON
8782  041102  104002                      ERROR   .2                  ;MMU ERROR
8783                                                                  ;NO GO TO ERROR
8784  041104  005037  177572      6$:     CLR     @#177572                ;TURN MMU OFF
8785  041110  020327  000000              CMP     R3,#0                   ;IS R3 CORRECT
8786  041114  001401                      BEQ     7$                      ;YES GO ON
8787  041116  104002                      ERROR   .2                  ;MMU ERROR
8788                                                                  ;NO GO TO ERROR
8789  041120  022627  000000      7$:     CMP     (SP)+,#0                ;IS STACK CORRECT
8790  041124  001401                      BEQ     8$                      ;YES GO ON
8791  041126  104002                      ERROR   .2                  ;MMU ERROR
8792                                                                  ;NO GO TO ERROR
8793  041130  022627  177777      8$:     CMP     (SP)+,#177777           ;IS STACK CORRECT
8794  041134  001401                      BEQ     9$                      ;YES GO ON
8795  041136  104002                      ERROR   .2                  ;MMU ERROR
8796                                                                  ;NO GO TO ERROR
8797  041140  012637  000244      9$:     MOV     (SP)+,@#244             ;RESTORE TEST LOCATION
8798
8799                                      ;
8800  041144                      TSMM6C:
8801                                      ;       TEST MTPI (MOVE TO PREVIOUS INSTRUCTION SPACE)
8802  041144  005037  177572              CLR     @#177572                ;MMU OFF
8803  041150  005037  002776              CLR     @#FLAG                  ;CLEAR MMU ABORT FLAG
8804  041154  012737  140000  177776      MOV     #140000,@#177776        ;POINT TO USER SPACE
8805  041162  010637  003002              MOV     R6,@#SAVUSE             ;SAVE USER SP
8806  041166  012737  040000  177776      MOV     #40000,@#177776         ;POINT TO SUPERVISOR SPACE
8807  041174  010637  003000              MOV     R6,@#SAVSUP             ;SAVE SUPERVISOR SP
```

```
8808  041200  012737  030000  177776          MOV     #30000,@#177776         ;SETUP PSW
8809  041206  004767  071722                  JSR     PC,MMU                  ;INIT MMU
8810  041212  012737  000027  172516          MOV     #27,@#172516            ;SETUP MMR3
8811  041220  013746  000244                  MOV     @#244, (SP)             ;SAVE DATA AT TEST LOCATION
8812  041224  012746  177777                  MOV     #177777,-(SP)           ;PUT KNOWN DATA ON STACK
8813  041230  012746  120413                  MOV     #120413,-(SP)           ;PUT TEST DATA ON STACK
8814  041234  012737  177777  000244          MOV     #177777,@#244           ;PUT KNOWN DATA AT TEST LOCATION
8815  041242  012767  077400  136350          MOV     #77400,UDPDRO           ;SETUP UDPDRO TO ABORT
8816  041250  012703  000244                  MOV     #244,R3                 ;SETUP POINTER TO TEST LOCATION
8817  041254  005237  177572                  INC     @#177572                ;TURN MMU ON
8818  041260  006623                          MTPI    (R3)+                   ; TEST INSTRUCTION
8819  041262  022737  030010  177776          CMP     #30010,@#177776         ;IS PSW CORRECT
8820  041270  001401                          BEQ     1$                      ;YES GO ON
8821  041272  104002                          ERROR   .2              ;MMU ERROR
8822                                                                  ;NO GO TO ERROR
8823  041274  005037  177572          1$:     CLR     @#177572                ;TURN MMU OFF
8824  041300  012737  140000  177776          MOV     #140000,@#177776        ;POINT TO USER SPACE
8825  041306  020637  003002                  CMP     R6,@#SAVUSE             ;IS USER SP CORRECT
8826  041312  001401                          BEQ     100$                    ;YES GO ON
8827  041314  104002                          ERROR   .2              ;MMU ERROR
8828                                                                  ;NO GO TO ERROR
8829  041316  012737  040000  177776  100$:   MOV     #40000,@#177776         ;POINT TO SUPERVISOR SPACE
8830  041324  020637  003000                  CMP     R6,@#SAVSUP             ;IS SUPERVISOR SP CORRECT
8831  041330  001401                          BEQ     200$                    ;YES GO ON
8832  041332  104002                          ERROR   .2              ;MMU ERROR
8833                                                                  ;NO GO TO ERROR
8834  041334  023727  000244  120413  200$:   CMP     @#244,#120413           ;IS TEST LOCATION CORRECT
8835  041342  001401                          BEQ     2$                      ;YES GO ON
8836  041344  104002                          ERROR   .2              ;MMU ERROR
8837                                                                  ;NO GO TO ERROR
8838  041346  020327  000246          2$:     CMP     R3,#246                 ;IS R3 CORRECT
8839  041352  001401                          BEQ     3$                      ;YES GO ON
8840  041354  104002                          ERROR   .2              ;MMU ERROR
8841                                                                  ;NO GO TO ERROR
8842  041356  005037  177776          3$:     CLR     @#177776                ;SET PSW TO KERNEL MODE
8843  041362  021627  177777                  CMP     (SP),#177777            ;IS KERNEL STACK CORRECT
8844  041366  001401                          BEQ     4$                      ;YES GO ON
8845  041370  104002                          ERROR   .2              ;MMU ERROR
8846                                                                  ;NO GO TO ERROR
8847  041372  012737  030017  177776  4$:     MOV     #30017,@#177776         ;SETUP PSW
8848  041400  012746  145121                  MOV     #145121,-(SP)           ;SETUP TEST DATA
8849  041404  012701  000244                  MOV     #244,R1                 ;SETUP R1
8850  041410  005237  177572                  INC     @#177572                ;TURN MMU ON
8851  041414  006611                          MTPI    (R1)                    ;TEST INSTRUCTION
8852  041416  022737  030011  177776          CMP     #30011,@#177776         ;IS PSW CORRECT
8853  041424  001401                          BEQ     300$                    ;YES GO ON
8854  041426  104002                          ERROR   .2              ;MMU ERROR
8855                                                                  ;NO GO TO ERROR
8856  041430  005037  177572          300$:   CLR     @#177572                ;TURN MMU OFF
8857  041434  023727  000244  145121          CMP     @#244,#145121           ;IS TEST LOCATION CORRECT
8858  041442  001401                          BEQ     301$                    ;YES GO ON
8859  041444  104002                          ERROR   .2              ;MMU ERROR
8860                                                                  ;NO GO TO ERROR
8861  041446  020127  000244          301$:   CMP     R1,#244                 ;IS R1 CORRECT
8862  041452  001401                          BEQ     302$                    ;YES GO ON
8863  041454  104002                          ERROR   .2              ;MMU ERROR
```

```
8864
8865  041456  005037  177776          302$:   CLR     @#177776                ;NO GO TO ERROR
8866  041462  021627  177777                  CMP     (SP),#177777            ;SET PSW TO KERNEL MODE
8867  041466  001401                          BEQ     304$                    ;IS STACK CORRECT
8868  041470  104002                          ERROR   .2                      ;YES GO ON
8869                                                                          ;MMU ERROR
8870  041472  012737  030017  177776  304$:   MOV     #030017,@#177776        ;NO GO TO ERROR
8871  041500  012746  122347                  MOV     #122347,-(SP)           ;SETUP PSW
8872  041504  012701  000246                  MOV     #246,R1                 ;SETUP TEST DATA
8873  041510  005237  177572                  INC     @#177572                ;SETUP R1
8874  041514  006641                          MTPI    -(R1)                   ;TURN MMU ON
8875  041516  022737  030011  177776          CMP     #30011,@#177776         ;TEST INSTRUCTION
8876  041524  001401                          BEQ     400$                    ;IS PSW CORRECT
8877  041526  104002                          ERROR   .2                      ;YES GO ON
8878                                                                          ;MMU ERROR
8879  041530  005037  177572          400$:   CLR     @#177572                ;NO GO TO ERROR
8880  041534  023727  000244  122347          CMP     @#244,#122347           ;TURN MMU OFF
8881  041542  001401                          BEQ     401$                    ;IS TEST LOCATION CORRECT
8882  041544  104002                          ERROR   .2                      ;YES GO ON
8883                                                                          ;MMU ERROR
8884  041546  020127  000244          401$:   CMP     R1,#244                 ;NO GO TO ERROR
8885  041552  001401                          BEQ     402$                    ;IS R1 CORRECT
8886  041554  104002                          ERROR   .2                      ;YES GO ON
8887                                                                          ;MMU ERROR
8888  041556  005037  177776          402$:   CLR     @#177776                ;NO GO TO ERROR
8889  041562  021627  177777                  CMP     (SP),#177777            ;SET PSW TO KERNEL MODE
8890  041566  001401                          BEQ     404$                    ;IS STACK CORRECT
8891  041570  104002                          ERROR   .2                      ;YES GO ON
8892                                                                          ;MMU ERROR
8893  041572  005046          404$:   CLR     -(SP)                   ;NO GO TO ERROR
8894  041574  005237  177572                  INC     @#177572                ;SETUP STACK FOR NEXT TEST
8895  041600  006603                          MTPI    R3                      ;TURN MMU ON
8896  041602  022737  000004  177776          CMP     #4,@#177776             ; TEST INSTRUCTION
8897  041610  001401                          BEQ     5$                      ;IS PSW CORRECT
8898  041612  104002                          ERROR   .2                      ;YES GO ON
8899                                                                          ;MMU ERROR
8900  041614  005037  177572          5$:     CLR     @#177572                ;NO GO TO ERROR
8901  041620  020327  000000                  CMP     R3,#0                   ;TURN MMU OFF
8902  041624  001401                          BEQ     6$                      ;IS R3 CORRECT
8903  041626  104002                          ERROR   .2                      ;YES GO ON
8904                                                                          ;MMU ERROR
8905  041630  022627  177777          6$:     CMP     (SP)+,#177777           ;NO GO TO ERROR
8906  041634  001401                          BEQ     7$                      ;IS STACK CORRECT
8907  041636  104002                          ERROR   .2                      ;YES GO ON
8908                                                                          ;MMU ERROR
8909  041640  012637  000244          7$:     MOV     (SP)+,@#244             ;NO GO TO ERROR
8910                                                                          ;RESTORE TEST LOCATION
8911                                  ;
8912  041644                          ;SMM6D:
8913                                  ;       TEST MTPD (MOVE TO PREVIOUS DATA SPACE)
8914  041644  005037  177572                  CLR     @#177572                ;MMU OFF
8915  041650  005037  002776                  CLR     @#FLAG                  ;CLEAR MMU ABORT FLAG
8916  041654  012737  140000  177776          MOV     #140000,@#177776        ;POINT TO USER SPACE
8917  041662  010637  003002                  MOV     R6,@#SAVUSE             ;SAVE USER SP
8918  041666  012737  040000  177776          MOV     #40000,@#177776         ;POINT TO SUPERVISOR SPACE
8919  041674  010637  003000                  MOV     R6,@#SAVSUP             ;SAVE SUPERVISOR SP
```

```
8920  041700  012737  030000  177776          MOV     #30000,@#177776         ;SETUP PSW
8921  041706  004767  071222                  JSR     PC,MMU                  ;INIT MMU
8922  041712  012737  000027  172516          MOV     #27,@#172516            ;SETUP MMR3
8923  041720  013746  000244                  MOV     @#244, (SP)             ;SAVE DATA AT TEST LOCATION
8924  041724  012746  177777                  MOV     #177777,-(SP)           ;PUT KNOWN DATA ON STACK
8925  041730  012746  100004                  MOV     #100004,-(SP)           ;PUT TEST DATA ON STACK
8926  041734  012737  177777  000244          MOV     #177777,@#244           ;PUT KNOWN DATA AT TEST LOCATION
8927  041742  012767  077400  135630          MOV     #77400,UIPDR0           ;SETUP UIPDR0 TO ABORT
8928  041750  012703  000244                  MOV     #244,R3                 ;SETUP POINTER TO TEST LOCATION
8929  041754  005237  177572                  INC     @#177572                ;TURN MMU ON
8930  041760  106623                          MTPD    (R3)+                   ; TEST INSTRUCTION
8931  041762  022737  030010  177776          CMP     #30010,@#177776         ;IS PSW CORRECT
8932  041770  001401                          BEQ     1$                      ;YES GO ON
8933  041772  104002                          ERROR   .+2             ;MMU ERROR
8934                                                                  ;NO GO TO ERROR
8935  041774  005037  177572          1$:     CLR     @#177572                ;TURN MMU OFF
8936  042000  012737  140000  177776          MOV     #140000,@#177776        ;POINT TO USER SPACE
8937  042006  020637  003002                  CMP     R6,@#SAVUSE             ;IS USER SP CORRECT
8938  042012  001401                          BEQ     100$                    ;YES GO ON
8939  042014  104002                          ERROR   .+2             ;MMU ERROR
8940                                                                  ;NO GO TO ERROR
8941  042016  012737  040000  177776  100$:   MOV     #40000,@#177776         ;POINT TO SUPERVISOR SPACE
8942  042024  020637  003000                  CMP     R6,@#SAVSUP             ;IS SUPERVISOR SP CORRECT
8943  042030  001401                          BEQ     200$                    ;YES GO ON
8944  042032  104002                          ERROR   .+2             ;MMU ERROR
8945                                                                  ;NO GO TO ERROR
8946  042034  023727  000244  100004  200$:   CMP     @#244,#100004           ;IS TEST LOCATION CORRECT
8947  042042  001401                          BEQ     2$                      ;YES GO ON
8948  042044  104002                          ERROR   .+2             ;MMU ERROR
8949                                                                  ;NO GO TO ERROR
8950  042046  020327  000246          2$:     CMP     R3,#246                 ;IS R3 CORRECT
8951  042052  001401                          BEQ     3$                      ;YES GO ON
8952  042054  104002                          ERROR   .+2             ;MMU ERROR
8953                                                                  ;NO GO TO ERROR
8954  042056  005037  177776          3$:     CLR     @#177776                ;SET PSW TO KERNEL MODE
8955  042062  021627  177777                  CMP     (SP),#177777            ;IS KERNEL STACK CORRECT
8956  042066  001401                          BEQ     4$                      ;YES GO ON
8957  042070  104002                          ERROR   .+2             ;MMU ERROR
8958                                                                  ;NO GO TO ERROR
8959  042072  012737  030017  177776  4$:     MOV     #30017,@#177776         ;SETUP PSW
8960  042100  012746  100737                  MOV     #100737,-(SP)           ;SETUP TEST DATA
8961  042104  012701  000244                  MOV     #244,R1                 ;SETUP R1
8962  042110  005237  177572                  INC     @#177572                ;TURN MMU ON
8963  042114  106611                          MTPD    (R1)                    ;TEST INSTRUCTION
8964  042116  022737  030011  177776          CMP     #30011,@#177776         ;IS PSW CORRECT
8965  042124  001401                          BEQ     300$                    ;YES GO ON
8966  042126  104002                          ERROR   .+2             ;MMU ERROR
8967                                                                  ;NO GO TO ERROR
8968  042130  005037  177572          300$:   CLR     @#177572                ;TURN MMU OFF
8969  042134  023727  000244  100737          CMP     @#244,#100737           ;IS TEST LOCATION CORRECT
8970  042142  001401                          BEQ     301$                    ;YES GO ON
8971  042144  104002                          ERROR   .+2             ;MMU ERROR
8972                                                                  ;NO GO TO ERROR
8973  042146  020127  000244          301$:   CMP     R1,#244                 ;IS R1 CORRECT
8974  042152  001401                          BEQ     302$                    ;YES GO ON
8975  042154  104002                          ERROR   .+2             ;MMU ERROR
```

```
8976
8977  042156  005037  177776          302$:   CLR     @#177776                ;NO GO TO ERROR
8978  042162  021627  177777                  CMP     (SP),#177777            ;SET PSW TO KERNEL MODE
8979  042166  001401                          BEQ     304$                    ;IS STACK CORRECT
8980  042170  104002                          ERROR   .2                      ;YES GO ON
8981                                                                          ;MMU ERROR
8982  042172  012737  030017  177776  304$:   MOV     #30017,@#177776         ;NO GO TO ERROR
8983  042200  012746  156711                  MOV     #156711,-(SP)           ;SETUP PSW
8984  042204  012701  000246                  MOV     #246,R1                 ;SETUP TEST DATA
8985  042210  005237  177572                  INC     @#177572                ;SETUP R1
8986  042214  106641                          MTPD    -(R1)                   ;TURN MMU ON
8987  042216  022737  030011  177776          CMP     #30011,@#177776         ;TEST INSTRUCTION
8988  042224  001401                          BEQ     400$                    ;IS PSW CORRECT
8989  042226  104002                          ERROR   .2                      ;YES GO ON
8990                                                                          ;MMU ERROR
8991  042230  005037  177572          400$:   CLR     @#177572                ;NO GO TO ERROR
8992  042234  023727  000244  156711          CMP     @#244,#156711           ;TURN MMU OFF
8993  042242  001401                          BEQ     401$                    ;IS TEST LOCATION CORRECT
8994  042244  104002                          ERROR   .2                      ;YES GO ON
8995                                                                          ;MMU ERROR
8996  042246  020127  000244          401$:   CMP     R1,#244                 ;NO GO TO ERROR
8997  042252  001401                          BEQ     402$                    ;IS R1 CORRECT
8998  042254  104002                          ERROR   .2                      ;YES GO ON
8999                                                                          ;MMU ERROR
9000  042256  005037  177776          402$:   CLR     @#177776                ;NO GO TO ERROR
9001  042262  021627  177777                  CMP     (SP),#177777            ;SET PSW TO KERNEL MODE
9002  042266  001401                          BEQ     404$                    ;IS STACK CORRECT
9003  042270  104002                          ERROR   .2                      ;YES GO ON
9004                                                                          ;MMU ERROR
9005  042272  005046          404$:   CLR     -(SP)                   ;NO GO TO ERROR
9006  042274  005237  177572                  INC     @#177572                ;SETUP STACK FOR NEXT TEST
9007  042300  106603                          MTPD    R3                      ;TURN MMU ON
9008  042302  022737  000004  177776          CMP     #4,@#177776             ; TEST INSTRUCTION
9009  042310  001401                          BEQ     5$                      ;IS PSW CORRECT
9010  042312  104002                          ERROR   .2                      ;YES GO ON
9011                                                                          ;MMU ERROR
9012  042314  005037  177572          5$:     CLR     @#177572                ;NO GO TO ERROR
9013  042320  020327  000000                  CMP     R3,#0                   ;TURN MMU OFF
9014  042324  001401                          BEQ     6$                      ;IS R3 CORRECT
9015  042326  104002                          ERROR   .2                      ;YES GO ON
9016                                                                          ;MMU ERROR
9017  042330  022627  177777          6$:     CMP     (SP)+,#177777           ;NO GO TO ERROR
9018  042334  001401                          BEQ     7$                      ;IS STACK CORRECT
9019  042336  104002                          ERROR   .2                      ;YES GO ON
9020                                                                          ;MMU ERROR
9021  042340  012637  000244          7$:     MOV     (SP)+,@#244             ;NO GO TO ERROR
9022                                                                          ;RESTORE TEST LOCATION
9023                                  ;
9024  042344                          TSMMU7:
9025                                  ;       TEST NON-RESIDENT ABORT
9026  042344  005037  177572                  CLR     @#177572                ;MMU OFF
9027  042350  005067  140422                  CLR     FLAG                    ;CLEAR MMU ABORT FLAG
9028  042354  013746  000214                  MOV     @#214,-(SP)             ;SAVE DATA AT TEST LOCATIONS
9029  042360  013746  000216                  MOV     @#216,-(SP)             ;
9030  042364  005067  140414                  CLR     SAVMR0                  ;CLEAR STATUS REGS SAVE AREAS
9031  042370  005067  140412                  CLR     SAVMR1                  ;
```

J13


```
 9032  042374  005067  140410                  CLR     SAVMR2                  ;
 9033  042400  004767  070530                  JSR     PC,MMU                  ;INIT MMU
 9034  042404  012737  030000  177776          MOV     #30000,@#177776         ;SETUP PSW
 9035  042412  012702  000200                  MOV     #200,R2                 ;
 9036  042416  012737  077400  177600          MOV     #77400,@#177600         ;SETUP FOR AN ABORT
 9037  042424  004767  000164                  JSR     PC,TS7                  ;CAUSE AN ABORT TO OCCUR AND
 9038                                                                          ;THEN CHECK IF ABORT FLAG REGISTERED
 9039                                                                          ;THIS EVENT AND CHECK IF STATUS REGS
 9040                                                                          ;CONTAINED EXPECTED VALUES.
 9041                                                                          ;IF NO ABORT OCCURRED THEN GO TO ERROR
 9042                                                                          ;OTHERWISE CONTINUE.
 9043  042430  012737  077404  177600          MOV     #77404,@#177600         ;SETUP FOR AN ABORT
 9044  042436  004767  000152                  JSR     PC,TS7                  ;CAUSE AN ABORT TO OCCUR AND
 9045                                                                          ;THEN CHECK IF ABORT FLAG REGISTERED
 9046                                                                          ;THIS EVENT AND CHECK IF STATUS REGS
 9047                                                                          ;CONTAINED EXPECTED VALUES.
 9048                                                                          ;IF NO ABORT OCCURRED THEN GO TO ERROR
 9049                                                                          ;OTHERWISE CONTINUE.
 9050  042442  012701  000220                  MOV     #220,R1                 ;
 9051  042446  004767  070462                  JSR     PC,MMU                  ;INIT MMU
 9052  042452  005003                          CLR     R3                      ;SETUP MMR1 EXPECTED DATA
 9053  042454  012767  000001  140314          MOV     #1,FLAG                 ;SETUP FLAG FOR AN ABORT
 9054  042462  012737  000001  177572          MOV     #1,@#177572             ;TURN MMU ON
 9055  042470  012737  100000  177776          MOV     #100000,@#177776        ;SETUP PSW FOR AN ABORT (ILLEGAL MODE)
 9056  042476  012241                          MOV     (R2)+,-(R1)             ;CAUSE AN ABORT
 9057  042500  004767  000220                  JSR     PC,TSM7                 ;CHECK IF AN ABORT OCCURRED BY
 9058                                                                          ;CHECKING ABORT FLAG AND STATUS REGS
 9059                                                                          ;IF NO ABORT OCCURRED THEN GO TO ERROR
 9060                                                                          ;OTHERWISE CONTINUE.
 9061  042504  005067  140274                  CLR     SAVMR0                  ;CLEAR STATUS REGS SAVE AREAS
 9062  042510  005067  140272                  CLR     SAVMR1                  ;
 9063  042514  005067  140270                  CLR     SAVMR2                  ;
 9064  042520  012703  000022                  MOV     #22,R3                  ;SETUP MMR1 EXPECTED DATA
 9065  042524  012767  000001  140244          MOV     #1,FLAG                 ;SETUP FLAG FOR AN ABORT
 9066  042532  012737  000001  177572          MOV     #1,@#177572             ;TURN MMU ON
 9067  042540  012737  020000  177776          MOV     #20000,@#177776         ;SETUP PSW FOR AN ABORT (ILLEGAL MODE)
 9068  042546  006522                          MFPI    (R2)+                   ;CAUSE AN ABORT
 9069  042550  004767  000150                  JSR     PC,TSM7                 ;CHECK IF AN ABORT OCCURRED BY
 9070                                                                          ;CHECKING ABORT FLAG AND STATUS REGS
 9071                                                                          ;IF NO ABORT OCCURRED THEN GO TO ERROR
 9072                                                                          ;OTHERWISE CONTINUE.
 9073  042554  012737  030000  177776          MOV     #30000,@#177776         ;SETUP PSW
 9074  042562  012737  077400  177600          MOV     #77400,@#177600         ;SETUP FOR AN ABORT
 9075  042570  005037  177572                  CLR     @#177572                ;MMU OFF
 9076  042574  006522                          MFPI    (R2)+                   ;TRY TO CAUSE AN ABORT
 9077  042576  012603                          MOV     (SP)+,R3                ;POP THE STACK
 9078  042600  012637  000216                  MOV     (SP)+,@#216             ;RESTORE DATA AT TEST LOCATIONS
 9079  042604  012637  000214                  MOV     (SP)+,@#214             ;
 9080
 9081  042610  000167  000154                  JMP     TS7FIN
 9082                                  ;
 9083                                  ;ROUTINE TO CAUSE AND CHECK  NONRESIDENT ABORTS
 9084                                  ;
 9085  042614  012767  000001  140154  TS7:    MOV     #1,FLAG                 ;SETUP FOR AN ABORT
 9086  042622  012737  000001  177572          MOV     #1,@#177572             ;TURN MMU ON
 9087  042630  010701                          MOV     R7,R1                   ;SAVE PC
```

```
 9088  042632  006522                          MFPI    (R2)+                   ;CAUSE AN ABORT
 9089  042634  022767  000000  140134          CMP     #0,FLAG                 ;DID AN ABORT OCCUR
 9090  042642  001401                          BEQ     OK7                     ;IF YES GO ON
 9091  042644  104002                          ERROR   +2              ;MMU ERROR
 9092                                          ;IF NO GO TO ERROR
 9093  042646  105067  140132          OK7:    CLRB    SAVMR0                  ;SETUP EXPECTED DATA
 9094  042652  022767  100000  140124          CMP     #100000,SAVMR0          ; TEST MMR0 FOR EXPECTED VALUE
 9095  042660  001401                          BEQ     OKA7                    ;IF OK THEN CONTINUE
 9096  042662  104002                          ERROR   +2              ;MMU ERROR
 9097                                          ;NOT OK THEN GO TO ERROR
 9098  042664  026727  140116  000022  OKA7:   CMP     SAVMR1,#22              ; TEST MMR1 FOR EXPECTED VALUE
 9099  042672  001401                          BEQ     OKAY7                   ;IF OK THEN CONTINUE
 9100  042674  104002                          ERROR   +2              ;MMU ERROR
 9101                                          ;NOT OK THEN GO TO ERROR
 9102  042676  026701  140106          OKAY7:  CMP     SAVMR2,R1               ; TEST MMR2 FOR EXPECTED VALUE
 9103  042702  001401                          BEQ     OKAY7A                  ;IF OK THEN CONTINUE
 9104  042704  104002                          ERROR   +2              ;MMU ERROR
 9105                                          ;NOT OK THEN GO TO ERROR
 9106  042706  005067  140072          OKAY7A: CLR     SAVMR0                  ;CLEAR STATUS REGS SAVE AREAS
 9107  042712  005067  140070                  CLR     SAVMR1                  ;
 9108  042716  005067  140066                  CLR     SAVMR2                  ;
 9109  042722  000207                          RTS     PC                      ;RETURN
 9110                                  ;
 9111                                  ;ROUTINE TO CHECK IF A NONRESIDENT ABORT OCCURRED
 9112                                  ;
 9113  042724  022767  000000  140044  TSM7:   CMP     #0,FLAG                 ;DID AN ABORT OCCUR
 9114  042732  001401                          BEQ     TSMA                    ;IF YES GO ON
 9115  042734  104002                          ERROR   +2              ;MMU ERROR
 9116                                          ;IF NO THEN GO TO ERROR
 9117  042736  042737  040377  003004  TSMA:   BIC     #40377,@#SAVMR0         ;SETUP EXPECTED DATA
 9118  042744  022767  100000  140032          CMP     #100000,SAVMR0          ; TEST MMR0 FOR EXPECTED VALUE
 9119  042752  001401                          BEQ     TSMB                    ;IF OK THEN CONTINUE
 9120  042754  104002                          ERROR   +2              ;MMU ERROR
 9121                                          ;IF NO THEN GO TO ERROR
 9122  042756  020367  140024          TSMB:   CMP     R3,SAVMR1               ; TEST MMR1 FOR EXPECTED VALUE
 9123  042762  001401                          BEQ     TSMC                    ;IF OK THEN CONTINUE
 9124  042764  104002                          ERROR   +2              ;MMU ERROR
 9125                                          ;IF NOT OK THEN GO TO ERROR
 9126  042766  000207                  TSMC:   RTS     PC                      ;RETURN
 9127                                  ;
 9128  042770  000240                  TS7FIN: NOP
 9129  042772                          TSMMU8:
 9130                                  ;       TEST READ ONLY ABORTS
 9131  042772  005037  177572                  CLR     @#177572                ;MMU OFF
 9132  042776  005067  137774                  CLR     FLAG                    ;CLEAR MMU ABORT FLAG
 9133  043002  013746  000244                  MOV     @#244,-(SP)             ;SAVE DATA AT TEST LOCATIONS
 9134  043006  013746  000246                  MOV     @#246,-(SP)             ;
 9135  043012  005067  137766                  CLR     SAVMR0                  ;CLEAR STATUS REGS SAVE AREAS
 9136  043016  005067  137764                  CLR     SAVMR1                  ;
 9137  043022  005067  137762                  CLR     SAVMR2                  ;
 9138  043026  004767  070102                  JSR     PC,MMU                  ;INIT MMU
 9139  043032  012737  030000  177776          MOV     #30000,@#177776         ;SETUP PSW
 9140  043040  012702  000244                  MOV     #244,R2                 ;
 9141  043044  012737  077402  177600          MOV     #77402,@#177600         ;SETUP FOR AN ABORT
 9142  043052  012746  000246                  MOV     #246,-(SP)              ;PUSH DATA ONTO THE STACK
 9143  043056  012767  000001  137712          MOV     #1,FLAG                 ;SETUP FLAG FOR AN ABORT
```

```
9144  043064  012737  000001  177572          MOV     #1,@#177572             ;TURN MMU ON
9145  043072  010701                          MOV     R7,R1                   ;SAVE PC
9146  043074  006622                          MTPI    (R2)+                   ;CAUSE ABORT
9147  043076  022767  000000  137672          CMP     #0,FLAG                 ;DID ABORT OCCUR
9148  043104  001401                          BEQ     1$                      ;IF YES THEN GO ON
9149  043106  104002                          ERROR   .2              ;MMU ERROR
9150                                          ;IF NO THEN GO TO ERROR
9151  043110  105067  137670          1$:     CLRB    SAVMR0                  ;SETUP EXPECTED DATA
9152  043114  022767  020000  137662          CMP     #20000,SAVMR0           ; TEST MMR0 FOR EXPECTED VALUE
9153  043122  001401                          BEQ     2$                      ;IF OK THEN CONTINUE
9154  043124  104002                          ERROR   .2              ;MMU ERROR
9155                                          ;OTHERWISE GO TO ERROR
9156  043126  022767  011026  137652  2$:     CMP     #11026,SAVMR1           ; TEST MMR1 FOR EXPECTED VALUE
9157  043134  001401                          BEQ     3$                      ;IF OK THEN CONTINUE
9158  043136  104002                          ERROR   .2              ;MMU ERROR
9159                                          ;OTHERWISE GO TO ERROR
9160  043140  020167  137644          3$:     CMP     R1,SAVMR2               ; TEST MMR2 FOR EXPECTED VALUE
9161  043144  001401                          BEQ     4$                      ;IF OK THEN CONTINUE
9162  043146  104002                          ERROR   .2              ;MMU ERROR
9163                                          ;OTHERWISE GO TO ERROR
9164  043150  012737  030000  177776  4$:     MOV     #30000,@#177776         ;SETUP PSW
9165  043156  012746  000002                  MOV     #2,-(SP)                ;PUSH DATA ONTO STACK
9166  043162  006622                          MTPI    (R2)+                   ;TRY TO CAUSE ABORT
9167  043164  012637  000246                  MOV     (SP)+,@#246             ;RESTORE DATA AT TEST LOCATIONS
9168  043170  012637  000244                  MOV     (SP)+,@#244             ;
9169
9170  043174                          TSMMU9:
9171                                  ;       TEST PAGE LENGTH ERROR ABORTS
9172  043174  005037  177572                  CLR     @#177572                ;MMU OFF
9173  043200  005067  137572                  CLR     FLAG                    ;CLEAR MMU ABORT FLAG
9174  043204  005067  137574                  CLR     SAVMR0                  ;CLEAR STATUS REGS SAVE AREAS
9175  043210  005067  137572                  CLR     SAVMR1                  ;
9176  043214  005067  137570                  CLR     SAVMR2                  ;
9177  043220  012737  030000  177776          MOV     #30000,@#177776         ;SETUP PSW
9178  043226  004767  067702                  JSR     PC,MMU                  ;INIT MMU
9179  043232  012703  043514                  MOV     #PLF0,R3                ;LET R3, R1, AND R2 POINT TO THE
9180  043236  012701  043566                  MOV     #BN0,R1                 ;UPWARD EXPANSION TABLES
9181  043242  012702  043636                  MOV     #ABORT0,R2              ;
9182  043246  012737  000026  172516          MOV     #26,@#172516            ;DISABLE USER DATA SPACE
9183  043254  004767  000050                  JSR     PC,TSM9                 ;TURN MMU ON
9184                                                                          ;DO RELOCATIONS FOR THE DIFFERENT
9185                                                                          ;VALUES OF THE PAGE LENGTH FIELD AND
9186                                                                          ;BLOCK NUMBER.  IF AN ABORT OCCURS
9187                                                                          ;CHECK TO SEE IF IT WAS SUPPOSED TO,
9188                                                                          ;AND IF YES CHECK ABORT FLAG AND
9189                                                                          ;STATUS REGISTERS.
9190  043260  012703  043706                  MOV     #PLF1,R3                ;LET R3, R1, AND R2 POINT TO THE
9191  043264  012701  043756                  MOV     #BN1,R1                 ;DOWNWARD EXPANSION TABLES
9192  043270  012702  044024                  MOV     #ABORT7,R2              ;
9193  043274  004767  000030                  JSR     PC,TSM9                 ;TURN MMU ON
9194                                                                          ;DO RELOCATIONS FOR THE DIFFERENT
9195                                                                          ;VALUES OF THE PAGE LENGTH FIELD AND
9196                                                                          ;BLOCK NUMBER.  IF AN ABORT OCCURS
9197                                                                          ;CHECK TO SEE IF IT WAS SUPPOSED TO,
9198                                                                          ;AND IF YES CHECK ABORT FLAG AND
9199                                                                          ;STATUS REGISTERS.
```

```
9200  043300  005037  177572            CLR     @#177572                ;MMU OFF
9201  043304  012703  043716            MOV     #PLF1+10,R3             ;POINT TO A VALUE WHICH SHOULD CAUSE
9202  043310  012701  043766            MOV     #BN1+10,R1              ;AN ABORT IF MMU IS ON.
9203  043314  011337  177600            MOV     (R3),@#177600           ;SETUP UIPDR0
9204  043320  006521                    MFPI    (R1)+                   ;DO A RELOCATION
9205  043322  012605                    MOV     (SP)+,R5                ;POP THE STACK
9206
9207  043324  000167  000542            JMP     TS9FIN
9208                                ;
9209                                ;ROUTINE TO CAUSE AND CHECK PAGE LENGTH ERROR ABORTS
9210                                ;
9211  043330  012337  177600        TSM9:   MOV     (R3)+,@#177600          ;SETUP UIPDR0
9212  043334  010100                    MOV     R1,R0                   ;SAVE A COPY OF R1
9213  043336  012767  000001  137432    MOV     #1,FLAG                 ;SETUP FOR AN ABORT
9214  043344  012737  000001  177572    MOV     #1,@#177572             ;TURN MMU ON
9215  043352  010704                    MOV     R7,R4                   ;SAVE PC
9216  043354  006530                    MFPI    @(R0)+                  ;DO A RELOCATION OPERATION
9217  043356  021227  000000            CMP     (R2),#0                 ;WAS AN ABORT SUPPOSED TO OCCUR
9218  043362  001007                    BNE     2$                      ;IF YES GO TO 2$
9219  043364  012605                    MOV     (SP)+,R5                ;POP THE STACK
9220  043366  022767  000001  137402    CMP     #1,FLAG                 ;DID AN ABORT OCCUR
9221  043374  001401                    BEQ     1$                      ;NO GO ON
9222  043376  104002                    ERROR   +2                  ;MMU ERROR
9223                                                                ;YES GO TO ERROR
9224  043400  000425                1$: BR      6$                          ;
9225  043402  022767  000000  137366  2$: CMP     #0,FLAG                 ;DID AN ABORT OCCUR
9226  043410  001401                    BEQ     3$                      ;YES GO ON
9227  043412  104002                    ERROR   +2                  ;MMU ERROR
9228                                                                ;NO GO TO ERROR
9229  043414  105067  137364        3$: CLRB    SAVMR0                  ;SETUP EXPECTED DATA
9230  043420  022767  040000  137356    CMP     #40000,SAVMR0           ; TEST MMR0 FOR EXPECTED VALUE
9231  043426  001401                    BEQ     4$                      ;IF OK THEN CONTINUE
9232  043430  104002                    ERROR   +2                  ;MMU ERROR
9233                                                                ;NOT OK THEN GO TO ERROR
9234  043432  022767  000020  137346  4$: CMP     #20,SAVMR1              ; TEST MMR1 FOR EXPECTED VALUE
9235  043440  001401                    BEQ     5$                      ;IF OK THEN CONTINUE
9236  043442  104002                    ERROR   +2                  ;MMU ERROR
9237                                                                ;NOT OK THEN GO TO ERROR
9238  043444  020467  137340        5$: CMP     R4,SAVMR2               ; TEST MMR2 FOR EXPECTED VALUE
9239  043450  001401                    BEQ     6$                      ;IF OK THEN CONTINUE
9240  043452  104002                    ERROR   +2                  ;MMU ERROR
9241                                                                ;NOT OK THEN GO TO ERROR
9242  043454  005067  137316        6$: CLR     FLAG                    ;CLEAR MMU ABORT FLAG
9243  043460  005067  137320            CLR     SAVMR0                  ;CLEAR STATUS REGS SAVE AREAS
9244  043464  005067  137316            CLR     SAVMR1                  ;
9245  043470  005067  137314            CLR     SAVMR2                  ;
9246  043474  005201                    INC     R1                      ;POINT TO NEXT ENTRY
9247  043476  005201                    INC     R1                      ;
9248  043500  005202                    INC     R2                      ;
9249  043502  005202                    INC     R2                      ;
9250  043504  021327  000777            CMP     (R3),#777               ;HAVE ALL ENTRIES BEEN TRIED
9251  043510  001307                    BNE     TSM9                    ;NO REPEAT
9252  043512  000207                    RTS     PC                      ;YES RETURN
9253                                ;
9254                                ;UPWARD EXPANSION TABLES
9255                                ;
```

```
 9256  043514  070006                 PLF0:    .WORD   70006
 9257  043516  070006                          .WORD   70006
 9258  043520  070006                          .WORD   70006
 9259  043522  013406                          .WORD   13406
 9260  043524  020006                          .WORD   20006
 9261  043526  004006                          .WORD   04006
 9262  043530  040006                          .WORD   40006
 9263  043532  070006                          .WORD   70006
 9264  043534  024006                          .WORD   24006
 9265  043536  004006                          .WORD   04006
 9266  043540  014006                          .WORD   14006
 9267  043542  012006                          .WORD   12006
 9268  043544  002006                          .WORD   02006
 9269  043546  001406                          .WORD   01406
 9270  043550  004006                          .WORD   04006
 9271  043552  002006                          .WORD   02006
 9272  043554  000406                          .WORD   00406
 9273  043556  007406                          .WORD   07406
 9274  043560  001006                          .WORD   01006
 9275  043562  003406                          .WORD   03406
 9276  043564  000777                          .WORD   777
 9277  043566  013000                 BN0:     .WORD   013000
 9278  043570  016000                          .WORD   016000
 9279  043572  017000                          .WORD   017000
 9280  043574  002700                          .WORD   002700
 9281  043576  014000                          .WORD   014000
 9282  043600  002000                          .WORD   002000
 9283  043602  004000                          .WORD   004000
 9284  043604  007000                          .WORD   007000
 9285  043606  002000                          .WORD   002000
 9286  043610  000700                          .WORD   000700
 9287  043612  004000                          .WORD   004000
 9288  043614  001000                          .WORD   001000
 9289  043616  000300                          .WORD   000300
 9290  043620  000400                          .WORD   000400
 9291  043622  001400                          .WORD   001400
 9292  043624  000600                          .WORD   000600
 9293  043626  000200                          .WORD   000200
 9294  043630  001700                          .WORD   001700
 9295  043632  000300                          .WORD   000300
 9296  043634  000700                          .WORD   000700
 9297  043636  000000                 ABORT0:  .WORD   0
 9298  043640  000000                          .WORD   0
 9299  043642  000001                          .WORD   1
 9300  043644  000000                          .WORD   0
 9301  043646  000001                          .WORD   1
 9302  043650  000001                          .WORD   1
 9303  043652  000000                          .WORD   0
 9304  043654  000000                          .WORD   0
 9305  043656  000000                          .WORD   0
 9306  043660  000000                          .WORD   0
 9307  043662  000001                          .WORD   1
 9308  043664  000000                          .WORD   0
 9309  043666  000000                          .WORD   0
 9310  043670  000001                          .WORD   1
 9311  043672  000001                          .WORD   1
```

```
9312  043674  000001                          .WORD   1
9313  043676  000001                          .WORD   1
9314  043700  000000                          .WORD   0
9315  043702  000001                          .WORD   1
9316  043704  000000                          .WORD   0
9317                                  ;
9318                                  ;DOWNWARD EXPANSION TABLES
9319                                  ;
9320  043706  000416                  PLF1:   .WORD   00416
9321  043710  020016                          .WORD   20016
9322  043712  024016                          .WORD   24016
9323  043714  034016                          .WORD   34016
9324  043716  074016                          .WORD   74016
9325  043720  040016                          .WORD   40016
9326  043722  020016                          .WORD   20016
9327  043724  000016                          .WORD   00016
9328  043726  030016                          .WORD   30016
9329  043730  010016                          .WORD   10016
9330  043732  014016                          .WORD   14016
9331  043734  004016                          .WORD   04016
9332  043736  002016                          .WORD   02016
9333  043740  000416                          .WORD   00416
9334  043742  000016                          .WORD   00016
9335  043744  003416                          .WORD   03416
9336  043746  001016                          .WORD   01016
9337  043750  001416                          .WORD   01416
9338  043752  000416                          .WORD   00416
9339  043754  000777                          .WORD   777
9340  043756  000100                  BN1:    .WORD   000100
9341  043760  010000                          .WORD   010000
9342  043762  006000                          .WORD   006000
9343  043764  016000                          .WORD   016000
9344  043766  016000                          .WORD   016000
9345  043770  004000                          .WORD   004000
9346  043772  000000                          .WORD   000000
9347  043774  000000                          .WORD   000000
9348  043776  004000                          .WORD   004000
9349  044000  004000                          .WORD   004000
9350  044002  004000                          .WORD   004000
9351  044004  000000                          .WORD   000000
9352  044006  000300                          .WORD   000300
9353  044010  000000                          .WORD   000000
9354  044012  000400                          .WORD   000400
9355  044014  001000                          .WORD   001000
9356  044016  000100                          .WORD   000100
9357  044020  000400                          .WORD   000400
9358  044022  000200                          .WORD   000200
9359  044024  000000                  ABORT7: .WORD   0
9360  044026  000000                          .WORD   0
9361  044030  000000                          .WORD   0
9362  044032  000000                          .WORD   0
9363  044034  000001                          .WORD   1
9364  044036  000001                          .WORD   1
9365  044040  000001                          .WORD   1
9366  044042  000000                          .WORD   0
9367  044044  000001                          .WORD   1
```

```
9368  044046  000000                          .WORD   0
9369  044050  000000                          .WORD   0
9370  044052  000001                          .WORD   1
9371  044054  000001                          .WORD   1
9372  044056  000001                          .WORD   1
9373  044060  000000                          .WORD   0
9374  044062  000000                          .WORD   0
9375  044064  000001                          .WORD   1
9376  044066  000000                          .WORD   0
9377  044070  000000                          .WORD   0
9378                                  ;
9379  044072  000240                  TS9FIN: NOP
9380  044074                          TSMM10:
9381                                  ;       FUNCTIONAL TEST OF BITS <6:1> OF MMR0
9382  044074  005037  177572                  CLR     @#177572                ;MMU OFF
9383  044100  005067  136672                  CLR     FLAG                    ;CLEAR MMU ABORT FLAG
9384  044104  005067  136674                  CLR     SAVMR0                  ;CLEAR STATUS REGS SAVE AREAS
9385  044110  005067  136672                  CLR     SAVMR1                  ;
9386  044114  005067  136670                  CLR     SAVMR2                  ;
9387  044120  004767  067010                  JSR     PC,MMU                  ;INIT MMU
9388  044124  005037  177776                  CLR     @#177776                ;INIT PSW: PREVIOUS MODE = KERNAL
9389  044130  012702  020200                  MOV     #020200,R2              ;
9390  044134  012737  077400  172302          MOV     #077400,@#172302        ;SETUP KIPDR1 TO ABORT
9391  044142  012767  000001  136626          MOV     #1,FLAG                 ;SETUP FLAG FOR AN ABORT
9392  044150  012737  000001  177572          MOV     #1,@#177572             ;TURN MMU ON
9393  044156  010701                          MOV     R7,R1                   ;SAVE PC
9394  044160  006522                          MFPI    (R2)+                   ;DO A RELOCATION VIA KIPAR1
9395  044162  012704  100003                  MOV     #100003,R4              ;SETUP EXPECTED DATA
9396  044166  004767  000202                  JSR     PC,TS10                 ;CHECK IF AN ABORT OCCURRED AND
9397                                                                          ;IF YES CHECK BITS <6:1> OF MMR0.
9398  044172  012737  030000  177776          MOV     #030000,@#177776        ;INIT PSW: PREVIOUS MODE = USER
9399  044200  004767  066730                  JSR     PC,MMU                  ;INIT MMU
9400  044204  012737  077400  177636          MOV     #077400,@#177636        ;SETUP UDPDR7 TO ABORT
9401  044212  012702  160000                  MOV     #160000,R2              ;
9402  044216  012767  000001  136552          MOV     #1,FLAG                 ;SETUP FLAG FOR AN ABORT
9403  044224  012737  000001  177572          MOV     #1,@#177572             ;TURN MMU ON
9404  044232  010701                          MOV     R7,R1                   ;SAVE PC
9405  044234  106522                          MFPD    (R2)+                   ;DO A RELOCATION VIA UDPAR7
9406  044236  012704  100177                  MOV     #100177,R4              ;SETUP EXPECTED DATA
9407  044242  004767  000126                  JSR     PC,TS10                 ;CHECK IF AN ABORT OCCURRED AND
9408                                                                          ;IF YES CHECK BITS <6:1> OF MMR0.
9409  044246  012737  010000  177776          MOV     #010000,@#177776        ;INIT PSW: PREVIOUS MODE= SUPERVISOR
9410  044254  004767  066654                  JSR     PC,MMU                  ;INIT MMU
9411  044260  012737  077400  172212          MOV     #077400,@#172212        ;SETUP SIPDR5 TO ABORT
9412  044266  012702  120000                  MOV     #120000,R2              ;ACCESS PAGE 05
9413  044272  012767  000001  136476          MOV     #1,FLAG                 ;SETUP FLAG FOR AN ABORT
9414  044300  012737  000001  177572          MOV     #1,@#177572             ;TURN MMU ON
9415  044306  010701                          MOV     R7,R1                   ;SAVE PC
9416  044310  006522                          MFPI    (R2)+                   ;DO A RELOCATION VIA SIPAR5
9417  044312  012704  100053                  MOV     #100053,R4              ;SETUP EXPECTED DATA:ABORT, PAGE 05
9418  044316  004767  000052                  JSR     PC,TS10                 ;CHECK IF AN ABORT OCCURRED AND
9419                                                                          ;IF YES CHECK BITS <6:1> OF MMR0.
9420                                  ;
9421                                  ;TEST THAT ILLEGAL MODE CAUSES MMU ABORT
9422                                  ;
9423  044322  012737  020000  177776          MOV     #020000,@#177776        ;INIT PSW:SET ILLEGAL PREVIOUS MODE
```

```
9424  044330  004767  066600            JSR     PC,MMU                  ;INIT MMU
9425  044334  012702  040000            MOV     #40000,R2               ;SET UP ACCESS TO PAGE 2
9426  044340  012767  000001  136430    MOV     #1,FLAG                 ;SETUP FLAG FOR AN ABORT
9427  044346  012737  000001  177572    MOV     #1,@#177572             ;TURN MMU ON
9428  044354  010701                    MOV     R7,R1                   ;SAVE PC
9429  044356  106522                    MFPD    (R2)+                   ;DO A RELOCATION
9430  044360  012704  100105            MOV     #100105,R4              ;SETUP EXPECTED DATA:ABORT, ILLEGAL
9431                                                                    ; PROCESSOR MODE, PAGE 02
9432  044364  004767  000004            JSR     PC,TS10                 ;CHECK IF AN ABORT OCCURRED AND
9433                                                                    ;IF YES CHECK BITS <6:1> OF MMR0.
9434
9435  044370  000167  000062            JMP     T10FIN
9436                                    ;
9437                                    ;ROUTINE TO CHECK IF A MMU ABORT OCCURRED AND IF STATUS REG MMR0
9438                                    ;CONTAINS EXPECTED DATA
9439                                    ;
9440  044374  022767  000000  136374    TS10:   CMP     #0,FLAG                 ;DID AN ABORT OCCUR
9441  044402  001401                            BEQ     1$                      ;YES GO ON
9442  044404  104002                            ERROR   .2              ;MMU ERROR
9443                                                                    ;NO GO TO ERROR
9444  044406  020467  136372            1$:     CMP     R4,SAVMR0               ; TEST MMR0 FOR EXPECTED DATA
9445  044412  001401                            BEQ     2$                      ;OK GO ON
9446  044414  104002                            ERROR   .2              ;MMU ERROR
9447                                                                    ;NO GO TO ERROR
9448  044416  022767  000022  136362    2$:     CMP     #22,SAVMR1              ; TEST MMR1 FOR EXPECTED DATA
9449  044424  001401                            BEQ     3$                      ;OK GO ON
9450  044426  104002                            ERROR   .2              ;MMU ERROR
9451                                                                    ;NO GO TO ERROR
9452  044430  020167  136354            3$:     CMP     R1,SAVMR2               ; TEST MMR2 FOR EXPECTED DATA
9453  044434  001401                            BEQ     4$                      ;OK GO ON
9454  044436  104002                            ERROR   .2              ;MMU ERROR
9455                                                                    ;NO GO TO ERROR
9456  044440  005067  136340            4$:     CLR     SAVMR0                  ;CLEAR MMU STATUS REGS SAVE AREAS
9457  044444  005067  136336                    CLR     SAVMR1                  ;
9458  044450  005067  136334                    CLR     SAVMR2                  ;
9459  044454  000207                            RTS     PC                      ;RETURN
9460                                    ;
9461  044456                            T10FIN:
9462  044456                            TSMM11:
9463                                    ;       TEST DATA SPACE BITS MMR3
9464  044456  005037  177572                    CLR     @#177572                ;MMU OFF
9465  044462  005067  136310                    CLR     FLAG                    ;CLEAR MMU ABORT FLAG
9466  044466  012737  030000  177776    MOV     #30000,@#177776         ;SETUP PSW
9467  044474  012701  000026            MOV     #26,R1                  ;SETUP FIRST MMR3 VALUE
9468  044500  012703  177610            MOV     #177610,R3              ;POINT TO UIPDR4
9469  044504  012704  000021            MOV     #21,R4                  ;SETUP SECOND MMR3 VALUE
9470  044510  004767  000060            JSR     PC,TS11                 ; TEST ENABLE USER DATA SPACE BIT
9471  044514  012737  000000  177776    MOV     #0,@#177776             ;SETUP PSW
9472  044522  012701  000023            MOV     #23,R1                  ;SETUP FIRST MMR3 VALUE
9473  044526  012703  172310            MOV     #172310,R3              ;POINT TO KIPDR4
9474  044532  012704  000024            MOV     #24,R4                  ;SETUP SECOND MMR3 VALUE
9475  044536  004767  000032            JSR     PC,TS11                 ; TEST ENABLE KERNEL DATA SPACE BIT
9476  044542  012737  010000  177776    MOV     #10000,@#177776         ;SETUP PSW
9477  044550  012701  000025            MOV     #25,R1                  ;SETUP FIRST MMR3 VALUE
9478  044554  012703  172210            MOV     #172210,R3              ;POINT TO SIPDR4
9479  044560  012704  000022            MOV     #22,R4                  ;SETUP SECOND MMR3 VALUE
```

```
9480  044564  004767  000004              JSR     PC,TS11                 ; TEST ENABLE SUPERVISOR DATA SPACE BIT
9481
9482  044570  000167  000120              JMP     T11FIN
9483                                  ;
9484                                  ;ROUTINE TO TEST ENABLE DATA SPACE BITS OF MMR3
9485                                  ;
9486  044574  004767  066334          TS11: JSR     PC,MMU                  ;INIT MMU
9487  044600  010137  172516                MOV     R1,@#172516             ;DISABLE DATA SPACE OF MODE UNDER TEST
9488  044604  012713  077400                MOV     #77400,(R3)             ;SETUP IPDR TO ABORT
9489  044610  012702  100000                MOV     #100000,R2              ;
9490  044614  012767  000001  136154        MOV     #1,FLAG                 ;SETUP FLAG FOR AN ABORT
9491  044622  012737  000001  177572        MOV     #1,@#177572             ;MMU ON
9492  044630  106522                        MFPD    (R2)+                   ;DO A RELOCATION
9493  044632  022767  000000  136136        CMP     #0,FLAG                 ;DID AN ABORT OCCUR
9494  044640  001401                        BEQ     1$                      ;YES GO ON
9495  044642  104002                        ERROR   .2              ;MMU ERROR
9496                                                                ;NO GO TO ERROR
9497  044644  010437  172516          1$:   MOV     R4,@#172516             ;ENABLE DATA SPACE OF MODE UNDER TEST
9498  044650  012702  100000                MOV     #100000,R2              ;
9499  044654  012767  000001  136114        MOV     #1,FLAG                 ;SETUP FLAG FOR AN ABORT
9500  044662  012737  000001  177572        MOV     #1,@#177572             ;MMU ON
9501  044670  106522                        MFPD    (R2)+                   ;DO A RELOCATION
9502  044672  005726                        TST     (SP)+                   ;POP THE STACK
9503  044674  022767  000001  136074        CMP     #1,FLAG                 ;DID AN ABORT OCCUR
9504  044702  001401                        BEQ     2$                      ;NO GO ON
9505  044704  104002                        ERROR   .2              ;MMU ERROR
9506                                                                ;YES GO TO ERROR
9507  044706  005067  136064          2$:   CLR     FLAG                    ;CLEAR MMU ABORT FLAG
9508  044712  000207                        RTS     PC                      ;RETURN
9509
9510  044714                          T11FIN:
9511  044714                          TSMM12:
9512                                  ;       MMR1 FUNCTIONAL TEST
9513  044714  005037  177572                CLR     @#177572                ;MMU OFF
9514  044720  005067  136052                CLR     FLAG                    ;CLEAR MMU ABORT FLAG
9515  044724  005067  136056                CLR     SAVMR1                  ;CLEAR STATUS REG SAVE AREA
9516  044730  004767  066200                JSR     PC,MMU                  ;INIT MMU
9517  044734  012737  030000  177776        MOV     #30000,@#177776         ;INIT PSW
9518  044742  012704  100200                MOV     #100200,R4              ;SETUP TEST LOCATIONS
9519  044746  010401                        MOV     R4,R1                   ;
9520  044750  012705  100101                MOV     #100101,R5              ;
9521  044754  010502                        MOV     R5,R2                   ;
9522  044756  012737  000020  172516        MOV     #20,@#172516            ;INIT MMR3
9523  044764  012737  077402  172310        MOV     #77402,@#172310         ;SETUP KIPDR4 TO ABORT
9524  044772  012703  006414                MOV     #6414,R3                ;SETUP EXPECTED DATA FOR MMR1
9525  044776  012767  000001  135772        MOV     #1,FLAG                 ;SETUP FLAG FOR AN ABORT
9526  045004  012737  000001  177572        MOV     #1,@#177572             ;TURN MMU ON
9527  045012  010767  135744                MOV     R7,SLOC00               ;SAVE PC
9528  045016  112425                        MOVB    (R4)+,(R5)+             ;DO A RELOCATION
9529  045020  004767  000206                JSR     PC,TS12                 ;CHECK IF AN ABORT OCCURRED AND IF
9530                                                                        ;YES IF MMR1 EQUALS EXPECTED DATA
9531  045024  012703  175011                MOV     #175011,R3              ;SETUP EXPECTED DATA FOR MMR1
9532  045030  012767  000001  135740        MOV     #1,FLAG                 ;SETUP FLAG FOR AN ABORT
9533  045036  012737  000001  177572        MOV     #1,@#177572             ;TURN MMU ON
9534  045044  010767  135712                MOV     R7,SLOC00               ;SAVE PC
9535  045050  112142                        MOVB    (R1)+,-(R2)             ;DO A RELOCATION
```

```
9536  045052  004767  000154                  JSR     PC,TS12         ;CHECK IF AN ABORT OCCURRED AND IF
9537                                                                    ;YES IF MMR1 EQUALS EXPECTED DATA
9538  045056  012703  006771                  MOV     #6771,R3        ;SETUP EXPECTED DATA FOR MMR1
9539  045062  012767  000001  135706          MOV     #1,FLAG         ;SETUP FLAG FOR AN ABORT
9540  045070  012737  000001  177572          MOV     #1,@#177572     ;TURN MMU ON
9541  045076  010767  135660                  MOV     R7,SLOC00       ;SAVE PC
9542  045102  114125                          MOVB    -(R1),(R5)+     ;DO A RELOCATION
9543  045104  004767  000122                  JSR     PC,TS12         ;CHECK IF AN ABORT OCCURRED AND IF
9544                                                                    ;YES IF MMR1 EQUALS EXPECTED DATA
9545  045110  012703  006411                  MOV     #6411,R3        ;SETUP EXPECTED DATA FOR MMR1
9546  045114  012767  000001  135654          MOV     #1,FLAG         ;SETUP FLAG FOR AN ABORT
9547  045122  012737  000001  177572          MOV     #1,@#177572     ;TURN MMU ON
9548  045130  010767  135626                  MOV     R7,SLOC00       ;SAVE PC
9549  045134  112125                          MOVB    (R1)+,(R5)+     ;DO A RELOCATION
9550  045136  004767  000070                  JSR     PC,TS12         ;CHECK IF AN ABORT OCCURRED AND IF
9551                                                                    ;YES IF MMR1 EQUALS EXPECTED DATA
9552  045142  012703  171025                  MOV     #171025,R3      ;SETUP EXPECTED DATA FOR MMR1
9553  045146  012767  000001  135622          MOV     #1,FLAG         ;SETUP FLAG FOR AN ABORT
9554  045154  012737  000001  177572          MOV     #1,@#177572     ;TURN MMU ON
9555  045162  010767  135574                  MOV     R7,SLOC00       ;SAVE PC
9556  045166  012542                          MOV     (R5)+,-(R2)     ;DO A RELOCATION
9557  045170  004767  000036                  JSR     PC,TS12         ;CHECK IF AN ABORT OCCURRED AND IF
9558                                                                    ;YES IF MMR1 EQUALS EXPECTED DATA
9559  045174  012703  012762                  MOV     #12762,R3       ;SETUP EXPECTED DATA FOR MMR1
9560  045200  012767  000001  135570          MOV     #1,FLAG         ;SETUP FLAG FOR AN ABORT
9561  045206  012737  000001  177572          MOV     #1,@#177572     ;TURN MMU ON
9562  045214  010767  135542                  MOV     R7,SLOC00       ;SAVE PC
9563  045220  014225                          MOV     -(R2),(R5)+     ;DO A RELOCATION
9564  045222  004767  000004                  JSR     PC,TS12         ;CHECK IF AN ABORT OCCURRED AND IF
9565                                                                    ;YES IF MMR1 EQUALS EXPECTED DATA
9566
9567  045226  000167  000046                  JMP     T12FIN
9568                                          ;
9569                                          ;ROUTINE TO CHECK IF AN ABORT OCCURRED AND IF MMR1 EQUALS EXPECTED DATA
9570                                          ;
9571  045232  022767  000000  135536  TS12:   CMP     #0,FLAG                 ;DID AN ABORT OCCUR
9572  045240  001401                          BEQ     1$                      ;YES GO ON
9573  045242  104002                          ERROR   .2              ;MMU ERROR
9574                                                                  ;NO GO TO ERROR
9575  045244  020367  135536          1$:     CMP     R3,SAVMR1               ; TEST MMR1 FOR EXPECTED DATA
9576  045250  001401                          BEQ     2$                      ;OK GO ON
9577  045252  104002                          ERROR   .2              ;MMU ERROR
9578                                                                  ;NO GO TO ERROR
9579  045254  026767  135502  135526  2$:     CMP     SLOC00,SAVMR2           ; TEST MMR2 FOR EXPECTED DATA
9580  045262  001401                          BEQ     3$                      ;OK GO ON
9581  045264  104002                          ERROR   .2              ;MMU ERROR
9582                                                                  ;NO GO TO ERROR
9583  045266  005067  135514          3$:     CLR     SAVMR1                  ;CLEAR STATUS REG SAVE AREA
9584  045272  005067  135512                  CLR     SAVMR2                  ;
9585  045276  000207                          RTS     PC                      ;RETURN
9586                                          ;
9587  045300  000240                  T12FIN: NOP
9588  045302                          TSMM13:
9589                                          ;       ADDER RELOCATION TEST PART A
9590                                          ;;**************************************************************
9591                                          ;(NEED 16 BITS OF MEMORY ADDRESSING)
```

```
9592  045302  005037  177572              CLR     @#177572                ;MMU OFF
9593  045306  005067  135464              CLR     FLAG                    ;CLEAR MMU ABORT FLAG
9594  045312  005037  177776              CLR     @#177776                ;INIT PSW
9595  045316  004767  065612              JSR     PC,MMU                  ;INIT MMU
9596  045322  012737  000020  172516      MOV     #20,@#172516            ;INIT MMR3
9597  045330  012703  045504              MOV     #PARAD1,R3              ;SETUP PARS WITH TEST VALUES
9598  045334  012701  045536              MOV     #PARVA1,R1              ;
9599  045340  012133              1$:     MOV     (R1)+,@(R3)+            ;
9600  045342  021127  000333              CMP     (R1),#333               ;
9601  045346  001374                      BNE     1$                      ;
9602  045350  012703  045622              MOV     #PHY1,R3                ;SET POINTERS TO ADDER PART A
9603  045354  012701  045570              MOV     #VIR1,R1                ; TEST TABLES.
9604  045360  012702  045654              MOV     #MODE1,R2               ;
9605  045364  012237  177776      2$:     MOV     (R2)+,@#177776          ;INIT PSW
9606  045370  013305                      MOV     @(R3)+,R5               ;SAVE DATA AT PHYSICAL ADDRESS
9607  045372  012737  000001  177572      MOV     #1,@#177572             ;TURN MMU ON
9608  045400  006531                      MFPI    @(R1)+                  ;SAVE DATA AT RELOCATED VIRTUAL ADDRESS
9609  045402  012604                      MOV     (SP)+,R4                ;
9610  045404  005037  177572              CLR     @#177572                ;TURN MMU OFF
9611  045410  020504                      CMP     R5,R4                   ;IS DATA EQUAL TO EXPECTED
9612  045412  001401                      BEQ     3$                      ;YES GO ON
9613  045414  104002                      ERROR   +2              ;MMU ERROR
9614                                                          ;NO IT IS AN ADDER ERROR
9615  045416  021327  000111      3$:     CMP     (R3),#111               ;ARE WE READY TO TEST DATA SPACE
9616  045422  001360                      BNE     2$                      ;NO GO TO 2$
9617  045424  005203                      INC     R3                      ;POINT TO DATA SPACE VALUES
9618  045426  005203                      INC     R3                      ;
9619  045430  005201                      INC     R1                      ;
9620  045432  005201                      INC     R1                      ;
9621  045434  005202                      INC     R2                      ;
9622  045436  005202                      INC     R2                      ;
9623  045440  012237  177776              MOV     (R2)+,@#177776          ;INIT PSW
9624  045444  013305                      MOV     @(R3)+,R5               ;SAVE DATA AT PHYSICAL ADDRESS
9625  045446  012737  000027  172516      MOV     #27,@#172516            ;INIT MMR3
9626  045454  012737  000001  177572      MOV     #1,@#177572             ;TURN MMU ON
9627  045462  106531                      MFPD    @(R1)+                  ;SAVE DATA AT RELOCATED VIRTUAL ADDRESS
9628  045464  012604                      MOV     (SP)+,R4                ;POP THE STACK
9629  045466  005037  177572              CLR     @#177572                ;TURN MMU OFF
9630  045472  020504                      CMP     R5,R4                   ;IS DATA EQUAL TO EXPECTED
9631  045474  001401                      BEQ     4$                      ;YES GO ON
9632  045476  104002                      ERROR   +2              ;MMU ERROR
9633                                                          ;NO IT IS AN ADDER ERROR
9634  045500                      4$:
9635  045500  000167  000202              JMP     T13FIN
9636                              ;
9637                              ;ADDER TEST PART A TABLES
9638                              ;
9639  045504  172240              PARAD1: .WORD   172240
9640  045506  177642                      .WORD   177642
9641  045510  172252                      .WORD   172252
9642  045512  177640                      .WORD   177640
9643  045514  172242                      .WORD   172242
9644  045516  172254                      .WORD   172254
9645  045520  177652                      .WORD   177652
9646  045522  177644                      .WORD   177644
9647  045524  172246                      .WORD   172246
```

```
9648  045526  177654                  .WORD   177654
9649  045530  172250                  .WORD   172250
9650  045532  177660                  .WORD   177660
9651  045534  000333                  .WORD   333
9652  045536  000000          PARVA1: .WORD   000000
9653  045540  000010                  .WORD   000010
9654  045542  177777                  .WORD   177777
9655  045544  177601                  .WORD   177601
9656  045546  000010                  .WORD   000010
9657  045550  000052                  .WORD   000052
9658  045552  000070                  .WORD   000070
9659  045554  000010                  .WORD   000010
9660  045556  000010                  .WORD   000010
9661  045560  000060                  .WORD   000060
9662  045562  000000                  .WORD   000000
9663  045564  000010                  .WORD   000010
9664  045566  000333                  .WORD   333
9665  045570  000000          VIR1:   .WORD   000000
9666  045572  025000                  .WORD   025000
9667  045574  135224                  .WORD   135224
9668  045576  017700                  .WORD   017700
9669  045600  033000                  .WORD   033000
9670  045602  145252                  .WORD   145252
9671  045604  121000                  .WORD   121000
9672  045606  043000                  .WORD   043000
9673  045610  075000                  .WORD   075000
9674  045612  142000                  .WORD   142000
9675  045614  117700                  .WORD   117700
9676  045616  000111                  .WORD   111
9677  045€    007000                  .WORD   007000
9678  045622  000000          PHY1:   .WORD   000000
9679  045624  006000                  .WORD   006000
9680  045626  015124                  .WORD   015124
9681  045630  000000                  .WORD   000000
9682  045632  014000                  .WORD   014000
9683  045634  012452                  .WORD   012452
9684  045636  010000                  .WORD   010000
9685  045640  004000                  .WORD   004000
9686  045642  016000                  .WORD   016000
9687  045644  010000                  .WORD   010000
9688  045646  017700                  .WORD   017700
9689  045650  000111                  .WORD   111
9690  045652  010000                  .WORD   010000
9691  045654  010000          MODE1:  .WORD   010000
9692  045656  030000                  .WORD   030000
9693  045660  010000                  .WORD   010000
9694  045662  030000                  .WORD   030000
9695  045664  010000                  .WORD   010000
9696  045666  010000                  .WORD   010000
9697  045670  030000                  .WORD   030000
9698  045672  030000                  .WORD   030000
9699  045674  010000                  .WORD   010000
9700  045676  030000                  .WORD   030000
9701  045700  010000                  .WORD   010000
9702  045702  000111                  .WORD   111
9703  045704  030000                  .WORD   030000
```

```
9704                                    ;
9705  045706                            T13FIN:
9706  045706                            TS1822:
9707                                    ;       TEST 22/18 BIT ADDRESS OPTION
9708                                    ;;*****************************************************************
9709                                    ;CHECK THE SOFTWARE SWITCH REGISTER TO DETERMINE IF THIS IS A 22 BIT OR AN
9710                                    ;18 BIT ADDRESS SYSTEM. BIT 08 IN THE SWR=1 INDICATES AN 18 BIT SYSTEM.
9711                                    ;IF WE'RE IN A 22 BIT SYSTEM WE CAN PERFORM SOME EXTRA TESTS.
9712  045706  032777  000400  133224            BIT     #BIT08,@SWR             ;IS BIT 08 SET?
9713  045714  001405                            BEQ     100$                    ;BRANCH IF ITS NOT
9714  045716  062737  000001  001204            ADD     #1,@#$TESTN             ;KEEP TEST NUMBERS IN ORDER
9715                                                                            ;ADD 1 FOR THE TESTS WE'RE SKIPPING
9716  045724  000167  001422                    JMP     T14FIN                  ;SKIP OVER THESE TESTS IF IT IS
9717                                    ;
9718                                    ;IF THIS IS A 22 BIT SYSTEM CHECK THE 22/18 BIT ADDRESS OPTION
9719                                    ;
9720                                    ;TO TEST 22 BIT ADDRESSING WE DO THE FOLLOWING:
9721                                    ;       A. ENABLE 22 BIT ADDRESSING MODE
9722                                    ;       B. CLEAR ADDRESS 0
9723                                    ;       C. WRITE PHYSICAL ADDRESS 17000000 WITH ALL ONES
9724                                    ;       D. CHECK ADDRESS 0
9725                                    ;IF ADDRESS 0 IS UNCHANGED (=0) OR A TIME OUT OCCURRED, IT INDICATES
9726                                    ;22 BIT MODE IS FUNCTIONING.
9727                                    ;IF ADDRESS 0 =177777 IT INDICATES THAT 22 BIT MODE IS NOT FUNCTIONING.
9728
9729  045730  013767  000004  135024  100$:   MOV     @#4,SLOC00              ;SAVE VECTOR
9730  045736  013767  000006  135020           MOV     @#6,SLOC01              ;SAVE VECTOR
9731  045744  012737  046024  000004           MOV     #1$,@#4                 ;SET VECTOR FOR NXM TRAP
9732  045752  012737  000340  000006           MOV     #340,@#6                ;
9733  045760  012767  000020  124530           MOV     #20,SR3                 ;ENABLE 22 BIT MODE ADDRESSING
9734  045766  012767  170000  124360           MOV     #170000,KIPAR6          ;SET KIPAR6 FOR 1920-1924KW ADDR RANGE
9735  045774  012767  000001  131570           MOV     #1,SR0                  ;ENABLE MMU
9736  046002  005067  131772                   CLR     0                       ;CLEAR ADDR 0
9737  046006  012737  177777  140000           MOV     #177777,@#140000        ;MOVE ALL ONES TO ADDR 17000000 VIA KIPAR6
9738                                                                            ;A TIME OUT ERROR OR
9739  046014  005767  131760                   TST     0                       ;ADDR 0 REMAINING CLEAR INDICATES
9740                                                                            ;THAT 22 BIT ADDRESS MODE IS WORKING AND
9741                                                                            ;THAT SOME FURTHER TESTS SHOULD BE PERFORMED
9742  046020  001405                           BEQ     2$                      ;IF ADDR 0 =177777
9743                                                                            ;ERROR! 22 BIT ADDRESS MODE BAD
9744  046022  104002                           ERROR   .2              ;MMU ERROR
9745  046024  012706  001000           1$:     MOV     #STBOT,SP               ;GOT HERE AS A RESULT OF NXM TRAP---
9746                                                                            ;CLEAN UP THE STACK
9747  046030  005037  177766                   CLR     @#177766                ;CLEAR CPU ERROR REGISTER
9748  046034  012737  046074  000004  2$:      MOV     #3$,@#4                 ;SET UP VECTOR FOR NXM TRAP
9749  046042  042767  000020  124446           BIC     #BIT04,SR3              ;SET 18 BIT ADDRESSING MODE IN SR3
9750  046050  012767  170000  124276           MOV     #170000,KIPAR6          ;SET KIPAR6 SO THAT BITS 18-21 SHOULD
9751                                                                            ;BE ASSERTED IF 22 BIT ADR WAS ENABLED
9752  046056  012737  177777  140000           MOV     #177777,@#140000        ;TRY TO WRITE ADDR 17000000 VIA KIPAR6
9753                                                                            ;ADDR 0 SHOULD = 177777. A TIME OUT
9754  046064  022737  177777  000000           CMP     #177777,@#0             ;OR ADDR 0 = ZERO INDICATES AN ERROR
9755  046072  001403                           BEQ     4$                      ;GO TO NEXT TEST IF ADDR 0=177777
9756  046074  005067  131472           3$:     CLR     SR0                     ;DISABLE MMU BEFORE ERROR.
9757                                                                            ;ERROR! 18 BIT ADDR OPTION IS N.G.
9758  046100  104002                           ERROR   .2              ;MMU ERROR
9759                                    ;
```

```
9760                                    ;TEST ADDRESS BITS 18 THRU 21
9761                                    ;
9762
9763  046102  052767  000020  124406  4$:     BIS     #BIT04,SR3              ;ENABLE 22 BIT ADDRESSING MODE
9764  046110  012767  046152  131666          MOV     #5$,4                   ;SET UP FOR NXM TRAP
9765  046116  005067  131656                  CLR     0                       ;CLEAR ADDRESS 0
9766  046122  012767  010000  124224          MOV     #10000,KIPAR6           ;TEST ADDRESS BIT 18
9767  046130  012737  177777  140000          MOV     #177777,@#140000        ;WRITE ALL ONES TO ADDR 1000000
9768  046136  005767  131636                  TST     0                       ;TEST ADDRESS 0. SHOULD = ZERO
9769  046142  001403                          BEQ     5$                      ;BRANCH IF ADDRESS 0=0
9770  046144  005067  131422                  CLR     SR0                     ;DISABLE MMU BEFORE ERROR
9771  046150  104002                          ERROR   .2                  ;MMU ERROR
9772                                                                      ;ERROR! BIT 18 DID NOT ASSERT
9773  046152  012737  046214  000004  5$:     MOV     #6$,@#4                 ;SET UP FOR NXM TRAP
9774  046160  005067  131614                  CLR     0                       ;CLEAR ADDR 0
9775  046164  012767  020000  124162          MOV     #20000,KIPAR6           ;TEST ADDRESS BIT 19
9776  046172  012737  177777  140000          MOV     #177777,@#140000        ;WRITE ALL ONES TO ADDR 2000000
9777  046200  005767  131574                  TST     0                       ;TEST ADDR 0. SHOULD= ZERO
9778  046204  001403                          BEQ     6$                      ;BRANCH IF ADDRESS 0=0
9779  046206  005067  131360                  CLR     SR0                     ;DISABLE MMU BEFORE ERROR
9780  046212  104002                          ERROR   .2                  ;MMU ERROR
9781                                                                      ;ERROR! BIT 19 DID NOT ASSERT
9782  046214  012737  046256  000004  6$:     MOV     #7$,@#4                 ;SET UP FOR NXM TRAP
9783  046222  005067  131552                  CLR     0                       ;CLEAR ADDR 0
9784  046226  012767  040000  124120          MOV     #40000,KIPAR6           ;TEST ADDRESS BIT 20
9785  046234  012737  177777  140000          MOV     #177777,@#140000        ;WRITE ALL ONES TO ADDR 4000000
9786  046242  005767  131532                  TST     0                       ;TEST ADDR 0. SHOULD =0
9787  046246  001403                          BEQ     7$                      ;BRANCH IF ADDRESS 0 =0
9788  046250  005067  131316                  CLR     SR0                     ;DISABLE MMU BEFORE ERROR
9789  046254  104002                          ERROR   .2                  ;MMU ERROR
9790                                                                      ;ERROR! BIT 20 DID NOT ASSERT
9791  046256  012737  046320  000004  7$:     MOV     #8$,@#4                 ;SET UP FOR NXM
9792  046264  005067  131510                  CLR     0                       ;CLEAR ADDRESS 0
9793  046270  012767  100000  124056          MOV     #100000,KIPAR6          ;TEST ADDRESS BIT 21
9794  046276  012737  177777  140000          MOV     #177777,@#140000        ;WRITE ALL ONES AT ADDR 10000000
9795  046304  005767  131470                  TST     0                       ;CHECK ADDRESS 0. SHOULD = 0
9796  046310  001403                          BEQ     8$                      ;BRANCH IF ADDR 0 = 0
9797  046312  005067  131254                  CLR     SR0                     ;DISABLE MMU    BEFORE ERROR
9798  046316  104002                          ERROR   .2                  ;MMU ERROR
9799                                                                      ;ERROR! ADDR BIT 21 DID NOT ASSERT
9800  046320  005067  131246          8$:     CLR     SR0                     ;DISABLE MMU
9801  046324  005037  177766                  CLR     @#177766                ;CLEAR CPU ERROR REGISTER
9802  046330  012706  001000                  MOV     #STBOT,R6               ;RESET STACK POINTER
9803  046334  013737  002762  000004          MOV     @#SLOC00,@#4            ;RESTORE VECTORS
9804  046342  013737  002764  000006          MOV     @#SLOC01,@#6            ;
9805
9806  046350                                TSMM14:
9807                                        ;       ADDER RELOCATION TEST PART B
9808                                        ;;**************************************************************
9809                                        ;(NEED 22 BITS OF MEMORY ADDRESSING)
9810  046350  005037  177572                  CLR     @#SR0                   ;TURN OFF MMU.
9811  046354  005067  131406                  CLR     CPEREG                  ;CLEAR THE CPU ERROR REGISTER
9812  046360  013737  000004  002762          MOV     @#4,@#SLOC00            ;SAVE LOC 4 IN SLOC00.
9813  046366  013737  000006  002764          MOV     @#6,@#SLOC01            ;SAVE LOC 6 IN SLOC01.
9814  046374  012737  047064  000004          MOV     #NXMTRP,@#4             ;SET UP FOR TIMEOUT TRAP
9815  046402  012737  000340  000006          MOV     #340,@#6                ;SET UP FOR TIMEOUT TRAP
```

```
9816  046410  005037  172340          CLR     @#KIPAR0                ;SET KER PARO FOR 1ST 4KW OF MEMORY.
9817  046414  012767  077406  123656  MOV     #77406,KIPDR0           ;SET KER PDR FOR 4KW R/W ACCESS.
9818  046422  012737  177500  172354  MOV     #177500,@#KIPAR6        ;SET UP KERNEL PAGE ADDR REG 6
9819                                                                  ;FOR HIGHEST 4K WORDS OF NON-I/O
9820                                                                  ;FOR 2 MEG WORDS OF MEMORY.
9821  046430  012767  077406  123656  MOV     #77406,KIPDR6           ;SET KER PDR6 FOR 4KW R/W ACCESS.
9822  046436  012737  000020  172516  MOV     #20,@#SR3               ;ENABLE 22 BIT ADDRESSING.
9823  046444  012737  000001  177572  MOV     #1,@#SR0                ;TURN ON THE MMU.
9824  046452  005737  157776          TST     @#157776                ;ATTEMPT TO ADDRESS LAST MEMORY ADDR.
9825                  ;**************WILL TRAP TO 4 IF 2 MEG WORDS OF MEMORY NOT AVAILABLE*********
9826  046456  013737  002762  000004  MOV     @#SLOC00,@#4            ;RESTORE LOC 4
9827  046464  013737  002764  000006  MOV     @#SLOC01,@#6            ;RESTORE LOC 6
9828  046472  005037  177572          CLR     @#177572                ;MMU OFF
9829  046476  005037  002776          CLR     @#FLAG                  ;CLEAR MMU ABORT FLAG
9830  046502  004767  064426          JSR     PC,MMU                  ;INIT MMU
9831  046506  012737  010000  177776  MOV     #10000,@#177776         ;INIT PSW
9832  046514  012737  000020  172516  MOV     #20,@#172516            ;INIT MMR3
9833  046522  052737  001000  177746  BIS     #1000,@#177746          ;TURN CACHE TEST FEATURE ON
9834  046530  012704  047266          MOV     #PARVA3,R4              ;SET POINTERS TO INIT TABLES
9835  046534  012701  047320          MOV     #VIR3,R1                ;
9836  046540  012437  172246  1$:     MOV     (R4)+,@#172246          ;INIT SIPAR3
9837  046544  012737  000001  177572  MOV     #1,@#177572             ;TURN MMU ON
9838  046552  012746  125252          MOV     #125252,-(SP)           ;PUSH BACKGROUND DATA ON TO THE STACK
9839  046556  006671  000000          MTPI    @0(R1)                  ;WRITE DATA TO PHYSICAL ADDRESS
9840  046562  006531                  MFPI    @(R1)+                  ;WRITE DATA AT PHYSICAL ADDRESS TO STACK
9841  046564  022726  125252          CMP     #125252,(SP)+           ;IS DATA EQUAL TO EXPECTED
9842  046570  001403                  BEQ     2$                      ;YES GO ON
9843  046572  005037  177572          CLR     @#177572                ;TURN MMU OFF
9844  046576  104002                  ERROR   +2                      ;MMU ERROR
9845                                                                  ;NOT EQUAL GO TO ERROR
9846  046600  005037  177572  2$:     CLR     @#177572                ;TURN MMU OFF
9847  046604  021427  000333          CMP     (R4),#333               ;ARE WE DONE
9848  046610  001353                  BNE     1$                      ;NO GO TO 1$
9849  046612  012704  047150          MOV     #PARVA2,R4              ;SET POINTERS TO PAR INIT TABLES
9850  046616  012701  047116          MOV     #PARAD2,R1              ;
9851  046622  012431          3$:     MOV     (R4)+,@(R1)+            ;INIT PARS
9852  046624  021127  000333          CMP     (R1),#333               ;ARE WE DONE
9853  046630  001374                  BNE     3$                      ;NO, GO TO 3$
9854  046632  012704  047234          MOV     #MODE2,R4               ;SET POINTERS TO ADDER PART B TABLES
9855  046636  012701  047202          MOV     #VIR2,R1                ;
9856  046642  012702  047266          MOV     #PARVA3,R2              ;
9857  046646  012703  047320          MOV     #VIR3,R3                ;
9858  046652  004767  000076  4$:     JSR     PC,TS14                 ;WRITE DATA TO PHYSICAL ADDRESS AND THEN
9859                                                                  ;CHECK IF DATA AT PHYSICAL ADDRESS IS
9860                                                                  ;EQUAL TO EXPECTED AND IF NOT DETERMINE
9861                                                                  ;IF IT IS AN ADDER ERROR OR A MEMORY ERROR
9862  046656  021127  000111          CMP     (R1),#111               ;HAVE WE DONE ALL THE 22 BIT MODE I SPACE
9863                                                                  ;CASES
9864  046662  001373                  BNE     4$                      ;NO GO TO 4$
9865  046664  005201                  INC     R1                      ;POINT TO 22 BIT MODE D SPACE CASE
9866  046666  005201                  INC     R1                      ;
9867  046670  005204                  INC     R4                      ;
9868  046672  005204                  INC     R4                      ;
9869  046674  012737  000027  172516  MOV     #27,@#172516            ;INIT MMR3
9870  046702  012437  177776          MOV     (R4)+,@#177776          ;INIT PSW
9871  046706  012746  052525          MOV     #52525,-(SP)            ;PUSH DATA ONTO STACK
```

```
9872  046712  012737  000001  177572          MOV     #1,@#177572             ;TURN MMU ON
9873  046720  106631                          MTPD    @(R1)+                  ;WRITE DATA TO PHYSICAL ADDRESS
9874  046722  005037  177572                  CLR     @#177572                ;TURN MMU OFF
9875  046726  012737  000020  172516          MOV     #20,@#172516            ;INIT MMR3
9876  046734  004767  000040                  JSR     PC,T14                  ;CHECK IF DATA AT PHYSICAL ADDRESS IS EQUAL
9877                                                                          ;TO EXPECTED AND IF NOT DETERMINE IF IT
9878                                                                          ;IS AN ADDER ERROR OR A MEMORY ERROR
9879  046740  005037  172516                  CLR     @#172516                ;INIT MMR3 FOR 18 BIT MODE
9880  046744  004767  000004                  JSR     PC,TS14                 ;WRITE DATA TO PHYSICAL ADDRESS AND THEN
9881                                                                          ;CHECK IF DATA AT PHYSICAL ADDRESS IS
9882                                                                          ;EQUAL TO EXPECTED AND IF NOT DETERMINE IF
9883                                                                          ;IT IS AN ADDER ERROR OR A MEMORY ERROR
9884
9885  046750  000167  000376                  JMP     T14FIN
9886                                  ;
9887                                  ;ROUTINE TO WRITE DATA TO PHYSICAL ADDRESS AND TO CHECK IF DATA AT
9888                                  ;PHYSICAL ADDRESS IS EQUAL TO EXPECTED AND IF NOT DETERMINE IF IT IS
9889                                  ;AN ADDER ERROR OR A MEMORY ERROR
9890                                  ;
9891  046754  012437  177776          TS14:   MOV     (R4)+,@#177776          ;INIT PSW
9892  046760  012737  000001  177572          MOV     #1,@#177572             ;TURN MMU ON
9893  046766  012746  052525                  MOV     #52525,-(SP)            ;WRITE DATA ONTO STACK
9894  046772  006631                          MTPI    @(R1)+                  ;WRITE DATA TO PHYSICAL ADDRESS VIA STACK
9895  046774  005037  177572                  CLR     @#177572                ;TURN MMU OFF
9896  047000  012737  010000  177776  T14:    MOV     #10000,@#177776         ;INIT PSW
9897  047006  012237  172246                  MOV     (R2)+,@#172246          ;INIT SIPAR3
9898  047012  012737  000001  177572          MOV     #1,@#177572             ;TURN MMU ON
9899  047020  006573  000000                  MFPI    @0(R3)                  ;DO RELOCATION
9900  047024  022726  052525                  CMP     #52525,(SP)+            ;IS DATA EQUAL TO EXPECTED
9901  047030  001410                          BEQ     2$                      ;YES GO ON
9902  047032  006573  000000                  MFPI    @0(R3)                  ;WHAT TYPE OF ERROR IS IT
9903  047036  022726  125252                  CMP     #125252,(SP)+           ;
9904  047042  001402                          BEQ     1$                      ;
9905  047044  104002                          ERROR   +2                      ;MMU ERROR
9906                                                                          ;IT IS A MEMORY ERROR
9907  047046  000401                          BR      2$                      ;
9908  047050  104002                  1$:     ERROR   +2                      ;MMU ERROR
9909                                                                          ;IT IS AN ADDER ERROR
9910  047052  005037  177572          2$:     CLR     @#177572                ;TURN MMU OFF
9911  047056  005203                          INC     R3                      ;
9912  047060  005203                          INC     R3                      ;
9913  047062  000207                          RTS     PC                      ;RETURN
9914                                  ;
9915                                  ;NON-EXISTANT MEMORY TRAP ROUTINE
9916                                  ;
9917  047064  005037  177572          NXMTRP: CLR     @#SR0                   ;TURN OFF MMU.
9918  047070  012716  047352                  MOV     #T14FIN,(SP)            ;SET UP STACK WITH RETURN ADDR.
9919  047074  013737  002762  000004          MOV     @#SLOC00,@#4            ;RESTORE LOC 4
9920  047102  013737  002764  000006          MOV     @#SLOC01,@#6            ;RESTORE LOC 6
9921  047110  005037  177766                  CLR     @#177766                ;CLEAR TIME OUT INDICATION FROM
9922                                                                          ;CPU ERROR REGISTER.
9923  047114  000006                          RTT                             ;RETURN FROM TRAP; GO TO NEXT TEST.
9924                                  ;
9925                                  ;ADDER TEST PART B TABLES
9926                                  ;
9927  047116  177646                  PARAD2: .WORD   177646
```

```
 9928  047120  177650                          .WORD   177650
 9929  047122  177652                          .WORD   177652
 9930  047124  172240                          .WORD   172240
 9931  047126  177640                          .WORD   177640
 9932  047130  177642                          .WORD   177642
 9933  047132  172244                          .WORD   172244
 9934  047134  177644                          .WORD   177644
 9935  047136  172252                          .WORD   172252
 9936  047140  172352                          .WORD   172352
 9937  047142  177662                          .WORD   177662
 9938  047144  172242                          .WORD   172242
 9939  047146  000333                          .WORD   333
 9940  047150  157700                PARVA2:   .WORD   157700
 9941  047152  137700                          .WORD   137700
 9942  047154  077700                          .WORD   077700
 9943  047156  176777                          .WORD   176777
 9944  047160  007600                          .WORD   007600
 9945  047162  167700                          .WORD   167700
 9946  047164  175700                          .WORD   175700
 9947  047166  177425                          .WORD   177425
 9948  047170  177220                          .WORD   177220
 9949  047172  173700                          .WORD   173700
 9950  047174  176700                          .WORD   176700
 9951  047176  077400                          .WORD   077400
 9952  047200  000333                          .WORD   333
 9953  047202  070000                VIR2:     .WORD   070000
 9954  047204  110000                          .WORD   110000
 9955  047206  130000                          .WORD   130000
 9956  047210  000000                          .WORD   000000
 9957  047212  000000                          .WORD   000000
 9958  047214  030000                          .WORD   030000
 9959  047216  050000                          .WORD   050000
 9960  047220  052524                          .WORD   052524
 9961  047222  136000                          .WORD   136000
 9962  047224  130000                          .WORD   130000
 9963  047226  000111                          .WORD   111
 9964  047230  030000                          .WORD   030000
 9965  047232  030000                          .WORD   030000
 9966  047234  030000                MODE2:    .WORD   030000
 9967  047236  030000                          .WORD   030000
 9968  047240  030000                          .WORD   030000
 9969  047242  010000                          .WORD   010000
 9970  047244  030000                          .WORD   030000
 9971  047246  030000                          .WORD   030000
 9972  047250  010000                          .WORD   010000
 9973  047252  030000                          .WORD   030000
 9974  047254  010000                          .WORD   010000
 9975  047256  000000                          .WORD   000000
 9976  047260  000111                          .WORD   111
 9977  047262  030000                          .WORD   030000
 9978  047264  010000                          .WORD   010000
 9979  047266  160000                PARVA3:   .WORD   160000
 9980  047270  140000                          .WORD   140000
 9981  047272  100000                          .WORD   100000
 9982  047274  176770                          .WORD   176770
 9983  047276  007600                          .WORD   007600
```

```
 9984  047300  170000                          .WORD   170000
 9985  047302  176000                          .WORD   176000
 9986  047304  177552                          .WORD   177552
 9987  047306  177400                          .WORD   177400
 9988  047310  174000                          .WORD   174000
 9989  047312  177000                          .WORD   177000
 9990  047314  007500                          .WORD   007500
 9991  047316  000333                          .WORD   333
 9992  047320  060000                  VIR3:   .WORD   060000
 9993  047322  060000                          .WORD   060000
 9994  047324  060000                          .WORD   060000
 9995  047326  060700                          .WORD   060700
 9996  047330  060000                          .WORD   060000
 9997  047332  060000                          .WORD   060000
 9998  047334  060000                          .WORD   060000
 9999  047336  060024                          .WORD   060024
10000  047340  060000                          .WORD   060000
10001  047342  060000                          .WORD   060000
10002  047344  060000                          .WORD   060000
10003  047346  060000                          .WORD   060000
10004  047350  000333                          .WORD   333
10005
10006
10007                                  ;
10008  047352                          T14FIN:
10009
10010                                  ;       TEST NON-EXISTANT MEMORY TRAP
10011                                  ;;**************************************************************************
10012                                  ;WE ARE ASSUMING THAT THE NON-EXISTANT MEMORY TIME OUT
10013                                  ;FEATURE IS WORKING SINCE WE CAN'T GUARANTEE THAT
10014                                  ;THE SYSTEM BEING TESTED HAS A NON-EXISTANT MEMORY LOCATION.
10015                                  ;AT THIS TIME WE WILL ATTEMPT TO TEST THE NXM FUNCTION
10016  047352  004767  063556                  JSR     PC,MMU                  ;INIT THE MMU
10017  047356  012737  177400  172354          MOV     #177400,@#KIPAR6        ;SET KIPAR6 TO RELOCATE TO HIGHEST MEMORY
10018  047364  016767  130414  133370          MOV     4,SLOC00                ;SAVE VECTOR
10019  047372  016767  000026  130404          MOV     2$,4                    ;LOAD VEC WITH ADDR OF TRAP HANDLER
10020  047400  052767  000001  130164          BIS     #BIT00,SR0              ;TURN ON THE MMU
10021  047406  005067  130354                  CLR     CPEREG                  ;CLEAR THE CPU ERROR REGISTER
10022  047412  005067  130360                  CLR     PS                      ;CLEAR THE PSW
10023  047416  005737  157776                  TST     @#157776                ;ACCESS PHYSICAL ADDR 17757776
10024  047422  000415                  1$:     BR      NXMFIN                  ;IF IT DOESN'T TRAP WE'LL ASSUME
10025                                                                          ;THAT THIS IA A 4 MEGABYTE SYSTEM
10026                                                                          ;AND GO TO THE NEXT TEST
10027  047424  022767  000040  130334  2$:     CMP     #BIT05,CPEREG           ;IS CPU ERROR REGISTER CORRECT?
10028  047432  001401                          BEQ     3$                      ;
10029  047434  104002                          ERROR   .+2             ;MMU ERROR
10030  047436  022726  047422          3$:     CMP     #1$,(SP)+               ;IS CONTENTS OF STACK CORRECT?
10031  047442  001401                          BEQ     4$                      ;
10032  047444  104002                          ERROR   .+2             ;MMU ERROR
10033  047446  022726  000000          4$:     CMP     #0,(SP)+                ;IS CONTENTS OF STACK CORRECT?
10034  047452  001401                          BEQ     NXMFIN                  ;
10035  047454  104002                          ERROR   .+2             ;MMU ERROR
10036  047456  005067  130110          NXMFIN: CLR     SR0                     ;TURN OFF THE MMU
10037  047462  005067  130300                  CLR     CPEREG                  ;CLEAR THE CPU ERROR REGISTER
10038  047466  016767  133270  130310          MOV     SLOC00,4                ;RESTORE THE VECTOR
10039
```

```
10040
10041  047474                          TSMM15:
10042                                  ;       PAGE WRITTEN BIT TEST
10043  047474  005037  177572                  CLR     @#177572        ;MMU OFF
10044  047500  005067  133272                  CLR     FLAG            ;CLEAR MMU ABORT FLAG
10045  047504  004767  063424                  JSR     PC,MMU          ;INIT MMU
10046  047510  005037  177776                  CLR     @#177776        ;INIT PSW
10047  047514  012704  172300                  MOV     #172300,R4      ;SET POINTER TO KPDRS
10048  047520  004767  000114                  JSR     PC,TS15         ;DO RELOCATIONS AND TEST KIPDRS FOR
10049                                          ;PAGE WRITTEN BIT BEING SET AND IF
10050                                          ;NOT SET GO TO ERROR
10051  047524  004767  000160                  JSR     PC,T15          ;DO RELOCATIONS AND TEST KDPDRS FOR
10052                                          ;PAGE WRITTEN BIT BEING SET AND IF NOT
10053                                          ;SET GO TO ERROR
10054  047530  012737  050000  177776          MOV     #50000,@#177776 ;INIT PSW
10055  047536  012704  172200                  MOV     #172200,R4      ;SET POINTER TO SPDRS
10056  047542  004767  000072                  JSR     PC,TS15         ;DO RELOCATIONS AND TEST SIPDRS FOR
10057                                          ;PAGE WRITTEN BIT BEING SET AND IF NOT
10058                                          ;SET GO TO ERROR
10059  047546  004767  000136                  JSR     PC,T15          ;DO RELOCATIONS AND TEST SDPDRS FOR
10060                                          ;PAGE WRITTEN BIT BEING SET AND IF NOT
10061                                          ;SET GO TO ERROR
10062  047552  005037  177776                  CLR     @#177776        ;INIT PSW TO A KNOWN STATE
10063  047556  012737  170000  177776          MOV     #170000,@#177776 ;INIT PSW
10064  047564  012704  177600                  MOV     #177600,R4      ;SET POINTER TO UPDRS
10065  047570  004767  000044                  JSR     PC,TS15         ;DO RELOCATIONS AND TEST UIPDRS FOR
10066                                          ;PAGE WRITTEN BIT BEING SET AND IF
10067                                          ;NOT SET GO TO ERROR
10068  047574  004767  000110                  JSR     PC,T15          ;DO RELOCATIONS AND TEST UDPDRS FOR
10069                                          ;PAGE WRITTEN BIT BEING SET AND IF NOT
10070                                          ;SET GO TO ERROR
10071  047600  005037  177776                  CLR     @#177776        ;INIT PSW TO A KNOWN STATE
10072  047604  012704  172300                  MOV     #172300,R4      ;SET POINTER TO KPDRS
10073  047610  004767  000152                  JSR     PC,T15A         ;EXPLICITLY WRITE TO KPDRS AND TEST
10074                                          ;FOR PAGE WRITTEN BIT BEING CLEARED
10075                                          ;AND IF NOT CLEARED GO TO ERROR
10076  047614  012704  172200                  MOV     #172200,R4      ;SET POINTER TO SPDRS
10077  047620  004767  000142                  JSR     PC,T15A         ;EXPLICITLY WRITE TO SPDRS AND TEST
10078                                          ;FOR PAGE WRITTEN BIT BEING CLEARED
10079                                          ;AND IF NOT CLEARED GO TO ERROR
10080  047624  012704  177600                  MOV     #177600,R4      ;SET POINTER TO UPDRS
10081  047630  004767  000132                  JSR     PC,T15A         ;EXPLICITLY WRITE TO UPDRS AND TEST
10082                                          ;FOR PAGE WRITTEN BIT BEING CLEARED
10083                                          ;AND IF NOT CLEARED GO TO ERROR
10084
10085  047634  000167  000154                  JMP     T15FIN
10086                                  ;
10087                                  ;ROUTINE TO DO RELOCATIONS AND TEST IPDRS FOR PAGE WRITTEN BIT BEING
10088                                  ;SET AND IF NOT SET REPORT AN ERROR
10089                                  ;
10090  047640  005001                  TS15:   CLR     R1              ;SET POINTER TO VIRTUAL ADDRESS
10091  047642  012737  000020  172516          MOV     #20,@#172516    ;INIT MMR3
10092  047650  012737  000001  177572  1$:     MOV     #1,@#177572     ;TURN MMU ON
10093  047656  011111                          MOV     (R1),(R1)       ;DO A RELOCATION
10094  047660  005037  177572                  CLR     @#177572        ;TURN MMU OFF
10095  047664  022427  077506                  CMP     (R4)+,#77506    ;IS DATA EQUAL TO EXPECTED
```

```
10096  047670  001401                        BEQ     2$                      ;OK GO ON
10097  047672  104002                        ERROR   .2              ;MMU ERROR
10098                                                                 ;NO GO TO ERROR
10099  047674  062701  020000        2$:     ADD     #20000,R1               ;POINT TO NEXT VIRTUAL ADDRESS
10100  047700  020127  160000              CMP     R1,#160000              ;ARE WE DONE
10101  047704  001361                      BNE     1$                      ;NO GO TO 1$
10102  047706  000207                      RTS     PC                      ;RETURN
10103                                  ;
10104                                  ;ROUTINE TO DO RELOCATIONS AND TEST DPDRS FOR PAGE WRITTEN BIT BEING SET
10105                                  ;AND IF NOT SET REPORT AN ERROR
10106                                  ;
10107  047710  005001                T15:    CLR     R1                      ;SET POINTER TO VIRTUAL ADDRESS
10108  047712  062704  000002              ADD     #2,R4                   ;POINT TO FIRST DPDR
10109  047716  012737  000027  172516      MOV     #27,@#172516            ;INIT MMR3
10110  047724  012737  000001  177572 1$:   MOV     #1,@#177572             ;TURN MMU ON
10111  047732  011146                      MOV     (R1),-(SP)              ;PUSH DATA ONTO THE STACK
10112  047734  106611                      MTPD    (R1)                    ;DO A RELOCATION
10113  047736  005037  177572              CLR     @#177572                ;TURN MMU OFF
10114  047742  022427  077506              CMP     (R4)+,#77506            ;IS DATA EQUAL TO EXPECTED
10115  047746  001401                      BEQ     2$                      ;OK GO ON
10116  047750  104002                      ERROR   .2              ;MMU ERROR
10117                                                                 ;NO GO TO ERROR
10118  047752  062701  020000        2$:     ADD     #20000,R1               ;POINT TO NEXT VIRTUAL ADDRESS
10119  047756  020127  160000              CMP     R1,#160000              ;ARE WE DONE
10120  047762  001360                      BNE     1$                      ;NO GO TO 1$
10121  047764  000207                      RTS     PC                      ;RETURN
10122                                  ;
10123                                  ;ROUTINE TO EXPLICITLY WRITE TO PDRS AND TEST PAGE WRITTEN BIT FOR BEING
10124                                  ;CLEARED AND IF NOT CLEARED REPORT AN ERROR
10125                                  ;
10126  047766  005002                T15A:   CLR     R2                      ;CLEAR COUNTER
10127  047770  011414                1$:     MOV     (R4),(R4)               ;DO AN EXPLICIT WRITE TO PDR
10128  047772  022427  077406              CMP     (R4)+,#77406            ;IS DATA EQUAL TO EXPECTED
10129  047776  001401                      BEQ     2$                      ;OK GO ON
10130  050000  104002                      ERROR   .2              ;MMU ERROR
10131                                                                 ;NO GO TO ERROR
10132  050002  005202                2$:     INC     R2                      ;INCREMENT POINTER
10133  050004  020227  000020              CMP     R2,#20                  ;ARE WE DONE
10134  050010  001367                      BNE     1$                      ;NO GO TO 1$
10135  050012  000207                      RTS     PC                      ;RETURN
10136  050014                        T15FIN:
10137                                  ;
10138  050014                        TSMM16:
10139                                  ;       TEST CSM (CALL SUPERVISOR MODE)
10140  050014  005037  177572              CLR     @#177572                ;MMU OFF
10141  050020  005037  002776              CLR     @#FLAG                  ;CLEAR MMU ABORT FLAG
10142  050024  012704  050344              MOV     #TMM16E,R4              ;INIT R4
10143  050030  004767  063100              JSR     PC,MMU                  ;INIT MMU
10144  050034  012737  000037  172516      MOV     #37,@#172516            ;ENABLE CSM INSTRUCTION
10145  050042  005037  177776              CLR     @#177776                ;SET PS TO KER MODE
10146  050046  013746  000010              MOV     @#10,-(SP)              ;SAVE VECTORS
10147  050052  013746  000014              MOV     @#14,-(SP)              ;
10148  050056  013746  000016              MOV     @#16,-(SP)              ;
10149  050062  012737  050234  000010      MOV     #TMM16B,@#10            ;SETUP NEW VECTORS
10150  050070  012737  000137  000014      MOV     #137,@#14               ;
10151  050076  012737  050354  000016      MOV     #TMM16A,@#16            ;
```

```
10152  050104  007014                          .WORD   7014                    ; TEST INSTRUCTION
10153  050106  104002                          ERROR   .2              ;MMU ERROR
10154                                                                  ;GO TO ERROR IF NOT TRAPPED
10155  050110  012737  050264  000010  TSM16A: MOV     #TMM16C,@#10            ;SETUP NEW VECTOR
10156  050116  012737  000027  172516          MOV     #27,@#172516            ;DISABLE CSM INSTRUCTION
10157  050124  012737  140000  177776          MOV     #140000,@#177776        ;SET PS TO USER MODE
10158  050132  007014                          .WORD   7014                    ; TEST INSTRUCTION
10159  050134  104002                          ERROR   .2              ;MMU ERROR
10160                                                                  ;GO TO ERROR IF NOT TRAPPED
10161  050136  012737  050314  000010  TSM16B: MOV     #TMM16D,@#10            ;SETUP NEW VECTOR
10162  050144  012737  000027  172516          MOV     #27,@#172516            ;DISABLE CSM INSTRUCTION
10163  050152  005037  177776                  CLR     @#177776                ;SET PS TO KER MODE
10164  050156  007014                          .WORD   7014                    ; TEST INSTRUCTION
10165  050160  104002                          ERROR   .2              ;MMU ERROR
10166                                                                  ;GO TO ERROR IF NOT TRAPPED
10167  050162  012737  000037  172516  TSM16C: MOV     #37,@#172516            ;ENABLE CSM INSTRUCTION
10168  050170  012737  040000  177776          MOV     #40000,@#177776         ;SET PS TO SUP MODE
10169  050176  012706  000700                  MOV     #700,R6                 ;INIT SUP SP
10170  050202  012737  140000  177776          MOV     #140000,@#177776        ;SET PS TO USER MODE
10171  050210  012706  000600                  MOV     #600,R6                 ;INIT USER SP
10172  050214  012737  000014  000010          MOV     #14,@#10                ;SETUP NEW VECTOR
10173  050222  000277                          SCC                             ;SET ALL CC BITS
10174  050224  007024                          .WORD   7024                    ; TEST INSTRUCTION
10175  050226  104002                  TSM16D: ERROR   .2              ;MMU ERROR
10176                                                                  ;GO TO ERROR IF NOT TRAPPED
10177  050230  000167  000504                  JMP     TM16A
10178                                  ;
10179                                  ;
10180  050234  042737  007777  177776  TMM16B: BIC     #7777,@#177776          ;CLEAR UNWANTED BITS
10181  050242  022737  000000  177776          CMP     #0,@#177776             ;IS PS CORRECT
10182  050250  001401                          BEQ     1$                      ;YES GO ON
10183  050252  104002                          ERROR   .2              ;MMU ERROR
10184                                                                  ;NO GO TO ERROR
10185  050254  005726                  1$:     TST     (SP)+                   ;CLEAN UP STACK
10186  050256  005726                          TST     (SP)+                   ;
10187  050260  000167  177624                  JMP     TSM16A                  ;CONTINUE TESTING
10188  050264  042737  007777  177776  TMM16C: BIC     #7777,@#177776          ;CLEAR UNWANTED BITS
10189  050272  022737  030000  177776          CMP     #30000,@#177776         ;IS PS CORRECT
10190  050300  001401                          BEQ     1$                      ;YES GO ON
10191  050302  104002                          ERROR   .2              ;MMU ERROR
10192                                                                  ;NO GO TO ERROR
10193  050304  005726                  1$:     TST     (SP)+                   ;CLEAN UP STACK
10194  050306  005726                          TST     (SP)+                   ;
10195  050310  000167  177622                  JMP     TSM16B                  ;CONTINUE TESTING
10196  050314  042737  007777  177776  TMM16D: BIC     #7777,@#177776          ;CLEAR UNWANTED BITS
10197  050322  022737  000000  177776          CMP     #0,@#177776             ;IS PS CORRECT
10198  050330  001401                          BEQ     1$                      ;YES GO ON
10199  050332  104002                          ERROR   .2              ;MMU ERROR
10200                                                                  ;NO GO TO ERROR
10201  050334  005726                  1$:     TST     (SP)+                   ;CLEAN UP STACK
10202  050336  005726                          TST     (SP)+                   ;
10203  050340  000167  177616                  JMP     TSM16C                  ;CONTINUE TESTING
10204  050344  156430                  TMM16E: .WORD   156430                  ; TEST LOCATION
10205  050346  104002                  TMM16F: ERROR   .2              ;MMU ERROR
10206                                                                  ;GO TO ERROR IF DIDN'T ABORT
10207  050350  000167  000364                  JMP     TM16A
```

```
10208  050354  022737  070017  177776  TMM16A: CMP     #70017,@#177776         ;IS PS CORRECT
10209  050362  001401                          BEQ     1$                      ;YES GO ON
10210  050364  104002                          ERROR   .2              ;MMU ERROR
10211                                                                  ;NO GO TO ERROR
10212  050366  020627  000572          1$:     CMP     R6,#572                 ;IS SP CORRECT
10213  050372  001401                          BEQ     2$                      ;YES GO ON
10214  050374  104002                          ERROR   .2              ;MMU ERROR
10215                                                                  ;NO GO TO ERROR
10216  050376  020427  050346          2$:     CMP     R4,#TMM16E+2            ;IS R4 CORRECT
10217  050402  001401                          BEQ     3$                      ;YES GO ON
10218  050404  104002                          ERROR   .2              ;MMU ERROR
10219                                                                  ;NO GO TO ERROR
10220  050406  023727  050344  156430  3$:     CMP     @#TMM16E,#156430        ;IS TEST LOCATION OK
10221  050414  001401                          BEQ     4$                      ;YES GO ON
10222  050416  104002                          ERROR   .2              ;MMU ERROR
10223                                                                  ;NO GO TO ERROR
10224  050420  022627  156430          4$:     CMP     (SP)+,#156430           ;IS STACK CORRECT
10225  050424  001401                          BEQ     5$                      ;YES GO ON
10226  050426  104002                          ERROR   .2              ;MMU ERROR
10227                                                                  ;NO GO TO ERROR
10228  050430  022627  050226          5$:     CMP     (SP)+,#TSM16D           ;IS STACK CORRECT
10229  050434  001401                          BEQ     6$                      ;YES GO ON
10230  050436  104002                          ERROR   .2              ;MMU ERROR
10231                                                                  ;NO GO TO ERROR
10232  050440  022627  140000          6$:     CMP     (SP)+,#140000           ;IS STACK CORRECT
10233  050444  001401                          BEQ     7$                      ;YES GO ON
10234  050446  104002                          ERROR   .2              ;MMU ERROR
10235                                                                  ;NO GO TO ERROR
10236  050450  012706  000700          7$:     MOV     #700,R6                 ;RESTORE SUP SP
10237  050454  012737  140000  177776          MOV     #140000,@#177776        ;SET PS TO USER MODE
10238  050462  020627  000600                  CMP     R6,#600                 ;IS USER SP CORRECT
10239  050466  001401                          BEQ     8$                      ;YES GO ON
10240  050470  104002                          ERROR   .2              ;MMU ERROR
10241                                                                  ;NO GO TO ERROR
10242  050472  012767  077400  121500  8$:     MOV     #77400,SIPDR0           ;SETUP SIPDR0 TO ABORT
10243  050500  012737  050346  000016          MOV     #TMM16F,@#16            ;SETUP VECTOR
10244  050506  012737  000001  002776          MOV     #1,@#FLAG               ;SETUP FLAG FOR AN ABORT
10245  050514  012737  000001  177572          MOV     #1,@#177572             ;TURN MMU ON
10246  050522  010701                          MOV     R7,R1                   ;SAVE OLD PC
10247  050524  007014                          .WORD   7014                    ; TEST INSTRUCTION
10248  050526  022737  000000  002776          CMP     #0,@#FLAG               ;DID AN ABORT OCCUR
10249  050534  001401                          BEQ     9$                      ;YES GO ON
10250  050536  104002                          ERROR   .2              ;MMU ERROR
10251                                                                  ;NO GO TO ERROR
10252  050540  023701  003010          9$:     CMP     @#SAVMR2,R1             ;IS MMR2 CORRECT
10253  050544  001401                          BEQ     10$                     ;YES GO ON
10254  050546  104002                          ERROR   .2              ;MMU ERROR
10255                                                                  ;NO GO TO ERROR
10256  050550  023727  003004  100041  10$:     CMP     @#SAVMR0,#100041        ;IS MMR0 CORRECT
10257  050556  001401                          BEQ     11$                     ;YES GO ON
10258  050560  104002                          ERROR   .2              ;MMU ERROR
10259                                                                  ;NO GO TO ERROR
10260  050562  012737  000037  172516  11$:     MOV     #37,@#172516            ;ENABLE CSM
10261  050570  012737  040000  177776          MOV     #40000,@#177776         ;SET PSW TO SUP
10262  050576  012706  000700                  MOV     #700,R6                 ;SETUP SUP SP
10263  050602  012737  140000  177776          MOV     #140000,@#177776        ;SET PSW TO USE
```

```
10264  050610  012706  000600                    MOV     #600,R6                 ;SETUP USE SP
10265  050614  012737  000014  000010            MOV     #14,@#10                ;SETUP NEW VECTOR
10266  050622  012737  050644  000016            MOV     #TS16,@#16              ;SETUP NEW VECTOR
10267  050630  000277                            SCC                             ;SET ALL CC BITS
10268  050632  007027                            .WORD   7027                    ;TEST INSTRUCTION
10269  050634  045712                            .WORD   45712
10270  050636  104002                  TS16A:    ERROR   .+2             ;MMU ERROR
10271                                                                    ;GO TO ERROR IF DIDN'T TRAP
10272  050640  000167  000074                    JMP     TM16A
10273  050644  022737  070017  177776  TS16:     CMP     #70017,@#177776         ;IS PSW CORRECT
10274  050652  001401                            BEQ     200$                    ;YES GO ON
10275  050654  104002                            ERROR   .+2             ;MMU ERROR
10276                                                                    ;NO GO TO ERROR
10277  050656  020627  000572            200$:   CMP     R6,#572                 ;IS SP CORRECT
10278  050662  001401                            BEQ     201$                    ;YES GO ON
10279  050664  104002                            ERROR   .+2             ;MMU ERROR
10280                                                                    ;NO GO TO ERROR
10281  050666  022627  045712            201$:   CMP     (SP)+,#045712           ;IS STACK CORRECT
10282  050672  001401                            BEQ     202$                    ;YES GO ON
10283  050674  104002                            ERROR   .+2             ;MMU ERROR
10284                                                                    ;NO GO TO ERROR
10285  050676  022627  050636            202$:   CMP     (SP)+,#TS16A            ;IS STACK CORRECT
10286  050702  001401                            BEQ     203$                    ;YES GO ON
10287  050704  104002                            ERROR   .+2             ;MMU ERROR
10288                                                                    ;NO GO TO ERROR
10289  050706  022627  140000            203$:   CMP     (SP)+,#140000           ;IS STACK CORRECT
10290  050712  001401                            BEQ     204$                    ;YES GO ON
10291  050714  104002                            ERROR   .+2             ;MMU ERROR
10292                                                                    ;NO GO TO ERROR
10293  050716  012706  000700            204$:   MOV     #700,R6                 ;RESTORE SUP SP
10294  050722  012737  140000  177776            MOV     #140000,@#177776        ;SET PSW TO USER MODE
10295  050730  020627  000600                    CMP     R6,#600                 ;IS USER SP CORRECT
10296  050734  001401                            BEQ     TM16A                   ;YES GO ON
10297  050736  104002                            ERROR   .+2             ;MMU ERROR
10298                                                                    ;NO GO TO ERROR
10299  050740  005037  177776          TM16A:    CLR     @#177776                ;SET PS TO KER MODE
10300  050744  012637  000016                    MOV     (SP)+,@#16              ;RESTORE VECTORS
10301  050750  012637  000014                    MOV     (SP)+,@#14              ;
10302  050754  012637  000010                    MOV     (SP)+,@#10              ;
10303
10304                                    ;*****************************************************************************
10305                                    .SBTTL  FLOATING POINT TESTS
10306                                    ;*****************************************************************************
10307                                    ;                       BEGIN FLOATING POINT TESTING
10308                                    ;*****************************************************************************
10309                                    ;*****************************************************************************
10310  050760                            MBT1:
10311
10312                                    ;       FPP REGISTER BIT TESTS
10313                                    ;;*************************************************************
10314                                    ;R5=FPP POINTER
10315                                    ;R1=TEMPORARY COUNTER
10316                                    ;R2=POINTER TO EXPECTED DATA
10317                                    ;R3=POINTER TO RECEIVED DATA
10318                                    ;R4=ODD/EVEN COUNTER
10319  050760  170011                            SETD
```

```
050762  005005                  MBT2:   CLR     R5                      ;SETUP FPP ACC POINTER
050764  012702  003040                  MOV     #BTEXP,R2               ;POINT TO TEST DATA
050770  012703  003050                  MOV     #BTRES,R3               ;POINT TO RECEIVED DATA
050774  170400                  MBT2A:  CLRD    AC0                     ;SETUP FPP REGISTER VALUES
050776  174012                          STD     AC0,(R2)                ;CLEAR EXPECTED VALUE
051000  005004                          CLR     R4
051002  170400                  BTG0:   CLRD    AC0                     ;SETUP FPP REGISTER VALUES
051004  170401                          CLRD    AC1
051006  170402                          CLRD    AC2
051010  170403                          CLRD    AC3
051012  170404                          CLRD    AC4
051014  170405                          CLRD    AC5

051016  010501                          MOV     R5,R1                   ;GET FPP AC NUMBER INTO R1
051020  070127  000014                  MUL     #14,R1                  ;ALLOW 10 LOCATIONS FOR OPERATION
051024  062701  051032                  ADD     #MAC0,R1                ;SETUP JMP LOCATION
051030  000111                          JMP     (R1)
051032  172467  132002          MAC0:   LDD     BTEXP,AC0               ;LOAD TEST DATA INTO TEST REGISTER
051036  174067  132006          MAC0A:  STD     AC0,BTRES               ;SAVE TEST RESULT
051042  000167  000074                  JMP     MACE                    ;GET OUT
051046  172567  131766          MAC1:   LDD     BTEXP,AC1               ;LOAD TEST DATA INTO TEST REGISTER
051052  174167  131772                  STD     AC1,BTRES               ;SAVE TEST RESULT
051056  000167  000060                  JMP     MACE                    ;GET OUT
051062  172667  131752          MAC2:   LDD     BTEXP,AC2               ;LOAD TEST DATA INTO TEST REGISTER
051066  174267  131756                  STD     AC2,BTRES               ;SAVE TEST RESULT
051072  000167  000044                  JMP     MACE                    ;GET OUT
051076  172767  131736          MAC3:   LDD     BTEXP,AC3               ;LOAD TEST DATA INTO TEST REGISTER
051102  174367  131742                  STD     AC3,BTRES               ;SAVE TEST RESULT
051106  000167  000030                  JMP     MACE                    ;GET OUT
051112  172467  131722          MAC4:   LDD     BTEXP,AC0
051116  174004                          STD     AC0,AC4                 ;LOAD TEST DATA INTO TEST REGISTER
051120  172404                          LDD     AC4,AC0                 ;SAVE TEST RESULT
051122  000167  177710                  JMP     MAC0A                   ;GET OUT
051126  172467  131706          MAC5:   LDD     BTEXP,AC0               ;LOAD TEST DATA INTO TEST XFER REGISTER
051132  174005                          STD     AC0,AC5                 ;LOAD TEST REGISTER
051134  172405                          LDD     AC5,AC0                 ;STORE RESULT INTO XFER FPP REGISTER
051136  000167  177674                  JMP     MAC0A                   ;GET OUT
051142  026767  131672  131700  MACE:   CMP     BTEXP,BTRES
051150  001014                          BNE     BTER                    ;BRANCH IF REGISTER ERROR
051152  026767  131664  131672          CMP     BTEXP+2,BTRES+2
051160  001010                          BNE     BTER
051162  026767  131656  131664          CMP     BTEXP+4,BTRES+4
051170  001004                          BNE     BTER
051172  026767  131650  131656          CMP     BTEXP+6,BTRES+6
051200  001401                          BEQ     MBT8                    ;GOOD RESULT
051202  104003                  BTER:   ERROR   +3                ;FPP ERROR
                                                                  ;FPP AC LOADED INCORRECTLY
                                ;NOW VERIFY THE OTHER REGISTERS REMAINED ZERO
051204                          MBT8:
051204  005001                          CLR     R1                      ;CLEAR TEMPORARY COUNTER
051206  005705                          TST     R5                      ;SEE IF R0 UNDER TEST
051210  001411                          BEQ     MBT8A                   ;BRANCH IF TEST ING R0
051212  020527  000004                  CMP     R5,#4                   ;SEE IF TESTING FPP REGISTER >R4
051216  100006                          BPL     MBT8A                   ;SKIP R0 TESTING
051220  174067  131624                  STD     AC0,BTRES               ;SAVE AC TEST RESULT
051224  004767  000216                  JSR     R7,BTTST                ;VERIFY THAT CONTENTS REMAINED ZERO
```

```
10376  051230  001401                    BEQ     MBT8A           ;BRANCH IF EXPECTED RESULT
10377  051232  104003                    ERROR   .3             ;FPP ERROR
10378                                                          ;BAD AC0
10379  051234  020527  000001  MBT8A:    CMP     R5,#1           ;SEE IF R1 UNDER TEST
10380  051240  001406                    BEQ     MBT8B           ;BRANCH IF R1 UNDER TEST
10381  051242  174167  131602            STD     AC1,BTRES       ;SAVE AC TEST RESULT
10382  051246  004767  000174            JSR     R7,BTTST        ;VERIFY THAT CONTENTS REMAINED ZERO
10383  051252  001401                    BEQ     MBT8B           ;BRANCH IF GOOD
10384  051254  104003                    ERROR   .3             ;FPP ERROR
10385                                                          ;BAD AC1
10386  051256  020527  000002  MBT8B:    CMP     R5,#2           ;SEE IF TESTING FPP REGISTER AC2
10387  051262  001406                    BEQ     MBT8C           ;BRANCH IF R2 UNDER TEST
10388  051264  174267  131560            STD     AC2,BTRES       ;SAVE AC TEST RESULT
10389  051270  004767  000152            JSR     R7,BTTST        ;VERIFY THAT CONTENTS REMAINED ZERO
10390  051274  001401                    BEQ     MBT8C           ;BRANCH IF GOOD
10391  051276  104003                    ERROR   .3             ;FPP ERROR
10392                                                          ;BAD AC2
10393  051300  020527  000003  MBT8C:    CMP     R5,#3           ;SEE IF R3 UNDER TEST
10394  051304  001406                    BEQ     MBT8D           ;BRANCH IF R3 UNDER TEST
10395  051306  174367  131536            STD     AC3,BTRES       ;SAVE AC TEST RESULT
10396  051312  004767  000130            JSR     R7,BTTST        ;VERIFY THAT CONTENTS REMAINED ZERO
10397  051316  001401                    BEQ     MBT8D           ;BRANCH IF GOOD
10398  051320  104003                    ERROR   .3             ;FPP ERROR
10399                                                          ;BAD AC3
10400  051322  020527  000004  MBT8D:    CMP     R5,#4           ;SEE IF R4 UNDER TEST
10401  051326  001407                    BEQ     MBT8E           ;BRANCH IF R4 UNDER TEST
10402  051330  172404                    LDD     AC4,AC0         ;MOVE REGISTER CONTENT
10403  051332  174067  131512            STD     AC0,BTRES       ;SAVE AC TEST RESULT
10404  051336  004767  000104            JSR     R7,BTTST        ;VERIFY THAT CONTENTS REMAINED ZERO
10405  051342  001401                    BEQ     MBT8E           ;BRANCH IF GOOD
10406  051344  104003                    ERROR   .3             ;FPP ERROR
10407                                                          ;BAD AC4
10408  051346  020527  000005  MBT8E:    CMP     R5,#5           ;SEE IF R0 UNDER TEST
10409  051352  001407                    BEQ     MBT8F           ;BRANCH IF R0 UNDER TEST
10410  051354  172405                    LDD     AC5,AC0         ;MOVE REGISTER CONTENTS
10411  051356  174067  131466            STD     AC0,BTRES       ;SAVE AC TEST RESULT
10412  051362  004767  000060            JSR     R7,BTTST        ;VERIFY THAT CONTENTS REMAINED ZERO
10413  051366  001401                    BEQ     MBT8F           ;BRANCH IF GOOD
10414  051370  104003                    ERROR   .3             ;FPP ERROR
10415                                                          ;BAD AC5
10416  051372  005204  MBT8F:    INC     R4              ;INCREMENT PATTERN COUNTER
10417  051374  000241                    CLC
10418  051376  042704  177776            BIC     #177776,R4      ; TEST FOR ODD /EVEN
10419  051402  001401                    BEQ     MBT8FG          ;BRANCH IF EVEN
10420  051404  000261                    SEC                     ;SET CARRY FOR TEST PATTERN SHIFT
10421  051406  006112  MBT8FG:  ROL     (R2)            ;ROTATE LSW OF TEST PATTERN
10422  051410  006162  000002            ROL     2(R2)           ;ROTATE 2 WORD OF TEST PATTERN
10423  051414  006162  000004            ROL     4(R2)           ;ROTATE 3 WORD OF TEST PATTERN
10424  051420  006162  000006            ROL     6(R2)           ;ROTATE 4 WORD OF TEST PATTERN
10425  051424  103402                    BCS     MBT8I           ;JUMP IF THROUGH WITH TEST PATTERN
10426  051426  000167  177350            JMP     BTGO            ;CONTINUE WITH NEW TEST PATTERN
10427
10428  051432  005205  MBT8I:    INC     R5              ;GO TO NEXT REGISTER TEST
10429  051434  020527  000006            CMP     R5,#6           ;SEE IF THROUGH TESTING
10430  051440  100016                    BPL     MBTE            ;JUMP IF THROUGH
10431  051442  000167  177326            JMP     MBT2A           ;CONTINUE TESTING WITH NEW PATTERN
```

```
10432                                   ;
10433                                   ;
10434  051446  005767  131376  BTTST:   TST     BTRES           ;VERIFY CONTENTS AS ZERO
10435  051452  001010                   BNE     BTTSTE          ;EXIT IF NOT ZERO
10436  051454  005767  131372           TST     BTRES+2         ;VERIFY CONTENTS AS ZERO
10437  051460  001005                   BNE     BTTSTE          ;EXIT IF NOT ZERO
10438  051462  005767  131366           TST     BTRES+4         ;VERIFY CONTENTS AS ZERO
10439  051466  001002                   BNE     BTTSTE          ;EXIT IF NOT ZERO
10440  051470  005767  131362           TST     BTRES+6         ;VERIFY CONTENTS AS ZERO
10441  051474  000207           BTTSTE: RTS     R7              ;GO BACK TO CALLING ROUTINE
10442
10443
10444                                           ;
10445                                           ;
10446  051476                   MBTE:
10447
10448                                   ;
10449  051476                   MFACU:
10450
10451                                   ;       TEST UNIQUENESS OF FPP ACCUMULATORS
10452                                   ;;*****************************************************************
10453                                   ;THIS TEST LOADS UNIQUE PATTERNS INTO EACH ACCUMULATOR SIMULTANEOUSLY.
10454                                   ;R2=POINTER TO EXPECTED DATA
10455                                   ;R3=POINTER TO RECEIVED DATA
10456                                           ;
10457                                           ;
10458                                           ;
10459  051476  170011                           SETD
10460  051500  005000           MFA:    CLR     R0                      ;SETUP FPP ACC POINTER
10461  051502  005004                   CLR     R4
10462  051504  012702  003040                   MOV     #BTEXP,R2               ;POINT TO TEST DATA
10463  051510  012703  003050                   MOV     #BTRES,R3               ;POINT TO RECEIVED DATA
10464  051514  012767  000051  131316           MOV     #51,BTEXP               ;SETUP EXPECTED DATA
10465  051522  012767  000052  131312           MOV     #52,BTEXP+2             ;
10466  051530  012767  000053  131306           MOV     #53,BTEXP+4             ;
10467  051536  012767  000054  131302           MOV     #54,BTEXP+6             ;
10468  051544  172467  131270                   LDD     BTEXP,AC0               ;MOVE DATA TEMPORARILY
10469  051550  174005                           STD     AC0,AC5                 ;PUT DATA INTO TEST REGISTER
10470  051552  004567  000210                   JSR     R5,SUBT                 ;SUBTRACT TEN FROM EACH EXPECTED DATA
10471  051556  172467  131256                   LDD     BTEXP,AC0               ;MOVE DATA TEMPORARILY
10472  051562  174004                           STD     AC0,AC4                 ;MOVE DATA INTO TEST REGISTER
10473  051564  004567  000176                   JSR     R5,SUBT                 ;SUBTRACT 10 FROM TEST DATA WORDS
10474  051570  172767  131244                   LDD     BTEXP,AC3               ;STORE INTO TEST REGISTER
10475  051574  004567  000166                   JSR     R5,SUBT                 ;GET NEXT SET OF UNIQUE DATA WORDS
10476  051600  172667  131234                   LDD     BTEXP,AC2               ;STORE INTO TEST REGISTER
10477  051604  004567  000156                   JSR     R5,SUBT                 ;GET NEXT SET OF TEST DATAS
10478  051610  172567  131224                   LDD     BTEXP,AC1               ;LOAD TEST REGISTER
10479  051614  004567  000146                   JSR     R5,SUBT                 ;GET NEXT SET OF TEST WORDS
10480  051620  172467  131214                   LDD     BTEXP,AC0               ;LOAD FINAL TEST REGISTER
10481                                           ;ALL REGISTER CONTAIN UNIQUE TEST WORDS
10482  051624  174067  131220                   STD     AC0,BTRES               ;STORE AC0,RESULT
10483  051630  004567  000216                   JSR     R5,BFA                  ;CHECK RESULT
10484  051634  001401                           BEQ     BFAC1                   ;BRANCH IF GOOD
10485  051636  104003                           ERROR   +3              ;FPP ERROR
10486                                                                   ;BAD AC0
10487  051640  004567  000154           BFAC1:  JSR     R5,ADDT                 ;UPDATE EXPECTED RESULT
```

J15


```
10488  051644  174167  131200          STD     AC1,BTRES       ;STORE AC1 RESULT
10489  051650  004567  000176          JSR     R5,BFA          ;CHECK RESULT
10490  051654  001401                  BEQ     BFAC2           ;BRANCH IF GOOD
10491  051656  104003                  ERROR   .+3         ;FPP ERROR
10492                                                      ;BAD RESULT AC1
10493  051660  004567  000134  BFAC2:  JSR     R5,ADDT         ;UPDATE EXPECTED RESULT
10494  051664  174267  131160          STD     AC2,BTRES       ;STORE AC2 RESULT
10495  051670  004567  000156          JSR     R5,BFA          ;CHECK RESULT
10496  051674  001401                  BEQ     BFAC3           ;BRANCH IF GOOD
10497  051676  104003                  ERROR   .+3         ;FPP ERROR
10498                                                      ;BAD AC2 RESLUT
10499  051700  004567  000114  BFAC3:  JSR     R5,ADDT         ;UPDATE EXPECTED RESULT
10500  051704  174367  131140          STD     AC3,BTRES       ;SAVE TEST RESULT
10501  051710  004567  000136          JSR     R5,BFA          ;CHECK RESULT
10502  051714  001401                  BEQ     BFAC4           ;BRANCH IF GOOD
10503  051716  104003                  ERROR   .+3         ;FPP ERROR
10504                                                      ;BAD AC3 RESULT
10505  051720  004567  000074  BFAC4:  JSR     R5,ADDT         ;UPDATE EXPECTED RESULT
10506  051724  172704                  LDD     AC4,AC3         ;SAVE TEMPORARY
10507  051726  174367  131116          STD     AC3,BTRES       ;STORE AC4 RESULT
10508  051732  004567  000114          JSR     R5,BFA          ;CHECK RESULT
10509  051736  001401                  BEQ     BFAC5           ;BRANCH IF GOOD
10510  051740  104003                  ERROR   .+3         ;FPP ERROR
10511                                                      ;BAD AC4 RESULT
10512  051742  004567  000052  BFAC5:  JSR     R5,ADDT         ;UPDATE EXPECTED RESULT
10513  051746  172605                  LDD     AC5,AC2         ;SAVE TEMPORARY COPY
10514  051750  174267  131074          STD     AC2,BTRES       ;MOVE AC5 RESULT
10515  051754  004567  000072          JSR     R5,BFA          ;CHECK RESULT
10516  051760  001454                  BEQ     BFAE            ;BRANCH IF GOOD
10517  051762  104003                  ERROR   .+3         ;FPP ERROR
10518                                                      ;BAD AC5 RESULT
10519  051764  000452                  BR      BFAE            ;EXIT MODULE
10520
10521  051766  162767  000010  131044  SUBT:   SUB     #10,BTEXP           ;UPDATE EXPECTED CONTENTS
10522  051774  162767  000010  131040          SUB     #10,BTEXP+2         ;UPDATE EXPECTED CONTENTS
10523  052002  162767  000010  131034          SUB     #10,BTEXP+4         ;UPDATE EXPECTED CONTENTS
10524  052010  162767  000010  131030          SUB     #10,BTEXP+6         ;UPDATE EXPECTED CONTENTS
10525  052016  000205                  RTS     R5                  ;
10526  052020  062767  000010  131012  ADDT:   ADD     #10,BTEXP           ;UPDATE EXPECTED CONTENTS
10527  052026  062767  000010  131006          ADD     #10,BTEXP+2         ;UPDATE EXPECTED CONTENTS
10528  052034  062767  000010  131002          ADD     #10,BTEXP+4         ;UPDATE EXPECTED CONTENTS
10529  052042  062767  000010  130776          ADD     #10,BTEXP+6         ;UPDATE EXPECTED CONTENTS
10530  052050  000205                  RTS     R5                  ;
10531
10532  052052  026767  130762  130770  BFA:    CMP     BTEXP,BTRES         ;VERIFY CONTENTS
10533  052060  001013                  BNE     BFB                 ;EXIT IF NOT ZERO
10534  052062  026767  130754  130762          CMP     BTEXP+2,BTRES+2     ;VERIFY CONTENTS
10535  052070  001007                  BNE     BFB                 ;EXIT IF NOT ZERO
10536  052072  026767  130746  130754          CMP     BTEXP+4,BTRES+4     ;VERIFY CONTENTS
10537  052100  001003                  BNE     BFB                 ;EXIT IF NOT ZERO
10538  052102  026767  130740  130746          CMP     BTEXP+6,BTRES+6     ;VERIFY CONTENTS
10539  052110  000205                  BFB:    RTS     R5              ;GO BACK TO CALLING ROUTINE
10540
10541
10542  052112                          BFAE:
10543
```

```
10544
10545
10546  052112                          TSFP1:
10547                                  ;       TEST LDFPS AND STFPS MODE 0
10548  052112  005037  133436                  CLR     @#TRPFLG                ;CLEAR TRAP FLAG
10549  052116  012704  147757                  MOV     #147757,R4              ;SETUP DATA TO BE LOADED
10550  052122  004767  000032                  JSR     PC,LOST                 ;LOAD AND STORE FPS WITH DATA
10551  052126  012704  105252                  MOV     #105252,R4              ;SETUP DATA TO BE LOADED
10552  052132  004767  000022                  JSR     PC,LOST                 ;LOAD AND STORE FPS WITH DATA
10553  052136  012704  042505                  MOV     #42505,R4               ;SETUP DATA TO BE LOADED
10554  052142  004767  000012                  JSR     PC,LOST                 ;LOAD AND STORE FPS WITH DATA
10555  052146  005004                          CLR     R4                      ;SETUP DATA TO BE LOADED
10556  052150  004767  000004                  JSR     PC,LOST                 ;LOAD AND STORE FPS WITH DATA
10557
10558                                  ;
10559  052154  000167  000014                  JMP     FIN1
10560                                  ;
10561  052160  170104                  LOST:   LDFPS   R4                      ;LOAD FPS WITH DATA
10562  052162  170201                          STFPS   R1                      ;LOAD R1 WITH (FPS)
10563  052164  020401                          CMP     R4,R1                   ;DID THE INSTRUCTIONS WORK
10564  052166  001401                          BEQ     1$                      ;YES GO ON
10565  052170  104003                          ERROR   +3              ;FPP ERROR
10566                                                                  ;NO GO TO ERROR
10567  052172  000207                  1$:     RTS     PC                      ;RETURN
10568  052174  000240                  FIN1:   NOP
10569                                  ;
10570  052176                          TSFP2:
10571                                  ;       TEST CFCC
10572  052176  005037  133436                  CLR     @#TRPFLG                ;CLEAR TRAP FLAG
10573  052202  012704  000017                  MOV     #17,R4                  ;SETUP DATA TO BE LOADED
10574  052206  004767  000032                  JSR     PC,TSF2                 ;LOAD FPS AND COPY CONDITION CODES TO PS
10575  052212  012704  000012                  MOV     #12,R4                  ;SETUP DATA TO BE LOADED
10576  052216  004767  000022                  JSR     PC,TSF2                 ;LOAD FPS AND COPY CONDITION CODES TO PS
10577  052222  012704  000005                  MOV     #5,R4                   ;SETUP DATA TO BE LOADED
10578  052226  004767  000012                  JSR     PC,TSF2                 ;LOAD FPS AND COPY CONDITION CODES TO PS
10579  052232  005004                          CLR     R4                      ;SETUP DATA TO BE LOADED
10580  052234  004767  000004                  JSR     PC,TSF2                 ;LOAD FPS AND COPY CONDITION CODES TO PS
10581
10582                                  ;
10583  052240  000167  000024                  JMP     FIN2
10584                                  ;
10585  052244  170104                  TSF2:   LDFPS   R4                      ;LOAD FPS
10586  052246  170000                          CFCC                            ;COPY CONDITION CODES TO PS
10587  052250  013701  177776                  MOV     @#177776,R1             ;SAVE PS TO R1
10588  052254  042701  177760                  BIC     #177760,R1              ;MASK OUT UNWANTED BITS
10589  052260  020401                          CMP     R4,R1                   ;WAS CONDITION CODE BITS TRANSFERRED
10590                                                                          ;CORRECTLY
10591  052262  001401                          BEQ     1$                      ;YES GO ON
10592  052264  104003                          ERROR   +3              ;FPP ERROR
10593                                                                  ;NO GO TO ERROR
10594  052266  000207                  1$:     RTS     PC                      ;RETURN
10595  052270                          FIN2:
10596                                  ;
10597  052270                          TSFP3:
10598                                  ;       TEST SETF, SETD, SETI, SETL
10599  052270  005037  133436                  CLR     @#TRPFLG                ;CLEAR TRAP FLAG
```

```
10600  052274  012704  000200          MOV     #200,R4                 ;SETUP DATA TO BE LOADED
10601  052300  170104                  LDFPS   R4                      ;LOAD FPS
10602  052302  170001                  SETF                            ;MAKE FD=0
10603  052304  170201                  STFPS   R1                      ;STORE FPS
10604  052306  020127  000000          CMP     R1,#0                   ;IS FD=0
10605  052312  001401                  BEQ     1$                      ;YES GO ON
10606  052314  104003                  ERROR   .+3                 ;FPP ERROR
10607                                                               ;NO GO TO ERROR
10608  052316  170011          1$:     SETD                            ;MAKE FD=1
10609  052320  170201                  STFPS   R1                      ;STORE FPS
10610  052322  020104                  CMP     R1,R4                   ;IS FD=1
10611  052324  001401                  BEQ     2$                      ;YES GO ON
10612  052326  104003                  ERROR   .+3                 ;FPP ERROR
10613                                                               ;NO GO TO ERROR
10614  052330  012704  000100  2$:     MOV     #100,R4                 ;SETUP DATA TO BE LOADED
10615  052334  170104                  LDFPS   R4                      ;LOAD FPS
10616  052336  170002                  SETI                            ;MAKE FL=0
10617  052340  170201                  STFPS   R1                      ;STORE FPS
10618  052342  020127  000000          CMP     R1,#0                   ;IS FL=0
10619  052346  001401                  BEQ     3$                      ;YES GO ON
10620  052350  104003                  ERROR   .+3                 ;FPP ERROR
10621                                                               ;NO GO TO ERROR
10622  052352  170012          3$:     SETL                            ;MAKE FL=1
10623  052354  170201                  STFPS   R1                      ;STORE FPS
10624  052356  020104                  CMP     R1,R4                   ;IS FL=1
10625  052360  001401                  BEQ     4$                      ;YES GO ON
10626  052362  104003                  ERROR   .+3                 ;FPP ERROR
10627                                                               ;NO GO TO ERROR
10628  052364          4$:
10629                  ;
10630  052364          TSFP4:
10631                  ;       TEST ILLEGAL OP CODES AND STST
10632  052364  005037  133436          CLR     @#TRPFLG                ;CLEAR TRAP FLAG
10633  052370  012705  170003          MOV     #170003,R5              ;INIT OP CODE
10634  052374  013746  000244          MOV     @#244,-(SP)             ;SAVE FP VECTOR
10635  052400  012737  052530  000244  MOV     #ILLOP1,@#244           ;SETUP NEW VECTOR
10636  052406  013746  000004          MOV     @#4,-(SP)               ;SAVE TIME OUT VECTOR
10637  052412  012737  052600  000004  MOV     #TIMEOU,@#4             ;SETUP NEW VECTOR
10638  052420  013746  000010          MOV     @#10,-(SP)              ;SAVE ILLEGAL VECTOR
10639  052424  012737  052604  000010  MOV     #ILLOP2,@#10            ;SETUP NEW VECTOR
10640  052432  005003          D1:     CLR     R3                      ;
10641  052434  170103                  LDFPS   R3                      ;CLEAR FPS
10642  052436  005002                  CLR     R2                      ;
10643  052440  010537  052444          MOV     R5,@#D2                 ;SETUP THE ILLEGAL INST
10644  052444  000000          D2:     .WORD   0                       ;
10645  052446  170000          D3:     CFCC                            ;MEMORY WORDS TO BE USED WITH
10646  052450  005202                  INC     R2                      ;EXECUTION OF ILLEGAL OP CODE
10647  052452  005202                  INC     R2                      ;
10648  052454  170201                  STFPS   R1                      ;SAVE FPS
10649  052456  104003                  ERROR   .+3                 ;FPP ERROR
10650                                                               ;GO TO ERROR
10651  052460  022705  170010  D4:     CMP     #170010,R5              ;COMPUTE NEXT OP CODE
10652  052464  001003                  BNE     D5                      ;
10653  052466  012705  170013          MOV     #170013,R5              ;
10654  052472  000757                  BR      D1                      ;
10655  052474  022705  170077  D5:     CMP     #170077,R5              ;
```

```
10656  052500  001001                    BNE     D6                      ;
10657  052502  000402                    BR      D7                      ;
10658  052504  005205            D6:     INC     R5                      ;
10659  052506  000751                    BR      D1                      ;
10660  052510  012637  000010    D7:     MOV     (SP)+,@#10              ;RESTORE VECTORS
10661  052514  012637  000004            MOV     (SP)+,@#4               ;
10662  052520  012637  000244            MOV     (SP)+,@#244             ;
10663
10664                            ;
10665  052524  000167  000060            JMP     FIN4
10666                            ;
10667  052530  022716  052446    ILLOP1: CMP     #D3,(SP)                ;DID TRAP OCCUR ON TEST INST
10668  052534  001401                    BEQ     1$                      ;YES GO ON
10669  052536  104003                    ERROR   +3              ;FPP ERROR
10670                                                           ;NO GO TO ERROR
10671  052540  022626            1$:     CMP     (SP)+,(SP)+             ;CLEAN UP STACK
10672  052542  170201                    STFPS   R1                      ;STORE FPS
10673  052544  022701  100000            CMP     #100000,R1              ;IS FPS CORRECT
10674  052550  001401                    BEQ     2$                      ;YES GO ON
10675  052552  104003                    ERROR   +3              ;FPP ERROR
10676                                                           ;NO GO TO ERROR
10677  052554  005004            2$:     CLR     R4                      ;INT R4 TO A KNOWN STATE
10678  052556  170304                    STST    R4                      ;STORE FEC AT R4
10679                                                                   ;IF THE DESTINATION MODE IS IMPROPERLY
10680                                                                   ;DECODED AN ODD ADDRESS TRAP TO 4
10681                                                                   ;SHOULD OCCUR
10682  052560  022704  000002            CMP     #2,R4                   ;IS FEC CORRECT
10683  052564  001002                    BNE     3$                      ;NO GO TO ERROR
10684  052566  000167  177666            JMP     D4                      ;YES GO ON
10685  052572  104003            3$:     ERROR   +3              ;FPP ERROR
10686                                                           ;GO TO ERROR
10687  052574  000167  177660            JMP     D4                      ;THEN GO ON
10688                            ;
10689  052600  104003            TIMEOU: ERROR   +3              ;FPP ERROR
10690                                                           ;ERROR BECAUSE OF TRAP TO 4
10691  052602  000006                    RTT                             ;RETURN
10692                            ;
10693  052604  104003            ILLOP2: ERROR   +3              ;FPP ERROR
10694                                                           ;ERROR BECAUSE OF TRAP TO 10
10695  052606  000006                    RTT                             ;RETURN
10696  052610                    FIN4:
10697                            ;
10698  052610                    TSFP5:
10699                            ;       TEST FID (INTERRUPT DISABLE BIT)
10700  052610  005037  133436            CLR     @#TRPFLG                ;CLEAR TRAP FLAG
10701  052614  013746  000244            MOV     @#244,-(SP)             ;SAVE FP VECTOR
10702  052620  012737  052674  000244    MOV     #ILL,@#244              ;SETUP NEW VECTOR
10703  052626  012703  040000            MOV     #40000,R3               ;SETUP DATA TO BE LOADED
10704  052632  170103                    LDFPS   R3                      ;LOAD FPS, FID=1
10705  052634  170020                    .WORD   170020                  ;ILLEGAL FP INSTRUCTION
10706  052636  170000                    CFCC
10707  052640  170201                    STFPS   R1                      ;SEE IF ERROR WAS RECORDED IN FPS
10708  052642  022701  140000            CMP     #140000,R1              ;
10709  052646  001401                    BEQ     1$                      ;YES GO ON
10710  052650  104003                    ERROR   +3              ;FPP ERROR
10711                                                           ;NO GO TO ERROR
```

```
10712  052652  170304                  1$:     STST    R4                      ;SEE IF FEC=2
10713  052654  022704  000002                  CMP     #2,R4                   ;
10714  052660  001401                          BEQ     2$                      ;YES GO ON
10715  052662  104003                          ERROR   .3              ;FPP ERROR
10716                                                                  ;NO GO TO ERROR
10717  052664  012637  000244          2$:     MOV     (SP)+,@#244             ;RESTORE VECTOR
10718
10719                                  ;
10720  052670  000167  000004                  JMP     FIN5
10721                                  ;
10722  052674  104003                  ILL:    ERROR   .3              ;FPP ERROR
10723                                                                  ;FID ERROR
10724  052676  000006                          RTT                             ;RETURN
10725  052700                          FIN5:
10726                                  ;
10727  052700                          TSFP6:
10728                                  ;       TEST LDD, STD FSRC AND FDST MODE 1
10729  052700  005037  133436                  CLR     @#TRPFLG                ;CLEAR TRAP FLAG
10730  052704  005004                          CLR     R4                      ;SETUP TO LOAD DATA
10731  052706  170104                          LDFPS   R4                      ;CLEAR FPS
10732  052710  170011                          SETD                            ;SET FD TO 1
10733  052712  013746  000004                  MOV     @#4,-(SP)               ;SAVE TIMEOUT VECTOR
10734  052716  012737  053050  000004          MOV     #TSF6,@#4               ;SETUP NEW VECTOR
10735  052724  012704  053040                  MOV     #TS6DAT,R4              ;SETUP POINTER TO DATA
10736  052730  172414                          LDD     (R4),AC0                ; TEST INSTRUCTION
10737  052732  020427  053040                  CMP     R4,#TS6DAT              ;IS R4 CORRECT
10738  052736  001401                          BEQ     1$                      ;YES GO ON
10739  052740  104003                          ERROR   .3              ;FPP ERROR
10740                                                                  ;NO GO TO ERROR
10741  052742  012701  053030          1$:     MOV     #TS6DA,R1               ;SETUP POINTER TO DATA
10742  052746  012703  000004                  MOV     #4,R3                   ;INIT COUNTER
10743  052752  022421                  2$:     CMP     (R4)+,(R1)+             ;WAS SOURCE DATA ALTERED
10744  052754  001401                          BEQ     3$                      ;NO GO ON
10745  052756  104003                          ERROR   .3              ;FPP ERROR
10746                                                                  ;YES GO TO ERROR
10747  052760  077304                  3$:     SOB     R3,2$                   ;ARE WE DONE
10748  052762  012704  003122                  MOV     #TSTLOC,R4              ;SETUP POINTER FOR DATA
10749  052766  174014                          STD     AC0,(R4)                ; TEST INSTRUCTION
10750  052770  020427  003122                  CMP     R4,#TSTLOC                      ;IS R4 CORRECT
10751  052774  001401                          BEQ     4$                      ;YES GO ON
10752  052776  104003                          ERROR   .3              ;FPP ERROR
10753                                                                  ;NO GO TO ERROR
10754  053000  012701  053030          4$:     MOV     #TS6DA,R1               ;SETUP POINTER TO DATA
10755  053004  012703  000004                  MOV     #4,R3                   ;INIT COUNTER
10756  053010  022421                  5$:     CMP     (R4)+,(R1)+             ;IS DESTINATION DATA CORRECT
10757  053012  001401                          BEQ     6$                      ;YES GO ON
10758  053014  104003                          ERROR   .3              ;FPP ERROR
10759                                                                  ;NO GO TO ERROR
10760  053016  077304                  6$:     SOB     R3,5$                   ;ARE WE DONE
10761  053020  012637  000004                  MOV     (SP)+,@#4               ;RESTORE VECTOR
10762
10763                                  ;
10764  053024  000167  000024                  JMP     FIN6
10765                                  ;
10766  053030  177777                  TS6DA:  .WORD   177777
10767  053032  000000                          .WORD   000000
```

```
10768  053034  052525                          .WORD   052525
10769  053036  125252                          .WORD   125252
10770  053040  177777                  TS6DAT: .WORD   177777
10771  053042  000000                          .WORD   000000
10772  053044  052525                          .WORD   052525
10773  053046  125252                          .WORD   125252
10774                                  ;
10775  053050  104003                  TSF6:   ERROR   .3              ;FPP ERROR
10776                                                                  ;ODD ADDRESS TRAP
10777  053052  000006                          RTT                             ;RETURN
10778  053054                          FIN6:
10779                                  ;
10780  053054                          TSFP7:
10781                                  ;       TEST LDD, LDF FSRC MODE 0
10782  053054  005037  133436                  CLR     @#TRPFLG                ;CLEAR TRAP FLAG
10783  053060  012704  000200                  MOV     #200,R4                 ;SETUP TO LOAD FPS
10784  053064  170104                          LDFPS   R4                      ;LOAD FPS, FD=1
10785  053066  013746  000004                  MOV     @#4,-(SP)               ;SAVE TIMEOUT VECTOR
10786  053072  012737  053240  000004          MOV     #TSF7,@#4               ;SETUP NEW VECTOR
10787  053100  012704  053244                  MOV     #TS7DA1,R4              ;SETUP POINTER TO DATA
10788  053104  172414                          LDD     (R4),AC0                ;CLEAR AC0
10789  053106  012701  053254                  MOV     #TS7DA2,R1              ;SETUP POINTER TO DATA
10790  053112  172511                          LDD     (R1),AC1                ;LOAD AC1 WITH DATA
10791  053114  172401                          LDD     AC1,AC0                 ; TEST INSTRUCTION
10792  053116  012704  003122                  MOV     #TSTLOC,R4              ;
10793  053122  174114                          STD     AC1,(R4)                ;CHECK IF AC1 HAS BEEN ALTERED
10794  053124  004767  000072                  JSR     PC,CHECK7               ;
10795  053130  012704  003122                  MOV     #TSTLOC,R4                      ;SETUP POINTERS FOR DATA
10796  053134  012701  053254                  MOV     #TS7DA2,R1              ;
10797  053140  174014                          STD     AC0,(R4)                ;CHECK IF AC0 RECEIVED CORRECT DATA
10798  053142  004767  000054                  JSR     PC,CHECK7               ;
10799  053146  012701  053244                  MOV     #TS7DA1,R1              ;SETUP POINTER TO DATA
10800  053152  172511                          LDD     (R1),AC1                ;CLEAR AC1
10801  053154  170001                          SETF                            ;SET FD=0
10802  053156  172401                          LDF     AC1,AC0                 ; TEST INSTRUCTION
10803  053160  170011                          SETD                            ;SET FD=1
10804  053162  012704  003122                  MOV     #TSTLOC,R4              ;SETUP POINTER TO DATA
10805  053166  174114                          STD     AC1,(R4)                ;CHECK IF AC1 HAS BEEN ALTERED
10806  053170  004767  000026                  JSR     PC,CHECK7               ;
10807  053174  012704  053264                  MOV     #TS7DA4,R4              ;SETUP POINTERS FOR DATA
10808  053200  012701  003122                  MOV     #TSTLOC,R1                      ;
10809  053204  174011                          STD     AC0,(R1)                ;CHECK IF AC0 HAS CORRECT DATA
10810  053206  004767  000010                  JSR     PC,CHECK7               ;
10811  053212  012637  000004                  MOV     (SP)+,@#4               ;RESTORE VECTOR
10812
10813                                  ;
10814  053216  000167  000052                  JMP     FIN7
10815                                  ;
10816  053222  012703  000004          CHECK7: MOV     #4,R3                   ;INIT COUNTER
10817  053226  022421                  CHEK7:  CMP     (R4)+,(R1)+             ;IS DATA OK
10818  053230  001401                          BEQ     CHK7                    ;YES GO ON
10819  053232  104003                          ERROR   .3              ;FPP ERROR
10820                                                                  ;NO GO TO ERROR
10821  053234  077304                  CHK7:   SOB     R3,CHEK7                ;ARE WE DONE
10822  053236  000207                          RTS     PC                      ;YES RETURN
10823                                  ;
```

```
10824  053240  104003                  TSF7:   ERROR   +3              ;FPP ERROR
10825                                                                   ;ODD ADDRESS TRAP
10826  053242  000006                          RTI                             ;RETURN
10827                                  ;
10828  053244  000000                  TS7DA1: .WORD   0
10829  053246  000000                          .WORD   0
10830  053250  000000                          .WORD   0
10831  053252  000000                          .WORD   0
10832  053254  037641                  TS7DA2: .WORD   37641
10833  053256  065121                          .WORD   65121
10834  053260  037373                          .WORD   37373
10835  053262  022265                          .WORD   22265
10836  053264  000000                  TS7DA4: .WORD   0
10837  053266  000000                          .WORD   0
10838  053270  037373                          .WORD   37373
10839  053272  022265                          .WORD   22265
10840  053274                          FIN7:
10841                                  ;
10842  053274                          TSFP10:
10843                                  ;       TEST STD, STF FDST MODE 0
10844  053274  005037  133436                  CLR     @#TRPFLG                ;CLEAR TRAP FLAG
10845  053300  012704  000200                  MOV     #200,R4                 ;SETUP TO LOAD FPS
10846  053304  170104                          LDFPS   R4                      ;LOAD FPS, FD=1
10847  053306  013746  000004                  MOV     @#4,-(SP)               ;SAVE TIMEOUT VECTOR
10848  053312  012737  053460  000004          MOV     #TSF10,@#4              ;SETUP NEW VECTOR
10849  053320  012704  053464                  MOV     #TS10D1,R4              ;SETUP POINTER TO DATA
10850  053324  172414                          LDD     (R4),AC0                ;CLEAR AC0
10851  053326  012701  053474                  MOV     #TS10D2,R1              ;SETUP POINTER TO DATA
10852  053332  172511                          LDD     (R1),AC1                ;LOAD AC1 WITH DATA
10853  053334  174100                          STD     AC1,AC0                 ; TEST INSTRUCTION
10854  053336  012704  003122                  MOV     #TSTLOC,R4              ;
10855  053342  174114                          STD     AC1,(R4)                ;CHECK IF AC1 HAS BEEN ALTERED
10856  053344  004767  000072                  JSR     PC,CHEC10               ;
10857  053350  012704  003122                  MOV     #TSTLOC,R4              ;SETUP POINTERS FOR DATA
10858  053354  012701  053474                  MOV     #TS10D2,R1              ;
10859  053360  174014                          STD     AC0,(R4)                ;CHECK IF AC0 RECEIVED CORRECT DATA
10860  053362  004767  000054                  JSR     PC,CHEC10               ;
10861  053366  012701  053464                  MOV     #TS10D1,R1              ;SETUP POINTER TO DATA
10862  053372  172511                          LDD     (R1),AC1                ;CLEAR AC1
10863  053374  170001                          SETF                            ;SET FD=0
10864  053376  174100                          STF     AC1,AC0                 ; TEST INSTRUCTION
10865  053400  170011                          SETD                            ;SET FD=1
10866  053402  012704  003122                  MOV     #TSTLOC,R4              ;SETUP POINTER TO DATA
10867  053406  174114                          STD     AC1,(R4)                ;CHECK IF AC1 HAS BEEN ALTERED
10868  053410  004767  000026                  JSR     PC,CHEC10               ;
10869  053414  012704  053504                  MOV     #TS10D4,R4              ;SETUP POINTERS FOR DATA
10870  053420  012701  003122                  MOV     #TSTLOC,R1              ;
10871  053424  174011                          STD     AC0,(R1)                ;CHECK IF AC0 HAS CORRECT DATA
10872  053426  004767  000010                  JSR     PC,CHEC10               ;
10873  053432  012637  000004                  MOV     (SP)+,@#4               ;RESTORE VECTOR
10874
10875                                  ;
10876  053436  000167  000052                  JMP     FIN10
10877                                  ;
10878  053442  012703  000004          CHEC10: MOV     #4,R3                   ;INIT COUNTER
10879  053446  022421                  CH10:   CMP     (R4)+,(R1)+             ;IS DATA OK
```

```
10880  053450  001401                  BEQ     CHK10                   ;YES GO ON
10881  053452  104003                  ERROR   .3              ;FPP ERROR
10882                                                          ;NO GO TO ERROR
10883  053454  077304          CHK10:  SOB     R3,CH10                 ;ARE WE DONE
10884  053456  000207                  RTS     PC                      ;YES RETURN
10885                          ;
10886  053460  104003          TSF10:  ERROR   .3              ;FPP ERROR
10887                                                          ;ODD ADDRESS TRAP
10888  053462  000006                  RTT                             ;
10889                          ;
10890  053464  000000          TS10D1: .WORD   0
10891  053466  000000                  .WORD   0
10892  053470  000000                  .WORD   0
10893  053472  000000                  .WORD   0
10894  053474  177777          TS10D2: .WORD   177777
10895  053476  111236                  .WORD   111236
10896  053500  100045                  .WORD   100045
10897  053502  003651                  .WORD   3651
10898  053504  000000          TS10D4: .WORD   0
10899  053506  000000                  .WORD   0
10900  053510  100045                  .WORD   100045
10901  053512  003651                  .WORD   3651
10902  053514                  FIN10:
10903                          ;
10904  053514                  TSFP11:
10905                          ;       TEST FDST SINGLE OPERAND MODE 0
10906  053514  005037  133436          CLR     @#TRPFLG                ;CLEAR TRAP FLAG
10907  053520  012704  000200          MOV     #200,R4                 ;SETUP TO LOAD FPS
10908  053524  170104                  LDFPS   R4                      ;SET FD=1
10909  053526  012704  053600          MOV     #TS11D1,R4              ;SETUP POINTER TO DATA
10910  053532  172414                  LDD     (R4),AC0                ;LOAD ALL ONES TO AC0
10911  053534  170400                  CLRD    AC0                     ; TEST INSTRUCTION
10912  053536  170203                  STFPS   R3                      ;GET FPS
10913  053540  012704  003122          MOV     #TSTLOC,R4              ;
10914  053544  174014                  STD     AC0,(R4)                ;CHECK AC0 FOR ALL ZEROES
10915  053546  012701  000004          MOV     #4,R1                   ;INIT COUNTER
10916  053552  022427  000000  1$:     CMP     (R4)+,#0                ;
10917  053556  001401                  BEQ     2$                      ;OK GO ON
10918  053560  104003                  ERROR   .3              ;FPP ERROR
10919                                                          ;NO GO TO ERROR
10920  053562  077105          2$:     SOB     R1,1$                   ;ARE WE DONE
10921  053564  020327  000204          CMP     R3,#204                 ;CHECK FPS
10922  053570  001401                  BEQ     3$                      ;OK GO ON
10923  053572  104003                  ERROR   .3              ;FPP ERROR
10924                                                          ;NO GO TO ERROR
10925  053574                  3$:
10926                          ;
10927  053574  000167  000010          JMP     FIN11
10928                          ;
10929  053600  177777          TS11D1: .WORD   177777
10930  053602  177777                  .WORD   177777
10931  053604  177777                  .WORD   177777
10932  053606  177777                  .WORD   177777
10933  053610                  FIN11:
10934                          ;
10935  053610                  TSFP12:
```

```
10936                                   ;       TEST FDST SOP MODE 0 WITH ILLEGAL AC7
10937  053610  005037  133436                   CLR     @#TRPFLG                ;CLEAR TRAP FLAG
10938  053614  012703  040200                   MOV     #40200,R3               ;SETUP TO LOAD FPS
10939  053620  170103                           LDFPS   R3                      ;SET FID=1, AND FD=1
10940  053622  170407                           CLRD    AC7                     ; TEST INSTRUCTION
10941  053624  170204                           STFPS   R4                      ;GET FPS
10942  053626  170305                           STST    R5                      ;GET FEC
10943  053630  022704  140200                   CMP     #140200,R4              ;IS FPS CORRECT
10944  053634  001401                           BEQ     1$                      ;YES GO ON
10945  053636  104003                           ERROR   .3              ;FPP ERROR
10946                                                                   ;NO GO TO ERROR
10947  053640  022705  000002           1$:     CMP     #2,R5                   ;IS FEC CORRECT
10948  053644  001401                           BEQ     2$                      ;YES GO ON
10949  053646  104003                           ERROR   .3              ;FPP ERROR
10950                                                                   ;NO GO TO ERROR
10951  053650                           2$:
10952                                   ;
10953  053650                           TSFP13:
10954                                   ;       TEST FDST SOP MODE 1
10955  053650  013746  000004                   MOV     @#4,-(SP)               ;SAVE TIMEOUT VECTOR
10956  053654  012737  053770  000004           MOV     #TSF13,@#4              ;SETUP NEW VECTOR
10957  053662  005037  133436                   CLR     @#TRPFLG                ;CLEAR TRAP FLAG
10958  053666  012702  000200                   MOV     #200,R2                 ;SETUP TO LOAD FPS
10959  053672  170102                           LDFPS   R2                      ;SET FD=1
10960  053674  012705  000004                   MOV     #4,R5                   ;INIT COUNTER
10961  053700  012704  003122                   MOV     #TSTLOC,R4              ;SETUP POINTER TO TEST LOCATION
10962  053704  012724  177777           100$:   MOV     #177777,(R4)+           ;MOVE ALL ONES TO TEST LOCATION
10963  053710  077503                           SOB     R5,100$                 ;ARE WE DONE
10964  053712  012702  003122                   MOV     #TSTLOC,R2                      ;SETUP POINTER TO DATA
10965  053716  170412                           CLRD    (R2)                    ; TEST INSTRUCTION
10966  053720  170203                           STFPS   R3                      ;GET FPS
10967  053722  020227  003122                   CMP     R2,#TSTLOC                      ;WAS R2 ALTERED
10968  053726  001401                           BEQ     1$                      ;NO GO ON
10969  053730  104003                           ERROR   .3              ;FPP ERROR
10970                                                                   ;YES GO TO ERROR
10971  053732  012701  000004           1$:     MOV     #4,R1                   ;INIT COUNTER
10972  053736  022227  000000           2$:     CMP     (R2)+,#0                ;CHECK LOCATION FOR 0
10973  053742  001401                           BEQ     3$                      ;OK GO ON
10974  053744  104003                           ERROR   .3              ;FPP ERROR
10975                                                                   ;NO GO TO ERROR
10976  053746  077105           3$:     SOB     R1,2$                   ;ARE WE DONE
10977  053750  020327  000204                   CMP     R3,#204                 ;CHECK FPS
10978  053754  001401                           BEQ     4$                      ;OK GO ON
10979  053756  104003                           ERROR   .3              ;FPP ERROR
10980                                                                   ;NO GO TO ERROR
10981  053760  012637  000004           4$:     MOV     (SP)+,@#4               ;RESTORE VECTOR
10982
10983                                   ;
10984  053764  000167  000004                   JMP     FIN13
10985                                   ;
10986  053770  104003                   TSF13:  ERROR   .3              ;FPP ERROR
10987                                                                   ;ODD ADDRESS TRAP
10988  053772  000006                           RTT                             ;RETURN
10989                                   ;
10990  053774                           FIN13:
10991                                   ;
```

```
10992  053774                                 TSFP14:
10993                                         ;       TEST FDST SOP MODE 2
10994  053774  013746  000004                        MOV     @#4,-(SP)       ;SAVE TIMEOUT VECTOR
10995  054000  012737  054120  000004                MOV     #TSF14,@#4      ;SETUP NEW VECTOR
10996  054006  005037  133436                        CLR     @#TRPFLG        ;CLEAR TRAP FLAG
10997  054012  012702  000200                        MOV     #200,R2         ;SETUP TO LOAD FPS
10998  054016  170102                          LDFPS   R2                      ;SET FD=1
10999  054020  012705  000004                        MOV     #4,R5           ;INIT COUNTER
11000  054024  012704  003122                        MOV     #TSTLOC,R4      ;SETUP POINTER TO TEST LOCATION
11001  054030  012724  177777          100$:   MOV     #177777,(R4)+   ;MOVE ALL ONES TO TEST LOCATION
11002  054034  077503                          SOB     R5,100$         ;ARE WE DONE
11003  054036  012702  003122                        MOV     #TSTLOC,R2              ;SETUP POINTER TO DATA
11004  054042  170422                          CLRD    (R2)+           ; TEST INSTRUCTION
11005  054044  170203                          STFPS   R3              ;GET FPS
11006  054046  020227  003132                        CMP     R2,#TSTLOC+10           ;IS R2 CORRECT
11007  054052  001401                          BEQ     1$              ;YES GO ON
11008  054054  104003                          ERROR   .3              ;FPP ERROR
11009                                                                   ;NO GO TO ERROR
11010  054056  012702  003122          1$:     MOV     #TSTLOC,R2              ;SETUP POINTER TO DATA
11011  054062  012701  000004                        MOV     #4,R1           ;INIT COUNTER
11012  054066  022227  000000          2$:     CMP     (R2)+,#0        ;CHECK LOCATION FOR 0
11013  054072  001401                          BEQ     3$              ;YES GO ON
11014  054074  104003                          ERROR   .3              ;FPP ERROR
11015                                                                   ;NO GO TO ERROR
11016  054076  077105                  3$:     SOB     R1,2$           ;ARE WE DONE
11017  054100  020327  000204                        CMP     R3,#204         ;CHECK FPS
11018  054104  001401                          BEQ     4$              ;OK GO ON
11019  054106  104003                          ERROR   .3              ;FPP ERROR
11020                                                                   ;NO GO TO ERROR
11021  054110  012637  000004          4$:     MOV     (SP)+,@#4               ;RESTORE VECTOR
11022
11023                                         ;
11024  054114  000167  000004                JMP     FIN14
11025                                         ;
11026  054120  104003                  TSF14:  ERROR   .3              ;FPP ERROR
11027                                                                   ;ODD ADDRESS TRAP
11028  054122  000006                          RTT                             ;RETURN
11029                                         ;
11030  054124  000240                  FIN14:  NOP
11031                                         ;
11032                                         ;
11033                                         ;-----------------------------------
11034                                         ;TEST  FDST SOP MODE 3
11035                                         ;
11036                                         ;
11037  054126                                 TSFP15:
11038                                         ;
11039  054126  013746  000004                        MOV     @#4,-(SP)               ;SAVE TIMEOUT VECTOR
11040  054132  012737  054306  000004                MOV     #TSF15,@#4              ;SETUP NEW VECTOR
11041  054140  005037  133436                        CLR     @#TRPFLG                ;CLEAR TRAP FLAG
11042  054144  012702  000200                        MOV     #200,R2                 ;SETUP TO LOAD FPS
11043  054150  170102                          LDFPS   R2                      ;SET FD=1
11044  054152  012705  000011                        MOV     #9.,R5                  ;INIT COUNTER
11045  054156  012704  003122                        MOV     #TSTLOC,R4              ;SETUP POINTER TO TEST LOCATION
11046  054162  012724  177777          100$:   MOV     #177777,(R4)+           ;INIT TEST LOCATION
11047  054166  077503                          SOB     R5,100$                 ;ARE WE DONE
```

```
11048  054170  012737  003134  003122          MOV     #TSTLOC+12,@#TSTLOC             ;INIT TEST LOCATION
11049  054176  012702  003122                  MOV     #TSTLOC,R2                      ;SETUP POINTER TO DATA
11050  054202  170432                          CLRD    @(R2)+                  ; TEST INSTRUCTION
11051  054204  170203                          STFPS   R3                      ;GET FPS
11052  054206  020227  003124                  CMP     R2,#TSTLOC+2                    ;IS R2 CORRECT
11053  054212  001401                          BEQ     1$                      ;YES GO ON
11054  054214  104003                          ERROR   +3              ;FPP ERROR
11055                                                                   ;NO GO TO ERROR
11056  054216  012702  003122          1$:     MOV     #TSTLOC,R2                      ;SETUP POINTER TO DATA
11057  054222  022227  003134                  CMP     (R2)+,#TSTLOC+12                ;IS DATA CORRECT
11058  054226  001401                          BEQ     2$                      ;YES GO ON
11059  054230  104003                          ERROR   +3              ;FPP ERROR
11060                                                                   ;NO GO TO ERROR
11061  054232  012701  000004          2$:     MOV     #4,R1                           ;INIT COUNTER
11062  054236  022227  177777          3$:     CMP     (R2)+,#177777                   ;IS LOCATION ALL ONES
11063  054242  001401                          BEQ     4$                      ;YES GO ON
11064  054244  104003                          ERROR   +3              ;FPP ERROR
11065                                                                   ;NO GO TO ERROR
11066  054246  077105          4$:     SOB     R1,3$                           ;ARE WE DONE
11067  054250  012701  000004                  MOV     #4,R1                           ;INIT COUNTER
11068  054254  022227  000000          5$:     CMP     (R2)+,#0                        ;IS LOCATION 0
11069  054260  001401                          BEQ     6$                      ;YES GO ON
11070  054262  104003                          ERROR   +3              ;FPP ERROR
11071                                                                   ;NO GO TO ERROR
11072  054264  077105          6$:     SOB     R1,5$                           ;ARE WE DONE
11073  054266  020327  000204                  CMP     R3,#204                         ;CHECK FPS
11074  054272  001401                          BEQ     7$                      ;OK GO ON
11075  054274  104003                          ERROR   +3              ;FPP ERROR
11076                                                                   ;NO GO TO ERROR
11077  054276  012637  000004          7$:     MOV     (SP)+,@#4                       ;RESTORE VECTOR
11078
11079                                  ;
11080  054302  000167  000004                  JMP     FIN15
11081                                  ;
11082  054306  104003                  TSF15:  ERROR   +3              ;FPP ERROR
11083                                                                   ;ODD ADDRESS TRAP
11084  054310  000006                          RTT                             ;RETURN
11085                                  ;
11086  054312  000240                  FIN15:  NOP
11087                                  ;
11088                                  ;
11089                                  ;-----------------------------------
11090                                  ;TEST  FDST SOP MODE 4
11091                                  ;
11092                                  ;
11093  054314                          TSFP16:
11094                                  ;
11095  054314  013746  000004                  MOV     @#4,-(SP)                       ;SAVE TIMEOUT VECTOR
11096  054320  012737  054452  000004          MOV     #TSF16,@#4                      ;SETUP NEW VECTOR
11097  054326  005037  133436                  CLR     @#TRPFLG                        ;CLEAR TRAP FLAG
11098  054332  012702  000200                  MOV     #200,R2                         ;SETUP TO LOAD FPS
11099  054336  170102                          LDFPS   R2                              ;SET FD=1
11100  054340  012705  000010                  MOV     #8.,R5                          ;INIT COUNTER
11101  054344  012704  003122                  MOV     #TSTLOC,R4                      ;SETUP POINTER TO TEST LOCATION
11102  054350  012724  177777          100$:   MOV     #177777,(R4)+                   ;INIT TEST LOCATION
11103  054354  077503                          SOB     R5,100$                         ;ARE WE DONE
```

```
11104  054356  012702  003132            MOV     #TSTLOC+10,R2           ;SETUP POINTER TO DATA
11105  054362  170442                    CLRD    -(R2)           ; TEST INSTRUCTION
11106  054364  170203                    STFPS   R3              ;GET FPS
11107  054366  020227  003122            CMP     R2,#TSTLOC              ;IS R2 CORRECT
11108  054372  001401                    BEQ     1$              ;YES GO ON
11109  054374  104003                    ERROR   .3              ;FPP ERROR
11110                                                           ;NO GO TO ERROR
11111  054376  012701  000004    1$:     MOV     #4,R1           ;INIT COUNTER
11112  054402  022227  000000    2$:     CMP     (R2)+,#0        ;IS LOCATION 0
11113  054406  001401                    BEQ     3$              ;YES GO ON
11114  054410  104003                    ERROR   .3              ;FPP ERROR
11115                                                           ;NO GO TO ERROR
11116  054412  077105            3$:     SOB     R1,2$           ;ARE WE DONE
11117  054414  012701  000004            MOV     #4,R1           ;INIT COUNTER
11118  054420  022227  177777    4$:     CMP     (R2)+,#177777   ;IS LOCATION UNCHANGED
11119  054424  001401                    BEQ     5$              ;YES GO ON
11120  054426  104003                    ERROR   .3              ;FPP ERROR
11121                                                           ;NO GO TO ERROR
11122  054430  077105            5$:     SOB     R1,4$           ;ARE WE DONE
11123  054432  020327  000204            CMP     R3,#204         ;CHECK FPS
11124  054436  001401                    BEQ     6$              ;OK GO ON
11125  054440  104003                    ERROR   .3              ;FPP ERROR
11126                                                           ;NO GO TO ERROR
11127  054442  012637  000004    6$:     MOV     (SP)+,@#4       ;RESTORE VECTOR
11128
11129                              ;
11130  054446  000167  000004            JMP     FIN16
11131                              ;
11132  054452  104003            TSF16:  ERROR   .3              ;FPP ERROR
11133                                                           ;ODD ADDRESS TRAP
11134  054454  000006                    RTT                     ;RETURN
11135                              ;
11136  054456  000240            FIN16:  NOP
11137                              ;
11138                              ;
11139                              ;-------------------------------
11140                              ;TEST  FDST SOP MODE 5
11141                              ;
11142                              ;
11143  054460                    TSFP17:
11144                              ;
11145  054460  013746  000004            MOV     @#4,-(SP)               ;SAVE TIMEOUT VECTOR
11146  054464  012737  054634  000004    MOV     #TSF17,@#4              ;SETUP NEW VECTOR
11147  054472  005037  133436            CLR     @#TRPFLG                ;CLEAR TRAP FLAG
11148  054476  012702  000200            MOV     #200,R2                 ;SETUP TO LOAD FPS
11149  054502  170102                    LDFPS   R2                      ;SET FD=1
11150  054504  012705  000011            MOV     #9.,R5                  ;INIT COUNTER
11151  054510  012704  003122            MOV     #TSTLOC,R4              ;SETUP POINTER TO TEST LOCATION
11152  054514  012724  177777    100$:   MOV     #177777,(R4)+           ;INIT TEST LOCATION
11153  054520  077503                    SOB     R5,100$                 ;ARE WE DONE
11154  054522  012737  003134  003122    MOV     #TSTLOC+12,@#TSTLOC             ;INIT TEST LOCATION
11155  054530  012702  003124            MOV     #TSTLOC+2,R2            ;SETUP POINTER TO DATA
11156  054534  170452                    CLRD    @-(R2)          ; TEST INSTRUCTION
11157  054536  170203                    STFPS   R3              ;GET FPS
11158  054540  020227  003122            CMP     R2,#TSTLOC              ;IS R2 CORRECT
11159  054544  001401                    BEQ     1$              ;YES GO ON
```

```
11160  054546  104003                          ERROR   ;3              ;FPP ERROR
11161                                                                  ;NO GO TO ERROR
11162  054550  022227  003134          1$:     CMP     (R2)+,#TSTLOC+12        ;IS DATA CORRECT
11163  054554  001401                          BEQ     2$                      ;YES GO ON
11164  054556  104003                          ERROR   ;3              ;FPP ERROR
11165                                                                  ;NO GO TO ERROR
11166  054560  012701  000004          2$:     MOV     #4,R1                   ;INIT COUNTER
11167  054564  022227  177777          3$:     CMP     (R2)+,#177777           ;IS LOCATION ALL ONES
11168  054570  001401                          BEQ     4$                      ;YES GO ON
11169  054572  104003                          ERROR   ;3              ;FPP ERROR
11170                                                                  ;NO GO TO ERROR
11171  054574  077105          4$:     SOB     R1,3$                   ;ARE WE DONE
11172  054576  012701  000004                  MOV     #4,R1                   ;INIT COUNTER
11173  054602  022227  000000          5$:     CMP     (R2)+,#0                ;IS LOCATION 0
11174  054606  001401                          BEQ     6$                      ;YES GO ON
11175  054610  104003                          ERROR   ;3              ;FPP ERROR
11176                                                                  ;NO GO TO ERROR
11177  054612  077105          6$:     SOB     R1,5$                   ;ARE WE DONE
11178  054614  020327  000204                  CMP     R3,#204                 ;CHECK FPS
11179  054620  001401                          BEQ     7$                      ;OK GO ON
11180  054622  104003                          ERROR   ;3              ;FPP ERROR
11181                                                                  ;NO GO TO ERROR
11182  054624  012637  000004          7$:     MOV     (SP)+,@#4               ;RESTORE VECTOR
11183
11184                                  ;
11185  054630  000167  000004                  JMP     FIN17
11186                                  ;
11187  054634  104003                  TSF17:  ERROR   ;3              ;FPP ERROR
11188                                                                  ;ODD ADDRESS TRAP
11189  054636  000006                          RTT                             ;RETURN
11190                                  ;
11191  054640  000240                  FIN17:  NOP
11192                                  ;
11193                                  ;
11194                                  ;---------------------------------
11195                                  ;TEST  FDST SOP MODE 6
11196                                  ;
11197                                  ;
11198  054642                          TSFP20:
11199                                  ;
11200  054642  005037  133436                  CLR     @#TRPFLG                ;CLEAR TRAP FLAG
11201  054646  013746  000004                  MOV     @#4,-(SP)               ;SAVE TIMEOUT VECTOR
11202  054652  012737  055006  000004          MOV     #TSF20,@#4              ;SETUP NEW VECTOR
11203  054660  012702  000200                  MOV     #200,R2                 ;SETUP TO LOAD FPS
11204  054664  170102                          LDFPS   R2                      ;SET FD=1
11205  054666  012705  000010                  MOV     #8.,R5                  ;INIT COUNTER
11206  054672  012704  003122                  MOV     #TSTLOC,R4              ;SETUP POINTER TO TEST LOCATION
11207  054676  012724  177777          100$:   MOV     #177777,(R4)+           ;INIT TEST LOCATION
11208  054702  077503                          SOB     R5,100$                 ;ARE WE DONE
11209  054704  012702  003123                  MOV     #TSTLOC+1,R2                    ;SETUP POINTER TO DATA
11210  054710  170462  000007                  CLRD    7(R2)                   ; TEST INSTRUCTION
11211  054714  170203                          STFPS   R3                      ;GET FPS
11212  054716  020227  003123                  CMP     R2,#TSTLOC+1                    ;IS R2 CORRECT
11213  054722  001401                          BEQ     1$                      ;YES GO ON
11214  054724  104003                          ERROR   ;3              ;FPP ERROR
11215                                                                  ;NO GO TO ERROR
```

```
11216  054726  012702  003122          1$:     MOV     #TSTLOC,R2                      ;SETUP POINTER TO DATA
11217  054732  012701  000004                  MOV     #4,R1                   ;INIT COUNTER
11218  054736  022227  177777          2$:     CMP     (R2)+,#177777           ;IS DATA CORRECT
11219  054742  001401                          BEQ     3$                      ;YES GO ON
11220  054744  104003                          ERROR   .+3             ;FPP ERROR
11221                                                                   ;NO GO TO ERROR
11222  054746  077105                  3$:     SOB     R1,2$                   ;ARE WE DONE
11223  054750  012701  000004                  MOV     #4,R1                   ;INIT COUNTER
11224  054754  022227  000000          4$:     CMP     (R2)+,#0                ;IS DATA CORRECT
11225  054760  001401                          BEQ     5$                      ;YES GO ON
11226  054762  104003                          ERROR   .+3             ;FPP ERROR
11227                                                                   ;NO GO TO ERROR
11228  054764  077105                  5$:     SOB     R1,4$                   ;ARE WE DONE
11229  054766  020327  000204                  CMP     R3,#204                 ;IS FPS CORRECT
11230  054772  001401                          BEQ     6$                      ;YES GO ON
11231  054774  104003                          ERROR   .+3             ;FPP ERROR
11232                                                                   ;NO GO TO ERROR
11233  054776  012637  000004          6$:     MOV     (SP)+,@#4               ;RESTORE VECTOR
11234
11235                                  ;
11236  055002  000167  000004                  JMP     FIN20
11237                                  ;
11238  055006  104003                  TSF20:  ERROR   .+3             ;FPP ERROR
11239                                                                   ;ODD ADDRESS TRAP
11240  055010  000006                          RTT                             ;RETURN
11241                                  ;
11242  055012  000240                  FIN20:  NOP
11243                                  ;
11244                                  ;
11245                                  ;-----------------------------------
11246                                  ;TEST  FDST SOP MODE 7
11247                                  ;
11248                                  ;
11249  055014                          TSFP21:
11250                                  ;
11251  055014  005037  133436                  CLR     @#TRPFLG                ;CLEAR TRAP FLAG
11252  055020  013746  000004                  MOV     @#4,-(SP)               ;SAVE TIMEOUT VECTOR
11253  055024  012737  055176  000004          MOV     #TSF21,@#4              ;SETUP NEW VECTOR
11254  055032  012702  000200                  MOV     #200,R2                 ;SETUP TO LOAD FPS
11255  055036  170102                          LDFPS   R2                      ;SET FD=1
11256  055040  012705  000010                  MOV     #8.,R5                  ;INIT COUNTER
11257  055044  012704  003122                  MOV     #TSTLOC,R4                      ;SETUP POINTER TO TEST LOCATION
11258  055050  012724  177777          100$:   MOV     #177777,(R4)+   ;INIT TEST LOCATION
11259  055054  077503                          SOB     R5,100$                 ;ARE WE DONE
11260  055056  012737  003122  003132          MOV     #TSTLOC,@#TSTLOC+10             ;INIT TEST LOCATION
11261  055064  012702  003125                  MOV     #TSTLOC+3,R2            ;SETUP POINTER TO DATA
11262  055070  170472  000005                  CLRD    @5(R2)                  ; TEST INSTRUCTION
11263  055074  170203                          STFPS   R3                      ;GET FPS
11264  055076  020227  003125                  CMP     R2,#TSTLOC+3                    ;IS R2 CORRECT
11265  055102  001401                          BEQ     1$                      ;YES GO ON
11266  055104  104003                          ERROR   .+3             ;FPP ERROR
11267                                                                   ;NO GO TO ERROR
11268  055106  012702  003122          1$:     MOV     #TSTLOC,R2                      ;SETUP POINTER TO DATA
11269  055112  012701  000004                  MOV     #4,R1                   ;INIT COUNTER
11270  055116  022227  000000          2$:     CMP     (R2)+,#0                ;IS DATA CORRECT
11271  055122  001401                          BEQ     3$                      ;YES GO ON
```

```
11272  055124  104003                          ERROR   .3              ;FPP ERROR
11273                                                                  ;NO GO TO ERROR
11274  055126  077105                  3$:     SOB     R1,2$                   ;ARE WE DONE
11275  055130  022227  003122                  CMP     (R2)+,#TSTLOC           ;IS DATA CORRECT
11276  055134  001401                          BEQ     4$                      ;YES GO ON
11277  055136  104003                          ERROR   .3              ;FPP ERROR
11278                                                                  ;NO GO TO ERROR
11279  055140  012701  000003          4$:     MOV     #3,R1                   ;INIT COUNTER
11280  055144  022227  177777          5$:     CMP     (R2)+,#177777           ;IS DATA CORRECT
11281  055150  001401                          BEQ     6$                      ;YES GO ON
11282  055152  104003                          ERROR   .3              ;FPP ERROR
11283                                                                  ;NO GO TO ERROR
11284  055154  077105                  6$:     SOB     R1,5$                   ;ARE WE DONE
11285  055156  020327  000204                  CMP     R3,#204                 ;CHECK FPS
11286  055162  001401                          BEQ     7$                      ;OK GO ON
11287  055164  104003                          ERROR   .3              ;FPP ERROR
11288                                                                  ;NO GO TO ERROR
11289  055166  012637  000004          7$:     MOV     (SP)+,@#4               ;RESTORE VECTOR
11290
11291                                  ;
11292  055172  000167  000004                  JMP     FIN21
11293                                  ;
11294  055176  104003                  TSF21:  ERROR   .3              ;FPP ERROR
11295                                                                  ;ODD ADDRESS TRAP
11296  055200  000006                          RTT                             ;RETURN
11297                                  ;
11298  055202  000240                  FIN21:  NOP
11299                                  ;
11300                                  ;
11301                                  ;---------------------------------
11302                                  ;TEST  FDST SOP MODE 3 GR7
11303                                  ;
11304                                  ;
11305  055204                          TSFP22:
11306                                  ;
11307  055204  005037  133436                  CLR     @#TRPFLG                ;CLEAR TRAP FLAG
11308  055210  013746  000004                  MOV     @#4,-(SP)               ;SAVE TIME OUT VECTOR
11309  055214  012737  055342  000004          MOV     #TSF22,@#4              ;SETUP NEW VECTOR
11310  055222  012702  000200                  MOV     #200,R2                 ;SETUP TO LOAD FPS
11311  055226  170102                          LDFPS   R2                      ;SET FD=1
11312  055230  012705  000010                  MOV     #8.,R5                  ;INIT COUNTER
11313  055234  012704  003122                  MOV     #TSTLOC,R4              ;SETUP POINTER TO TEST LOCATION
11314  055240  012724  177777          100$:   MOV     #177777,(R4)+           ;INIT TEST LOCATION
11315  055244  077503                          SOB     R5,100$                 ;ARE WE DONE
11316  055246  012737  003132  003122          MOV     #TSTLOC+10,@#TSTLOC             ;INIT TEST LOCATION
11317  055254  170437  003122                  CLRD    @#TSTLOC                        ; TEST INSTRUCTION
11318  055260  170203                          STFPS   R3                      ;GET FPS
11319  055262  012702  003122                  MOV     #TSTLOC,R2                      ;SETUP POINTER TO DATA
11320  055266  012701  000004                  MOV     #4,R1                   ;INIT COUNTER
11321  055272  022227  000000          1$:     CMP     (R2)+,#0                ;IS DATA CORRECT
11322  055276  001401                          BEQ     2$                      ;YES GO ON
11323  055300  104003                          ERROR   .3              ;FPP ERROR
11324                                                                  ;NO GO TO ERROR
11325  055302  077105                  2$:     SOB     R1,1$                   ;ARE WE DONE
11326  055304  012701  000004                  MOV     #4,R1                   ;INIT COUNTER
11327  055310  022227  177777          3$:     CMP     (R2)+,#177777           ;IS DATA CORRECT
```

```
11328  055314  001401                          BEQ     4$                      ;YES GO ON
11329  055316  104003                          ERROR   .3                     ;FPP ERROR
11330                                                                         ;NO GO TO ERROR
11331  055320  077105                  4$:     SOB     R1,3$                   ;ARE WE DONE
11332  055322  020327  000204                  CMP     R3,#204                 ;CHECK FPS
11333  055326  001401                          BEQ     5$                      ;OK GO ON
11334  055330  104003                          ERROR   .3                     ;FPP ERROR
11335                                                                         ;NO GO TO ERROR
11336  055332  012637  000004          5$:     MOV     (SP)+,@#4               ;RESTORE VECTOR
11337
11338                                  ;
11339  055336  000167  000004                  JMP     FIN22
11340                                  ;
11341  055342  104003                  TSF22:  ERROR   .3                     ;FPP ERROR
11342                                                                         ;ODD ADDRESS TRAP
11343  055344  000006                          RTT                             ;RETURN
11344                                  ;
11345  055346  000240                  FIN22:  NOP
11346                                  ;
11347                                  ;
11348                                  ;-----------------------------------
11349                                  ;TEST  FDST SOP MODE 6 GR7
11350                                  ;
11351                                  ;
11352  055350                          TSFP23:
11353                                  ;
11354  055350  005037  133436                  CLR     @#TRPFLG                ;CLEAR TRAP FLAG
11355  055354  013746  000004                  MOV     @#4,-(SP)               ;SAVE TIMEOUT VECTOR
11356  055360  012737  055462  000004          MOV     #TSF23,@#4              ;SETUP NEW VECTOR
11357  055366  012702  000200                  MOV     #200,R2                 ;SETUP TO LOAD FPS
11358  055372  170102                          LDFPS   R2                      ;SET FD=1
11359  055374  012705  000004                  MOV     #4,R5                   ;INIT COUNTER
11360  055400  012704  003122                  MOV     #TSTLOC,R4              ;SETUP POINTER TO TEST LOCATION
11361  055404  012724  177777          100$:   MOV     #177777,(R4)+           ;INIT TEST LOCATION
11362  055410  077503                          SOB     R5,100$                 ;ARE WE DONE
11363  055412  170467  125504                  CLRD    TSTLOC                  ; TEST INSTRUCTION
11364  055416  170203                          STFPS   R3                      ;GET FPS
11365  055420  012701  000004                  MOV     #4,R1                   ;INIT COUNTER
11366  055424  012702  003122                  MOV     #TSTLOC,R2                      ;SETUP POINTER TO DATA
11367  055430  022227  000000          1$:     CMP     (R2)+,#0                ;IS DATA CORRECT
11368  055434  001401                          BEQ     2$                      ;YES GO ON
11369  055436  104003                          ERROR   .3                     ;FPP ERROR
11370                                                                         ;NO GO TO ERROR
11371  055440  077105                  2$:     SOB     R1,1$                   ;ARE WE DONE
11372  055442  020327  000204                  CMP     R3,#204                 ;CHECK FPS
11373  055446  001401                          BEQ     3$                      ;OK GO ON
11374  055450  104003                          ERROR   .3                     ;FPP ERROR
11375                                                                         ;NO GO TO ERROR
11376  055452  012637  000004          3$:     MOV     (SP)+,@#4               ;RESTORE VECTOR
11377
11378                                  ;
11379  055456  000167  000004                  JMP     FIN23
11380                                  ;
11381  055462  104003                  TSF23:  ERROR   .3                     ;FPP ERROR
11382                                                                         ;ODD ADDRESS TRAP
11383  055464  000006                          RTT                             ;
```

```
11384  055466  000240                  FIN23:  NOP
11385                                  ;
11386                                  ;
11387                                  ;---------------------- ----------
11388                                  ;TEST  FDST SOP MODE 7 GR7
11389                                  ;
11390                                  ;
11391  055470                          TSFP24:
11392                                  ;
11393  055470  005037  133436                  CLR     @#TRPFLG                ;CLEAR TRAP FLAG
11394  055474  013746  000004                  MOV     @#4,-(SP)               ;SAVE TIMEOUT VECTOR
11395  055500  012737  055636  000004          MOV     #TSF24,@#4              ;SETUP NEW VECTOR
11396  055506  012702  000200                  MOV     #200,R2                 ;SETUP TO LOAD FPS
11397  055512  170102                          LDFPS   R2                      ;SET FD=1
11398  055514  012705  000010                  MOV     #8.,R5          ;INIT COUNTER
11399  055520  012704  003122                  MOV     #TSTLOC,R4              ;SETUP TEST LOCATION POINTER
11400  055524  012724  177777          100$:   MOV     #177777,(R4)+           ;INIT TEST LOCATION
11401  055530  077503                          SOB     R5,100$                 ;ARE WE DONE
11402  055532  012737  003122  003132          MOV     #TSTLOC,@#TSTLOC+10             ;INIT TEST LOCATION
11403  055540  170477  125366                  CLRD    @TSTLOC+10                      ; TEST INSTRUCTION
11404  055544  170203                          STFPS   R3                      ;GET FPS
11405  055546  012702  003122                  MOV     #TSTLOC,R2                      ;SETUP POINTER TO DATA
11406  055552  012701  000004                  MOV     #4,R1                   ;INIT COUNTER
11407  055556  022227  000000          1$:     CMP     (R2)+,#0                ;IS DATA CORRECT
11408  055562  001401                          BEQ     2$                      ;YES GO ON
11409  055564  104003                          ERROR   +3              ;FPP ERROR
11410                                                                  ;NO GO TO ERROR
11411  055566  077105                  2$:     SOB     R1,1$                   ;ARE WE DONE
11412  055570  022227  003122                  CMP     (R2)+,#TSTLOC           ;IS DATA CORRECT
11413  055574  001401                          BEQ     3$                      ;YES GO ON
11414  055576  104003                          ERROR   +3              ;FPP ERROR
11415                                                                  ;NO GO TO ERROR
11416  055600  012701  000003          3$:     MOV     #3,R1                   ;INIT COUNTER
11417  055604  022227  177777          4$:     CMP     (R2)+,#177777           ;IS DATA CORRECT
11418  055610  001401                          BEQ     5$                      ;YES GO ON
11419  055612  104003                          ERROR   +3              ;FPP ERROR
11420                                                                  ;NO GO TO ERROR
11421  055614  077105                  5$:     SOB     R1,4$                   ;ARE WE DONE
11422  055616  020327  000204                  CMP     R3,#204                 ;CHECK FPS
11423  055622  001401                          BEQ     6$                      ;OK GO ON
11424  055624  104003                          ERROR   +3              ;FPP ERROR
11425                                                                  ;NO GO TO ERROR
11426  055626  012637  000004          6$:     MOV     (SP)+,@#4               ;RESTORE VECTOR
11427
11428                                  ;
11429  055632  000167  000004                  JMP     FIN24
11430                                  ;
11431  055636  104003                  TSF24:  ERROR   +3              ;FPP ERROR
11432                                                                  ;ODD ADDRESS TRAP
11433  055640  000006                          RTT                             ;
11434                                  ;
11435  055642  000240                  FIN24:  NOP
11436                                  ;
11437                                  ;
11438                                  ;---------------------------------
11439                                  ;TEST  CLRF
```

```
                                        ;
                                        ;
055644                                  TSFP25:
                                        ;
055644  005037  133436                          CLR     @#TRPFLG                ;CLEAR TRAP FLAG
055650  005002                                  CLR     R2                      ;SETUP TO LOAD FPS
055652  170102                                  LDFPS   R2                      ;SET FD=0
055654  012705  000004                          MOV     #4,R5                   ;INIT COUNTER
055660  012704  003122                          MOV     #TSTLOC,R4              ;SETUP POINTER TO TEST LOCATION
055664  012724  177777                  100$:   MOV     #177777,(R4)+           ;INIT TEST LOCATION
055670  077503                                  SOB     R5,100$                 ;ARE WE DONE
055672  012702  003122                          MOV     #TSTLOC,R2                      ;SETUP POINTER TO DATA
055676  170422                                  CLRF    (R2)+                   ; TEST INSTRUCTION
055700  170203                                  STFPS   R3                      ;GET FPS
055702  020227  003126                          CMP     R2,#TSTLOC+4                    ;IS R2 CORRECT
055706  001401                                  BEQ     1$                      ;YES GO ON
055710  104003                                  ERROR   +3                      ;FPP ERROR
                                                                                ;NO GO TO ERROR
055712  012702  003122                  1$:     MOV     #TSTLOC,R2                      ;SETUP POINTER TO DATA
055716  012701  000002                          MOV     #2,R1                   ;INIT COUNTER
055722  022227  000000                  2$:     CMP     (R2)+,#0                ;IS DATA CORRECT
055726  001401                                  BEQ     3$                      ;YES GO ON
055730  104003                                  ERROR   +3                      ;FPP ERROR
                                                                                ;NO GO TO ERROR
055732  077105                          3$:     SOB     R1,2$                   ;ARE WE DONE
055734  012701  000002                          MOV     #2,R1                   ;INIT COUNTER
055740  022227  177777                  4$:     CMP     (R2)+,#177777           ;IS DATA CORRECT
055744  001401                                  BEQ     5$                      ;YES GO ON
055746  104003                                  ERROR   +3                      ;FPP ERROR
                                                                                ;NO GO TO ERROR
055750  077105                          5$:     SOB     R1,4$                   ;ARE WE DONE
055752  020327  000004                          CMP     R3,#4                   ;CHECK FPS
055756  001401                                  BEQ     6$                      ;OK GO ON
055760  104003                                  ERROR   +3                      ;FPP ERROR
                                                                                ;NO GO TO ERROR
055762                                  6$:
                                        ;
                                        ;
                                        ;-----------------------------------
                                        ;TEST  TSTF AND TSTD
                                        ;
                                        ;
055762                                  TSFP26:
                                        ;
055762  005037  133436                          CLR     @#TRPFLG                ;CLEAR TRAP FLAG
055766  005004                                  CLR     R4                      ;SETUP TO LOAD FPS
055770  170104                                  LDFPS   R4                      ;SET FD=0
055772  170567  000244                          TSTF    TS26D0                  ; TEST INSTRUCTION
055776  170203                                  STFPS   R3                      ;GET FPS
056000  020327  000004                          CMP     R3,#4                   ;CHECK FPS
056004  001401                                  BEQ     1$                      ;OK GO ON
056006  104003                                  ERROR   +3                      ;FPP ERROR
                                                                                ;NO GO TO ERROR
056010  012704  056242                  1$:     MOV     #TS26D0,R4              ;SETUP POINTERS TO DATA
056014  012702  056272                          MOV     #TS26D3,R2              ;
056020  004767  000200                          JSR     PC,CHEC26               ;CHECK IF DATA IS CORRECT
```

```
11496  056024  170537  056252          TSTF    @#TS26D1        ; TEST INSTRUCTION
11497  056030  170203                  STFPS   R3              ;GET FPS
11498  056032  020327  000010          CMP     R3,#10          ;CHECK FPS
11499  056036  001401                  BEQ     2$              ;OK GO ON
11500  056040  104003                  ERROR   .3      ;FPP ERROR
11501                                  ;NO GO TO ERROR
11502  056042  012704  056252  2$:     MOV     #TS26D1,R4      ;SETUP POINTERS TO DATA
11503  056046  012702  056302          MOV     #TS26D4,R2      ;
11504  056052  004767  000146          JSR     PC,CHEC26       ;CHECK IF DATA IS CORRECT
11505  056056  170567  000200          TSTF    TS26D2          ; TEST INSTRUCTION
11506  056062  170203                  STFPS   R3              ;GET FPS
11507  056064  020327  000000          CMP     R3,#0           ;CHECK FPS
11508  056070  001401                  BEQ     3$              ;OK GO ON
11509  056072  104003                  ERROR   .3      ;FPP ERROR
11510                                  ;NO GO TO ERROR
11511  056074  012704  056262  3$:     MOV     #TS26D2,R4      ;SETUP POINTERS TO DATA
11512  056100  012702  056312          MOV     #TS26D5,R2      ;
11513  056104  004767  000114          JSR     PC,CHEC26       ;CHECK IF DATA IS CORRECT
11514  056110  012704  000200          MOV     #200,R4         ;SETUP TO LOAD FPS
11515  056114  170104                  LDFPS   R4              ;SET FD=1
11516  056116  170537  056242          TSTD    @#TS26D0        ; TEST INSTRUCTION
11517  056122  170203                  STFPS   R3              ;GET FPS
11518  056124  020327  000204          CMP     R3,#204         ;CHECK FPS
11519  056130  001401                  BEQ     4$              ;OK GO ON
11520  056132  104003                  ERROR   .3      ;FPP ERROR
11521                                  ;NO GO TO ERROR
11522  056134  012704  056242  4$:     MOV     #TS26D0,R4      ;SETUP POINTERS TO DATA
11523  056140  012702  056272          MOV     #TS26D3,R2      ;
11524  056144  004767  000054          JSR     PC,CHEC26       ;CHECK IF DATA IS CORRECT
11525  056150  170567  000076          TSTD    TS26D1          ; TEST INSTRUCTION
11526  056154  170203                  STFPS   R3              ;GET FPS
11527  056156  020327  000210          CMP     R3,#210         ;CHECK FPS
11528  056162  001401                  BEQ     5$              ;OK GO ON
11529  056164  104003                  ERROR   .3      ;FPP ERROR
11530                                  ;NO GO TO ERROR
11531  056166  012704  056252  5$:     MOV     #TS26D1,R4      ;SETUP POINTERS TO DATA
11532  056172  012702  056302          MOV     #TS26D4,R2      ;
11533  056176  004767  000022          JSR     PC,CHEC26       ;CHECK IF DATA IS CORRECT
11534  056202  170567  000054          TSTD    TS26D2          ; TEST INSTRUCTION
11535  056206  170203                  STFPS   R3              ;GET FPS
11536  056210  020327  000200          CMP     R3,#200         ;CHECK FPS
11537  056214  001401                  BEQ     6$              ;OK GO ON
11538  056216  104003                  ERROR   .3      ;FPP ERROR
11539                                  ;NO GO TO ERROR
11540  056220          6$:
11541                                  ;
11542  056220  000167  000076          JMP     FIN26
11543                                  ;
11544  056224  012701  000004  CHEC26: MOV     #4,R1           ;INIT COUNTER
11545  056230  022422          1$:     CMP     (R4)+,(R2)+     ;IS DATA CORRECT
11546  056232  001401                  BEQ     2$              ;YES GO ON
11547  056234  104003                  ERROR   .3      ;FPP ERROR
11548                                  ;NO GO TO ERROR
11549  056236  077104          2$:     SOB     R1,1$           ;ARE WE DONE
11550  056240  000207                  RTS     PC              ;RETURN
11551                                  ;
```

```
11552  056242  000177                  TS26D0: .WORD   177
11553  056244  177777                          .WORD   177777
11554  056246  177777                          .WORD   177777
11555  056250  177777                          .WORD   177777
11556  056252  177777                  TS26D1: .WORD   177777
11557  056254  000000                          .WORD   0
11558  056256  000000                          .WORD   0
11559  056260  000000                          .WORD   0
11560  056262  077777                  TS26D2: .WORD   77777
11561  056264  000000                          .WORD   0
11562  056266  000000                          .WORD   0
11563  056270  000000                          .WORD   0
11564  056272  000177                  TS26D3: .WORD   177
11565  056274  177777                          .WORD   177777
11566  056276  177777                          .WORD   177777
11567  056300  177777                          .WORD   177777
11568  056302  177777                  TS26D4: .WORD   177777
11569  056304  000000                          .WORD   0
11570  056306  000000                          .WORD   0
11571  056310  000000                          .WORD   0
11572  056312  077777                  TS26D5: .WORD   77777
11573  056314  000000                          .WORD   0
11574  056316  000000                          .WORD   0
11575  056320  000000                          .WORD   0
11576  056322  000240                  FIN26:  NOP
11577                                  ;
11578                                  ;
11579                                  ;-------------------------------
11580                                  ;TEST  ABSF
11581                                  ;
11582                                  ;
11583  056324                          TSFP27:
11584                                  ;
11585  056324  005037  133436                  CLR     @#TRPFLG                ;CLEAR TRAP FLAG
11586  056330  005005                          CLR     R5                      ;SETUP TO LOAD FPS
11587  056332  170105                          LDFPS   R5                      ;SET FD=0
11588  056334  012701  000014                  MOV     #12.,R1                 ;INIT COUNTER
11589  056340  012704  003122                  MOV     #TSTLOC,R4                      ;SETUP POINTER TO TEST LOCATION
11590  056344  012703  056550                  MOV     #TS27D0,R3              ;SETUP POINTER TO TEST VALUE
11591  056350  012324          100$:   MOV     (R3)+,(R4)+             ;INIT TEST LOCATION
11592  056352  077102                  SOB     R1,100$                 ;ARE WE DONE
11593  056354  012705  003122                  MOV     #TSTLOC,R5                      ;SETUP POINTER TO DATA
11594  056360  170615                          ABSF    (R5)                    ; TEST INSTRUCTION
11595  056362  170203                          STFPS   R3                      ;GET FPS
11596  056364  020527  003122                  CMP     R5,#TSTLOC                      ;IS R5 CORRECT
11597  056370  001401                          BEQ     1$                      ;YES GO ON
11598  056372  104003                          ERROR   .3              ;FPP ERROR
11599                                          ;NO GO TO ERROR
11600  056374  012702  056600          1$:     MOV     #TS27D3,R2              ;SETUP POINTER TO DATA
11601  056400  004767  000126                  JSR     PC,CHEC27               ;CHECK IF DATA IS CORRECT
11602  056404  020327  000000                  CMP     R3,#0                   ;CHECK FPS
11603  056410  001401                          BEQ     2$                      ;OK GO ON
11604  056412  104003                          ERROR   .3              ;FPP ERROR
11605                                          ;NO GO TO ERROR
11606  056414  012705  003132          2$:     MOV     #TSTLOC+10,R5                   ;SETUP POINTER TO DATA
11607  056420  170625                          ABSF    (R5)+                   ; TEST INSTRUCTION
```

```
11608  056422  170203                  STFPS   R3              ;GET FPS
11609  056424  020527  003136          CMP     R5,@TSTLOC+14           ;IS R5 CORRECT
11610  056430  001401                  BEQ     3$              ;YES GO ON
11611  056432  104003                  ERROR   .3      ;FPP ERROR
11612                                  ;NO GO TO ERROR
11613  056434  012705  003132  3$:     MOV     @TSTLOC+10,R5           ;SETUP POINTER TO DATA
11614  056440  012702  056610          MOV     @TS27D4,R2      ;
11615  056444  004767  000062          JSR     PC,CHEC27       ;CHECK IF DATA IS CORRECT
11616  056450  020327  000000          CMP     R3,@0           ;CHECK FPS
11617  056454  001401                  BEQ     4$              ;OK GO ON
11618  056456  104003                  ERROR   .3      ;FPP ERROR
11619                                  ;NO GO TO ERROR
11620  056460  012705  003122  4$:     MOV     @TSTLOC,R5              ;SETUP POINTER TO DATA
11621  056464  170665  000020          ABSF    20(R5)          ; TEST INSTRUCTION
11622  056470  170203                  STFPS   R3              ;GET FPS
11623  056472  020527  003122          CMP     R5,@TSTLOC              ;IS R5 CORRECT
11624  056476  001401                  BEQ     5$              ;YES GO ON
11625  056500  104003                  ERROR   .3      ;FPP ERROR
11626                                  ;NO GO TO ERROR
11627  056502  012705  003142  5$:     MOV     @TSTLOC+20,R5           ;SETUP POINTERS TO DATA
11628  056506  012702  056620          MOV     @TS27D5,R2      ;
11629  056512  004767  000014          JSR     PC,CHEC27       ;CHECK IF DATA IS CORRECT
11630  056516  020327  000004          CMP     R3,@4           ;CHECK FPS
11631  056522  001401                  BEQ     6$              ;OK GO ON
11632  056524  104003                  ERROR   .3      ;FPP ERROR
11633                                  ;NO GO TO ERROR
11634  056526          6$:
11635                                  ;
11636  056526  000167  000076          JMP     FIN27
11637                                  ;
11638  056532  012701  000004  CHEC27: MOV     @4,R1           ;INIT COUNTER
11639  056536  022522          1$:     CMP     (R5)+,(R2)+             ;IS DATA CORRECT
11640  056540  001401                  BEQ     2$              ;YES GO ON
11641  056542  104003                  ERROR   .3      ;FPP ERROR
11642                                  ;NO GO TO ERROR
11643  056544  077104          2$:     SOB     R1,1$           ;ARE WE DONE
11644  056546  000207                  RTS     PC              ;RETURN
11645                                  ;
11646  056550  177777  TS27D0:  .WORD   177777
11647  056552  177777                  .WORD   177777
11648  056554  177777                  .WORD   177777
11649  056556  177777                  .WORD   177777
11650  056560  000377  TS27D1:  .WORD   377
11651  056562  175436                  .WORD   175436
11652  056564  136477                  .WORD   136477
11653  056566  000001                  .WORD   1
11654  056570  000177  TS27D2:  .WORD   177
11655  056572  175436                  .WORD   175436
11656  056574  136477                  .WORD   136477
11657  056576  000001                  .WORD   1
11658  056600  077777  TS27D3:  .WORD   77777
11659  056602  177777                  .WORD   177777
11660  056604  177777                  .WORD   177777
11661  056606  177777                  .WORD   177777
11662  056610  000377  TS27D4:  .WORD   377
11663  056612  175436                  .WORD   175436
```

```
11664  056614  136477                  .WORD   136477
11665  056616  000001                  .WORD   1
11666  056620  000000          TS27D5: .WORD   0
11667  056622  000000                  .WORD   0
11668  056624  136477                  .WORD   136477
11669  056626  000001                  .WORD   1
11670  056630  000240          FIN27:  NOP
11671                          ;
11672                          ;
11673                          ;-------------------------------
11674                          ;TEST  ABSD
11675                          ;
11676                          ;
11677  056632                  TSFP30:
11678                          ;
11679  056632  005037  133436          CLR     @#TRPFLG                ;CLEAR TRAP FLAG
11680  056636  012705  000200          MOV     #200,R5                 ;SETUP TO LOAD FPS
11681  056642  170105                  LDFPS   R5                      ;SET FD=1
11682  056644  012701  000014          MOV     #12.,R1                 ;INIT COUNTER
11683  056650  012704  003122          MOV     #TSTLOC,R4                      ;SETUP POINTER TO TEST LOCATION
11684  056654  012703  057060          MOV     #TS30D0,R3              ;SETUP POINTER TO TEST VALUE
11685  056660  012324          100$:   MOV     (R3)+,(R4)+             ;INIT TEST LOCATION
11686  056662  077102                  SOB     R1,100$                 ;ARE WE DONE
11687  056664  012705  003122          MOV     #TSTLOC,R5                      ;SETUP POINTER TO DATA
11688  056670  170615                  ABSD    (R5)                    ; TEST INSTRUCTION
11689  056672  170203                  STFPS   R3                      ;GET FPS
11690  056674  020527  003122          CMP     R5,#TSTLOC                      ;IS R5 CORRECT
11691  056700  001401                  BEQ     1$                      ;YES GO ON
11692  056702  104003                  ERROR   .3              ;FPP ERROR
11693                                  ;NO GO TO ERROR
11694  056704  012702  057110  1$:     MOV     #TS30D3,R2              ;SETUP POINTER TC DATA
11695  056710  004767  000126          JSR     PC,CHEC30               ;CHECK IF DATA IS CORRECT
11696  056714  020327  000200          CMP     R3,#200                 ;CHECK FPS
11697  056720  001401                  BEQ     2$                      ;OK GO ON
11698  056722  104003                  ERROR   .3              ;FPP ERROR
11699                                  ;NO GO TO ERROR
11700  056724  012705  003132  2$:     MOV     #TSTLOC+10,R5                   ;SETUP POINTER TO DATA
11701  056730  170625                  ABSD    (R5)+                   ; TEST INSTRUCTION
11702  056732  170203                  STFPS   R3                      ;GET FPS
11703  056734  020527  003142          CMP     R5,#TSTLOC+20                   ;IS R5 CORRECT
11704  056740  001401                  BEQ     3$                      ;YES GO ON
11705  056742  104003                  ERROR   .3              ;FPP ERROR
11706                                  ;NO GO TO ERROR
11707  056744  012705  003132  3$:     MOV     #TSTLOC+10,R5                   ;SETUP POINTERS TO DATA
11708  056750  012702  057120          MOV     #TS30D4,R2              ;
11709  056754  004767  000062          JSR     PC,CHEC30               ;CHECK IF DATA IS CORRECT
11710  056760  020327  000200          CMP     R3,#200                 ;CHECK FPS
11711  056764  001401                  BEQ     4$                      ;OK GO ON
11712  056766  104003                  ERROR   .3              ;FPP ERROR
11713                                  ;NO GO TO ERROR
11714  056770  012705  003122  4$:     MOV     #TSTLOC,R5                      ;SETUP POINTER TO DATA
11715  056774  170665  000020          ABSD    20(R5)                  ; TEST INSTRUCTION
11716  057000  170203                  STFPS   R3                      ;GET FPS
11717  057002  020527  003122          CMP     R5,#TSTLOC                      ;IS R5 CORRECT
11718  057006  001401                  BEQ     5$                      ;YES GO ON
11719  057010  104003                  ERROR   .3              ;FPP ERROR
```

```
                                                                        ;NO GO TO ERROR
057012  012705  003142          5$:     MOV     #TSTLOC+20,R5                   ;SETUP POINTERS TO DATA
057016  012702  057130                  MOV     #TS30D5,R2              ;
057022  004767  000014                  JSR     PC,CHEC30               ;CHECK IF DATA IS CORRECT
057026  020327  000204                  CMP     R3,#204                 ;CHECK FPS
057032  001401                          BEQ     6$                      ;OK GO ON
057034  104003                          ERROR   +3                      ;FPP ERROR
                                                                        ;NO GO TO ERROR
057036                          6$:
                                ;
057036  000167  000076                  JMP     FIN30
                                ;
057042  012701  000004          CHEC30: MOV     #4,R1                   ;INIT COUNTER
057046  022522                  1$:     CMP     (R5)+,(R2)+             ;IS DATA CORRECT
057050  001401                          BEQ     2$                      ;YES GO ON
057052  104003                          ERROR   +3                      ;FPP ERROR
                                                                        ;NO GO TO ERROR
057054  077104                  2$:     SOB     R1,1$                   ;ARE WE DONE
057056  000207                          RTS     PC                      ;RETURN
                                ;
057060  177777                  TS30D0: .WORD   177777
057062  177777                          .WORD   177777
057064  177777                          .WORD   177777
057066  177777                          .WORD   177777
057070  000377                  TS30D1: .WORD   377
057072  175436                          .WORD   175436
057074  136477                          .WORD   136477
057076  000001                          .WORD   1
057100  000177                  TS30D2: .WORD   177
057102  175436                          .WORD   175436
057104  136477                          .WORD   136477
057106  000001                          .WORD   1
057110  077777                  TS30D3: .WORD   77777
057112  177777                          .WORD   177777
057114  177777                          .WORD   177777
057116  177777                          .WORD   177777
057120  000377                  TS30D4: .WORD   377
057122  175436                          .WORD   175436
057124  136477                          .WORD   136477
057126  000001                          .WORD   1
057130  000000                  TS30D5: .WORD   0
057132  000000                          .WORD   0
057134  000000                          .WORD   0
057136  000000                          .WORD   0
057140  000240                  FIN30:  NOP
                                ;
                                ;
                                ;-----------------------------------
                                ;TEST  FDST SOP MODE 2 GR7
                                ;
                                ;
057142                          TSFP31:
                                ;
057142  005037  133436                  CLR     @#TRPFLG                ;CLEAR TRAP FLAG
057146  013746  000004                  MOV     @#4,-(SP)               ;SAVE TIMEOUT VECTOR
057152  012737  057254  000004          MOV     #TSF31,@#4              ;SETUP NEW VECTOR
```

```
11776  057160  012702  000200          MOV     #200,R2         ;SETUP TO LOAD FPS
11777  057164  170102                  LDFPS   R2              ;SET FD=1
11778  057166  170527  000005  TSD31:  TSTD    #5              ; TEST INSTRUCTION
11779  057172  000240                  NOP
11780  057174  000240                  NOP
11781  057176  000240                  NOP
11782  057200  170203                  STFPS   R3              ;GET FPS
11783  057202  020327  000204          CMP     R3,#204         ;CHECK FPS
11784  057206  001401                  BEQ     1$              ;OK GO ON
11785  057210  104003                  ERROR   .3      ;FPP ERROR
11786                                  ;NO GO TO ERROR
11787  057212  012702  057170  1$:     MOV     #TSD31+2,R2     ;SETUP POINTER TO DATA
11788  057216  022227  000005          CMP     (R2)+,#5        ;IS DATA CORRECT
11789  057222  001401                  BEQ     2$              ;YES GO ON
11790  057224  104003                  ERROR   .3      ;FPP ERROR
11791                                  ;NO GO TO ERROR
11792  057226  012701  000003  2$:     MOV     #3,R1           ;INIT COUNTER
11793  057232  022227  000240  3$:     CMP     (R2)+,#240      ;IS DATA CORRECT
11794  057236  001401                  BEQ     4$              ;YES GO ON
11795  057240  104003                  ERROR   .3      ;FPP ERROR
11796                                  ;NO GO TO ERROR
11797  057242  077105          4$:     SOB     R1,3$           ;ARE WE DONE
11798  057244  012637  000004          MOV     (SP)+,@#4       ;RESTORE VECTOR
11799
11800                          ;
11801  057250  000167  000004          JMP     FIN31
11802                          ;
11803  057254  104003          TSF31:  ERROR   .3      ;FPP ERROR
11804                                  ;ODD ADDRESS TRAP
11805  057256  000006                  RTT             ;RETURN
11806                          ;
11807  057260  000240          FIN31:  NOP
11808                          ;
11809                          ;
11810                          ;-----------------------------------
11811                          ;TEST  NEGF
11812                          ;
11813                          ;
11814  057262                  TSFP32:
11815                          ;
11816  057262  005037  133436          CLR     @#TRPFLG        ;CLEAR TRAP FLAG
11817  057266  005005                  CLR     R5              ;SETUP TO LOAD FPS
11818  057270  170105                  LDFPS   R5              ;SET FD=0
11819  057272  012701  000014          MOV     #12.,R1         ;INIT COUNTER
11820  057276  012704  003122          MOV     #TSTLOC,R4              ;SETUP POINTER TO TEST LOCATION
11821  057302  012703  057452          MOV     #TS32D0,R3      ;SETUP POINTER TO TEST VALUE
11822  057306  012324          100$:   MOV     (R3)+,(R4)+     ;INIT TEST LOCATION
11823  057310  077102                  SOB     R1,100$         ;ARE WE DONE
11824  057312  170767  123604          NEGF    TSTLOC          ; TEST INSTRUCTION
11825  057316  170203                  STFPS   R3              ;GET FPS
11826  057320  012705  003122          MOV     #TSTLOC,R5              ;SETUP POINTERS TO DATA
11827  057324  012702  057502          MOV     #TS32D3,R2      ;
11828  057330  004767  000100          JSR     PC,CHEC32       ;CHECK IF DATA IS CORRECT
11829  057334  020327  000000          CMP     R3,#0           ;CHECK FPS
11830  057340  001401                  BEQ     1$              ;YES GO ON
11831  057342  104003                  ERROR   .3      ;FPP ERROR
```

```
11832
11833  057344  170767  123562          1$:     NEGF    TSTLOC+10                ;NO GO TO ERROR
11834  057350  170203                          STFPS   R3                               ; TEST INSTRUCTION
11835  057352  012705  003132                  MOV     #TSTLOC+10,R5            ;GET FPS
11836  057356  012702  057512                  MOV     #TS32D4,R2                       ;SETUP POINTERS TO DATA
11837  057362  004767  000046                  JSR     PC,CHEC32                ;
11838  057366  020327  000010                  CMP     R3,#10                   ;CHECK IF DATA IS CORRECT
11839  057372  001401                          BEQ     2$                       ;CHECK FPS
11840  057374  104003                          ERROR   .3                       ;OK GO ON
11841                                                                           ;FPP ERROR
11842  057376  170767  123540          2$:     NEGF    TSTLOC+20                ;NO GO TO ERROR
11843  057402  170203                          STFPS   R3                               ; TEST INSTRUCTION
11844  057404  012705  003142                  MOV     #TSTLOC+20,R5            ;GET FPS
11845  057410  012702  057522                  MOV     #TS32D5,R2                       ;SETUP POINTERS TO DATA
11846  057414  004767  000014                  JSR     PC,CHEC32                ;
11847  057420  020327  000004                  CMP     R3,#4                    ;CHECK IF DATA IS CORRECT
11848  057424  001401                          BEQ     3$                       ;CHECK FPS
11849  057426  104003                          ERROR   .3                       ;OK GO ON
11850                                                                           ;FPP ERROR
11851  057430                          3$:                                      ;NO GO TO ERROR
11852                                  ;
11853  057430  000167  000076                  JMP     FIN32
11854                                  ;
11855  057434  012701  000004          CHEC32: MOV     #4,R1                    ;INIT COUNTER
11856  057440  022522                  1$:     CMP     (R5)+,(R2)+              ;IS DATA CORRECT
11857  057442  001401                          BEQ     2$                       ;YES GO ON
11858  057444  104003                          ERROR   .3                       ;FPP ERROR
11859                                                                           ;NO GO TO ERROR
11860  057446  077104                  2$:     SOB     R1,1$                    ;ARE WE DONE
11861  057450  000207                          RTS     PC                       ;RETURN
11862                                  ;
11863  057452  170000          TS32D0: .WORD   170000
11864  057454  003541                  .WORD   3541
11865  057456  177777                  .WORD   177777
11866  057460  172710                  .WORD   172710
11867  057462  070000          TS32D1: .WORD   70000
11868  057464  003541                  .WORD   3541
11869  057466  177777                  .WORD   177777
11870  057470  172710                  .WORD   172710
11871  057472  000177                  .WORD   177
11872  057474  100000                  .WORD   100000
11873  057476  177777                  .WORD   177777
11874  057500  177007                  .WORD   177007
11875  057502  070000          TS32D3: .WORD   70000
11876  057504  003541                  .WORD   3541
11877  057506  177777                  .WORD   177777
11878  057510  172710                  .WORD   172710
11879  057512  170000          TS32D4: .WORD   170000
11880  057514  003541                  .WORD   3541
11881  057516  177777                  .WORD   177777
11882  057520  172710                  .WORD   172710
11883  057522  000000          TS32D5: .WORD   0
11884  057524  000000                  .WORD   0
11885  057526  177777                  .WORD   177777
11886  057530  177007                  .WORD   177007
11887  057532  000240          FIN32:  NOP
```

```
                                        ;
                                        ;
                                        ;-----------------------------------
                                        ;TEST  NEGD
                                        ;
                                        ;
057534                                  TSFP33:
                                        ;
057534  005037  133436                        CLR     @#TRPFLG                ;CLEAR TRAP FLAG
057540  012705  000200                        MOV     #200,R5                 ;SETUP TO LOAD FPS
057544  170105                              LDFPS   R5                      ;SET FD=1
057546  012701  000014                        MOV     #12.,R1                 ;INIT COUNTER
057552  012704  003122                        MOV     #TSTLOC,R4                      ;SETUP POINTER TO TEST LOCATION
057556  012703  057726                        MOV     #TS33D0,R3              ;SETUP POINTER TO TEST VALUE
057562  012324                        100$:   MOV     (R3)+,(R4)+             ;INIT TEST LOCATION
057564  077102                              SOB     R1,100$                 ;ARE WE DONE
057566  170767  123330                        NEGD    TSTLOC                  ; TEST INSTRUCTION
057572  170203                              STFPS   R3                      ;GET FPS
057574  012705  003122                        MOV     #TSTLOC,R5                      ;SETUP POINTERS TO DATA
057600  012702  057756                        MOV     #TS33D3,R2              ;
057604  004767  000100                        JSR     PC,CHEC33               ;CHECK IF DATA IS CORRECT
057610  020327  000200                        CMP     R3,#200                 ;CHECK FPS
057614  001401                              BEQ     1$                      ;OK GO ON
057616  104003                              ERROR   .3                      ;FPP ERROR
                                                                            ;NO GO TO ERROR
057620  170767  123306                1$:     NEGD    TSTLOC+10                       ; TEST INSTRUCTION
057624  170203                              STFPS   R3                      ;GET FPS
057626  012705  003132                        MOV     #TSTLOC+10,R5                   ;SETUP POINTERS TO DATA
057632  012702  057766                        MOV     #TS33D4,R2              ;
057636  004767  000046                        JSR     PC,CHEC33               ;CHECK IF DATA IS CORRECT
057642  020327  000210                        CMP     R3,#210                 ;CHECK FPS
057646  001401                              BEQ     2$                      ;OK GO ON
057650  104003                              ERROR   .3                      ;FPP ERROR
                                                                            ;NO GO TO ERROR
057652  170767  123264                2$:     NEGD    TSTLOC+20                       ; TEST INSTRUCTION
057656  170203                              STFPS   R3                      ;GET FPS
057660  012705  003142                        MOV     #TSTLOC+20,R5                   ;SETUP POINTERS TO DATA
057664  012702  057776                        MOV     #TS33D5,R2              ;
057670  004767  000014                        JSR     PC,CHEC33               ;CHECK IF DATA IS CORRECT
057674  020327  000204                        CMP     R3,#204                 ;CHECK FPS
057700  001401                              BEQ     3$                      ;OK GO ON
057702  104003                              ERROR   .3                      ;FPP ERROR
                                                                            ;NO GO TO ERROR
057704                                  3$:
                                        ;
057704  000167  000076                        JMP     FIN33
                                        ;
057710  012701  000004                CHEC33: MOV     #4,R1                   ;INIT COUNTER
057714  022522                        1$:     CMP     (R5)+,(R2)+             ;IS DATA CORRECT
057716  001401                              BEQ     2$                      ;YES GO ON
057720  104003                              ERROR   .3                      ;FPP ERROR
                                                                            ;NO GO TO ERROR
057722  077104                        2$:     SOB     R1,1$                   ;ARE WE DONE
057724  000207                              RTS     PC                      ;RETURN
                                        ;
057726  170000                        TS33D0: .WORD   170000
```

```
11944  057730  003541                          .WORD   3541
11945  057732  177777                          .WORD   177777
11946  057734  172710                          .WORD   172710
11947  057736  070000                  TS33D1: .WORD   70000
11948  057740  003541                          .WORD   3541
11949  057742  177777                          .WORD   177777
11950  057744  172710                          .WORD   172710
11951  057746  000177                  TS33D2: .WORD   177
11952  057750  100000                          .WORD   100000
11953  057752  177777                          .WORD   177777
11954  057754  177007                          .WORD   177007
11955  057756  070000                  TS33D3: .WORD   70000
11956  057760  003541                          .WORD   3541
11957  057762  177777                          .WORD   177777
11958  057764  172710                          .WORD   172710
11959  057766  170000                  TS33D4: .WORD   170000
11960  057770  003541                          .WORD   3541
11961  057772  177777                          .WORD   177777
11962  057774  172710                          .WORD   172710
11963  057776  000000                  TS33D5: .WORD   0
11964  060000  000000                          .WORD   0
11965  060002  000000                          .WORD   0
11966  060004  000000                          .WORD   0
11967  060006  000240                  FIN33:  NOP
11968                                  ;
11969                                  ;
11970                                  ;
11971                                  ;-----------------------------------
11972                                  ;TEST LDD MODE 0, ILLEGAL AC7
11973                                  ;
11974                                          ;
11975                                          ;
11976                                          ;
11977  060010                          MFSRCM0:
11978
11979                                  ;
11980  060010  012704  047600                  MOV     #47600,R4               ;SETUP FPP STATUS
11981  060014  170104                          LDFPS   R4                      ;LOAD FP STATUS
11982  060016  012702  003062                  MOV     #RECFEC,R2              ;POINT TO RECEIVED FEC MEMORY
11983  060022  172407                  1$:     LDD     R7,AC0                  ;*TEST INSTRUCTION
11984                                                                          ;LOAD AC0 FROM ILLEGAL AC7
11985  060024  170201                          STFPS   R1                      ;SAVE FPP STATUS
11986  060026  022701  147600                  CMP     #147600,R1              ;VERIFY FER BIT SET
11987  060032  001401                          BEQ     2$                      ;BRANCH IF GOOD ERROR CONDITION
11988  060034  104003                          ERROR   .3                  ;FPP ERROR
11989                                                                      ;THE FER BIT DIDNT SET
11990  060036  170312                  2$:     STST    (R2)                        ;SAVE FEC AND FEA
11991  060040  022722  000002                  CMP     #2,(R2)+                    ;VERIFY FEC CONTENTS
11992  060044  001401                          BEQ     3$                          ;BRANCH IF GOOD
11993  060046  104003                          ERROR   .3                  ;FPP ERROR
11994                                                                      ;FEC NE 2 (OPCODE ERROR)
11995  060050  022722  060022          3$:     CMP     #1$,(R2)+                   ;VERFIY FEA CONTENTS
11996  060054  001401                          BEQ     4$                          ;BRANCH FI GOOD
11997  060056  104003                          ERROR   .3                  ;FPP ERROR
11998                                                                      ;FEA NOT CORRECT ERROR ADDRESS
11999  060060                          4$:
```

```
12000
12001
12002                                   ;
12003                                   ;
12004                                   ;-----------------------------------
12005                                   ;TEST LDD MODE2
12006                                   ;
12007                                           ;
12008                                           ;
12009                                           ;
12010  060060                           MLDDM2:
12011
12012                                   ;
12013  060060  012701  003102                   MOV     #RECDST,R1              ;POINT TO RECEIVED DATA LOCATION
12014  060064  012704  003174                   MOV     #TAB1,R4                ;POINT TO GOOD DATA
12015  060070  012702  047750                   MOV     #47750,R2               ;LOAD GOOD STATUS
12016  060074  170102                           LDFPS   R2                      ;LOAD FPP STATUS - DOUBLE,ID
12017  060076  172424                           LDD     (R4)+,AC0               ;*TEST INSTRUCTION - MODE 2
12018  060100  170203                           STFPS   R3                      ;SAVE TEST FPP STATUS
12019  060102  174021                           STD     AC0,(R1)+               ;SAVE TEST RESULT MODE 2
12020  060104  020203                           CMP     R2,R3                   ;VERIFY FPP STATUS
12021  060106  001401                           BEQ     1$                      ;BRANCH IF GOOD
12022  060110  104003                           ERROR   .3                      ;FPP ERROR
12023                                                                          ;BAD FPP STSTUS
12024  060112  022704  003204           1$:     CMP     #TAB1+10,R4                     ;VERFIY AUTO-INCR
12025  060116  001401                           BEQ     2$                      ;BRANCH IF GOOD
12026  060120  104003                           ERROR   .3                      ;FPP ERROR
12027                                                                          ;BAD AUTO-INCR
12028  060122  012704  003174           2$:     MOV     #TAB1,R4                        ;POINT TO RECEIVED DATA
12029  060126  162701  000010                   SUB     #10,R1                  ;RETURN R1 TO PROPER VALUE
12030  060132  004767  053324                   JSR     R7,DATVER               ;VERFIY DATA FROM FPP
12031  060136  005767  122716                   TST     COUNT                   ;SEE IF COUNTER=0
12032  060142  001401                           BEQ     3$                      ;BRANCH IF GOOD COMPARE
12033  060144  104003                           ERROR   .3                      ;FPP ERROR
12034                                                                          ;BAD DATA FROM FPP
12035  060146                           3$:
12036
12037                                   ;
12038                                   ;
12039                                   ;
12040                                   ;
12041                                   ;-----------------------------------
12042                                   ;TEST LDD MODE 3
12043                                   ;
12044                                           ;
12045                                           ;
12046                                           ;
12047  060146                           MLDDM3:
12048
12049                                   ;
12050  060146  012737  003102  003124           MOV     #RECDST,@#TSTLOC+2              ;POINT TO RECEIVED DATA LOCATION
12051  060154  012701  003124                   MOV     #TSTLOC+2,R1                    ;SETUP STD IN MODE 3
12052  060160  012737  003204  003122           MOV     #TAB2,@#TSTLOC          ;POINT TO DATA TABLE
12053  060166  012704  003122                   MOV     #TSTLOC,R4                      ;POINT TO GOOD DATA
12054  060172  012702  047750                   MOV     #47750,R2               ;LOAD GOOD STATUS
12055  060176  170102                           LDFPS   R2                      ;LOAD FPP STATUS - DOUBLE,ID
```

```
12056  060200  172434                          LDD     @(R4)+,AC0      ;*TEST INSTRUCTION - MODE 2
12057  060202  170203                          STFPS   R3              ;SAVE TEST FPP STATUS
12058  060204  174031                          STD     AC0,@(R1)+      ;SAVE TEST RESULT IN MODE 3
12059  060206  022703  047740                  CMP     #47740,R3       ;VERIFY FPP STATUS
12060  060212  001401                          BEQ     1$              ;BRANCH IF GOOD
12061  060214  104003                          ERROR   .3              ;FPP ERROR
12062                                                                  ;BAD FPP STSTUS
12063  060216  022704  003124          1$:     CMP     #TSTLOC+2,R4    ;VERFIY AUTO-INCR
12064  060222  001401                          BEQ     2$              ;BAD AUTO-DEC ON LDD
12065  060224  104003                          ERROR   .3              ;FPP ERROR
12066                                                                  ;BAD AUTO-INC
12067  060226  022701  003126          2$:     CMP     #TSTLOC+4,R1    ; TEST STD AUTO-INC
12068  060232  001401                          BEQ     3$              ;BRANCH IF GOOD
12069  060234  104003                          ERROR   .3              ;FPP ERROR
12070                                                                  ;BAD AUTO-INCR
12071  060236  012704  003204          3$:     MOV     #TAB2,R4        ;POINT TO RECEIVED DATA
12072  060242  012701  003102                  MOV     #RECDST,R1      ;POINT TO RECEIVED DATA
12073  060246  004767  053210                  JSR     R7,DATVER       ;VERFIY DATA FROM FPP
12074  060252  005767  122602                  TST     COUNT           ;SEE IF COUNTER=0
12075  060256  001401                          BEQ     4$              ;BRANCH IF GOOD COMPARE
12076  060260  104003                          ERROR   .3              ;FPP ERROR
12077                                                                  ;BAD DATA FROM FPP
12078  060262                          4$:
12079
12080                                  ;
12081                                  ;
12082                                  ;
12083                                  ;
12084                                  ;-----------------------------------
12085                                  ;TEST LDF, STD MODE 4
12086                                  ;
12087                                          ;
12088                                          ;
12089                                          ;
12090  060262                          MLDDM4:
12091
12092                                  ;
12093  060262  012701  003106                  MOV     #RECDST+4,R1            ;POINT TO RECEIVED DATA LOCATION
12094  060266  012704  003220                  MOV     #TAB3 4,R4              ;POINT TO GOOD DATA
12095  060272  012705  003254                  MOV     #TAB6,R5                ;CLEAR OUT AC0
12096  060276  170127  000200                  LDFPS   #200                    ;SET TO DOUBLE
12097  060302  172415                          LDD     (R5),AC0                ;AC0=0
12098  060304  012702  047550                  MOV     #47550,R2               ;LOAD GOOD STATUS FLOATING
12099  060310  170102                          LDFPS   R2                      ;LOAD FPP STATUS - DOUBLE,ID
12100  060312  172444                          LDF     -(R4),AC0               ;*TEST INSTRUCTION - MODE 4
12101  060314  170203                          STFPS   R3                      ;SAVE TEST FPP STATUS
12102  060316  012702  047750                  MOV     #47750,R2               ;SET TO DOUBLE MODE
12103  060322  170102                          LDFPS   R2                      ;SET FPP TO DOUBLE
12104  060324  174041                          STD     AC0,-(R1)               ;SAVE TEST RESULT
12105  060326  022703  047540                  CMP     #47540,R3               ;VERIFY FPP STATUS
12106  060332  001401                          BEQ     1$                      ;BRANCH IF GOOD
12107  060334  104003                          ERROR   .3              ;FPP ERROR
12108                                                                  ;BAD FPP STSTUS
12109  060336  022704  003214          1$:     CMP     #TAB3,R4                ;VERFIY AUTO-DEC
12110  060342  001401                          BEQ     2$                      ;BRANCH IF GOOD
12111  060344  104003                          ERROR   .3              ;FPP ERROR
```

N1


```
12112                                                                   ;BAD AUTO-INCR
12113  060346  012704  003214          2$:     MOV     #TAB3,R4                ;POINT TO RECEIVED DATA
12114  060352  004767  053104                  JSR     R7,DATVER       ;VERIFY DATA FROM FPP
12115  060356  005767  122476                  TST     COUNT           ;SEE IF COUNTER=0
12116  060362  001401                          BEQ     3$              ;BRANCH IF GOOD COMPARE
12117  060364  104003                          ERROR   .-3             ;FPP ERROR
12118                                                                   ;BAD DATA FROM FPP
12119  060366                          3$:
12120
12121                                  ;
12122                                  ;
12123                                  ;
12124                                  ;
12125                                  ;-------------------------------
12126                                  ;TEST LDD MODE 5
12127                                  ;
12128                                          ;
12129                                          ;
12130                                          ;
12131  060366                          MLDDM5:
12132
12133                                  ;
12134  060366  012701  003102                  MOV     @RECDST,R1      ;POINT TO RECEIVED DATA LOCATION
12135  060372  012704  003124                  MOV     #TSTLOC-2,R4            ;POINT TO GOOD DATA
12136  060376  012737  003174  003122          MOV     #TAB1,@#TSTLOC          ;SET UP MODE 5 POINTER TO DATA
12137  060404  012702  047750                  MOV     #47750,R2       ;LOAD GOOD STATUS
12138  060410  170102                          LDFPS   R2              ;LOAD FPP STATUS - DOUBLE,ID
12139  060412  172454                          LDD     -(R4),AC0       ;*TEST INSTRUCTION - MODE 5
12140  060414  170203                          STFPS   R3              ;SAVE TEST FPP STATUS
12141  060416  174011                          STD     AC0,(R1)        ;SAVE TEST RESULT
12142  060420  020203                          CMP     R2,R3           ;VERIFY FPP STATUS
12143  060422  001401                          BEQ     1$              ;BRANCH IF GOOD
12144  060424  104003                          ERROR   .-3             ;FPP ERROR
12145                                                                   ;BAD FPP STATUS
12146  060426  022704  003122          1$:     CMP     #TSTLOC,R4              ;VERIFY AUTO-DEC
12147  060432  001401                          BEQ     2$              ;BRANCH IF GOOD
12148  060434  104003                          ERROR   .-3             ;FPP ERROR
12149                                                                   ;BAD AUTO-DEC
12150  060436  012704  003174          2$:     MOV     #TAB1,R4                ;POINT TO EXPECTED DATA
12151  060442  004767  053014                  JSR     R7,DATVER       ;VERIFY DATA FROM FPP
12152  060446  005767  122406                  TST     COUNT           ;SEE IF COUNTER=0
12153  060452  001401                          BEQ     3$              ;BRANCH IF GOOD COMPARE
12154  060454  104003                          ERROR   .-3             ;FPP ERROR
12155                                                                   ;BAD DATA FROM FPP
12156  060456                          3$:
12157
12158                                  ;
12159                                  ;
12160                                  ;-------------------------------
12161                                  ;TEST LDD MODE 6
12162                                  ;
12163                                          ;
12164                                          ;
12165                                          ;
12166  060456                          MLDDM6:
12167
```

```
12168                                           ;
12169  060456  012701  003302                           MOV     #RECDST+200,R1          ;POINT TO RECEIVED DATA LOCATION
12170  060462  012704  003004                           MOV     #TAB2-200,R4                    ;SETUP R4 FOR MODE 6
12171  060466  012702  047750                           MOV     #47750,R2               ;LOAD GOOD STATUS
12172  060472  170102                                    LDFPS   R2                      ;LOAD FPP STATUS - DOUBLE,ID
12173  060474  172464  000200                           LDD     200(R4),AC0             ;LDD MODE 6
12174  060500  170203                                    STFPS   R3                      ;SAVE TEST FPP STATUS
12175  060502  174061  177600                           STD     AC0,-200(R1)                    ;SAVE TEST RESULT
12176  060506  022703  047740                           CMP     #47740,R3                       ;VERIFY FPP STATUS
12177  060512  001401                                    BEQ     1$                      ;BRANCH IF GOOD
12178  060514  104003                                    ERROR   .3              ;FPP ERROR
12179                                                   ;BAD FPP STATUS
12180  060516  162701  000200          1$:      SUB     #200,R1                         ;R1=RECDST
12181  060522  062704  000200          2$:      ADD     #200,R4                         ;POINT TO EXPECTED DATA
12182  060526  004767  052730                           JSR     R7,DATVER               ;VERFIY DATA FROM FPP
12183  060532  005767  122322                           TST     COUNT                   ;SEE IF COUNTER=0
12184  060536  001401                                    BEQ     3$                      ;BRANCH IF GOOD COMPARE
12185  060540  104003                                    ERROR   .3              ;FPP ERROR
12186                                                   ;BAD DATA FROM FPP
12187  060542                          3$:
12188
12189                                           ;
12190                                           ;
12191                                           ;-------------------------------
12192                                           ;TEST LDD MODE 7
12193                                           ;
12194                                                   ;
12195                                                   ;
12196                                                   ;
12197  060542                          MLDDM7:
12198
12199                                           ;
12200  060542  012701  003102                           MOV     #RECDST,R1              ;POINT TO RECEIVED DATA LOCATION
12201  060546  005004                                    CLR     R4                      ;R4=0
12202  060550  012727  003174  003122                    MOV     #TAB1,#TSTLOC           ;POINTER FOR MODE 7 GOOD DATA
12203  060556  012702  047750                           MOV     #47750,R2               ;LOAD GOOD STATUS
12204  060562  170102                                    LDFPS   R2                      ;LOAD FPP STATUS - DOUBLE,ID
12205  060564  172474  003122                           LDD     @TSTLOC(R4),AC0         ;*TEST INSTRUCTION - MODE 7
12206  060570  170203                                    STFPS   R3                      ;SAVE TEST FPP STATUS
12207  060572  174011                                    STD     AC0,(R1)                        ;SAVE TEST RESULT
12208  060574  020203                                    CMP     R2,R3                   ;VERIFY FPP STATUS
12209  060576  001401                                    BEQ     1$                      ;BRANCH IF GOOD
12210  060600  104003                                    ERROR   .3              ;FPP ERROR
12211                                                   ;BAD FPP STATUS
12212  060602  005704                  1$:      TST     R4                              ;VERFIY CONTENTS OF R4
12213  060604  001401                                    BEQ     2$                      ;BRANCH IF GOOD
12214  060606  104003                                    ERROR   .3              ;FPP ERROR
12215                                                   ;BAD R4
12216  060610  012704  003174          2$:      MOV     #TAB1,R4                        ;POINT TO RECEIVED DATA
12217  060614  004767  052642                           JSR     R7,DATVER               ;VERFIY DATA FROM FPP
12218  060620  005767  122234                           TST     COUNT                   ;SEE IF COUNTER=0
12219  060624  001401                                    BEQ     3$                      ;BRANCH IF GOOD COMPARE
12220  060626  104003                                    ERROR   .3              ;FPP ERROR
12221                                                   ;BAD DATA FROM FPP
12222  060630                          3$:
12223
```

C2


```
12224                                   ;
12225                                   ;
12226                                   ;-----------------------------------
12227                                   ;TEST LDD MODE 27 - ONLY 16 BITS ARE LOADED OR STORED
12228                                   ;
12229                                           ;
12230                                           ;
12231                                           ;
12232  060630                           MLDM27:
12233
12234                                   ;
12235  060630  012701  003102                   MOV     #RECDST,R1              ;POINT TO RECEIVED DATA LOCATION
12236  060634  012704  003234                   MOV     #TAB5,R4                        ;POINT TO GOOD DATA
12237  060640  012702  047750                   MOV     #047750,R2              ;LOAD GOOD STATUS
12238  060644  005005                           CLR     R5                      ;R5=0
12239  060646  170102                           LDFPS   R2                      ;LOAD FPP STATUS - DOUBLE,ID
12240  060650  172427  043243                   LDD     #5205,AC0               ;*TEST INSTRUCTION - MODE 27
12241  060654  005205                           INC     R5
12242  060656  005205                           INC     R5
12243  060660  005205                           INC     R5                      ; TEST PROPER PC PATH
12244  060662  022705  000003                   CMP     #3,R5                   ;VERIFY ONLY 3 PC INCREMENT
12245  060666  001401                           BEQ     1$                      ;BRANCH IF PROPER PC ACTION
12246  060670  104003                           ERROR   .3                  ;FPP ERROR
12247                                                                        ;BAD MODE 27 LOAD
12248  060672  170203                   1$:     STFPS   R3                      ;SAVE TEST FPP STATUS
12249  060674  174011                           STD     AC0,(R1)                        ;SAVE TEST RESULT
12250  060676  022703  047740                   CMP     #47740,R3                       ;VERIFY FPP STATUS
12251  060702  001401                           BEQ     2$                      ;BRANCH IF GOOD
12252  060704  104003                           ERROR   .3                  ;FPP ERROR
12253                                                                        ;BAD FPP STATUS
12254  060706  004767  052550           2$:     JSR     R7,DATVER                       ;VERFIY DATA FROM FPP
12255  060712  005767  122142                   TST     COUNT                           ;SEE IF COUNTER=0
12256  060716  001401                           BEQ     3$                      ;BRANCH IF GOOD COMPARE
12257  060720  104003                           ERROR   .3                  ;FPP ERROR
12258                                                                        ;BAD DATA FROM FPP
12259  060722                           3$:
12260
12261                                   ;
12262                                   ;
12263                                   ;-----------------------------------
12264                                   ;TEST ADDF, ADDD, SUBF, SUBD - AC0=0 FSRC=0;
12265                                           ;
12266                                           ;
12267                                           ;
12268  060722                           MNNRM1:
12269
12270                                   ;
12271  060722  012704  003254                   MOV     #TAB6,R4                ;POINT TO FSRC TEST DATA
12272  060726  005067  122154                   CLR     RECDST+4                ;CLEAR OUT RECEIVED DATA TABLE
12273  060732  005067  122152                   CLR     RECDST+6                ;
12274  060736  012702  040000                   MOV     #40000,R2               ;SET UP GOOD STATUS
12275  060742  170102                           LDFPS   R2                      ;LOAD FPP STATUS, FLOATING
12276  060744  172414                           LDF     (R4),AC0                ;LOAD AC0 WITH 0
12277  060746  172014                           ADDF    (R4),AC0                ;0+0
12278  060750  170203                           STFPS   R3                      ;SAVE STATUS
12279  060752  022703  040004                   CMP     #40004,R3               ;VERIFY STATUS
```

```
12280  060756  001401                    BEQ     1$              ;BRANCH IF GOOD
12281  060760  104003                    ERROR   .3          ;FPP ERROR
12282                                                       ;BAD FPP STATUS
12283  060762  012701  003102      1$:   MOV     @RECDST,R1      ;POINT TO RECEIVERD DATA
12284  060766  174011                    STF     AC0,(R1)        ;SAVE DATA
12285  060770  004767  052466            JSR     R7,DATVER       ;VERIFY DATA
12286  060774  005767  122060            TST     COUNT           ;
12287  061000  001401                    BEQ     2$              ;BRANCH IF GOOD
12288  061002  104003                    ERROR   .3          ;FPP ERROR
12289                                                       ;BAD DATA IN AC0
12290  061004  012702  040200      2$:   MOV     @40200,R2       ;LOAD FLOATING STATUS
12291  061010  170102                    LDFPS   R2              ;
12292  061012  172414                    LDD     (R4),AC0        ;LOAD AC0 WITH 0
12293  061014  172014                    ADDD    (R4),AC0        ;*TEST INSTRUCTION
12294  061016  174011                    STD     AC0,(R1)        ;SAVE DATA
12295  061020  170203                    STFPS   R3              ;SAVE FPS
12296  061022  022703  040204            CMP     @40204,R3       ;VERFIY STATUS
12297  061026  001401                    BEQ     3$              ;BRANCH IF GOOD
12298  061030  104003                    ERROR   .3          ;FPP ERROR
12299                                                       ;BAD FPS
12300  061032  004767  052424      3$:   JSR     R7,DATVER               ;VERFIY DATA
12301  061036  005737  003060            TST     @#COUNT         ;VERIFY RESULT
12302  061042  001401                    BEQ     44$             ;BRANCH IF GOOD
12303  061044  104003                    ERROR   .3          ;FPP ERROR
12304                                                       ;BAD AC0
12305  061046  172414            44$:    LDD     (R4),AC0        ;SETUP DATA
12306  061050  173014                    SUBD    (R4),AC0        ;*TEST INSTRUCTION
12307  061052  170203                    STFPS   R3              ;SAVE STATUS
12308  061054  022703  040204                    CMP     @40204,R3               ;VERFIY STATUS
12309  061060  001401                    BEQ     4$              ;BRANCH IF GOOD
12310  061062  104003                    ERROR   .3          ;FPP ERROR
12311                                                       ;BAD FPS
12312  061064  174011            4$:     STD     AC0,(R1)        ;SAVE AC0 DATA
12313  061066  004767  052370                    JSR     R7,DATVER               ;VERFIY DATA
12314  061072  005767  121762            TST     COUNT           ;
12315  061076  001401                    BEQ     5$              ;BRANCH IF GOOD
12316  061100  104003                    ERROR   .3          ;FPP ERROR
12317                                                       ;BAD AC0
12318  061102  170127  000000      5$:   LDFPS   @0              ;STORE FPP STATUS
12319  061106  172414                    LDD     (R4),AC0        ;LOAD AC0
12320  061110  173014                    SUBF    (R4),AC0                ;0-0
12321  061112  170203                    STFPS   R3              ;SAVE STATUS
12322  061114  174011                    STD     AC0,(R1)        ;SAVE AC0
12323  061116  022703  000004            CMP     @4,R3           ;VERFIY STATUS
12324  061122  001401                    BEQ     6$              ;BRANCH IF GOOD
12325  061124  104003                    ERROR   .3          ;FPP ERROR
12326                                                       ;BAD FPS
12327  061126  004767  052330      6$:   JSR     R7,DATVER               ;VERIFY DATA!
12328  061132  005767  121722            TST COUNT               ;
12329  061136  001401                    BEQ     7$              ;BRANC IF GOOD
12330  061140  104003                    ERROR   .3          ;FPP ERROR
12331                                                       ;BAD AC0
12332  061142            7$:
12333
12334
12335                                    ;
```

```
12336                                   ;
12337                                   ;-----------------------------------
12338                                   ;TEST ADDF,SUBD - FSRC=0, AC0 NE 0
12339                                   ;
12340                                           ;
12341                                           ;
12342                                           ;
12343  06 142                           MNNRM2:
12344
12345                                   ;
12346  061142  012701  003102                  MOV     #RECDST,R1              ;POINT TO RECEIVED DATA TABLE
12347  061146  012705  003254                  MOV     #TAB6,R5                ;POINT TO SOURCE DATA TABLE
12348  061152  012704  003204                  MOV     #TAB2,R4                ;POINT TO AC0 DATA
12349  061156  170127  000200                  LDFPS   #200                    ;SET TO DOUBLE FOR CLEAR
12350  061162  172415                          LDD     (R5),AC0                ;
12351  061164  005002                          CLR     R2                      ;SETUP FPP STATUS
12352  061166  170102                          LDFPS   R2                      ;LOAD FPS
12353  061170  172414                          LDF     (R4),AC0                ;LOAD AC0
12354  061172  172015                          ADDF    (R5),AC0                        ;*TEST INSTRUCTION
12355  061174  170203                          STFPS   R3                      ;SAVE STATUS
12356  061176  174011                          STF     AC0,(R1)                ;SAVE AC0
12357  061200  022703  000000                  CMP     #0,R3                   ;VERFIY NEGATIVE RESULT
12358  061204  001401                          BEQ     1$                      ;BRANCH IF GOOD
12359  061206  104003                          ERROR   .3              ;FPP ERROR
12360                                                                   ;BAD FPS
12361  061210  012704  003224          1$:     MOV     #TAB4,R4                ;POINT TO EXPECTED DATA
12362  061214  004767  052242                  JSR     R7,DATVER                       ;VERFIY AC0
12363  061220  005767  121634                  TST     COUNT                   ;CHECK RESULT
12364  061224  001401                          BEQ     2$                      ;BRANCH IF GOOD
12365  061226  104003                          ERROR   .3              ;FPP ERROR
12366                                                                   ;BAD AC0
12367  061230  170127  000200          2$:     LDFPS   #200                    ;SET STATUS TO DUOBLE NODE
12368  061234  172414                          LDD     (R4),AC0                ;LOAD AC0 WITH A VALUE
12369  061236  173015                          SUBD    (R5),AC0                ;*TEST INSTRUCTION
12370  061240  170203                          STFPS   R3                      ;SAVE FPP STATUS
12371  061242  174011                          STD     AC0,(R1)                ;SAVE AC0
12372  061244  022703  000200                  CMP     #200,R3         ;VERIFY RESULT
12373  061250  001401                          BEQ     3$                      ;BRANCH IF GOOD
12374  061252  104003                          ERROR   .3              ;FPP ERROR
12375                                                                   ;BAD SUBD
12376  061254  012704  003224          3$:     MOV     #TAB4,R4                ;POINT TO EXPECTED
12377  061260  004767  052176                  JSR     R7,DATVER               ;VERIFY AC0
12378  061264  005767  121570                  TST     COUNT                   ;
12379  061270  001401                          BEQ     4$                      ;BRANCH IF GOOD AC0
12380  061272  104003                          ERROR   .3              ;FPP ERROR
12381                                                                   ;BAD AC0
12382  061274                           4$:
12383
12384
12385                                   ;
12386                                   ;
12387                                   ;-----------------------------------
12388                                   ;TEST ADDD, SUBF - FSRC NE 0, AC0=0
12389                                   ;
12390                                           ;
12391                                           ;
```

```
12392                                           ;
12393  061274                           MNNRM3:
12394
12395                                   ;
12396  061274  012701  003102                   MOV     #RECDST,R1              ;POINT TO RECEIVED DATA TABLE
12397  061300  012705  003254                   MOV     #TAB6,R5                ;POINT TO ACO DATA TABLE
12398  061304  012704  003174                   MOV     #TAB1,R4                ;POINT TO FSRC DATA
12399  061310  012702  000200                   MOV     #200,R2                 ;SETUP FPP STATUS
12400  061314  170102                           LDFPS   R2                      ;LOAD FPS
12401  061316  172415                           LDD     (R5),AC0                ;LOAD AC0
12402  061320  172014                           ADDD    (R4),AC0                        ;*TEST INSTRUCTION
12403  061322  170203                           STFPS   R3                      ;SAVE STATUS
12404  061324  174011                           STD     AC0,(R1)                ;SAVE AC0
12405  061326  022703  000210                   CMP     #210,R3                 ;VERFIY NEGATIVE RESULT
12406  061332  001401                           BEQ     1$                      ;BRANCH IF GOOD
12407  061334  104003                           ERROR   .3              ;FPP ERROR
12408                                           ;BAD FPS
12409  061336  004767  052120           1$:     JSR     R7,DATVER                       ;VERFIY AC0
12410  061342  005767  121512                   TST     COUNT                   ;CHECK RESULT
12411  061346  001401                           BEQ     2$                      ;BRANCH IF GOOD
12412  061350  104003                           ERROR   .3              ;FPP ERROR
12413                                           ;BAD AC0
12414  061352  170127  000200           2$:     LDFPS   #200                    ;SET STATUS TO DUOBLE NODE
12415  061356  172415                           LDF     (R5),AC0                ;LOAD AC0 WITH A VALUE
12416  061360  173014                           SUBD    (R4),AC0                ;*TEST INSTRUCTION
12417  061362  170203                           STFPS   R3                      ;SAVE FPP STATUS
12418  061364  174011                           STF     AC0,(R1)                        ;SAVE AC0
12419  061366  022703  000200                   CMP     #200,R3                 ;VERIFY RESULT
12420  061372  001401                           BEQ     3$                      ;BRANCH IF GOOD
12421  061374  104003                           ERROR   .3              ;FPP ERROR
12422                                           ;BAD SUBD
12423  061376  012704  003444           3$:     MOV     #TAB18,R4               ;POINT TO EXPECTED DATA
12424  061402  004767  052054                   JSR     R7,DATVER               ;VERIFY AC0
12425  061406  005767  121446                   TST     COUNT                   ;
12426  061412  001401                           BEQ     4$                      ;BRANCH IF GOOD AC0
12427  061414  104003                           ERROR   .3              ;FPP ERROR
12428                                           ;BAD AC0
12429  061416                           4$:
12430
12431
12432                                   ;
12433                                   ;
12434                                   ;-----------------------------------
12435                                   ;TEST  ADDF, SUBD - EXP(AC0) = EXP(FSRC)
12436                                   ;
12437                                           ;
12438                                           ;
12439                                           ;
12440  061416                           MNNRM4:
12441
12442                                   ;
12443  061416  012702  003240                   MOV     #3240,R2                ;SET FIU,FD,FT
12444  061422  170102                           LDFPS   R2
12445  061424  012704  003274                   MOV     #TAB7,R4                        ;SET FSRC
12446  061430  012705  003304                   MOV     #TAB8,R5                ;SETUP AC0
12447  061434  012701  003102                   MOV     #RECDST,R1              ;POINT TO RECEIVED DATA
```

```
12448  061440  172415                          LDD     (R5),AC0                        ;LOAD AC0
12449  061442  172014                          ADDD    (R4),AC0                ;*TEST INSTRUCTION
12450  061444  174011                          STD     AC0,(R1)                ;SAVE TEST RESULT
12451  061446  012704  003314                  MOV     #TAB9,R4                ;POINT TO EXPECTED DATA
12452  061452  004767  052004                  JSR     R7,DATVER                       ;VERIFY AC0 DATA
12453  061456  005767  121376                  TST     COUNT                   ;
12454  061462  001401                          BEQ     1$                      ;BRANCH IF GOOD
12455  061464  104003                          ERROR   .3              ;FPP ERROR
12456                                                                   ;BAD ADDD
12457  061466  012704  003304          1$:     MOV     #TAB8,R4                ;
12458  061472  012703  003304                  MOV     #TAB8,R3                ;SETUP SAME AC0
12459  061476  012702  003200                  MOV     #3200,R2                ;ROUND MODE
12460  061502  170102                          LDFPS   R2
12461  061504  172413                          LDD     (R3),AC0                ;LOAD AC0
12462  061506  061400                          ADD     (R4),AC0                ;*TEST INSTRUCTION
12463  061510  174011                          STD     AC0,(R1)                ;SAVE DATA
12464  061512  004767  051744                  JSR     R7,DATVER               ;VERIFY AC0
12465  061516  005767  121336                  TST     COUNT
12466  061522  001401                          BEQ     2$                      ;BRANCH IF GOOD
12467  061524  104003                          ERROR   .3              ;FPP ERROR
12468                                                                   ;BAD ROUND RESULT
12469  061526                          2$:
12470
12471
12472                                  ;
12473                                  ;
12474                                  ;-----------------------------------
12475                                  ;TEST ADD - EXP(FSRC) > EXP(AC0)
12476                                  ;
12477                                          ;
12478                                          ;
12479                                          ;
12480  061526                          MXDF1:
12481
12482                                  ;
12483  061526  012702  003200                  MOV     #3200,R2                ;R2=FPP STATUS
12484  061532  170102                          LDFPS   R2                      ;LOAD FPS STATUS
12485  061534  012704  003334                  MOV     #TAB11,R4               ;POINT TO FSRC DATA
12486  061540  012701  003102                  MOV     #RECDST,R1              ;POINT TO AC0 RESULT
12487  061544  012705  003324                  MOV     #TAB10,R5               ;POINT TO AC0 DATA
12488  061550  172415                          LDD     (R5),AC0                ;LOAD AC0 DATA
12489  061552  172014                          ADDD    (R4),AC0                ;*TEST INSTRUCTIONS
12490  061554  170203                          STFPS   R3                      ;SAVE FPP STATUS
12491  061556  174011                          STD     AC0,(R1)                ;SAVE AC0 DATA
12492  061560  022703  003200                  CMP     #3200,R3                ;VERIFY FPP STATUS
12493  061564  001401                          BEQ     1$                      ;BRANCH IF GOOD
12494  061566  104003                          ERROR   .3              ;FPP ERROR
12495                                                                   ;BAD FPP STATUS
12496  061570  012704  003344          1$:     MOV     #TAB11A,R4              ;POINT TO EXPECTED DATA
12497  061574  004767  051662                  JSR     R7,DATVER               ;VERIFY CONTENTS OF AC0
12498  061600  005767  121254                  TST     COUNT                   ;
12499  061604  001401                          BEQ     2$                      ;BRANCH IF GOOD AC0
12500  061606  104003                          ERROR   .3              ;FPP ERROR
12501                                                                   ;BAD AC0, SHOULD = FSRC
12502  061610  012704  003354          2$:     MOV     #TAB12,R4               ;POINT TO FSRC DATA
12503  061614  172415                          LDD     (R5),AC0                ;AC0
```

```
12504  061616  172014                          ADDD    (R4),AC0        ;*TEST INSTRUCTION
12505  061620  012704  003374                  MOV     #TAB13B,R4      ;POINT TO EXPECTED RESULT
12506  061624  174011                          STD     AC0,(R1)        ;SAVE AC0 DATA INTO RECDAT
12507  061626  004767  051630                  JSR     R7,DATVER       ;VERIFY DATA
12508  061632  005767  121222                  TST     COUNT
12509  061636  001401                          BEQ     3$              ;BRANCH IF GOOD DATA
12510  061640  104003                          ERROR   +3          ;FPP ERROR
12511                                                               ;BAD AC0 DATA
12512  061642  012702  003000          3$:     MOV     #3000,R2        ;GET FPP STATUS DATA
12513  061646  012704  003404                  MOV     #TAB14,R4       ;POINT TO FSRC DATA
12514  061652  012705  003414                  MOV     #TAB15,R5       ;POINT TO AC0 DATA
12515  061656  172415                          LDD     (R5),AC0        ;LOAD AC0
12516  061660  170102                          LDFPS   R2              ;FPP STATUS = FLOAT, INTERRUPTS ENABLE
12517  061662  172014                          ADDF    (R4),AC0        ;*TEST INSTRUCTION
12518  061664  170127  000200                  LDFPS   #200            ;RESET TO DOUBLE
12519  061670  174011                          STD     AC0, (R1)       ;RECDST=AC0
12520  061672  012704  003324                  MOV     #TAB10,R4       ;POINT TO GOOD DATA
12521  061676  004767  051560                  JSR     R7,DATVER               ;VERFIY CONTENTS OF AC0
12522  061702  005767  121152                  TST     COUNT                   ;
12523  061706  001401                          BEQ     4$              ;BRANCH IF GOOD
12524  061710  104003                          ERROR   +3          ;FPP ERROR
12525                                                               ;BAD FLOATING ADD
12526  061712  012705  003424          4$:     MOV     #TAB16,R5       ;POINT TO AC0 DATA
12527  061716  170102                          LDFPS   R2              ;FPP STATUS = FLOAT
12528  061720  172415                          LDF     (R5),AC0        ;LOAD AC0
12529  061722  172014                          ADDF    (R4),AC0        ;*TEST INSTRUCTION
12530  061724  174011                          STD     AC0,(R1)                ;SAVE AC0  DATA
12531  061726  012704  003434                  MOV     #TAB17,R4       ;POINT TO GOOD DATA
12532  061732  004767  051524                  JSR     R7,DATVER
12533  061736  005767  121116                  TST     COUNT
12534  061742  001401                          BEQ     5$                      ;BRANCH IF GOOD
12535  061744  104003                          ERROR   +3                  ;FPP ERROR
12536                                                                       ;BAD FLOATING ADD
12537  061746                          5$:
12538
12539
12540
12541                                   ;
12542                                   ;
12543                                   ;-----------------------------------
12544                                   ;TEST ADDD WITH NEGATIVE OPERANDS
12545                                   ;
12546                                           ;
12547                                           ;
12548                                           ;
12549  061746                           MNGOP:
12550
12551                                   ;
12552  061746  012702  003200                  MOV     #3200,R2        ;LOAD FPS VALUE
12553  061752  170102                          LDFPS   R2              ;
12554  061754  012704  003454                  MOV     #TAB21,R4       ;DATA ADDRESS FOR AC0 AND FSR
12555  061760  172414                          LDD     (R4),AC0        ;AC0=100200 0 0 0
12556  061762  172014                          ADDD    (R4),AC0        ;*TEST INSTRUCTION
12557  061764  170203                          STFPS   R3              ;SAVE STATUS
12558  061766  012701  003102                  MOV     #RECDST,R1      ;POINT TO RECEIVED DATA TABLE
12559  061772  174011                          STD     AC0,(R1)        ;SAVE AC0 DATA
```

```
12560  061774  022703  003210            CMP     #3210,R3                ;VERIFY STATUS
12561  062000  001401                    BEQ     1$                      ;BRANCH IF GOOD
12562  062002  104003                    ERROR   .3              ;FPP ERROR
12563                                                          ;
12564  062004  012704  003464    1$:     MOV     #TAB22,R4               ;POINT TO EXPECTED DATA
12565  062010  004767  051446            JSR     R7,DATVER               ;
12566  062014  005767  121040            TST     COUNT                   ;VERIFY DATA
12567  062020  001401                    BEQ     2$                      ;BRANCH IF GOOD
12568  062022  104003                    ERROR   .3              ;FPP ERROR
12569                                                          ;
12570                                    ;       !-FSRC! = !AC0!
12571  062024  012704  003454    2$:     MOV     #TAB21,R4               ;POINT TO FSRC DATA
12572  062030  012701  003474            MOV     #TAB23,R1               ;POINT TO AC0 DATA
12573  062034  012737  062052  000244      MOV     #101$,@#FPVEC           ;SETUP FP VECTOR
12574  062042  172411                    LDD     (R1),AC0                ;LOAD AC0
12575  062044  172014                    ADDD    (R4),AC0                ;*TEST INSTRUCTION
12576  062046  170000            100$:   CFCC                            ;COPY FPP CC
12577  062050  104003                    ERROR   .3              ;FPP ERROR
12578                                                          ;GO TO ERROR
12579  062052  170203            101$:   STFPS   R3                      ;SAVE FPP STATUS
12580  062054  012701  003102            MOV     #RECDST,R1                      ;POINT TO RECEIVED DATA TABLE
12581  062060  174011                    STD     AC0,(R1)                ;SAVE AC0 DATA
12582  062062  022703  103200            CMP     #103200,R3              ;VERIFY STATUS
12583  062066  001401                    BEQ     3$                      ;BRANCH IF GOOD
12584  062070  104003                    ERROR   .3              ;FPP ERROR
12585                                                          ;BAD STATUS
12586  062072  012605            3$:     MOV     (SP)+,R5                ;GET ERROR PC
12587  062074  020527  062046            CMP     R5,#100$                ;VERIFY ERROR ADDRESS ON STACK
12588  062100  001401                    BEQ     102$                    ;BRANCH IF GOOD
12589  062102  104003                    ERROR   .3              ;FPP ERROR
12590                                                          ;BAD ERROR RETURN ON STACK
12591  062104  005726            102$:   TST     (SP)+                   ;RESTORE STACK
12592  062106  012704  003504            MOV     #TAB24,R4               ;POINT TO EXPECTED DATA TABLE
12593  062112  004767  051344            JSR     R7,DATVER               ;VERIFY DATA
12594  062116  005767  120736            TST     COUNT
12595  062122  001401                    BEQ     4$                      ;BRANC IF GOOD
12596  062124  104003                    ERROR   .3              ;FPP ERROR
12597                                                          ;BAD AC0 DATA
12598                                    ;       !-AC0! = !FSRC!
12599  062126  012704  003474    4$:     MOV     #TAB23,R4               ;POINT TO FSRC DATA
12600  062132  012701  003454            MOV     #TAB21,R1               ;POINT TO AC0 DATA
12601  062136  012737  062162  000244      MOV     #104$,@#FPVEC           ;SETUP FP VECTOR
12602  062144  012702  003200            MOV     #3200,R2                ;LOAD FPS VALUE
12603  062150  170102                    LDFPS   R2                      ;
12604  062152  172411                    LDD     (R1),AC0                ;LOAD AC0 DATA
12605  062154  172014                    ADDD    (R4),AC0                ;*TEST INSTRUCTION
12606  062156  170000            103$:   CFCC                            ;COPY FPP CC
12607  062160  104003                    ERROR   .3              ;FPP ERROR
12608                                                          ;GO TO ERROR
12609  062162  170203            104$:   STFPS   R3                      ;SAVE FPS
12610  062164  012701  003102            MOV     #RECDST,R1              ;SAVE AC0
12611  062170  174011                    STD     AC0,(R1)                ;
12612  062172  022703  103200            CMP     #103200,R3              ;VERFIY STATUS
12613  062176  001401                    BEQ     5$                      ;BRANCH IF GOOD
12614  062200  104003                    ERROR   .3              ;FPP ERROR
12615                                                          ;BAD FPS STATUS
```

J2


```
12616  062202  012605                  5$:     MOV     (SP)+,R5        ;GET ERROR PC
12617  062204  020527  062156                  CMP     R5,#103$        ;VERIFY ERROR ADDRESS ON STACK
12618  062210  001401                          BEQ     105$            ;BRANCH IF GOOD
12619  062212  104003                          ERROR   .+3             ;FPP ERROR
12620                                                                  ;BAD ERROR RETURN ON STACK
12621  062214  005726                  105$:   TST     (SP)+           ;RESTORE STACK
12622  062216  012704  003504                  MOV     #TAB24,R4       ;POINT TO EXPECTED DATA
12623  062222  004767  051234                  JSR     R7,DATVER
12624  062226  005767  120626                  TST     COUNT           ;
12625  062232  001401                          BEQ     6$              ;BRANCH IF  GOOD
12626  062234  104003                          ERROR   .+3             ;FPP ERROR
12627                                                                  ;BAD ACO
12628                                          ;       !-FSRC! <       !ACO!
12629  062236  012704  003524          6$:     MOV     #TAB26,R4       ;POINT TO FSRC DATA
12630  062242  012701  003514                  MOV     #TAB25,R1       ;POINT TO ACO DATA
12631  062246  012702  003200                  MOV     #3200,R2        ;LOAD FPS VALUE
12632  062252  170102                          LDFPS   R2              ;
12633  062254  012737  000246  000244          MOV     #246,@#FPVEC    ;SETUP FP VECTOR
12634  062262  172411                          LDD     (R1),AC0        ;LOAD AC0 DATA
12635  062264  172014                          ADDD    (R4),AC0        ;*TEST INSTRUCTION
12636  062266  170203                          STFPS   R3              ;SAVE STATUS
12637  062270  012701  003102                  MOV     #RECDST,R1      ;POINT TO RECEIVED DATA TABLE
12638  062274  174011                          STD     AC0,(R1)        ;SAVE AC0
12639  062276  020327  003200                  CMP     R3,#3200        ;VERIFY STATUS
12640  062302  001401                          BEQ     7$              ;BRANCH IF GOOD
12641  062304  104003                          ERROR   .+3             ;FPP ERROR
12642                                                                  ;BAD FPS
12643  062306  012704  003534          7$:     MOV     #TAB27,R4       ;POINT TO EXPECTED DSATA
12644  062312  004767  051144                  JSR     R7,DATVER       ;VERIFY DATA
12645  062316  005767  120536                  TST     COUNT           ;
12646  062322  001401                          BEQ     8$              ;BRANCH IF GOOD
12647  062324  104003                          ERROR   .+3             ;FPP ERROR
12648                                                                  ;
12649                                          ;       !FSRC!  >       !-AC!
12650  062326  012704  003514          8$:     MOV     #TAB25,R4       ;POINT TO FSRC DATA
12651  062332  012701  003524                  MOV     #TAB26,R1       ;POINT TO AC0 DATA
12652  062336  172411                          LDD     (R1),AC0        ;LOAD AC0 DATA
12653  062340  172014                          ADDD    (R4),AC0        ;*TEST INSTRUCTION
12654  062342  170203                          STFPS   R3              ;SAVE STATUS
12655  062344  012701  003102                  MOV     #RECDST,R1      ;POINT TO RECEIVED DATA TABLE
12656  062350  174011                          STD     AC0,(R1)        ;SAVE AC0
12657  062352  020327  003200                  CMP     R3,#3200        ;VERIFY STATUS
12658  062356  001401                          BEQ     9$              ;BRANCH IF GOOD
12659  062360  104003                          ERROR   .+3             ;FPP ERROR
12660                                                                  ;BAD FPS
12661  062362  012704  003534          9$:     MOV     #TAB27,R4       ;POINT TO EXPECTED DSATA
12662  062366  004767  051070                  JSR     R7,DATVER       ;VERIFY DATA
12663  062372  005767  120462                  TST     COUNT           ;
12664  062376  001401                          BEQ     10$             ;BRANCH IF GOOD
12665  062400  104003                          ERROR   .+3             ;FPP ERROR
12666                                                                  ;BAD AC0
12667                                          ;       !-FSRC! <       !ACO!
12668  062402  012704  003554          10$:    MOV     #TAB29,R4       ;POINT TO FSRC DATA
12669  062406  012701  003544                  MOV     #TAB28,R1       ;POINT TO AC0 DATA
12670  062412  172411                          LDD     (R1),AC0        ;LOAD AC0 DATA
12671  062414  172014                          ADDD    (R4),AC0        ;*TEST INSTRUCTION
```

K2


```
12672  062416  170203                          STFPS   R3                      ;SAVE STATUS
12673  062420  012701  003102                  MOV     #RECDST,R1              ;POINT TO RECEIVED DATA TABLE
12674  062424  174011                          STD     AC0,(R1)                        ;SAVE AC0
12675  062426  020327  003200                  CMP     R3,#3200                        ;VERIFY STATUS
12676  062432  001401                          BEQ     11$                     ;BRANCH IF GOOD
12677  062434  104003                          ERROR   .3              ;FPP ERROR
12678                                                                  ;BAD FPS
12679  062436  012704  003564          11$:    MOV     #TAB29A,R4              ;POINT TO EXPECTED DATA
12680  062442  004767  051014                  JSR     R7,DATVER                       ;VERIFY DATA
12681  062446  005767  120406                  TST     COUNT                   ;
12682  062452  001401                          BEQ     12$                     ;BRANCH IF GOOD
12683  062454  104003                          ERROR   .3              ;FPP ERROR
12684                                                                  ;
12685  062456                          12$:
12686
12687                                  ;
12688                                  ;
12689                                  ;
12690                                  ;
12691                                  ;-----------------------------------
12692                                  ;TEST SUB WITH EXP[AC0]=EXP[FSRC]
12693                                  ;
12694                                          ;
12695                                          ;
12696                                          ;
12697  062456                          MSB:
12698
12699                                  ;
12700  062456  012702  003200                  MOV     #3200,R2                ;LOAD FPS DATA
12701  062462  170102                          LDFPS   R2                      ;LOAD FPS
12702  062464  012704  003454                  MOV     #TAB21,R4               ;POINT TO FSRC DATA
12703  062470  012701  003102                  MOV     #RECDST,R1              ;POINT TO AC0 RECEIVED DATA TABLE
12704  062474  172414                          LDD     (R4),AC0                ;LOAD AC0
12705  062476  173014                          SUBD    (R4),AC0                        ;*TEST INSTRUCTION
12706  062500  170203                          STFPS   R3                      ;SAVE STATUS
12707  062502  174011                          STD     AC0,(R1)                ;SAVE AC0 INTO RECDST
12708  062504  022703  003204                  CMP     #3204,R3                ;VERIFY STATUS
12709  062510  001401                          BEQ     1$                      ;BRANCH IF GOOD
12710  062512  104003                          ERROR   .3              ;FPP ERROR
12711                                                                  ;BAD FPS STATUS
12712  062514  012704  003254          1$:     MOV     #TAB6,R4                ;POINT TO EXPECTED DATA
12713  062520  004767  050736                  JSR     R7,DATVER               ;VERIFY AC0
12714  062524  005767  120330                  TST     COUNT                   ;
12715  062530  001401                          BEQ     2$                      ;BRANCH IF GOOD
12716  062532  104003                          ERROR   .3              ;FPP ERROR
12717                                                                  ;BAD AC0
12718  062534  012704  003404          2$:     MOV     #TAB14,R4               ;POINT TO FSRC AND AC0 DATA
12719  062540  172414                          LDD     (R4),AC0                        ;LOAD AC0 DATA
12720  062542  173014                          SUBD    (R4),AC0                        ;*TEST  INSTRUCTION
12721  062544  170203                          STFPS   R3                      ;SAVE FPS
12722  062546  174011                          STD     AC0,(R1)                                ;SAVE AC0 INTO RECDST
12723  062550  022703  003204                  CMP     #3204,R3                        ;VERIFY FPS
12724  062554  001401                          BEQ     3$                      ;BRANCH IF GOOD
12725  062556  104003                          ERROR   .3              ;FPP ERROR
12726                                                                  ;BAD AC0
12727  062560  012704  003254          3$:     MOV     #TAB6,R4                ;POINT TO EXPECTED DATA
```

```
12728  062564  004767  050672                  JSR     R7,DATVER                       ;VERIFY ACO
12729  062570  005767  120264                  TST     COUNT
12730  062574  001401                          BEQ     4$                      ;BRANCH IF GOOD
12731  062576  104003                          ERROR   .3              ;FPP ERROR
12732                                                                   ;BAD ACO
12733  062600                          4$:
12734
12735                                   ;
12736                                   ;
12737                                   ;
12738                                   ;-------------------------------
12739                                   ;TEST NORMALIZE
12740                                   ;
12741                                           ;
12742                                           ;
12743                                           ;
12744  062600                          MNRM:
12745
12746                                   ;
12747  062600  012702  003200                  MOV     #3200,R2                ;LOAD FPS
12748  062604  170102                          LDFPS   R2                      ;
12749  062606  012705  003604                  MOV     #TAB31,R5               ;POINT TO FSRC DATA
12750  062612  012701  003574                  MOV     #TAB30,R1               ;POINT TO ACO DATA
12751  062616  172411                          LDD     (R1),AC0                ;LOAD ACO
12752  062620  173015                          SUBD    (R5),AC0                ;*TEST INSTRUCTION
12753                                                                           ;1 LEFT SHIFT
12754  062622  170203                          STFPS   R3                      ;SAVE STATUS
12755  062624  012704  003102                  MOV     #RECDST,R4              ;POINT TO RECDATA
12756  062630  174014                          STD     AC0,(R4)                        ;SAVE ACO
12757  062632  012701  003634                  MOV     #TAB34,R1               ;POINT TO EXPECTED DATA
12758  062636  004767  050620                  JSR     R7,DATVER                       ;VERIFY DATA
12759  062642  005767  120212                  TST     COUNT
12760  062646  001401                          BEQ     1$                      ;BRANCH IF GOOD
12761  062650  104003                          ERROR   .3              ;FPP ERROR
12762                                                                   ;
12763  062652  012701  003614          1$:     MOV     #TAB32,R1               ;ACO DATA
12764  062656  012705  003624                  MOV     #TAB33,R5                       ;FSRC DATA
12765  062662  172411                          LDD     (R1),AC0                        ;LOAD ACO
12766  062664  173015                          SUBD    (R5),AC0                ;*TST INSTRUCTION
12767                                                                           ;56 LEFT SHIFTS
12768  062666  012701  003102                  MOV     #RECDST,R1                      ;SAVE DATA
12769  062672  174011                          STD     AC0,(R1)                        ;
12770  062674  004767  050562                  JSR     R7,DATVER
12771  062700  005767  120154                  TST     COUNT
12772  062704  001401                          BEQ     2$                      ;
12773  062706  104003                          ERROR   .3              ;FPP ERROR
12774                                                                   ;
12775  062710                          2$:
12776
12777                                   ;
12778                                   ;
12779                                   ;-------------------------------------
12780                                   ;TEST ADDD WITH OVERFLOW AND UNDERFLOW
12781                                   ;
12782                                           ;
12783                                           ;
```

M2


```
12784                                                   ;
12785  062710                                   MUVAD:
12786
12787                                           ;
12788  062710  012702  000200                          MOV     #200,R2         ;SETUP FLOATING POINT STATUS
12789  062714  170102                                  LDFPS   R2              ;LOAD FPS
12790  062716  012704  003644                          MOV     #TAB40,R4               ;POINT TO FSRC DATA
12791  062722  012701  003644                          MOV     #TAB40,R1               ;POINT TO ACO DATA
12792  062726  172411                                  LDD     (R1),AC0                ;LOAD AC0 WITH TEST DATA
12793  062730  172014                                  ADDD    (R4),AC0        ;*TEST INSTRUCTION
12794  062732  170203                                  STFPS   R3              ;SAVE FPS
12795  062734  012701  003102                          MOV     #RECDST,R1      ;POINT TO RECEIVED DATA TABLE
12796  062740  174011                                  STD     AC0,(R1)        ;SAVE AC0  RESULT
12797  062742  022703  000206                          CMP     #206,R3         ;VERIFY STATUS
12798  062746  001401                                  BEQ     1$              ;BRANCH IF GOOD
12799  062750  104003                                  ERROR   .3      ;FPP ERROR
12800                                                   ;BAD FPS
12801  062752  012704  003254          1$:             MOV     #TAB6,R4                ;POINT TO EXPECTED DATA
12802  062756  004767  050500                          JSR     R7,DATVER               ;VERIFY DATA
12803  062762  005767  120072                          TST     COUNT
12804  062766  001401                                  BEQ     2$              ;BRANCH IF GOOD
12805  062770  104003                                  ERROR   .3      ;FPP ERROR
12806                                                   ;BAD AC0
12807                                           ;OVERFLOW TRAPS ENABLED
12808  062772  012702  001200          2$:             MOV     #1200,R2                ;SETUP FLOATING POINT STATUS
12809  062776  170102                                  LDFPS   R2              ;LOAD FPS
12810  063000  012704  003644                          MOV     #TAB40,R4               ;POINT TO FSRC DATA
12811  063004  012701  003644                          MOV     #TAB40,R1               ;POINT TO ACO DATA
12812  063010  172411                                  LDD     (R1),AC0                ;LOAD AC0 WITH TEST DATA
12813  063012  012737  063026  000244                  MOV     #3$,@#FPVEC             ;CHANGE TRAP VECTOR
12814  063020  172014                                  ADDD    (R4),AC0        ;*TEST INSTRUCTION
12815  063022  170000                  23$:            CFCC
12816  063024  104003                                  ERROR   .3      ;FPP ERROR
12817                                                   ;FAILED TO TRAP ON OVERFLOW
12818  063026  170203                  3$:             STFPS   R3              ;SAVE FPS
12819  063030  012701  003102                          MOV     #RECDST,R1      ;POINT TO RECEIVED DATA TABLE
12820  063034  174011                                  STD     AC0,(R1)        ;SAVE AC0  RESULT
12821  063036  022703  101206                          CMP     #101206,R3              ;VERIFY STATUS
12822  063042  001401                                  BEQ     4$              ;BRANCH IF GOOD
12823  063044  104003                                  ERROR   .3      ;FPP ERROR
12824                                                   ;BAD FPS
12825  063046  012600                  4$:             MOV     (SP)+,R0        ;CHECK STORED PC
12826  063050  022700  063022                          CMP     #23$,R0         ;
12827  063054  001401                                  BEQ     5$              ;BRANCH IF RETURN ADDRESS IS GOOD
12828  063056  104003                                  ERROR   .3      ;FPP ERROR
12829                                                   ;BAD RETURN ADDRESS
12830  063060  012600                  5$:             MOV     (SP)+,R0        ;CLEAN UP STACK
12831  063062  012704  003254                          MOV     #TAB6,R4                ;POINT TO EXPECTED DATA
12832  063066  004767  050370                          JSR     R7,DATVER               ;VERIFY DATA
12833  063072  005767  117762                          TST     COUNT
12834  063076  001401                                  BEQ     7$              ;BRANCH IF GOOD
12835  063100  104003                                  ERROR   .3      ;FPP ERROR
12836                                                   ;BAD AC0
12837                                           ;UNDERFLOW TRAPS DISABLED
12838  063102  012702  000200          7$:             MOV     #200,R2         ;SETUP FLOATING POINT STATUS
12839  063106  170102                                  LDFPS   R2              ;LOAD FPS
```

N2


```
12840  063110  012737  155426  000244          MOV     #WLDTRP,@#FPVEC                 ;REPLACE WILD TRAP VECTOR
12841  063116  012704  003274                  MOV     #TAB7,R4                        ;POINT TO FSRC DATA
12842  063122  012701  003654                  MOV     #TAB41,R1                       ;POINT TO ACO DATA
12843  063126  172411                          LDD     (R1),AC0                        ;LOAD AC0 WITH TEST DATA
12844  063130  172014                          ADDD    (R4),AC0                ;*TEST INSTRUCTION
12845  063132  170203                          STFPS   R3                      ;SAVE FPS
12846  063134  012701  003102                  MOV     #RECDST,R1              ;POINT TO RECEIVED DATA TABLE
12847  063140  174011                          STD     AC0,(R1)                ;SAVE AC0  RESULT
12848  063142  022703  000204                  CMP     #204,R3                 ;VERIFY STATUS
12849  063146  001401                          BEQ     8$                      ;BRANCH IF GOOD
12850  063150  104003                          ERROR   .-3                     ;FPP ERROR
12851                                                                          ;BAD FPS
12852  063152  012704  003254          8$:     MOV     #TAB6,R4                        ;POINT TO EXPECTED DATA
12853  063156  004767  050300                  JSR     R7,DATVER                       ;VERIFY DATA
12854  063162  005767  117672                  TST     COUNT
12855  063166  001401                          BEQ     9$                      ;BRANCH IF GOOD
12856  063170  104003                          ERROR   .-3                     ;FPP ERROR
12857                                                                          ;BAD AC0
12858                          ;UNDERFLOW TRAPS ENABLED
12859  063172  012702  002200          9$:     MOV     #2200,R2                        ;SETUP FLOATING POINT STATUS
12860  063176  170102                          LDFPS   R2                      ;LOAD FPS
12861  063200  012737  063226  000244          MOV     #11$,@#FPVEC            ;REPOSITION TRAP VECTOR
12862  063206  012704  003274                  MOV     #TAB7,R4                        ;POINT TO FSRC DATA
12863  063212  012701  003654                  MOV     #TAB41,R1                       ;POINT TO ACO DATA
12864  063216  172411                          LDD     (R1),AC0                        ;LOAD AC0 WITH TEST DATA
12865  063220  172014                          ADDD    (R4),AC0                ;*TEST INSTRUCTION
12866  063222  170000          10$:    CFCC                            ;COPY FPP CC
12867  063224  104003                          ERROR   .-3                     ;FPP ERROR
12868                                                                          ;FAILED TO TRAP ON UNDERFLOW
12869  063226  170203          11$:    STFPS   R3                      ;SAVE FPS
12870  063230  012701  003102                  MOV     #RECDST,R1              ;POINT TO RECEIVED DATA TABLE
12871  063234  174011                          STD     AC0,(R1)                ;SAVE AC0  RESULT
12872  063236  022703  102210                  CMP     #102210,R3                      ;VERIFY STATUS
12873  063242  001401                          BEQ     12$                     ;BRANCH IF GOOD
12874  063244  104003                          ERROR   .-3                     ;FPP ERROR
12875                                                                          ;BAD FPS
12876  063246  012605          12$:    MOV     (SP)+,R5                        ;GET ERROR PC
12877  063250  020527  063222                  CMP     R5,#10$                 ;VERIFY ERROR ADDRESS ON STACK
12878  063254  001401                          BEQ     13$                     ;BRANCH IF GOOD
12879  063256  104003                          ERROR   .-3                     ;FPP ERROR
12880                                                                          ;BAD ERROR RETURN ON STACK
12881  063260  005726          13$:    TST     (SP)+                           ;RESTORE STACK
12882  063262  012704  003244                  MOV     #TAB5A,R4                       ;POINT TO EXPECTED DATA
12883  063266  004767  050170                  JSR     R7,DATVER                       ;VERIFY DATA
12884  063272  005767  117562                  TST     COUNT
12885  063276  001401                          BEQ     14$                     ;BRANCH IF GOOD
12886  063300  104003                          ERROR   .-3                     ;FPP ERROR
12887                                                                          ;BAD AC0
12888                          ;UNDERFLOW WITH TRAPS DISBLED - NON-ZERO RESULT
12889  063302  012702  000200          14$:    MOV     #200,R2                 ;SETUP FLOATING POINT STATUS
12890  063306  170102                          LDFPS   R2                      ;LOAD FPS
12891  063310  012737  155426  000244          MOV     #WLDTRP,@#FPVEC                 ;RESTORE TRAP VECTOR
12892  063316  012704  003654                  MOV     #TAB41,R4                       ;POINT TO FSRC DATA
12893  063322  012701  003664                  MOV     #TAB42,R1                       ;POINT TO ACO DATA
12894  063326  172411                          LDD     (R1),AC0                        ;LOAD AC0 WITH TEST DATA
12895  063330  172014                          ADDD    (R4),AC0                ;*TEST INSTRUCTION
```

```
12896  063332  170203                          STFPS   R3              ;SAVE FPS
12897  063334  012701  003102                  MOV     #RECDST,R1      ;POINT TO RECEIVED DATA TABLE
12898  063340  174011                          STD     AC0,(R1)        ;SAVE AC0  RESULT
12899  063342  022703  000204                  CMP     #204,R3         ;VERIFY STATUS
12900  063346  001401                          BEQ     15$             ;BRANCH IF GOOD
12901  063350  104003                          ERROR   .3              ;FPP ERROR
12902                                                                  ;BAD FPS
12903  063352  012704  003254          15$:    MOV     #TAB6,R4                ;POINT TO EXPECTED DATA
12904  063356  004767  050100                  JSR     R7,DATVER               ;VERIFY DATA
12905  063362  005767  117472                  TST     COUNT
12906  063366  001401                          BEQ     16$             ;BRANCH IF GOOD
12907  063370  104003                          ERROR   .3              ;FPP ERROR
12908                                                                  ;BAD AC0
12909                                  ;UNERFLOW WITH TRAPS ENABLED - NON-ZERO  RESULT
12910  063372  012702  102200          16$:    MOV     #102200,R2              ;SETUP FLOATING POINT STATUS
12911  063376  170102                          LDFPS   R2              ;LOAD FPS
12912  063400  012737  063426  000244          MOV     #18$,@#FPVEC    ;RESTORE  TRAP VECTOR
12913  063406  012704  003654                  MOV     #TAB41,R4               ;POINT TO FSRC DATA
12914  063412  012701  003664                  MOV     #TAB42,R1               ;POINT TO AC0 DATA
12915  063416  172411                          LDD     (R1),AC0                ;LOAD AC0 WITH TEST DATA
12916  063420  172014                          ADDD    (R4),AC0        ;*TEST INSTRUCTION
12917  063422  170000          17$:    CFCC
12918  063424  104003                          ERROR   .3              ;FPP ERROR
12919                                                                  ;NO TRAP ON UNDERFLOW
12920  063426  170203          18$:    STFPS   R3              ;SAVE FPS
12921  063430  012701  003102                  MOV     #RECDST,R1      ;POINT TO RECEIVED DATA TABLE
12922  063434  174011                          STD     AC0,(R1)        ;SAVE AC0  RESULT
12923  063436  012600                          MOV     (SP)+,R0        ;SAVE STACK CONTENTS
12924  063440  005726                          TST     (SP)+           ;CLEAN UP STACK
12925  063442  022700  063422                  CMP     #17$,R0         ;VERIFY RETURN ADDRESS
12926  063446  001401                          BEQ     19$             ;BRANCH IF GOOD
12927  063450  104003                          ERROR   .3              ;FPP ERROR
12928                                                                  ;BAD RETURN ADDRESS
12929  063452  022703  102204          19$:    CMP     #102204,R3              ;VERIFY STATUS
12930  063456  001401                          BEQ     20$             ;BRANCH IF GOOD
12931  063460  104003                          ERROR   .3              ;FPP ERROR
12932                                                                  ;BAD FPS
12933  063462  012704  003674          20$:    MOV     #TAB43,R4               ;POINT TO EXPECTED DATA
12934  063466  004767  047770                  JSR     R7,DATVER               ;VERIFY DATA
12935  063472  005767  117362                  TST     COUNT
12936  063476  001401                          BEQ     21$             ;BRANCH IF GOOD
12937  063500  104003                          ERROR   .3              ;FPP ERROR
12938                                                                  ;BAD AC0
12939  063502          21$:
12940
12941
12942                                  ;
12943                                  ;
12944                                  ;-------------------------------
12945                                  ;TEST LDCFD, LDCDF
12946                                  ;
12947                                          ;
12948                                          ;
12949                                          ;
12950  063502          MLDC:
12951
```

```
12952                                   ;
12953                                   ;TRUNCATE
12954  063502  012702  000300                   MOV     #300,R2         ;SETUP FLOATING POINT STATUS
12955  063506  170102                           LDFPS   R2              ;LOAD FPS
12956  063510  012704  003704                   MOV     #TAB45,R4               ;POINT TO FSRC DATA
12957  063514  012701  003254                   MOV     #TAB6,R1                ;POINT TO ACO DATA
12958  063520  172411                           LDD     (R1),AC0                ;LOAD ACO WITH TEST DATA
12959  063522  177424                           LDCFD   (R4)+,AC0       ;*TEST INSTRUCTION
12960  063524  012701  003102                   MOV     #RECDST,R1      ;POINT TO RECEIVED DATA TABLE
12961  063530  174011                           STD     AC0,(R1)        ;SAVE ACO  RESULT
12962  063532  022704  003710                   CMP     #TAB45+4,R4     ;VERIFY AUTO-INC
12963  063536  001401                           BEQ     1$              ;BRANCH IF GOOD AUTO-INC
12964  063540  104003                           ERROR   .3      ;FPP ERROR
                                                               ;BAD AUTO-INC
12965
12966  063542  012704  003714   1$:     MOV     #TAB46,R4               ;POINT TO EXPECTED DATA
12967  063546  004767  047710           JSR     R7,DATVER               ;VERIFY DATA
12968  063552  005767  117302           TST     COUNT
12969  063556  001401                   BEQ     2$              ;BRANCH IF GOOD
12970  063560  104003                   ERROR   .3      ;FPP ERROR
                                                       ;BAD ACO
12971
12972                                   ;AUTO-INC DOUBLE MODE
12973  063562  005002   2$:     CLR     R2              ;SETUP FLOATING POINT STATUS
12974  063564  170102                   LDFPS   R2              ;LOAD FPS
12975  063566  012704  003704           MOV     #TAB45,R4               ;POINT TO FSRC DATA
12976  063572  012701  003424           MOV     #TAB16,R1               ;POINT TO ACO DATA
12977  063576  172411                   LDD     (R1),AC0                ;LOAD ACO WITH TEST DATA
12978  063600  177424                   LDCDF   (R4)+,AC0       ;*TEST INSTRUCTION
12979  063602  020427  003714           CMP     R4,#TAB45+10    ;VERIFY AUTO-INC
12980  063606  001401                   BEQ     3$              ;BRANCH IF GOOD
12981  063610  104003                   ERROR   .3      ;FPP ERROR
                                                       ;BAD AUTO-INC ON DOUBLE
12982
12983  063612  170203   3$:     STFPS   R3              ;SAVE FPS
12984  063614  012701  003102           MOV     #RECDST,R1              ;POINT TO RECEIVED DATA TABLE
12985  063620  174011                   STD     AC0,(R1)        ;SAVE ACO  RESULT
12986  063622  022703  000000           CMP     #0,R3           ;VERIFY STATUS
12987  063626  001401                   BEQ     4$              ;BRANCH IF GOOD
12988  063630  104003                   ERROR   .3      ;FPP ERROR
                                                       ;BAD FPS
12989
12990  063632  012704  003754   4$:     MOV     #TAB49,R4               ;POINT TO EXPECTED DATA
12991  063636  004767  047620           JSR     R7,DATVER               ;VERIFY DATA
12992  063642  005767  117212           TST     COUNT
12993  063646  001401                   BEQ     5$              ;BRANCH IF GOOD
12994  063650  104003                   ERROR   .3      ;FPP ERROR
                                                       ;BAD ACO
12995
12996                                   ;LDCFD GR7
12997  063652  012702  000200   5$:     MOV     #200,R2         ;SETUP FLOATING POINT STATUS
12998  063656  170102                   LDFPS   R2              ;LOAD FPS
12999  063660  005003                   CLR     R3
13000  063662  177427  043243           LDCFD   #5203,AC0       ;*TEST INSTRUCTION
13001  063666  005203                   INC     R3
13002  063670  005203                   INC     R3
13003  063672  005203                   INC     R3              ;IF LDCFD WORKED, R3 SHOULD=3
13004  063674  022703  000003           CMP     #3,R3           ;VERIFY CORRECT PROGRAM FLOW
13005  063700  001401                   BEQ     6$              ;BRANCH IF GOOD
13006  063702  104003                   ERROR   .3      ;FPP ERROR
                                                       ;BAD PROGRAM FLOW
13007
```

```
                                        ;NEGATIVE OPERANDS
063704  012702  000200          6$:     MOV     #200,R2                 ;SETUP FLOATING POINT STATUS
063710  170102                          LDFPS   R2                      ;LOAD FPS
063712  012704  003724                  MOV     #TAB47,R4                       ;POINT TO FSRC DATA
063716  012701  003704                  MOV     #TAB45,R1                       ;POINT TO AC0 DATA
063722  172411                          LDD     (R1),AC0                        ;LOAD AC0 WITH TEST DATA
063724  177414                          LDCFD   (R4),AC0                ;*TEST INSTRUCTION
063726  170203                          STFPS   R3                      ;SAVE FPS
063730  012701  003102                  MOV     #RECDST,R1              ;POINT TO RECEIVED DATA TABLE
063734  174011                          STD     AC0,(R1)                ;SAVE AC0  RESULT
063736  022703  000210                  CMP     #210,R3                 ;VERIFY STATUS
063742  001401                          BEQ     7$                      ;BRANCH IF GOOD
063744  104003                          ERROR   +3                  ;FPP ERROR
                                                                    ;BAD FPS
063746  012704  003744          7$:     MOV     #TAB48,R4                       ;POINT TO EXPECTED DATA
063752  004767  047504                  JSR     R7,DATVER                       ;VERIFY DATA
063756  005767  117076                  TST     COUNT
063762  001401                          BEQ     8$                      ;BRANCH IF GOOD
063764  104003                          ERROR   +3                  ;FPP ERROR
                                                                    ;BAD AC0
                                        ;LOAD A ZERO
063766  012702  000200          8$:     MOV     #200,R2                 ;SETUP FLOATING POINT STATUS
063772  170102                          LDFPS   R2                      ;LOAD FPS
063774  012704  003254                  MOV     #TAB6,R4                        ;POINT TO FSRC DATA
064000  012701  003744                  MOV     #TAB48,R1                       ;POINT TO AC0 DATA
064004  172411                          LDD     (R1),AC0                        ;LOAD AC0 WITH TEST DATA
064006  177414                          LDCFD   (R4),AC0                        ;*TEST INSTRUCTION
064010  170203                          STFPS   R3                      ;SAVE FPS
064012  012701  003102                  MOV     #RECDST,R1              ;POINT TO RECEIVED DATA TABLE
064016  174011                          STD     AC0,(R1)                ;SAVE AC0  RESULT
064020  022703  000204                  CMP     #204,R3                 ;VERIFY STATUS
064024  001401                          BEQ     9$                      ;BRANCH IF GOOD
064026  104003                          ERROR   +3                  ;FPP ERROR
                                                                    ;BAD FPS
064030  012704  003254          9$:     MOV     #TAB6,R4                        ;POINT TO EXPECTED DATA
064034  004767  047422                  JSR     R7,DATVER                       ;VERIFY DATA
064040  005767  117014                  TST     COUNT
064044  001401                          BEQ     10$                     ;BRANCH IF GOOD
064046  104003                          ERROR   +3                  ;FPP ERROR
                                                                    ;BAD AC0
064050                          10$:

                                        ;
                                        ;
                                        ;
                                        ;-----------------------------------
                                        ;TEST CMPD
                                        ;
                                                ;
                                                ;
                                                ;
064050                          MCMPD:

                                        ;
                                        ;  CMPD WITH FRSC=AC0=0
064050  005037  002776                  CLR     @#FLAG                  ;SIGNAL THAT AC0 REMAINS CONSTANT
```

E3


```
13064  064054  004767  000152                  JSR     R7,CMPRTN               ;ROUTINE TO TEST DATA
13065  064060  000000  000000  000000          .WORD   0,0,0,0                 ;AC0 AT START
13066  064066  000000
13067  064070  000000  000000  000000          .WORD   0,0,0,0                 ;FSRC AT START
13068  064076  000000
13069  064100  000200                          .WORD   200                     ;FPS AT START (D)
13070  064102  000204                          .WORD   204                     ;FPS AT END
13071                                  ;  CMPD WITH EXP[FSRC]=0, EXP[AC0]=0
13072  064104  012737  000001  002776          MOV     #1,@#FLAG               ;SIGNAL THAT AC0 WILL = 0
13073  064112  004767  000114                  JSR     R7,CMPRTN               ;ROUTINE TO TEST DATA
13074  064116  000000  000000  000000          .WORD   0,0,0,125252            ;AC0 AT START
13075  064124  125252
13076  064126  000100  000022  000123          .WORD   100,22,123,123          ;FSRC AT START
13077  064134  000123
13078  064136  000200                          .WORD   200                     ;FPS AT START (D)
13079  064140  000204                          .WORD   204                     ;FPS AT END
13080                                  ;  CMPD FSRC>EXP[AC0]=0
13081  064142  005037  002776                  CLR     @#FLAG                  ;AC0 REMAINS UNCHANGED
13082  064146  004767  000060                  JSR     R7,CMPRTN               ;ROUTINE TO TEST DATA
13083  064152  000400  012346  012346          .WORD   400,12346,12346,23      ;AC0 AT START
13084  064160  000023
13085  064162  000200  000000  000000          .WORD   200,0,0,0               ;FSRC AT START
13086  064170  000000
13087  064172  000200                          .WORD   200                     ;FPS AT START (D)
13088  064174  000210                          .WORD   210                     ;FPS AT END
13089                                  ; CMPD FSRC=AC0>0
13090  064176  004767  000030                  JSR     R7,CMPRTN               ;ROUTINE TO TEST DATA
13091  064202  077777  177777  177777          .WORD   77777,-1,-1,-1          ;AC0 AT START
13092  064210  177777
13093  064212  077777  177777  177777          .WORD   77777,-1,-1,-1          ;FSRC AT START
13094  064220  177777
13095  064222  000200                          .WORD   200                     ;FPS AT START (D)
13096  064224  000204                          .WORD   204                     ;FPS AT END
13097  064226  000167  000116                  JMP     HOP44                   ;HOP OVER SUBROUTINE
13098
13099                                  ;*X*X*X
```

```
13120  064260  170102                          LDFPS   R2                      ;LOAD TEST FPS
13121  064262  173414                  1$:     CMPD    (R4),AC0                ;*TEST INSTRUCTION
13122  064264  170203                          STFPS   R3                      ;SAVE FPS
13123  064266  012702  000200                  MOV     #200,R2         ;SET FPP TO DOUBLE
13124  064272  170102                          LDFPS   R2
13125  064274  012701  003102                  MOV     #RECDST,R1              ;POINT TO RECEIVED DATA TABLE
13126  064300  174011                          STD     AC0,(R1)                ;SAVE AC0  RESULT
13127  064302  026503  000022                  CMP     22(R5),R3               ;VERIFY STATUS
13128  064306  001401                          BEQ     2$                      ;BRANCH IF GOOD
13129  064310  104003                          ERROR   .3              ;FPP ERROR
13130                                          ;BAD FPS
13131  064312  005737  002776          2$:     TST     @#FLAG                  ;SEE IF AC0 REMAINS UNCHANGED
13132  064316  001403                          BEQ     3$                      ;BRANCH IF AC0 STAYS THE SAME
13133  064320  012704  003254                  MOV     #TAB6,R4                ;AC0=0
13134  064324  000401                          BR      4$                      ;GO VERIFY  DATA
13135  064326  010504                  3$:     MOV     R5,R4                   ;POINT TO EXPECTED DATA
13136  064330  004767  047126          4$:     JSR     R7,DATVER                       ;VERIFY DATA
13137  064334  005767  116520                  TST     COUNT
13138  064340  001401                          BEQ     5$                      ;BRANCH IF GOOD
13139  064342  104003                          ERROR   .3              ;FPP ERROR
13140                                          ;BAD AC0
13141  064344  000165  000024          5$:     JMP     24(R5)                  ;RETURN
13142  064350                          HOP44:
13143
13144
13145
13146                                  ;
13147                                  ;
13148                                  ;-----------------------------------
13149                                  ;TEST DIVF
13150                                  ;
13151                                          ;
13152                                          ;
13153                                          ;
13154  064350                          MDIVF:
13155
13156                                  ;
13157                                  ;1/EXP[AC]=FSRC=0
13158  064350  012737  000002  002776          MOV     #2,@#FLAG               ;NO INTERRUPT, BUT FEC
13159  064356  004767  000706                  JSR     R7,DVFSUB               ;DO TEST
13160  064362  000100  000027                  .WORD   100,27          ;AC0
13161  064366  000000  000000                  .WORD   0,0             ;FSRC
13162  064372  000100  000027                  .WORD   100,27          ;RESULT
13163  064376  040000                          .WORD   40000           ; TEST FPS
13164  064400  140000                          .WORD   140000          ;RESULT FPS
13165  064402  000004                          .WORD   4               ;FEC
13166                                  ;2/AC=EXP[FSRC]=0
13167                                  ;TRAPS ENABLED
13168  064404  012737  000001  002776          MOV     #1,@#FLAG               ;INTERRUPT
13169  064412  004767  000652                  JSR     R7,DVFSUB               ;DO TEST
13170  064416  000000  000000                  .WORD   0,0             ;AC0
13171  064422  000100  000000                  .WORD   100,0           ;FSRC
13172  064426  000000  000000                  .WORD   0,0             ;RESULT
13173  064432  000000                          .WORD   0               ; TEST FPS
13174  064434  100000                          .WORD   100000          ;RESULT FPS
13175  064436  000004                          .WORD   4               ;FEC
```

```
                                        ;3/FSRC>AC0=0
064440  005037  002776                          CLR     @#FLAG          ;NO INTERRUPT
064444  004767  000620                          JSR     R7,DVFSUB                       ;DO TEST
064450  000177  000234                          .WORD   177,234         ;AC0
064454  004100  000000                          .WORD   4100,0          ;FSRC
064460  000000  000000                          .WORD   0,0             ;RESULT
064464  007400                                  .WORD   7400            ; TEST FPS
064466  007404                                  .WORD   7404            ;RESULT FPS
                                        ;4/AC0>EXP[FSRC]=0
064470  012737  000001  002776                  MOV     #1,@#FLAG                       ;INTERRUPT
064476  004767  000566                          JSR     R7,DVFSUB                       ;DO TEST
064502  040200  104210                          .WORD   40200,104210    ;AC0
064506  000125  025252                          .WORD   125,25252       ;FSRC
064512  040200  104210                          .WORD   40200,104210    ;RESULT
064516  007557                                  .WORD   7557            ; TEST FPS
064520  107557                                  .WORD   107557          ;RESULT FPS
064522  000004                                  .WORD   4               ;FEC
                                        ;5/EXP[AC]=EXP[FSRC]
064524  005037  002776                          CLR     @#FLAG          ;NO INTERRUPT
064530  004767  000534                          JSR     R7,DVFSUB                       ;DO TEST
064534  077760  177777                          .WORD   77760,-1                        ;AC0
064540  077760  000000                          .WORD   77760,0         ;FSRC
064544  040200  104210                          .WORD   40200,104210    ;RESULT
064550  007414                                  .WORD   7414            ; TEST FPS
064552  007400                                  .WORD   7400            ;RESULT FPS
                                        ;6/AC=FSRC
064554  005037  002776                          CLR     @#FLAG          ;NO INTERRUPT
064560  004767  000504                          JSR     R7,DVFSUB                       ;DO TEST
064564  052525  052525                          .WORD   52525,52525                     ;AC0
064570  052525  052525                          .WORD   52525,52525                     ;FSRC
064574  040200  000000                          .WORD   40200,0         ;RESULT
064600  007400                                  .WORD   7400            ; TEST FPS
064602  007400                                  .WORD   7400            ;RESULT FPS
                                        ;7/FSRC>0<AC0, ROUND
064604  005037  002776                          CLR     @#FLAG          ;NO INTERRUPT
064610  004767  000454                          JSR     R7,DVFSUB                       ;DO TEST
064614  077777  125252                          .WORD   77777,125252    ;AC0
064620  040300  000000                          .WORD   40300,0         ;FSRC
064624  077652  070707                          .WORD   77652,070707                    ;RESULT
064630  007400                                  .WORD   7400            ; TEST FPS
064632  007400                                  .WORD   7400            ;RESULT FPS
                                        ;8/AC>0<FSRC
064634  005037  002776                          CLR     @#FLAG          ;NO INTERRUPT
064640  004767  000424                          JSR     R7,DVFSUB                       ;DO TEST
064644  055377  177777                          .WORD   55377,-1        ;AC0
064650  055300  000000                          .WORD   55300,0         ;FSRC
064654  040252  125252                          .WORD   40252,125252    ;RESULT
064660  000000                                  .WORD   0               ; TEST FPS
064662  000000                                  .WORD   0               ;RESULT FPS
                                        ;9/FSRC>AC>0
064664  005037  002776                          CLR     @#FLAG          ;NO INTERRUPT
064670  004767  000374                          JSR     R7,DVFSUB                       ;DO TEST
064674  064600  000001                          .WORD   64600,1         ;AC0
064700  066600  000000                          .WORD   66600,0         ;FSRC
064704  036200  000001                          .WORD   36200,1         ;RESULT
064710  000000                                  .WORD   0               ; TEST FPS
```

```
13232  064712  000000                          .WORD   0               ;RESULT FPS
13233                                  ;10/AC>FSRC>0
13234  064714  005037  002776                  CLR     @#FLAG          ;NO INTERRUPT
13235  064720  004767  000344                  JSR     R7,DVFSUB               ;DO TEST
13236  064724  012345  156024                  .WORD   12345,156024    ;AC0
13237  064730  005600  000000                  .WORD   05600,0         ;FSRC
13238  064734  044745  156024                  .WORD   44745,156024            ;RESULT
13239  064740  000017                          .WORD   17              ; TEST FPS
13240  064742  000000                          .WORD   0               ;RESULT FPS
13241                                  ;11/FSRC<0
13242  064744  005037  002776                  CLR     @#FLAG          ;NO INTERRUPT
13243  064750  004767  000314                  JSR     R7,DVFSUB               ;DO TEST
13244  064754  040422  101010                  .WORD   40422,101010    ;AC0
13245  064760  140511  101010                  .WORD   140511,101010   ;FSRC
13246  064764  140072  020167                  .WORD   140072,20167            ;RESULT
13247  064770  000057                          .WORD   57              ; TEST FPS
13248  064772  000050                          .WORD   50              ;RESULT FPS
13249                                  ;12/AC<0
13250  064774  005037  002776                  CLR     @#FLAG          ;NO INTERRUPT
13251  065000  004767  000264                  JSR     R7,DVFSUB               ;DO TEST
13252  065004  160077  000101                  .WORD   160077,101      ;AC0
13253  065010  040417  177777                  .WORD   40417,-1        ;FSRC
13254  065014  157651  143527                  .WORD   157651,143527           ;RESULT
13255  065020  000007                          .WORD   7               ; TEST FPS
13256  065022  000010                          .WORD   10              ;RESULT FPS
13257                                  ;13/TRUNCATE TEST
13258  065024  005037  002776                  CLR     @#FLAG          ;NO INTERRUPT
13259  065030  004767  000234                  JSR     R7,DVFSUB               ;DO TEST
13260  065034  060100  000177                  .WORD   60100,177       ;AC0
13261  065040  040300  000000                  .WORD   40300,0         ;FSRC
13262  065044  060000  000124                  .WORD   60000,124               ;RESULT
13263  065050  000040                          .WORD   40              ; TEST FPS
13264  065052  000040                          .WORD   40              ;RESULT FPS
13265                                  ;14/ROUND TEST
13266
13267  065054  005037  002776                  CLR     @#FLAG          ;NO INTERRUPT
13268  065060  004767  000204                  JSR     R7,DVFSUB               ;DO TEST
13269  065064  060100  000177                  .WORD   60100,177       ;AC0
13270  065070  040300  000000                  .WORD   40300,0 ;FSRC
13271  065074  060000  000125                  .WORD   60000,125               ;RESULT
13272  065100  000000                          .WORD   0               ; TEST FPS
13273  065102  000000                          .WORD   0               ;RESULT FPS
13274                                  ;15/OVERFLOW, INTERRUPTS ENABLED
13275  065104  012737  000001  002776          MOV     #1,@#FLAG               ;INTERRUPT
13276  065112  004767  000152                  JSR     R7,DVFSUB               ;DO TEST
13277  065116  177700  000000                  .WORD   177700,0                ;AC0
13278  065122  000200  000000                  .WORD   200,0           ;FSRC
13279  065126  137700  000000                  .WORD   137700,0                ;RESULT
13280  065132  001100                          .WORD   1100            ; TEST FPS
13281  065134  101112                          .WORD   101112          ;RESULT FPS
13282  065136  000010                          .WORD   10              ;FEC
13283                                  ;16/OVERFLOW, TRAPS DISABLED
13284  065140  012737  000002  002776          MOV     #2,@#FLAG               ;NO INTERRUPT
13285  065146  004767  000116                  JSR     R7,DVFSUB               ;DO TEST
13286  065152  000200  000000                  .WORD   200,0           ;AC0
13287  065156  177700  000000                  .WORD   177700,0                ;FSRC
```

```
13288  065162  000000  000000                  .WORD   0,0             ;RESULT
13289  065166  041100                          .WORD   41100           ; TEST FPS
13290  065170  041104                          .WORD   41104           ;RESULT FPS
13291  065172  000010                          .WORD   10              ;FEC OVERFLOW
13292                                  ;17/UNDERFLOW, TRAPS ENABLED, UV RESULT
13293  065174  012737  000001  002776          MOV     #1,@#FLAG               ;INTERRUPT
13294  065202  004767  000062                  JSR     R7,DVFSUB               ;DO TEST
13295  065206  100200  000000                  .WORD   100200,0                ;ACO
13296  065212  040377  177777                  .WORD   40377,-1                ;FSRC
13297  065216  100000  000001                  .WORD   100000,1                ;RESULT
13298  065222  002000                          .WORD   2000            ; TEST FPS
13299  065224  102014                          .WORD   102014          ;RESULT FPS
13300  065226  000012                          .WORD   12              ;FEC
13301                                  ;18/UNDERFLOW,TRAPS ENABLED, ROUND
13302  065230  012737  000001  002776          MOV     #1,@#FLAG               ;INTERRUPT
13303  065236  004767  000026                  JSR     R7,DVFSUB               ;DO TEST
13304  065242  030325  025252                  .WORD   30325,25252             ;ACO
13305  065246  076777  023456                  .WORD   76777,23456             ;FSRC
13306  065252  071525  157716                  .WORD   71525,157716            ;RESULT
13307  065256  002537                          .WORD   2537            ; TEST FPS
13308  065260  102500                          .WORD   102500          ;RESULT FPS
13309  065262  000012                          .WORD   12              ;FEC
13310                                  ;
13311
13312  065264  000167  000212                  JMP     HOP10           ;GO TO NEXT TEST
13313                                  ;*X*X*X
```

```
                                        ;
                                        ;INSTRUCTION TRAPPED
065346  032737  000001  002776  50$:    BIT     #1,@#FLAG                   ;SEE IF EXPECTING A TRAP
065354  001003                          BNE     51$                         ;BRANCH IF EXPECTING A TRAP
065356  104003                          ERROR   +3                  ;FPP ERROR
                                                                    ;INSTRUCTION WASNT SUPPOSE TO TRAP
065360  000167  000014                  JMP     2$                          ;REJOIN CODE
065364  012604                  51$:    MOV     (SP)+,R4                    ;SEE IF PC = INSTRUCTION
065366  005726                          TST     (SP)+                       ;CLEAN UP STACK
065370  022704  065326                  CMP     #1$,R4                      ;
065374  001401                          BEQ     2$                          ;BRANCH IF GOOD COMPARE
065376  104003                          ERROR   +3                  ;FPP ERROR
                                                                    ;PC WAS INCORRECT
                                        ;
                                        ;COMMON CODE FOR TRAP AND NO TRAP
065400  170203                  2$:     STFPS   R3                          ;SAVE FPS
065402  012702  000200                  MOV     #200,R2                     ;SET FPP TO DOUBLE
065406  170102                          LDFPS   R2
065410  012701  003102                  MOV     #RECDST,R1                  ;POINT TO RECEIVED DATA TABLE
065414  174011                          STD     AC0,(R1)                    ;SAVE AC0  RESULT
065416  026503  000016                  CMP     16(R5),R3                   ;VERIFY STATUS
065422  001401                          BEQ     3$                          ;BRANCH IF GOOD
065424  104003                          ERROR   +3                  ;FPP ERROR
                                                                    ;BAD FPS
065426  010504                  3$:     MOV     R5,R4                       ;POINT TO EXPECTED DATA
065430  062704  000010                  ADD     #10,R4
065434  004767  046004          4$:     JSR     R7,DATVFR                           ;VERIFY DATA
065440  005767  115414                  TST     COUNT
065444  001401                          BEQ     5$                          ;BRANCH IF GOOD
065446  104003                          ERROR   +3                  ;FPP ERROR
                                                                    ;BAD AC0
065450  005737  002776          5$:     TST     @#FLAG                      ;SEE IF NEED TO CHECK FEC
065454  001002                          BNE     6$                          ;BRANCH IF NEED TO CHECK
065456  000165  000020                  JMP     20(R5)                      ;RETURN FROM TEST
065462  170301                  6$:     STST    R1                          ;SAVE FEC
065464  016504  000020                  MOV     20(R5),R4                   ;GET FEC
065470  020401                          CMP     R4,R1                       ;VERIFY FEC
065472  001401                          BEQ     7$                          ;BRANCH IF GOOD
065474  104003                          ERROR   +3                  ;FPP ERROR
                                                                            ;BAD FEC
065476  000165  000022          7$:     JMP     22(R5)                      ;RETURN FROM TEST
                                        ;
065502                                  HOP10:
                                        ;
                                        ;
                                        ;-----------------------------------
                                        ;TEST DIVD
                                        ;
                                                ;
                                                ;
                                                ;
065502                                  MDIVD:

                                        ;
                                        ;1/AC=FSRC=0 TRAPS DISABLED
065502  012737  000002  002776                  MOV     #2,@#FLAG                   ;NO INTERRUPT
```

K3


```
13400  065510  004767  000516                  JSR     R7,DVDSUB               ;DO TEST
13401  065514  000000  000000  000000          .WORD   0,0,0,1         ;ACO
13402  065522  000001
13403  065524  000100  000000  000000          .WORD   100,0,0,0       ;FSRC
13404  065532  000000
13405  065534  000000  000000  000000          .WORD   0,0,0,1         ;RESULT
13406  065542  000001
13407  065544  040000                          .WORD   40000           ; TEST FPS
13408  065546  140000                          .WORD   140000          ;RESULT FPS
13409  065550  000004                          .WORD   4               ;FEC
13410                          ;2/FSRC=0, TRAPS ENABLED
13411  065552  012737  000001  002776          MOV     #1,@#FLAG               ;INTERRUPT
13412  065560  004767  000446                  JSR     R7,DVDSUB               ;DO TEST
13413  065564  000402  000000  000000          .WORD   402,0,0,0       ;ACO
13414  065572  000000
13415  065574  000000  000000  000000          .WORD   0,0,0,0         ;FSRC
13416  065602  000000
13417  065604  000402  000000  000000          .WORD   402,0,0,0               ;RESULT
13418  065612  000000
13419  065614  000200                          .WORD   200             ; TEST FPS
13420  065616  100200                          .WORD   100200          ;RESULT FPS
13421  065620  000004                          .WORD   4               ;FEC
13422                          ;3/ROUND
13423  065622  005037  002776                  CLR     @#FLAG          ;NO INTERRUPT
13424  065626  004767  000400                  JSR     R7,DVDSUB               ;DO TEST
13425  065632  034300  000000  000000          .WORD   34300,0,0,1     ;ACO
13426  065640  000001
13427  065642  140300  000000  000000          .WORD   140300,0,0,0    ;FSRC
13428  065650  000000
13429  065652  134200  000000  000000          .WORD   134200,0,0,1    ;RESULT
13430  065660  000001
13431  065662  000200                          .WORD   200             ; TEST FPS
13432  065664  000210                          .WORD   210             ;RESULT FPS .
13433                          ;4/TRUNCATE
13434  065666  005037  002776                  CLR     @#FLAG          ;NO INTERRUPT
13435  065672  004767  000334                  JSR     R7,DVDSUB               ;DO TEST
13436  065676  034300  000000  000000          .WORD   34300,0,0,1     ;ACO
13437  065704  000001
13438  065706  140300  000000  000000          .WORD   140300,0,0,0    ;FSRC
13439  065714  000000
13440  065716  134200  000000  000000          .WORD   134200,0,0,0    ;RESULT
13441  065724  000000
13442  065726  000240                          .WORD   240             ; TEST FPS
13443  065730  000250                          .WORD   250             ;RESULT FPS
13444                          ;5/ROUND NEGATIVE AC, FSRC
13445  065732  005037  002776                  CLR     @#FLAG          ;NO INTERRUPT
13446  065736  004767  000270                  JSR     R7,DVDSUB               ;DO TEST
13447  065742  177642  000000  000000          .WORD   177642,0,0,151  ;ACO
13448  065750  000151
13449  065752  166600  000000  000000          .WORD   166600,0,0,123  ;FSRC
13450  065760  000123
13451  065762  051242  000000  000000          .WORD   51242,0,0,0             ;RESULT
13452  065770  000000
13453  065772  000200                          .WORD   200             ; TEST FPS
13454  065774  000200                          .WORD   200             ;RESULT FPS
13455                          ;6/TRUNCATE NEAGTIVE AC, FSRC
```

```
13456  065776  005037  002776                  CLR     @#FLAG          ;NO INTERRUPT
13457  066002  004767  000224                  JSR     R7,DVDSUB               ;DO TEST
13458  066006  177642  000000  000000          .WORD   177642,0,0,151  ;AC0
13459  066014  000151
13460  066016  166600  000000  000000          .WORD   166600,0,0,123  ;FSRC
13461  066024  000123
13462  066026  051241  177777  177777          .WORD   51241,-1,-1,-1          ;RESULT
13463  066034  177777
13464  066036  000240                          .WORD   240             ; TEST FPS
13465  066040  000240                          .WORD   240             ;RESULT FPS
13466                          ;7/AC=FSRC
13467  066042  005037  002776                  CLR     @#FLAG          ;NO INTERRUPT
13468  066046  004767  000160                  JSR     R7,DVDSUB               ;DO TEST
13469  066052  055521  047621  100333          .WORD   55521,47621,100333,-1   ;AC0
13470  066060  177777
13471  066062  055521  047621  100333          .WORD   55521,47621,100333,-1   ;FSRC
13472  066070  177777
13473  066072  040200  000000  000000          .WORD   40200,0,0,0             ;RESULT
13474  066100  000000
13475  066102  007717                          .WORD   7717            ; TEST FPS
13476  066104  007700                          .WORD   7700            ;RESULT FPS
13477                          ;8/UNDERFLOW TRAPS ENABLED, UV RESULT
13478  066106  012737  000001  002776          MOV     #1,@#FLAG               ;INTERRUPT
13479  066114  004767  000112                  JSR     R7,DVDSUB               ;DO TEST
13480  066120  100200  000000  000000          .WORD   100200,0,0,0            ;AC0
13481  066126  000000
13482  066130  077777  000000  000000          .WORD   77777,0,0,0             ;FSRC
13483  066136  000000
13484  066140  140400  100200  100200          .WORD   140400,100200,100200,100201     ;RESULT
13485  066146  100201
13486  066150  002200                          .WORD   2200            ; TEST FPS
13487  066152  102210                          .WORD   102210          ;RESULT FPS
13488  066154  000012                          .WORD   12              ;FEC
13489                          ;9/OVERFLOW TRAPS ENABLED
13490  066156  012737  000001  002776          MOV     #1,@#FLAG               ;INTERRUPT
13491  066164  004767  000042                  JSR     R7,DVDSUB               ;DO TEST
13492  066170  077000  123465  012346          .WORD   77000,123465,12346,525          ;AC0
13493  066176  000525
13494  066200  000303  000001  140000          .WORD   303,1,140000,140001             ;FSRC
13495  066206  140001
13496  066210  036650  163002  103645          .WORD   36650,163002,103645,64003       ;RESULT
13497  066216  064003
13498  066220  001700                          .WORD   1700            ; TEST FPS
13499  066222  101702                          .WORD   101702          ;RESULT FPS
13500  066224  000010                          .WORD   10              ;FEC
13501                          ;
13502
13503  066226  000167  000212                  JMP     HOP11                   ;HOP OVER SUBROUTINE
13504                          ;*X*X*X
```

```
13512                                   ;               (FEC)
13513                                   ;
13514                                   ;*X*X*X
```

```
13568  066416  001002                  BNE     6$                      ;BRANCH IF NEED TO CHECK
13569  066420  000165  000034          JMP     34(R5)                  ;RETURN FROM TEST
13570  066424  170301          6$:     STST    R1                      ;SAVE FEC
13571  066426  016504  000034          MOV     34(R5),R4               ;GET FEC
13572  066432  020401                  CMP     R4,R1                   ;VERIFY FEC
13573  066434  001401                  BEQ     7$                      ;BRANCH IF GOOD
13574  066436  104003                  ERROR   .3              ;FPP ERROR
13575                                                                  ;BAD FEC
13576  066440  000165  000036  7$:     JMP     36(R5)                  ;RETURN FROM TEST
13577                          ;
13578  066444                  MOP11:
13579                          ;
13580                          ;
13581                          ;-----------------------------------
13582                          ;TEST MULF
13583                          ;
13584                                  ;
13585                                  ;
13586                                  ;
13587  066444                  MMULF:
13588
13589                          ;
13590                          ;1/AC0=FSRC=0 -INTERRUPTS DISABLED
13591  066444  005037  002776          CLR     @#FLAG                  ;NO INTERRUPT
13592  066450  004767  000564          JSR     R7,MLFSUB                       ;DO TEST
13593  066454  000000  000000          .WORD   0.0     ;AC0
13594  066460  000000  000000          .WORD   0.0     ;FSRC
13595  066464  000000  000000          .WORD   0.0                     ;RESULT
13596  066470  007517                  .WORD   7517                    ; TEST FPS
13597  066472  007504                  .WORD   7504                    ;RESULTANT FPS
13598
13599  066474  005037  002776  ;2/AC>FSRC=0 - INTERRUPTS ON
13600  066500  004767  000534          CLR     @#FLAG                  ;NO INTERRUPT
13601  066504  000200  000000          JSR     R7,MLFSUB                       ;DO TEST
13602  066510  000000  000000          .WORD   200.0   ;AC0
13603  066514  000000  000000          .WORD   0.0     ;FSRC
13604  066520  000013                  .WORD   0.0                     ;RESULT
13605  066522  000004                  .WORD   13                      ; TEST FPS
13606                                  .WORD   4                       ;RESULTANT FPS
13607  066524  005037  002776  ;3/AC=0 FSRC>0 -
13608  066530  004767  000504          CLR     @#FLAG                  ;NO INTERRUPT
13609  066534  000100  000000          JSR     R7,MLFSUB                       ;DO TEST
13610  066540  000300  000000          .WORD   100.0   ;AC0
13611  066544  000000  000000          .WORD   300.0   ;FSRC
13612  066550  007500                  .WORD   0.0                     ;RESULT
13613  066552  007504                  .WORD   7500                    ; TEST FPS
13614                                  .WORD   7504                    ;RESULTANT FPS
13615  066554  005037  002776  ;4/AC=1 >FSRC - ROUND
13616  066560  004767  000454          CLR     @#FLAG                  ;NO INTERRUPT
13617  066564  040200  000000          JSR     R7,MLFSUB                       ;DO TEST
13618  066570  040177  177777          .WORD   40200.0 ;AC0
13619  066574  040177  177777          .WORD   40177.-1                ;FSRC
13620  066600  000000                  .WORD   40177.-1                        ;RESULT
13621  066602  000000                  .WORD   0                       ; TEST FPS
13622                                  .WORD   0                       ;RESULTANT FPS
13623  066604  005037  002776  ;5/TRUNCATE
                                        CLR     @#FLAG                  ;NO INTERRUPT
```

```
066610  004767  000424                  JSR     R7,MLFSUB                ;DO TEST
066614  040177  177777                  .WORD   40177,-1         ;AC0
066620  040200  000000                  .WORD   40200,0 ;FSRC
066624  040177  177777                  .WORD   40177, 1                 ;RESULT
066630  000040                          .WORD   40               ; TEST FPS
066632  000040                          .WORD   40               ;RESULTANT FPS
                                ;6/NORMALIZE
066634  005037  002776                  CLR     @#FLAG           ;NO INTERRUPT
066640  004767  000374                  JSR     R7,MLFSUB                ;DO TEST
066644  040100  000000                  .WORD   40100,0 ;AC0
066650  040100  000000                  .WORD   40100,0 ;FSRC
066654  040020  000000                  .WORD   40020,0          ;RESULT
066660  000012                          .WORD   12               ; TEST FPS
066662  000000                          .WORD   0                ;RESULTANT FPS
                                ;7/ROUND
066664  005037  002776                  CLR     @#FLAG           ;NO INTERRUPT
066670  004767  000344                  JSR     R7,MLFSUB                ;DO TEST
066674  017500  000000                  .WORD   17500,0 ;AC0
066700  023652  125252                  .WORD   23652,125252     ;FSRC
066704  003177  177777                  .WORD   3177,-1          ;RESULT
066710  007417                          .WORD   7417             ; TEST FPS
066712  007400                          .WORD   7400             ;RESULTANT FPS
                                ;8/AC>0>FSRC ROUND
066714  005037  002776                  CLR     @#FLAG           ;NO INTERRUPT
066720  004767  000314                  JSR     R7,MLFSUB                ;DO TEST
066724  040342  177777                  .WORD   40342,-1         ;AC0
066730  176543  025252                  .WORD   176543,025252    ;FSRC
066734  176711  067324                  .WORD   176711,67324             ;RESULT
066740  007500                          .WORD   7500             ; TEST FPS
066742  007510                          .WORD   7510             ;RESULTANT FPS
                                ;9/IAC<FSRC<0, ROUND
066744  005037  002776                  CLR     @#FLAG           ;NO INTERRUPT
066750  004767  000264                  JSR     R7,MLFSUB                ;DO TEST
066754  144600  000000                  .WORD   144600,0                 ;AC0
066760  154000  000000                  .WORD   154000,0                 ;FSRC
066764  060400  000000                  .WORD   60400,0          ;RESULT
066770  000017                          .WORD   17               ; TEST FPS
066772  000000                          .WORD   0                ;RESULT FPS
                                ;10/AC<FSRC, ROUND
066774  005037  002776                  CLR     @#FLAG           ;NO INTERRUPT
067000  004767  000234                  JSR     R7,MLFSUB                ;DO TEST
067004  060000  000000                  .WORD   60000,0          ;AC0
067010  140377  177776                  .WORD   140377,177776            ;FSRC
067014  160177  177776                  .WORD   160177,177776            ;RESULT
067020  000017                          .WORD   17               ; TEST FPS
067022  000010                          .WORD   10               ;RESULT FPS
                                ;11/AC>0>FSRC, TRUNCATE
067024  005037  002776                  CLR     @#FLAG           ;NO INTERRUPT
067030  004767  000204                  JSR     R7,MLFSUB                ;DO TEST
067034  060000  000000                  .WORD   60000,0          ;AC0
067040  140377  177776                  .WORD   140377,177776            ;FSRC
067044  160177  177776                  .WORD   160177,177776            ;RESULT
067050  007547                          .WORD   7547             ; TEST FPS
067052  007550                          .WORD   7550             ;RESULT FPS
                                ;12/UNDERFLOW, NO INTERRUPTS
067054  012737  000002  002776          MOV     #2,@#FLAG                ;NO INTERRUPT
```

```
13680  067062  004767  000152                JSR     R7,MLFSUB               ;DO TEST
13681  067066  000200  000001                .WORD   200,1           ;ACO
13682  067072  000200  000001                .WORD   200,1           ;FSRC
13683  067076  040200  000002                .WORD   40200,2         ;RESULT
13684  067102  042117                        .WORD   42117           ; TEST FPS
13685  067104  142100                        .WORD   142100          ;RESULT FPS
13686  067106  000012                        .WORD   12              ;FEC
13687                          ;13/OVERFLOW, TRAP
13688  067110  012737  000001  002776        MOV     #1,@#FLAG               ;INTERRUPT
13689  067116  004767  000116                JSR     R7,MLFSUB               ;DO TEST
13690  067122  177777  177777                .WORD   177777,-1               ;ACO
13691  067126  040300  000000                .WORD   40300,0         ;FSRC
13692  067132  100077  177777                .WORD   100077,-1               ;RESULT
13693  067136  001117                        .WORD   1117            ; TEST FPS
13694  067140  101116                        .WORD   101116          ;RESULT FPS
13695  067142  000010                        .WORD   10              ;FEC
13696                          ;14/OVERFLOW NO TRAP
13697  067144  012737  000002  002776        MOV     #2,@#FLAG               ;NO INTERRUPT
13698  067152  004767  000062                JSR     R7,MLFSUB               ;DO TEST
13699  067156  077700  000000                .WORD   77700,0         ;ACO
13700  067162  077700  000000                .WORD   77700,0         ;FSRC
13701  067166  000000  000000                .WORD   0,0             ;RESULT
13702  067172  040117                        .WORD   40117           ; TEST FPS
13703  067174  040106                        .WORD   40106           ;RESULT FPS
13704  067176  000010                        .WORD   10              ;FEC
13705                          ;15/UNDEFINED VARIABLE IN FSRC, TRAP ENABLED
13706  067200  012737  000001  002776        MOV     #1,@#FLAG               ;INTERRUPT
13707  067206  004767  000026                JSR     R7,MLFSUB               ;DO TEST
13708  067212  123465  000000                .WORD   123465,0                ;ACO
13709  067216  100022  000000                .WORD   100022,0                ;FSRC
13710  067222  123465  000000                .WORD   123465,0                ;RESULT
13711  067226  004000                        .WORD   4000            ; TEST FPS
13712  067230  104000                        .WORD   104000          ;RESULT FPS
13713  067232  000014                        .WORD   14              ;FEC
13714                          ;
13715
13716  067234  000167  000212                JMP     HOP12
13717                          ;*X*X*X
```

```
13736  067266  016502  000014          MOV     14(R5),R2               ;GET TEST FPS
13737  067272  170102                  LDFPS   R2                      ;LOAD TEST FPS
13738                                  ;
13739  067274  171014                  MULF    (R4),AC0                ;*TEST INSTRUCTION
13740  067276  170001                  1$:     SETF                    ;WAIT FOR POSSIBLE FPA TRAP.
13741                                  ;
13742                                  ;INSTRUCTION DIDNT TRAP
13743  067300  032737  000001  002776          BIT     #1,@#FLAG               ;VERIFY A NO TRAP CONDITION
13744  067306  001420                  BEQ     2$                      ;BRANCH IF GOOD
13745  067310  104003                  ERROR   .3                      ;FPP ERROR
13746                                  ;INSTRUCTION SHOULD HAVE TRAPPED
13747  067312  000167  000032          JMP     2$                      ;REJOIN CODE
13748                                  ;
13749                                  ;INSTRUCTION TRAPPED
13750  067316  032737  000001  002776  50$:    BIT     #1,@#FLAG               ;SEE IF EXPECTING A TRAP
13751  067324  001003                  BNE     51$                     ;BRANCH IF EXPECTING A TRAP
13752  067326  104003                  ERROR   .3                      ;FPP ERROR
13753                                  ;INSTRUCTION WASNT SUPPOSE TO TRAP
13754  067330  000167  000014          JMP     2$                      ;REJOIN CODE
13755  067334  012604                  51$:    MOV     (SP)+,R4                ;SEE IF PC = INSTRUCTION
13756  067336  005726                  TST     (SP)+                   ;CLEAN UP STACK
13757  067340  022704  067276          CMP     #1$,R4                  ;
13758  067344  001401                  BEQ     2$                      ;BRANCH IF GOOD COMPARE
13759  067346  104003                  ERROR   .3                      ;FPP ERROR
13760                                  ;PC WAS INCORRECT
13761
13762                                  ;COMMON CODE FOR TRAP AND NO TRAP
13763  067350  170203                  2$:     STFPS   R3                      ;SAVE FPS
13764  067352  012702  000200          MOV     #200,R2                 ;SET FPP TO DOUBLE
13765  067356  170102                  LDFPS   R2
13766  067360  012701  003102          MOV     #RECDST,R1              ;POINT TO RECEIVED DATA TABLE
13767  067364  174011                  STD     AC0,(R1)                ;SAVE AC0  RESULT
13768  067366  026503  000016          CMP     16(R5),R3               ;VERIFY STATUS
13769  067372  001401                  BEQ     3$                      ;BRANCH IF GOOD
13770  067374  104003                  ERROR   .3                      ;FPP ERROR
13771                                  ;BAD FPS
13772  067376  010504                  3$:     MOV     R5,R4                   ;POINT TO EXPECTED DATA
13773  067400  062704  000010          ADD     #10,R4
13774  067404  004767  044034          4$:     JSR     R7,DATVFR               ;VERIFY DATA
13775  067410  005767  113444          TST     COUNT
13776  067414  001401                  BEQ     5$                      ;BRANCH IF GOOD
13777  067416  104003                  ERROR   .3                      ;FPP ERROR
13778                                  ;BAD AC0
13779  067420  005737  002776          5$:     TST     @#FLAG                  ;SEE IF NEED TO CHECK FEC
13780  067424  001002                  BNE     6$                      ;BRANCH IF NEED TO CHECK
13781  067426  000165  000020          JMP     20(R5)                  ;RETURN FROM TEST
13782                                  ;VERIFY ERROR STATUS
13783  067432  170301                  6$:     STST    R1                      ;SAVE FEC
13784  067434  016504  000020          MOV     20(R5),R4               ;GET FEC
13785  067440  020401                  CMP     R4,R1                   ;VERIFY FEC
13786  067442  001401                  BEQ     7$                      ;BRANCH IF GOOD
13787  067444  104003                  ERROR   .3                      ;FPP ERROR
13788                                  ;BAD FEC
13789  067446  000165  000022          7$:     JMP     22(R5)          ;RETURN FROM TEST
13790  067452                          MOP12:
13791                                  ;
```

```
13792                                   ;
13793                                   ;-------------------------------
13794                                   ;TEST MULD
13795                                   ;
13796                                           ;
13797                                           ;
13798                                           ;
13799   067452                          MMULD:
13800
13801                                   ;
13802                                   ;1/AC=0
13803   067452  005037  002776                   CLR     @#FLAG          ;NO INTERRUPT
13804   067456  004767  000554                   JSR     R7,MLDSUB               ;DO TEST
13805   067462  000100  000000  000000           .WORD   100,0,0,0       ;AC0
13806   067470  000000
13807   067472  000411  177777  000000           .WORD   411,-1,0,1      ;FSRC
13808   067500  000001
13809   067502  000000  000000  000000           .WORD   0,0,0,0         ;RESULT
13810   067510  000000
13811   067512  000200                           .WORD   200             ; TEST FPS
13812   067514  000204                           .WORD   204             ;RESULTANT FPS
13813                                   ;2/FSRC=0
13814   067516  005037  002776                   CLR     @#FLAG          ;NO INTERRUPT
13815   067522  004767  000510                   JSR     R7,MLDSUB               ;DO TEST
13816   067526  077777  000000  000000           .WORD   77777,0,0,0     ;AC0
13817   067534  000000
13818   067536  000000  000000  000000           .WORD   0,0,0,0 ;FSRC
13819   067544  000000
13820   067546  000000  000000  000000           .WORD   0,0,0,0         ;RESULT
13821   067554  000000
13822   067556  007700                           .WORD   7700            ; TEST FPS
13823   067560  007704                           .WORD   7704            ;RESULTANT FPS
13824                                   ;3/AC=1
13825   067562  005037  002776                   CLR     @#FLAG          ;NO INTERRUPT
13826   067566  004767  000444                   JSR     R7,MLDSUB               ;DO TEST
13827   067572  040200  000000  000000           .WORD   40200,0,0,0     ;AC0
13828   067600  000000
13829   067602  000277  177777  177777           .WORD   277,-1,-1,-1    ;FSRC
13830   067610  177777
13831   067612  000277  177777  177777           .WORD   277,-1,-1,-1            ;RESULT
13832   067620  177777
13833   067622  007717                           .WORD   7717            ; TEST FPS
13834   067624  007700                           .WORD   7700            ;RESULTANT FPS
13835                                   ;4/AC>FSRC>0, TRUNCATE
13836   067626  005037  002776                   CLR     @#FLAG          ;NO INTERRUPT
13837   067632  004767  000400                   JSR     R7,MLDSUB               ;DO TEST
13838   067636  065500  000000  000000           .WORD   65500,0,0,1     ;AC0
13839   067644  000001
13840   067646  037577  177777  177777           .WORD   37577,-1,-1,-2  ;FSRC
13841   067654  177776
13842   067656  065077  177777  177777           .WORD   65077,-1,-1,-1          ;RESULT
13843   067664  177777
13844   067666  007717                           .WORD   7717            ; TEST FPS
13845   067670  007700                           .WORD   7700            ;RESULTANT FPS
13846                                   ;5/AC<FSRC<0
13847   067672  005037  002776                   CLR     @#FLAG          ;NO INTERRUPT
```

```
13848  067676  004767  000334                  JSR     R7,MLDSUB               ;DO TEST
13849  067702  137577  177777  177777          .WORD   137577,-1,-1,-2 ;ACO
13850  067710  177776
13851  067712  165400  000000  000000          .WORD   165400,0,0,1    ;FSRC
13852  067720  000001
13853  067722  065000  000000  000000          .WORD   65000,0,0,0             ;RESULT
13854  067730  000000
13855  067732  007717                          .WORD   7717            ; TEST FPS
13856  067734  007700                          .WORD   7700            ;RESULTANT FPS
13857                          ;6/AC>FSRC>0
13858  067736  005037  002776                  CLR     @#FLAG          ;NO INTERRUPT
13859  067742  004767  000270                  JSR     R7,MLDSUB               ;DO TEST
13860  067746  017500  000000  000000          .WORD   17500,0,0,0     ;ACO
13861  067754  000000
13862  067756  123652  125252  125252          .WORD   123652,125252,125252,125252     ;FSRC
13863  067764  125252
13864  067766  103177  177777  177777          .WORD   103177,-1,-1,-1         ;RESULT
13865  067774  177777
13866  067776  000200                          .WORD   200             ; TEST FPS
13867  070000  000210                          .WORD   210             ;RESULTANT FPS
13868                          ;7/UNDERFLOW, TRAPS DISABLED
13869  070002  005037  002776                  CLR     @#FLAG                  ;NO INTERRUPT
13870  070006  004767  000224                  JSR     R7,MLDSUB               ;DO TEST
13871  070012  000300  000000  000000          .WORD   300,0,0,252             ;ACO
13872  070020  000252
13873  070022  000377  000001  000002          .WORD   377,1,2,3               ;FSRC
13874  070030  000003
13875  070032  000000  000000  000000          .WORD   0,0,0,0         ;RESULT
13876  070040  000000
13877  070042  005740                          .WORD   5740            ; TEST FPS
13878  070044  005744                          .WORD   5744            ;RESULT FPS
13879                          ;8/UNDERFLOW, TRAP ENABLED
13880  070046  012737  000001  002776          MOV     #1,@#FLAG               ;INTERRUPT
13881  070054  004767  000156                  JSR     R7,MLDSUB               ;DO TEST
13882  070060  100277  000001  000002          .WORD   100277,1,2,-1           ;ACO
13883  070066  177777
13884  070070  100300  000001  000001          .WORD   100300,1,1,1            ;FSRC
13885  070076  000001
13886  070100  040417  040001  077403          .WORD   40417,40001,77403,0             ;RESULT
13887  070106  000000
13888  070110  002217                          .WORD   2217            ; TEST FPS
13889  070112  102200                          .WORD   102200          ;RESULT FPS
13890  070114  000012                          .WORD   12              ;FEC
13891                          ;9/OVERFLOW, TRAPS DISABLED
13892  070116  005037  002776                  CLR     @#FLAG                  ;NO INTERRUPT
13893  070122  004767  000110                  JSR     R7,MLDSUB               ;DO TEST
13894  070126  177777  177777  177777          .WORD   -1,-1,-1,-1             ;ACO
13895  070134  177777
13896  070136  040200  177777  177777          .WORD   40200,-1,-1,-1          ;FSRC
13897  070144  177777
13898  070146  000000  000000  000000          .WORD   0,0,0,0         ;RESULT
13899  070154  000000
13900  070156  006740                          .WORD   6740            ; TEST FPS
13901  070160  006746                          .WORD   6746            ;RESULT FPS
13902                          ;10/OVERFLOW, TRAPS ENABLED
13903  070162  012737  000001  002776          MOV     #1,@#FLAG               ;INTERRUPT
```

```
13904  070170  004767  000042                  JSR     R7,MLDSUB               ;DO TEST
13905  070174  157700  025252  025252          .WORD   157700,25252,25252,25252        ;AC0
13906  070202  025252
13907  070204  167700  000000  000000          .WORD   167700,0,0,0            ;FSRC
13908  070212  000000
13909  070214  007420  017777  117777          .WORD   7420,017777,117777,117777       ;RESULT
13910  070222  117777
13911  070224  001240                          .WORD   1240            ; TEST FPS
13912  070226  101242                          .WORD   101242          ;RESULT FPS
13913  070230  000010                          .WORD   10              ;FEC
13914                                  ;
13915
13916  070232  000167  000212                  JMP     HOP13
13917
13918                                  ;*X*X*X
```

```
13960  070344  104003                    ERROR   .+3             ;FPP ERROR
13961                                                             ;PC WAS INCORRECT
13962                                    ;
13963                                    ;COMMON CODE FOR TRAP AND NO TRAP
13964  070346  170203                    2$:     STFPS   R3                      ;SAVE FPS
13965  070350  012702  000200                    MOV     #200,R2                 ;SET FPP TO DOUBLE
13966  070354  170102                            LDFPS   R2
13967  070356  012701  003102                    MOV     #RECDST,R1              ;POINT TO RECEIVED DATA TABLE
13968  070362  174011                            STD     AC0,(R1)                ;SAVE AC0  RESULT
13969  070364  026503  000032                    CMP     32(R5),R3               ;VERIFY STATUS
13970  070370  001401                            BEQ     3$                      ;BRANCH IF GOOD
13971  070372  104003                            ERROR   .+3             ;FPP ERROR
13972                                                             ;BAD FPS
13973  070374  010504                    3$:     MOV     R5,R4                   ;POINT TO EXPECTED DATA
13974  070376  062704  000020                    ADD     #20,R4
13975  070402  004767  043054            4$:     JSR     R7,DATVER                       ;VERIFY DATA
13976  070406  005767  112446                    TST     COUNT
13977  070412  001401                            BEQ     5$                      ;BRANCH IF GOOD
13978  070414  104003                            ERROR   .+3             ;FPP ERROR
13979                                                             ;BAD AC0
13980  070416  005737  002776            5$:     TST     @#FLAG                  ;SEE IF NEED TO CHECK FEC
13981  070422  001002                            BNE     6$                      ;BRANCH IF NEED TO CHECK
13982  070424  000165  000034                    JMP     34(R5)                  ;RETURN FROM TEST
13983                                    ;VERIFY ERROR STATUS
13984  070430  170301                    6$:     STST    R1                      ;SAVE FEC
13985  070432  016504  000034                    MOV     34(R5),R4               ;GET FEC
13986  070436  020401                            CMP     R4,R1                   ;VERIFY FEC
13987  070440  001401                            BEQ     7$                      ;BRANCH IF GOOD
13988  070442  104003                            ERROR   .+3             ;FPP ERROR
13989                                                             ;BAD FEC
13990  070444  000165  000036            7$:     JMP     36(R5)                  ;RETURN FROM TEST
13991  070450                            HOP13:
13992                                    ;
13993                                    ;
13994                                    ;---------------------------------
13995                                    ;TEST MODF
13996                                    ;
13997                                            ;
13998                                            ;
13999                                            ;
14000  070450                            MMODF:
14001
14002                                    ;
14003                                    ;1/AC=0 FSRC=0
14004  070450  005037  002776                    CLR     @#FLAG                  ;NO INTERRUPT
14005  070454  004767  000554                    JSR     R7,MDFSUB                       ;DO TEST
14006  070460  000100  000000                    .WORD   100,0   ;AC0
14007  070464  012346  177777                    .WORD   12346,-1                ;FSRC
14008  070470  000000  000000                    .WORD   0,0                     ;FRACTIONAL RESULT
14009  070474  000000  000000                    .WORD   0,0                     ;INTEGER RESULT
14010  070500  000013                            .WORD   13                      ; TEST FPS
14011  070502  000004                            .WORD   4                       ;RESULTANT FPS
14012                                    ;2/FSRC=0
14013  070504  005037  002776                    CLR     @#FLAG                  ;NO INTERRUPT
14014  070510  004767  000520                    JSR     R7,MDFSUB                       ;DO TEST
14015  070514  012356  177777                    .WORD   12356,-1                ;AC0
```

```
14016  070520  000000  000000                .WORD   0,0     ;FSRC
14017  070524  000000  000000                .WORD   0,0             ;FRACTIONAL RESULT
14018  070530  000000  000000                .WORD   0,0             ;INTEGER RESULT
14019  070534  000003                        .WORD   3               ; TEST FPS
14020  070536  000004                        .WORD   4       ;RESULTANT FPS
14021                          ;3/AC=0
14022  070540  005037  002776                CLR     @#FLAG          ;NO INTERRUPT
14023  070544  004767  000464                JSR     R7,MDFSUB               ;DO TEST
14024  070550  000000  000000                .WORD   0,0             ;AC0
14025  070554  177777  177777                .WORD   -1,-1           ;FSRC
14026  070560  000000  000000                .WORD   0,0             ;FRACTIONAL  RESULT
14027  070564  000000  000000                .WORD   0,0             ;INTEGER RESULT
14028  070570  007500                        .WORD   7500            ; TEST FPS
14029  070572  007504                        .WORD   7504            ;RESULT FPS
14030                          ;4/AC>FSRC>0
14031  070574  005037  002776                CLR     @#FLAG          ;NO INTERRUPT
14032  070600  004767  000430                JSR     R7,MDFSUB               ;DO TEST
14033  070604  046252  125252                .WORD   46252,125252    ;AC0
14034  070610  040300  000000                .WORD   40300,0 ;FSRC
14035  070614  000000  000000                .WORD   0,0             ;FRACTIONAL RESULT
14036  070620  046377  177777                .WORD   46377,-1                ;INTEGER RESULT
14037  070624  000013                        .WORD   13              ; TEST FPS
14038  070626  000004                        .WORD   4               ;RESULTANT FPS
14039                          ;5/AC>FSRC>0
14040  070630  005037  002776                CLR     @#FLAG          ;NO INTERRUPT
14041  070634  004767  000374                JSR     R7,MDFSUB               ;DO TEST
14042  070640  077652  125252                .WORD   77652,125252    ;AC0
14043  070644  040300  000000                .WORD   40300,0 ;FSRC
14044  070650  000000  000000                .WORD   0,0             ;FRACTIONAL RESULT
14045  070654  077777  177777                .WORD   77777,-1                ;INTEGER RESULT
14046  070660  000000                        .WORD   0               ; TEST FPS
14047  070662  000004                        .WORD   4               ;RESULTANT FPS
14048
14049                          ;6/AC>0<FSRC, INTEGERS
14050  070664  005037  002776                CLR     @#FLAG          ;NO INTERRUPT
14051  070670  004767  000340                JSR     R7,MDFSUB               ;DO TEST
14052  070674  060600  000000                .WORD   60600,0         ;AC0
14053  070700  147400  025700                .WORD   147400,25700            ;FSRC
14054  070704  000000  000000                .WORD   0,0             ;FRACTIONAL  RESULT
14055  070710  170000  025700                .WORD   170000,25700            ;INTEGER RESULT
14056  070714  007400                        .WORD   7400            ; TEST FPS
14057  070716  007404                        .WORD   7404            ;RESULT FPS
14058                          ;7/AC<0<FSRC, FRACTIONAL
14059  070720  005037  002776                CLR     @#FLAG          ;NO INTERRUPT
14060  070724  004767  000304                JSR     R7,MDFSUB               ;DO TEST
14061  070730  100227  177777                .WORD   100227,-1               ;AC0
14062  070734  044025  025252                .WORD   44025,25252             ;FSRC
14063  070740  104061  021251                .WORD   104061,21251            ;FRACTIONAL  RESULT
14064  070744  000000  000000                .WORD   0,0             ;INTEGER RESULT
14065  070750  000000                        .WORD   0               ; TEST FPS
14066  070752  000010                        .WORD   10              ;RESULT FPS
14067                          ;8/AC<0>FSRC, TRUNCATE
14068  070754  005037  002776                CLR     @#FLAG          ;NO INTERRUPT
14069  070760  004767  000250                JSR     R7,MDFSUB               ;DO TEST
14070  070764  046252  125252                .WORD   46252,125252            ;AC0
14071  070770  040300  000000                .WORD   40300,0         ;FSRC
```

```
14072  070774  000000  000000                  .WORD   0,0             ;FRACTIONAL  RESULT
14073  071000  046377  177777                  .WORD   46377,-1                ;INTEGER RESULT
14074  071004  000053                          .WORD   53              ; TEST FPS
14075  071006  000044                          .WORD   44              ;RESULT FPS
14076                                  ;9/ROUND INTEGER
14077  071010  005037  002776                  CLR     @#FLAG          ;NO INTERRUPT
14078  071014  004767  000214                  JSR     R7,MDFSUB               ;DO TEST
14079  071020  046252  125252                  .WORD   46252,125252            ;ACO
14080  071024  040300  000000                  .WORD   40300,0         ;FSRC
14081  071030  000000  000000                  .WORD   0,0             ;FRACTIONAL  RESULT
14082  071034  046377  177777                  .WORD   46377,-1                ;INTEGER RESULT
14083  071040  000013                          .WORD   13              ; TEST FPS
14084  071042  000004                          .WORD   4               ;RESULT FPS
14085                                  ;10/TRUNCATE FRACTION
14086  071044  005037  002776                  CLR     @#FLAG          ;NO INTERRUPT
14087  071050  004767  000160                  JSR     R7,MDFSUB               ;DO TEST
14088  071054  040777  177777                  .WORD   40777,-1                ;ACO
14089  071060  040200  000000                  .WORD   40200,0         ;FSRC
14090  071064  040177  177770                  .WORD   40177,177770            ;FRACTIONAL  RESULT
14091  071070  040740  000000                  .WORD   40740,0         ;INTEGER RESULT
14092  071074  000000                          .WORD   0               ; TEST FPS
14093  071076  000000                          .WORD   0               ;RESULT FPS
14094                                  ;11/ROUND INTEGER
14095  071100  005037  002776                  CLR     @#FLAG          ;NO INTERRUPT
14096  071104  004767  000124                  JSR     R7,MDFSUB               ;DO TEST
14097  071110  000000  000000                  .WORD   0,0             ;ACO
14098  071114  000000  000000                  .WORD   0,0             ;FSRC
14099  071120  000000  000000                  .WORD   0,0             ;FRACTIONAL  RESULT
14100  071124  000000  000000                  .WORD   0,0             ;INTEGER RESULT
14101  071130  000000                          .WORD   0               ; TEST FPS
14102  071132  000004                          .WORD   4               ;RESULT FPS
14103                                  ;12/ROUND FRACTION
14104  071134  005037  002776                  CLR     @#FLAG          ;NO INTERRUPT
14105  071140  004767  000070                  JSR     R7,MDFSUB               ;DO TEST
14106  071144  040225  125252                  .WORD   40225,125252            ;ACO
14107  071150  066652  052525                  .WORD   66652,52525             ;FSRC
14108  071154  000000  000000                  .WORD   0,0             ;FRACTIONAL  RESULT
14109  071160  066707  025160                  .WORD   66707,25160             ;INTEGER RESULT
14110  071164  007027                          .WORD   7027            ; TEST FPS
14111  071166  007004                          .WORD   7004            ;RESULT FPS
14112                                  ;/OVERFLOW
14113  071170  012737  000001  002776          MOV     #1,@#FLAG               ;INTERRUPT
14114  071176  004767  000032                  JSR     R7,MDFSUB               ;DO TEST
14115  071202  076000  000000                  .WORD   76000,0         ;ACO
14116  071206  076000  000000                  .WORD   76000,0         ;FSRC
14117  071212  000000  000000                  .WORD   0,0             ;FRACTIONAL  RESULT
14118  071216  033600  000000                  .WORD   33600,0         ;INTEGER RESULT
14119  071222  001000                          .WORD   1000            ; TEST FPS
14120  071224  101006                          .WORD   101006          ;RESULT FPS
14121  071226  000010                          .WORD   10              ;FEC
14122                                  ;
14123  071230  000167  000254                  JMP     HOP14
14124
14125                                  ;*X*X*X
```

```
14128                                  ;               AC0
14129                                  ;               FSRC
14130                                  ;               FRACTIONAL RESULT
14131                                  ;               INTEGER RESULT
14132                                  ;               FPS BEFORE EXECUTION
14133                                  ;               FPS AFTER EXECUTION
14134                                  ;               (FEC)
14135                                  ;
14136                                  ;*X*X*X
```

```
14184
14185  071400  010504                  3$:     MOV     R5,R4           ;BAD FPS
14186  071402  062704  000010                  ADD     #10,R4                  ;POINT TO EXPECTED DATA
14187  071406  004767  042032          4$:     JSR     R7,DATVFR                       ;VERIFY DATA
14188  071412  005767  111442                  TST     COUNT
14189  071416  001401                          BEQ     5$                      ;BRANCH IF GOOD
14190  071420  104003                          ERROR   .+3             ;FPP ERROR
14191                                                                  ;BAD AC0
14192                                  ;SAVE INTEGER RESULT
14193  071422  174111                  5$:     STD     AC1,(R1)                ;SAVE AC1 RESULT
14194  071424  010504                          MOV     R5,R4           ;POINT TO EXPECTED
14195  071426  062704  000014                  ADD     #14,R4
14196  071432  004767  042006                  JSR     R7,DATVFR               ;VERIFY DATA
14197  071436  005767  111416                  TST     COUNT                   ;
14198  071442  001401                          BEQ     6$                      ;BRANCH IF GOOD
14199  071444  104003                          ERROR   .+3             ;FPP ERROR
14200                                                                  ;BAD AC1
14201  071446  005737  002776          6$:     TST     @#FLAG                  ;SEE IF NEED TO CHECK FEC
14202  071452  001002                          BNE     7$                      ;BRANCH IF NEED TO CHECK
14203  071454  000165  000024                  JMP     24(R5)                  ;RETURN FROM TEST
14204  071460  170301                  7$:     STST    R1                      ;SAVE FEC
14205  071462  016504  000024                  MOV     24(R5),R4               ;GET FEC
14206  071466  020401                          CMP     R4,R1                   ;VERIFY FEC
14207  071470  001401                          BEQ     8$                      ;BRANCH IF GOOD
14208  071472  104003                          ERROR   .+3             ;FPP ERROR
14209                                                                  ;BAD FEC
14210  071474  000165  000026          8$:     JMP     26(R5)          ;RETURN FROM TEST
14211                                  ;
14212  071500  177777  177777  177777  MODGAR: .WORD   -1,-1,-1,-1             ;KNOWN DATA FOR AC1
14213  071506  177777
14214  071510                          HOP14:
14215
14216                                  ;
14217                                  ;
14218                                  ;---------------------------------
14219                                  ;TEST MODD
14220                                  ;
14221                                          ;
14222                                          ;
14223                                          ;
14224  071510                          MMODD:
14225
14226                                  ;
14227                                  ;1/AC>FSRC=0
14228  071510  005037  002776                  CLR     @#FLAG                  ;NO INTERRUPT
14229  071514  004767  001164                  JSR     R7,MDDSUB                       ;DO TEST
14230  071520  012345  177777  177777          .WORD   12345,-1,-1,-1          ;AC0
14231  071526  177777
14232  071530  000100  000000  000000          .WORD   100,0,0,0               ;FSRC
14233  071536  000000
14234  071540  000000  000000  000000          .WORD   0,0,0,0                 ;FRACTIONAL RESULT
14235  071546  000000
14236  071550  000000  000000  000000          .WORD   0,0,0,0                 ;INTEGER RESULT
14237  071556  000000
14238  071560  000200                          .WORD   200                     ; TEST FPS
14239  071562  000204                          .WORD   204                     ;RESULTANT FPS
```

```
14240                                   ;2/AC=0<FSRC
14241  071564  005037  002776                   CLR     @#FLAG          ;NO INTERRUPT
14242  071570  004767  001110                   JSR     R7,MDDSUB               ;DO TEST
14243  071574  000000  000000  000000           .WORD   0,0,0,0 ;AC0
14244  071602  000000
14245  071604  001234  177777  000000           .WORD   1234,-1,0,0     ;FSRC
14246  071612  000000
14247  071614  000000  000000  000000           .WORD   0,0,0,0         ;FRACTIONAL RESULT
14248  071622  000000
14249  071624  000000  000000  000000           .WORD   0,0,0,0         ;INTEGER RESULT
14250  071632  000000
14251  071634  007717                           .WORD   7717            ; TEST FPS
14252  071636  007704                           .WORD   7704            ;RESULTANT FPS
14253                                   ;3/AC>FSRC>0
14254  071640  005037  002776                   CLR     @#FLAG          ;NO INTERRUPT
14255  071644  004767  001034                   JSR     R7,MDDSUB               ;DO TEST
14256  071650  056252  125252  125252           .WORD   56252,125252,125252,125250      ;AC0
14257  071656  125250
14258  071660  040300  000000  000000           .WORD   40300,0,0,0     ;FSRC
14259  071666  000000
14260  071670  000000  000000  000000           .WORD   0,0,0,0         ;FRACTIONAL RESULT
14261  071676  000000
14262  071700  056377  177777  177777           .WORD   56377,-1,-1,-4          ;INTEGER RESULT
14263  071706  177774
14264  071710  000213                           .WORD   213             ; TEST FPS
14265  071712  000204                           .WORD   204             ;RESULTANT FPS
14266                                   ;4/AC<0>FSRC
14267  071714  005037  002776                   CLR     @#FLAG          ;NO INTERRUPT
14268  071720  004767  000760                   JSR     R7,MDDSUB               ;DO TEST
14269  071724  140240  000000  000000           .WORD   140240,0,0,0    ;AC0
14270  071732  000000
14271  071734  063714  146314  133572           .WORD   63714,146314,133572,167737      ;FSRC
14272  071742  167737
14273  071744  000000  000000  000000           .WORD   0,0,0,0         ;FRACTIONAL RESULT
14274  071752  000000
14275  071754  163777  177777  162531           .WORD   163777,-1,162531,125726         ;INTEGER RESULT
14276  071762  125726
14277  071764  000210                           .WORD   210             ; TEST FPS
14278  071766  000204                           .WORD   204             ;RESULTANT FPS
14279                                   ;5/AC>FSRC>0
14280  071770  005037  002776                   CLR     @#FLAG          ;NO INTERRUPT
14281  071774  004767  000704                   JSR     R7,MDDSUB               ;DO TEST
14282  072000  056200  000000  000000           .WORD   56200,0,0,1     ;AC0
14283  072006  000001
14284  072010  040340  000000  000000           .WORD   40340,0,0,0     ;FSRC
14285  072016  000000
14286  072020  000000  000000  000000           .WORD   0,0,0,0         ;FRACTIONAL RESULT
14287  072026  000000
14288  072030  056340  000000  000000           .WORD   56340,0,0,1             ;INTEGER RESULT
14289  072036  000001
14290  072040  000213                           .WORD   213             ; TEST FPS
14291  072042  000204                           .WORD   204             ;RESULTANT FPS
14292                                   ;6/TRUNCATE
14293  072044  005037  002776                   CLR     @#FLAG          ;NO INTERRUPT
14294  072050  004767  000630                   JSR     R7,MDDSUB               ;DO TEST
14295  072054  056252  125252  125252           .WORD   56252,125252,125252,125252              ;AC0
```

```
14296  072062  125252
14297  072064  040300  000000  000000          .WORD   40300,0,0,0             ;FSRC
14298  072072  000000
14299  072074  000000  000000  000000          .WORD   0,0,0,0          ;FRACTIONAL  RESULT
14300  072102  000000
14301  072104  056377  177777  177777          .WORD   56377,-1,-1,-1           ;INTEGER RESULT
14302  072112  177777
14303  072114  000253                          .WORD   253              ; TEST FPS
14304  072116  000244                          .WORD   244              ;RESULT FPS
14305                          ;7/TRUNCATE FRACTION
14306  072120  005037  002776                  CLR     @#FLAG           ;NO INTERRUPT
14307  072124  004767  000554                  JSR     R7,MDDSUB                ;DO TEST
14308  072130  023252  125252  125252          .WORD   23252,125252,125252,125252               ;AC0
14309  072136  125252
14310  072140  040300  000000  000000          .WORD   40300,0,0,0              ;FSRC
14311  072146  000000
14312  072150  023377  177777  177777          .WORD   23377,-1,-1,-1           ;FRACTIONAL  RESULT
14313  072156  177777
14314  072160  000000  000000  000000          .WORD   0,0,0,0          ;INTEGER RESULT
14315  072166  000000
14316  072170  000253                          .WORD   253              ; TEST FPS
14317  072172  000240                          .WORD   240              ;RESULT FPS
14318                          ;8/ROUND INTEGER
14319  072174  005037  002776                  CLR     @#FLAG           ;NO INTERRUPT
14320  072200  004767  000500                  JSR     R7,MDDSUB                ;DO TEST
14321  072204  076600  000000  000000          .WORD   76600,0,0,125252                 ;AC0
14322  072212  125252
14323  072214  040300  000000  000000          .WORD   40300,0,0,0              ;FSRC
14324  072222  000000
14325  072224  000000  000000  000000          .WORD   0,0,0,0          ;FRACTIONAL  RESULT
14326  072232  000000
14327  072234  076700  000000  000000          .WORD   76700,0,0,-1             ;INTEGER RESULT
14328  072242  177777
14329  072244  000200                          .WORD   200              ; TEST FPS
14330  072246  000204                          .WORD   204              ;RESULT FPS
14331                          ;9/ROUND THROUGH FRACTION
14332  072250  005037  002776                  CLR     @#FLAG           ;NO INTERRUPT
14333  072254  004767  000424                  JSR     R7,MDDSUB                ;DO TEST
14334  072260  041525  052525  052525          .WORD   41525,052525,52525,52525                 ;AC0
14335  072266  052525
14336  072270  040300  000000  000000          .WORD   40300,0,0,0              ;FSRC
14337  072276  000000
14338  072300  040177  177777  177777          .WORD   40177,-1,-1,177740               ;FRACTIONAL  RESULT
14339  072306  177740
14340  072310  041636  000000  000000          .WORD   41636,0,0,0              ;INTEGER RESULT
14341  072316  000000
14342  072320  007700                          .WORD   7700             ; TEST FPS
14343  072322  007700                          .WORD   7700             ;RESULT FPS
14344                          ;/OVERFLOW, TRAPS ENABLED
14345  072324  012737  000001  002776          MOV     #1,@#FLAG                ;INTERRUPT
14346  072332  004767  000346                  JSR     R7,MDDSUB                ;DO TEST
14347  072336  177777  177777  177777          .WORD   -1,-1,-1,-1              ;AC0
14348  072344  177777
14349  072346  040400  000000  000000          .WORD   40400,0,0,0              ;FSRC
14350  072354  000000
14351  072356  000000  000000  000000          .WORD   0,0,0,0          ;FRACTIONAL  RESULT
```

```
14352  072364  000000
14353  072366  100177  177777  177777          .WORD   100177,-1,-1,-1          ;INTEGER RESULT
14354  072374  177777
14355  072376  007700                          .WORD   7700              ; TEST FPS
14356  072400  107706                          .WORD   107706            ;RESULT FPS
14357  072402  000010                          .WORD   10                ;FEC
14358                                  ;/INTEGER CHOPPED TO 56 BITS
14359  072404  005037  002776                  CLR     @#FLAG            ;NO INTERRUPT
14360  072410  004767  000270                  JSR     R7,MDDSUB                 ;DO TEST
14361  072414  056700  000000  000000          .WORD   56700,0,0,-1              ;ACO
14362  072422  177777
14363  072424  044440  177777  177777          .WORD   44440,-1,-1,-1            ;FSRC
14364  072432  177777
14365  072434  000000  000000  000000          .WORD   0,0,0,0           ;FRACTIONAL  RESULT
14366  072442  000000
14367  072444  063161  100000  000001          .WORD   63161,100000,1,40775              ;INTEGER RESULT
14368  072452  040775
14369  072454  000200                          .WORD   200               ; TEST FPS
14370  072456  000204                          .WORD   204               ;RESULT FPS
14371                                  ;/OVERFLOW, TRAPS DISABLED
14372  072460  012737  000002  002776          MOV     #2,@#FLAG                 ;NO INTERRUPT
14373  072466  004767  000212                  JSR     R7,MDDSUB                 ;DO TEST
14374  072472  066600  000000  000000          .WORD   66600,0,0,0               ;ACO
14375  072500  000000
14376  072502  066600  000000  000000          .WORD   66600,0,0,0               ;FSRC
14377  072510  000000
14378  072512  000000  000000  000000          .WORD   0,0,0,0           ;FRACTIONAL  RESULT
14379  072520  000000
14380  072522  015200  000000  000000          .WORD   15200,0,0,0               ;INTEGER RESULT
14381  072530  000000
14382  072532  047700                          .WORD   47700             ; TEST FPS
14383  072534  147706                          .WORD   147706            ;RESULT FPS
14384  072536  000010                          .WORD   10                ;FEC
14385                                  ;/UNDERFLOW, TRAPS DISABLED
14386  072540  012737  000002  002776          MOV     #2,@#FLAG                 ;NO INTERRUPT
14387  072546  004767  000132                  JSR     R7,MDDSUB                 ;DO TEST
14388  072552  100277  000001  000002          .WORD   100277,1,2,-1             ;ACO
14389  072560  177777
14390  072562  100300  000001  000001          .WORD   100300,1,1,1              ;FSRC
14391  072570  000001
14392  072572  000000  000000  000000          .WORD   0,0,0,0           ;FRACTIONAL  RESULT
14393  072600  000000
14394  072602  000000  000000  000000          .WORD   0,0,0,0           ;INTEGER RESULT
14395  072610  000000
14396  072612  005200                          .WORD   5200              ; TEST FPS
14397  072614  005204                          .WORD   5204              ;RESULT FPS
14398  072616  000010                          .WORD   10                ;FEC
14399                                  ;/UNDERFLOW TRAPS ENABLED, UV AS RESULT
14400  072620  012737  000001  002776          MOV     #1,@#FLAG                 ;INTERRUPT
14401  072626  004767  000052                  JSR     R7,MDDSUB                 ;DO TEST
14402  072632  100277  000001  000002          .WORD   100277,1,2,-1             ;ACO
14403  072640  177777
14404  072642  100300  000001  000001          .WORD   100300,1,1,1              ;FSRC
14405  072650  000001
14406  072652  040417  040001  077403          .WORD   40417,40001,77403,0               ;FRACTIONAL  RESULT
14407  072660  000000
```

```
14408  072662  000000  000000  000000          .WORD   0,0,0,0         ;INTEGER RESULT
14409  072670  000000
14410  072672  002200                          .WORD   2200            ; TEST FPS
14411  072674  102200                          .WORD   102200          ;RESULT FPS
14412  072676  000012                          .WORD   12              ;FEC
14413                                  ;
14414
14415  072700  000167  000244                  JMP     MOP15           ;JUMP OVER SUBROUTINE
14416                                  ;*X*X*X
```

```
14464                                   ;
14465                                   ;COMMON CODE FOR TRAP AND NO TRAP
14466  073022  170203                   2$:     STFPS   R3                      ;SAVE FPS
14467  073024  012702  000200                   MOV     #200,R2                 ;SET FPP TO DOUBLE
14468  073030  170102                           LDFPS   R2
14469  073032  012701  003102                   MOV     #RECDST,R1              ;POINT TO RECEIVED DATA TABLE
14470                                   ;SAVE FRACTIONAL RESULT
14471  073036  174011                           STD     AC0,(R1)                ;SAVE AC0  RESULT
14472  073040  026503  000042                   CMP     42(R5),R3               ;VERIFY STATUS
14473  073044  001401                           BEQ     3$                      ;BRANCH IF GOOD
14474  073046  104003                           ERROR   .3                      ;FPP ERROR
14475                                                                           ;BAD FPS
14476  073050  010504                   3$:     MOV     R5,R4                   ;POINT TO EXPECTED DATA
14477  073052  062704  000020                   ADD     #20,R4
14478  073056  004767  040400           4$:     JSR     R7,DATVER                       ;VERIFY DATA
14479  073062  005767  107772                   TST     COUNT
14480  073066  001401                           BEQ     5$                      ;BRANCH IF GOOD
14481  073070  104003                           ERROR   .3                      ;FPP ERROR
14482                                                                           ;BAD AC0
14483                                   ;SAVE INTEGER RESULT
14484  073072  174111                   5$:     STD     AC1,(R1)                ;SAVE AC1 RESULT
14485  073074  010504                           MOV     R5,R4                   ;POINT TO EXPECTED
14486  073076  062704  000030                   ADD     #30,R4
14487  073102  004767  040354                   JSR     R7,DATVER               ;VERIFY DATA
14488  073106  005767  107746                   TST     COUNT                   ;
14489  073112  001401                           BEQ     6$                      ;BRANCH IF GOOD
14490  073114  104003                           ERROR   .3                      ;FPP ERROR
14491                                                                           ;BAD AC1
14492  073116  005737  002776           6$:     TST     @#FLAG                  ;SEE IF NEED TO CHECK FEC
14493  073122  001002                           BNE     7$                      ;BRANCH IF NEED TO CHECK
14494  073124  000165  000044                   JMP     44(R5)                  ;RETURN FROM TEST
14495  073130  170301                   7$:     STST    R1                      ;SAVE FEC
14496  073132  016504  000044                   MOV     44(R5),R4               ;GET FEC
14497  073136  020401                           CMP     R4,R1                   ;VERIFY FEC
14498  073140  001401                           BEQ     8$                      ;BRANCH IF GOOD
14499  073142  104003                           ERROR   .3                      ;FPP ERROR
14500                                                                           ;BAD FEC
14501  073144  000165  000046           8$:     JMP     46(R5)                  ;RETURN FROM TEST
14502                                   ;
14503  073150                           MOP15:
14504                                   ;
14505                                   ;
14506                                   ;---------------------------------
14507                                   ;TEST STCFD
14508                                   ;
14509                                           ;
14510                                           ;
14511                                           ;
14512  073150                           MSFD:
14513
14514                                   ;
14515                                   ;1/AC=0
14516  073150  004767  000170                   JSR     R7,SFDSUB               ;DO TEST
14517  073154  000177  000000  000000           .WORD   0177,0,0,1              ;AC0
14518  073162  000001
14519  073164  000000  000000  000000           .WORD   0,0,0,0                 ;RESULT
```

```
14520  073172  000000
14521  073174  047557                          .WORD   47557           ; TEST FPS
14522  073176  047544                          .WORD   47544           ;RESULT FPS
14523                                  ;2/AC>0, TRUNCATE
14524  073200  004767  000140                  JSR     R7,SFDSUB       ;DO TEST
14525  073204  077577  177777  177777          .WORD   77577,-1,-1,-1          ;AC0
14526  073212  177777
14527  073214  077577  177777  000000          .WORD   77577,-1,0,0            ;RESULT
14528  073222  000000
14529  073224  007540                          .WORD   7540            ; TEST FPS
14530  073226  007540                          .WORD   7540            ;RESULT FPS
14531                                  ;3/AC<0, ROUND
14532  073230  004767  000110                  JSR     R7,SFDSUB               ;DO TEST
14533  073234  100377  177777  100000          .WORD   100377,-1,100000,0              ;AC0
14534  073242  000000
14535  073244  100377  177777  000000          .WORD   100377,-1,0,0   ;RESULT
14536  073252  000000
14537  073254  007517                          .WORD   7517            ; TEST FPS
14538  073256  007510                          .WORD   7510            ;RESULT FPS
14539                                  ;4/AC=-0
14540  073260  004767  000060                  JSR     R7,SFDSUB               ;DO TEST
14541  073264  100000  000000  000000          .WORD   100000,0,0,0            ;AC0
14542  073272  000000
14543  073274  000000  000000  000000          .WORD   0,0,0,0         ;RESULT
14544  073302  000000
14545  073304  007757                          .WORD   7757            ; TEST FPS
14546  073306  007744                          .WORD   7744            ;RESULT FPS
14547                                  ;5/AC<0
14548  073310  004767  000030                  JSR     R7,SFDSUB               ;DO TEST
14549  073314  125252  125252  125252          .WORD   125252,125252,125252,125252             ;AC0
14550  073322  125252
14551  073324  125252  125252  000000          .WORD   125252,125252,0,0               ;RESULT
14552  073332  000000
14553  073334  000000                          .WORD   0               ; TEST FPS
14554  073336  000010                          .WORD   10              ;RESULT FPS
14555                                  ;
14556
14557  073340  000167  000104                  JMP     HOP16                   ;GET OVER SUBROUTINE
14558                                  ;*X*X*X
```

```
073370  016502  000020                  MOV     20(R5),R2                ;GET TEST FPS
073374  170102                          LDFPS   R2                       ;LOAD TEST FPS
                                ;
073376  176011                  40$:    STCFD   AC0,(R1)                 ;*TEST INSTRUCTION
                                ;
                                ;INSTRUCTION DIDNT TRAP
                                ;VERIFY STATUS
073400  170203                  2$:     STFPS   R3                       ;SAVE FPS
073402  016502  000022                  MOV     22(R5),R2                        ;GET EXPECTED STATUS
073406  020203                          CMP     R2,R3                    ;VERIFY STATUS
073410  001401                          BEQ     3$                       ;BRANCH IF GOOD
073412  104003                          ERROR   +3                 ;FPP ERROR
                                                                   ;BAD FPS
073414  010504                  3$:     MOV     R5,R4                    ;POINT TO EXPECTED DATA
073416  062704  000010                  ADD     #10,R4
073422  004767  040034          4$:     JSR     R7,DATVER                        ;VERIFY DATA
073426  005767  107426                  TST     COUNT
073432  001401                          BEQ     5$                       ;BRANCH IF GOOD
073434  104003                          ERROR   +3                 ;FPP ERROR
                                                                   ;BAD AC0
073436  000165  000024          5$:     JMP     24(R5)                   ;RETURN FROM TEST
                                ;INSTRUCTION TRAPPED
073442  104003                  50$:    ERROR   +3                 ;FPP ERROR
                                                                   ;INSTRUCTION WASNT SUPPOSE TO TRAP
073444  000165  000024                  JMP     24(R5)             ;RETURN FROM TEST
                                ;
073450                          HOP16:


                                ;
                                ;
                                ;-----------------------------------
                                ;TEST STCDF
                                ;
                                        ;
                                        ;
                                        ;
073450                          MSDF:

                                ;
                                ;1/AC=0
073450  005037  002776                  CLR     @#FLAG             ;NO INTERRUPT
073454  004767  000220                  JSR     R7,SDFSUB                ;DO TEST
073460  000177  000000  000000          .WORD   177,0,0,0                ;AC0
073466  000000
073470  000000  000000                  .WORD   0,0                ;RESULT
073474  000200                          .WORD   200                ; TEST FPS
073476  000204                          .WORD   204                ;RESULT FPS
                                ;2/AC=-0
073500  005037  002776                  CLR     @#FLAG             ;NO INTERRUPT
073504  004767  000170                  JSR     R7,SDFSUB                ;DO TEST
073510  100000  000300  000200          .WORD   100000,300,200,100               ;AC0
073516  000100
073520  000000  000000                  .WORD   0,0                ;RESULT
073524  007777                          .WORD   7777               ; TEST FPS
073526  007744                          .WORD   7744               ;RESULT FPS
                                ;3/AC>0, TRUNCATE
```

```
073530  005037  002776                  CLR     @#FLAG          ;NO INTERRUPT
073534  004767  000140                  JSR     R7,SDFSUB               ;DO TEST
073540  055555  055555  177777          .WORD   55555,55555,-1,-1               ;AC0
073546  177777
073550  055555  055555                  .WORD   55555,55555             ;RESULT
073554  000240                          .WORD   240             ; TEST FPS
073556  000240                          .WORD   240             ;RESULT FPS
                                ;4/AC<0, ROUND TO UNDEFINED VARIABLE
073560  012737  000001  002776          MOV     #1,@#FLAG               ;INTERRUPT
073566  004767  000106                  JSR     R7,SDFSUB               ;DO TEST
073572  077777  177777  100000          .WORD   77777,-1,100000,0               ;AC0
073600  000000
073602  000000  000000                  .WORD   0,0             ;RESULT
073606  001200                          .WORD   1200            ; TEST FPS
073610  101206                          .WORD   101206          ;RESULT FPS
                                ;5/AC<0, ROUND
073612  005037  002776                  CLR     @#FLAG          ;NO INTERRUPT
073616  004767  000056                  JSR     R7,SDFSUB               ;DO TEST
073622  125252  125252  125252          .WORD   125252,125252,125252,125252             ;AC0
073630  125252
073632  125252  125253                  .WORD   125252,125253           ;RESULT
073636  007700                          .WORD   7700            ; TEST FPS
073640  007710                          .WORD   7710            ;RESULT FPS
                                ;6/ROUND TO UV, TRAPS DISABLED
073642  012737  000002  002776          MOV     #2,@#FLAG               ;INTERRUPT
073650  004767  000024                  JSR     R7,SDFSUB               ;DO TEST
073654  077777  177777  177777          .WORD   77777,-1,-1,0           ;AC0
073662  000000
073664  000000  000000                  .WORD   0,0             ;RESULT
073670  006700                          .WORD   6700            ; TEST FPS
073672  006706                          .WORD   6706            ;RESULT FPS
                                ;
073674  000167  000202                  JMP     HOP17           ;GET OVER SUBROUTINE
                                ;
                                ;
                                ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
                                ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
                                ;STCDF
                                ;               AC0
                                ;               RESULT
                                ;               FPS BEFORE EXECUTION
                                ;               FPS AFTER EXECUTION
                                ;
                                ;A TRAP CAN ONLY OCCUR IF ROUNDING CAUSES OVERFLOW
                                ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
                                ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
                                ;
073700  012605                  SDFSUB: MOV     (SP)+,R5                ; RETURN ADDRESS TO USE AS POINTER
073702  012737  073756  000244          MOV     #50$,@#FPVEC            ;REDIRECT TRAP VECTOR
073710  012702  000200                  MOV     #200,R2                 ;SET TO DOUBLE MODE FOR LOAD
073714  170102                          LDFPS   R2                      ;LOAD FPS
073716  172415                          LDD     (R5),AC0                ;LOAD AC0 WITH TEST DATA
073720  012701  003102                  MOV     #RECDST,R1              ;POINT TO RESULT AREA
073724  016502  000014                  MOV     14(R5),R2               ;GET TEST FPS
073730  170102                          LDFPS   R2                      ;LOAD TEST FPS
```

```
14688                                   ;
14689  073732  176011                   40$:    STCDF   AC0,(R1)          ;*TEST INSTRUCTION
14690  073734  170327                   1$:     STST    (PC)+             ;WAIT FOR POSSIBLE FPA TRAP.
14691  073736  000000                           .WORD   0                 ;STORE STATUS HERE.
14692                                   ;
14693                                   ;INSTRUCTION DIDNT TRAP
14694  073740  032737  000001  002776           BIT     #1,@#FLAG                 ;VERIFY A NO TRAP CONDITION
14695  073746  001420                           BEQ     2$                ;BRANCH IF GOOD
14696  073750  104003                           ERROR   +3              ;FPP ERROR
14697                                           ;INSTRUCTION SHOULD HAVE TRAPPED
14698  073752  000167  000032                   JMP     2$                ;REJOIN CODE
14699                                   ;
14700                                   ;INSTRUCTION TRAPPED
14701  073756  032737  000001  002776   50$:    BIT     #1,@#FLAG         ;SEE IF EXPECTING A TRAP
14702  073764  001003                           BNE     51$               ;BRANCH IF EXPECTING A TRAP
14703  073766  104003                           ERROR   +3              ;FPP ERROR
14704                                           ;INSTRUCTION WASNT SUPPOSE TO TRAP
14705  073770  000167  000014                   JMP     2$                ;REJOIN CODE
14706  073774  012604                   51$:    MOV     (SP)+,R4          ;SEE IF PC = INSTRUCTION
14707  073776  005726                           TST     (SP)+             ;CLEAN UP STACK
14708  074000  022704  073734                   CMP     #1$,R4            ;
14709  074004  001401                           BEQ     2$                ;BRANCH IF GOOD COMPARE
14710  074006  104003                           ERROR   +3              ;FPP ERROR
14711                                           ;PC WAS INCORRECT
14712
14713                                   ;
14714                                   ;COMMON CODE FOR TRAP AND NO TRAP
14715                                   ;VERIFY STATUS
14715  074010  170203                   2$:     STFPS   R3                ;SAVE FPS
14716  074012  016502  000016                   MOV     16(R5),R2                 ;GET EXPECTED STATUS
14717  074016  020203                           CMP     R2,R3             ;VERIFY STATUS
14718  074020  001401                           BEQ     3$                ;BRANCH IF GOOD
14719  074022  104003                           ERROR   +3              ;FPP ERROR
14720                                           ;BAD FPS
14721  074024  010504                   3$:     MOV     R5,R4                     ;POINT TO EXPECTED DATA
14722  074026  062704  000010                   ADD     #10,R4
14723  074032  004767  037406           4$:     JSR     R7,DATVFR                 ;VERIFY DATA
14724  074036  005767  107016                   TST     COUNT
14725  074042  001401                           BEQ     5$                ;BRANCH IF GOOD
14726  074044  104003                           ERROR   +3              ;FPP ERROR
14727                                           ;BAD AC0
14728  074046  005737  002776           5$:     TST     @#FLAG                    ;SEE IF NEED TO CHECK FEC
14729  074052  001002                           BNE     7$                ;BRANCH IF NEED TO CHECK
14730  074054  000165  000020                   JMP     20(R5)            ;RETURN FROM TEST
14731                                   ;VERIFY FEC
14732  074060  012704  003062           7$:     MOV     #RECFEC,R4                ;POINT TO FEC AREA
14733  074064  170314                           STST    (R4)              ;SAVE FEC
14734  074066  021427  000010                   CMP     (R4),#10                  ;VERIFY FEC FOR OVERFLOW
14735  074072  001401                           BEQ     8$                ;BRANCH IF GOOD
14736  074074  104003                           ERROR   +3              ;FPP ERROR
14737                                           ;BAD FEC
14738  074076  000165  000020           8$:     JMP     20(R5)          ;RETURN FROM TEST
14739                                   ;
14740  074102                           HOP17:
14741                                   ;
14742                                   ;
14743                                   ;-------------------------------
```

```
14744                                   ;TEST STCFD - USING ILLEGAL ACCUMULATOR
14745                                   ;
14746                                           ;
14747                                           ;
14748                                           ;
14749  074102                           MSFDI:
14750
14751                                   ;
14752  074102  012701  040000                  MOV     #40000,R1               ;DISABLE INTERRUPTS
14753  074106  170101                           LDFPS   R1                      ;
14754  074110  176006                           STCFD   AC0,AC6                 ;*TEST ILLEGAL INSTRUCTION
14755  074112  170202                           STFPS   R2                      ;SAVE STATUS
14756  074114  170303                           STST    R3                      ;SAVE FEC
14757  074116  022702  140000                  CMP     #140000,R2              ;VERIFY FER SET
14758  074122  001401                           BEQ     1$                      ;BRANCH IF ERROR RECEIVED
14759  074124  104003                           ERROR   .3                  ;FPP ERROR
14760                                                                       ;FER BIT NOT SET ON ILLEGAL INST.
14761  074126  022703  000002          1$:     CMP     #2,R3                   ;VERIFY FEC = FLOATING OPCDOE ERROR
14762  074132  001401                           BEQ     2$                      ;BRANCH IF GOOD
14763  074134  104003                           ERROR   .3                  ;FPP ERROR
14764                                                                       ;FEC INCORRECT
14765  074136                           2$:
14766
14767                                   ;
14768                                   ;
14769                                   ;
14770                                   ;-----------------------------------
14771                                   ;TEST CLRD
14772                                   ;
14773                                           ;
14774                                           ;
14775                                           ;
14776  074136                           MCLRD:
14777
14778                                   ;
14779  074136  012701  003724                  MOV     #TAB47,R1               ;POINT TO DATA
14780  074142  012704  000200                  MOV     #200,R4                 ;SET FPP STATUS TO DOUBLE
14781  074146  170104                           LDFPS   R4
14782  074150  172411                           LDD     (R1),AC0                ;
14783  074152  012701  003102                  MOV     #RECDST,R1              ;POINT TO DATA BUFFER
14784  074156  174011                           STD     AC0,(R1)                ;STORE GARBAGE
14785  074160  170411                           CLRD    (R1)                    ;CLEAR DATA BUFFER
14786  074162  012704  003254                  MOV     #TAB6,R4                ;VERIFY BUFFER =0
14787  074166  004767  037270                  JSR     R7,DATVER               ;
14788  074172  005767  106662                  TST     COUNT                   ;
14789  074176  001401                           BEQ     1$                      ;BRANCH I RECDST = 0
14790  074200  104003                           ERROR   .3                  ;FPP ERROR
14791                                                                       ;RECDST NOT CLEARED
14792  074202  170202                  1$:     STFPS   R2                      ;SAVE STATUS
14793  074204  020227  000204                  CMP     R2,#204                 ;VERIFY STATUS
14794  074210  001401                           BEQ     2$                      ;BRANCH IF GOOD
14795  074212  104003                           ERROR   .3                  ;FPP ERROR
14796                                                                       ;BAD STATUS
14797  074214                           2$:
14798
14799
```

```
14800                                   ;
14801                                   ;
14802                                   ;-----------------------------------
14803                                   ;TEST CLRD, ILLEGAL ACCUMULATOR
14804                                   ;
14805                                           ;
14806                                           ;
14807                                           ;
14808  074214                           MCLRI:
14809
14810                                   ;
14811  074214  012704  040200                   MOV     #40200,R4               ;DISABLE INTERRUPTS
14812  074220  170104                           LDFPS   R4                      ;LOAD STATUS
14813  074222  170406                           CLRD    R6                      ;*TEST INSTRUCTION WITH ILLEGAL ACC
14814  074224  170203                           STFPS   R3                      ;SAVE STATUS
14815  074226  170305                           STST    R5                      ;SAVE FEC
14816  074230  022703  140200                   CMP     #140200,R3              ;VERIFY ERROR
14817  074234  001401                           BEQ     1$                      ;BRANCH IF FER SET
14818  074236  104003                           ERROR   .3              ;FPP ERROR
14819                                                                   ;ERROR IN FPS
14820  074240  022705  000002           1$:     CMP     #2,R5           ;VERIFY FEC =2 OPCODE ERROR
14821  074244  001401                           BEQ     2$              ;BRANCH IF GOOD
14822  074246  104003                           ERROR   .3              ;FPP ERROR
14823                                                                           ;BAD FEC
14824  074250                           2$:
14825
14826
14827                                   ;
14828                                   ;
14829                                   ;-----------------------------------
14830                                   ;TEST LDFPS, STFPS MODE 1
14831                                   ;
14832                                           ;
14833                                           ;
14834                                           ;
14835  074250                           MLS1:
14836
14837                                   ;
14838  074250  012704  003122                   MOV     #TSTLOC,R4              ;POINT R4 TO RAM
14839  074254  012714  147757                   MOV     #147757,(R4)            ;SETUP EXPECTED STATUS
14840  074260  012701  003072                   MOV     #RECST,R1               ;SET BUFFER FOR RECEIVED STATUS
14841  074264  012737  074334  000244           MOV     #10$,@#FPVEC            ;SETUP TRAP VECTOR
14842  074272  170114                           LDFPS   (R4)                    ;*TEST INSTRUCTION
14843  074274  170211                           STFPS   (R1)                    ;*TEST INSTRUCTION
14844  074276  020427  003122                   CMP     R4,#TSTLOC              ;VERIFY R4
14845  074302  001401                           BEQ     1$                      ;BRANCH IF GOOD
14846  074304  104003                           ERROR   .3              ;FPP ERROR
14847                                                                   ;
14848  074306  020127  003072           1$:     CMP     R1,#RECST               ;VERIFY R1
14849  074312  001401                           BEQ     2$                      ;BRANCH IF GOOD
14850  074314  104003                           ERROR   .3              ;FPP ERROR
14851                                                                   ;BAD R1
14852  074316  023727  003072  147757   2$:     CMP     @#RECST,#147757         ;VERIFY STATUS
14853  074324  001406                           BEQ     3$                      ;BRANCH F GOOD
14854  074326  104003                           ERROR   .3              ;FPP ERROR
14855                                                                   ;BAD STATUS\
```

```
14856  074330  000167  000006            JMP     3$                      ;GET OVER TRAP
14857                                  ;UNEXPECTED TRAP
14858  074334  012600                  10$:    MOV     (SP)+,R0                        ;SAVE PC
14859  074336  012605                          MOV     (SP)+,R5                        ;SAVE PS
14860  074340  104003                          ERROR   +3              ;FPP ERROR
14861                                                                  ;UNEXPECTED TRAP
14862  074342                          3$:
14863
14864
14865                                  ;
14866                                  ;
14867                                  ;-----------------------------------
14868                                  ;TEST LDFPS, STFPS MODE 2
14869                                  ;
14870                                          ;
14871                                          ;
14872                                          ;
14873  074342                          MLS2:
14874
14875                                  ;
14876  074342  012704  003122                  MOV     #TSTLOC,R4              ;POINT R4 TO RAM
14877  074346  012714  145557                  MOV     #145557,(R4)            ;SETUP EXPECTED STATUS
14878  074352  012701  003072                  MOV     #RECST,R1               ;SET BUFFER FOR RECEIVED STATUS
14879  074356  012737  074426  000244          MOV     #10$,@#FPVEC            ;SETUP TRAP VECTOR
14880  074364  170124                          LDFPS   (R4)+                   ;*TEST INSTRUCTION
14881  074366  170221                          STFPS   (R1)+                   ;*TEST INSTRUCTION
14882  074370  020427  003124                  CMP     R4,#TSTLOC+2            ;VERIFY R4
14883  074374  001401                          BEQ     1$                      ;BRANCH IF GOOD
14884  074376  104003                          ERROR   +3              ;FPP ERROR
14885                                                                  ;
14886  074400  020127  003074          1$:     CMP     R1,#RECST+2                     ;VERIFY R1
14887  074404  001401                          BEQ     2$                              ;BRANCH IF GOOD
14888  074406  104003                          ERROR   +3              ;FPP ERROR
14889                                                                  ;BAD R1
14890  074410  023727  003072  145557  2$:     CMP     @#RECST,#145557                 ;VERIFY STATUS
14891  074416  001406                          BEQ     3$                              ;BRANCH F GOOD
14892  074420  104003                          ERROR   +3              ;FPP ERROR
14893                                                                  ;BAD STATUS\
14894  074422  000167  000006                  JMP     3$                              ;GET OVER TRAP
14895                                  ;UNEXPECTED TRAP
14896  074426  012600                  10$:    MOV     (SP)+,R0                        ;SAVE PC
14897  074430  012605                          MOV     (SP)+,R5                        ;SAVE PS
14898  074432  104003                          ERROR   +3              ;FPP ERROR
14899                                                                  ;UNEXPECTED TRAP
14900  074434                          3$:
14901
14902
14903                                  ;
14904                                  ;
14905                                  ;-----------------------------------
14906                                  ;TEST LDFPS, STFPS MODE 3
14907                                  ;
14908                                          ;
14909                                          ;
14910                                          ;
14911  074434                          MLS3:
```

```
14912
14913                                   ;
14914  074434  012704  003122                   MOV     #TSTLOC,R4              ;POINT R4 TO RAM
14915  074440  012737  003126  003122           MOV     #TSTLOC+4,@#TSTLOC      ;TSTLOC= DEFERRED ADDRESS
14916  074446  012737  147501  003126           MOV     #147501,@#TSTLOC+4      ;SETUP EXPECTED STATUS
14917  074454  012701  003132                   MOV     #TSTLOC+10,R1           ;R1 POINTS TO TSTLOC+10
14918  074460  012737  003072  003132           MOV     #RECST,@#TSTLOC+10      ;SET DEFERRED BUFFER FOR RECEIVED STATUS
14919  074466  012737  074536  000244           MOV     #10$,@#FPVEC            ;SETUP TRAP VECTOR
14920  074474  170134                           LDFPS   @(R4)+                  ;*TEST INSTRUCTION
14921  074476  170231                           STFPS   @(R1)+                  ;*TEST INSTRUCTION
14922  074500  020427  003124                   CMP     R4,#TSTLOC+2            ;VERIFY R4
14923  074504  001401                           BEQ     1$                      ;BRANCH IF GOOD
14924  074506  104003                           ERROR   +3              ;FPP ERROR
14925                                                                   ;
14926  074510  020127  003134           1$:     CMP     R1,#TSTLOC+12           ;VERIFY R1
14927  074514  001401                           BEQ     2$                      ;BRANCH IF GOOD
14928  074516  104003                           ERROR   +3              ;FPP ERROR
14929                                                                   ;BAD R1
14930  074520  023727  003072  147501   2$:     CMP     @#RECST,#147501         ;VERIFY STATUS
14931  074526  001406                           BEQ     3$                      ;BRANCH F GOOD
14932  074530  104003                           ERROR   +3              ;FPP ERROR
14933                                                                   ;BAD STATUS
14934  074532  000167  000006                   JMP     3$                      ;GET OVER TRAP
14935                                   ;UNEXPECTED TRAP
14936  074536  012600                   10$:    MOV     (SP)+,R0                        ;SAVE PC
14937  074540  012605                           MOV     (SP)+,R5                        ;SAVE PS
14938  074542  104003                           ERROR   +3              ;FPP ERROR
14939                                                                   ;UNEXPECTED TRAP
14940  074544                           3$:
14941
14942
14943                                   ;
14944                                   ;
14945                                   ;---------------------------------
14946                                   ;TEST LDFPS, STFPS MODE 4
14947                                   ;
14948                                           ;
14949                                           ;
14950                                           ;
14951  074544                           MLS4:
14952
14953                                   ;
14954  074544  012704  003124                   MOV     #TSTLOC+2,R4            ;POINT R4 TO RAM
14955  074550  012737  147757  003122           MOV     #147757,@#TSTLOC        ;TSTLOC= STATUS ADDRESS
14956  074556  012701  003074                   MOV     #RECST+2,R1             ;SET BUFFER FOR RECEIVED STATUS
14957  074562  012737  074632  000244           MOV     #10$,@#FPVEC            ;SETUP TRAP VECTOR
14958  074570  170144                           LDFPS   -(R4)                   ;*TEST INSTRUCTION
14959  074572  170241                           STFPS   -(R1)                   ;*TEST INSTRUCTION
14960  074574  020427  003122                   CMP     R4,#TSTLOC              ;VERIFY R4
14961  074600  001401                           BEQ     1$                      ;BRANCH IF GOOD
14962  074602  104003                           ERROR   +3              ;FPP ERROR
14963                                                                   ;
14964  074604  020127  003072           1$:     CMP     R1,#RECST               ;VERIFY R1
14965  074610  001401                           BEQ     2$                      ;BRANCH IF GOOD
14966  074612  104003                           ERROR   +3              ;FPP ERROR
14967                                                                   ;BAD R1
```

```
14968  074614  023727  003072  147757  2$:     CMP     @#RECST,#147757         ;VERIFY STATUS
14969  074622  001406                          BEQ     3$                      ;BRANCH F GOOD
14970  074624  104003                          ERROR   .3              ;FPP ERROR
14971                                                                  ;BAD STATUS\
14972  074626  000167  000006                  JMP     3$                      ;GET OVER TRAP
14973                                  ;UNEXPECTED TRAP
14974  074632  012600                  10$:    MOV     (SP)+,R0                        ;SAVE PC
14975  074634  012605                          MOV     (SP)+,R5                        ;SAVE PS
14976  074636  104003                          ERROR   .3              ;FPP ERROR
14977                                                                  ;UNEXPECTED TRAP
14978  074640                          3$:
14979
14980
14981                                  ;
14982                                  ;
14983                                  ;-----------------------------------
14984                                  ;TEST LDFPS, STFPS MODE 5
14985                                  ;
14986                                          ;
14987                                          ;
14988                                          ;
14989  074640                          MLS5:
14990
14991                                  ;
14992  074640  012704  003124                  MOV     #TSTLOC+2,R4            ;POINT R4 TO RAM
14993  074644  012737  003126  003122          MOV     #TSTLOC+4,@#TSTLOC      ;TSTLOC= DEFERRED ADDRESS
14994  074652  012737  147501  003126          MOV     #147501,@#TSTLOC+4      ;SETUP EXPECTED STATUS
14995  074660  012701  003134                  MOV     #TSTLOC+12,R1           ;R1 POINTS TO 412
14996  074664  012737  003072  003132          MOV     #RECST,@#TSTLOC+10      ;SET DEFERRED BUFFER FOR RECEIVED STATUS
14997  074672  012737  074742  000244          MOV     #10$,@#FPVEC            ;SETUP TRAP VECTOR
14998  074700  170154                          LDFPS   @-(R4)                  ;*TEST INSTRUCTION
14999  074702  170251                          STFPS   @-(R1)                  ;*TEST INSTRUCTION
15000  074704  020427  003122                  CMP     R4,#TSTLOC              ;VERIFY R4
15001  074710  001401                          BEQ     1$                      ;BRANCH IF GOOD
15002  074712  104003                          ERROR   .3              ;FPP ERROR
15003                                                                  ;
15004  074714  020127  003132          1$:     CMP     R1,#TSTLOC+10           ;VERIFY R1
15005  074720  001401                          BEQ     2$                      ;BRANCH IF GOOD
15006  074722  104003                          ERROR   .3              ;FPP ERROR
15007                                                                  ;BAD R1
15008  074724  023727  003072  147501  2$:     CMP     @#RECST,#147501         ;VERIFY STATUS
15009  074732  001406                          BEQ     3$                      ;BRANCH F GOOD
15010  074734  104003                          ERROR   .3              ;FPP ERROR
15011                                                                  ;BAD STATUS\
15012  074736  000167  000006                  JMP     3$                      ;GET OVER TRAP
15013                                  ;UNEXPECTED TRAP
15014  074742  012600                  10$:    MOV     (SP)+,R0                        ;SAVE PC
15015  074744  012605                          MOV     (SP)+,R5                        ;SAVE PS
15016  074746  104003                          ERROR   .3              ;FPP ERROR
15017                                                                  ;UNEXPECTED TRAP
15018  074750                          3$:
15019
15020
15021                                  ;
15022                                  ;
15023                                  ;-----------------------------------
```

```
15024                                   ;TEST LDFPS, STFPS MODE 6
15025                                   ;
15026                                           ;
15027                                           ;
15028                                           ;
15029  074750                           MLS6:
15030
15031                                   ;
15032  074750  012704  003122                  MOV     #TSTLOC,R4              ;POINT R4 TO RAM
15033  074754  012737  140001  003126          MOV     #140001,@#TSTLOC+4      ;SETUP EXPECTED STATUS
15034  074762  012701  003232                  MOV     #TSTLOC+110,R1          ;R1 WILL POINT TO TESTLOC+10
15035  074766  012737  075042  000244          MOV     #10$,@#FPVEC            ;SETUP TRAP VECTOR
15036  074774  170164  000004                  LDFPS   4(R4)                   ;*TEST INSTRUCTION
15037  075000  170261  177700                  STFPS   -100(R1)                        ;*TEST INSTRUCTION
15038  075004  020427  003122                  CMP     R4,#TSTLOC              ;VERIFY R4
15039  075010  001401                          BEQ     1$                      ;BRANCH IF GOOD
15040  075012  104003                          ERROR   +3              ;FPP ERROR
15041                                                                   ;
15042  075014  020127  003232          1$:     CMP     R1,#TSTLOC+110          ;VERIFY R1
15043  075020  001401                          BEQ     2$                      ;BRANCH IF GOOD
15044  075022  104003                          ERROR   +3              ;FPP ERROR
15045                                                                   ;BAD R1
15046  075024  023727  003132  140001  2$:     CMP     @#TSTLOC+10,#140001     ;VERIFY STATUS
15047  075032  001406                          BEQ     3$                      ;BRANCH F GOOD
15048  075034  104003                          ERROR   +3              ;FPP ERROR
15049                                                                   ;BAD STATUS\
15050  075036  000167  000006                  JMP     3$                      ;GET OVER TRAP
15051                                   ;UNEXPECTED TRAP
15052  075042  012600                   10$:    MOV     (SP)+,R0                        ;SAVE PC
15053  075044  012605                           MOV     (SP)+,R5                        ;SAVE PS
15054  075046  104003                          ERROR   +3              ;FPP ERROR
15055                                                                   ;UNEXPECTED TRAP
15056  075050                           3$:
15057
15058
15059                                   ;
15060                                   ;
15061                                   ;-----------------------------------
15062                                   ;TEST LDFPS, STFPS MODE 7
15063                                   ;
15064                                           ;
15065                                           ;
15066                                           ;
15067  075050                           MLS7:
15068
15069                                   ;
15070  075050  012704  003222                  MOV     #TSTLOC+100,R4          ;POINT R4 TO RAM
15071  075054  012737  003126  003122          MOV     #TSTLOC+4,@#TSTLOC      ;TSTLOC= DEFERRED ADDRESS
15072  075062  012737  145501  003126          MOV     #145501,@#TSTLOC+4      ;SETUP EXPECTED STATUS
15073  075070  012701  003032                  MOV     #TSTLOC-70,R1           ;R1 POINTS TO TSTLOC+10
15074  075074  012737  003132  003124          MOV     #TSTLOC+10,@#TSTLOC+2   ;
15075  075102  012737  075156  000244          MOV     #10$,@#FPVEC            ;SETUP TRAP VECTOR
15076  075110  170174  177700                  LDFPS   @-100(R4)                       ;*TEST INSTRUCTION
15077  075114  170271  000072                  STFPS   @72(R1)                 ;*TEST INSTRUCTION
15078  075120  020427  003222                  CMP     R4,#TSTLOC+100          ;VERIFY R4
15079  075124  001401                          BEQ     1$                      ;BRANCH IF GOOD
```

```
15080  075126  104003                          ERROR   .3              ;FPP ERROR
15081                                                                  ;
15082  075130  020127  003032          1$:     CMP     R1,@TSTLOC-70           ;VERIFY R1
15083  075134  001401                          BEQ     2$                      ;BRANCH IF GOOD
15084  075136  104003                          ERROR   .3              ;FPP ERROR
15085                                                                  ;BAD R1
15086  075140  023727  003132  145501  2$:     CMP     @#TSTLOC+10,#145501     ;VERIFY STATUS
15087  075146  001406                          BEQ     3$                      ;BRANCH F GOOD
15088  075150  104003                          ERROR   .3              ;FPP ERROR
15089                                                                  ;BAD STATUS\
15090  075152  000167  000006                  JMP     3$                      ;GET OVER TRAP
15091                                  ;UNEXPECTED TRAP
15092  075156  012600                  10$:    MOV     (SP)+,R0                        ;SAVE PC
15093  075160  012605                          MOV     (SP)+,R5                        ;SAVE PS
15094  075162  104003                          ERROR   .3              ;FPP ERROR
15 95                                                                  ;UNEXPECTED TRAP
15096  075164                          3$:
15097
15098
15099                                   ;
15100                                   ;
15101                                   ;-------------------------------
15102                                   ;TEST LDCLD MODE 27
15103                                   ;
15104                                           ;
15105                                           ;
15106                                           ;
15107  075164                          MLDC2:
15108
15109                                   ;
15110  075164  005001                          CLR     R1                      ;INIT R1
15111  075166  012704  007700                  MOV     #7700,R4                ;FPS-DOUBLE, LONG
15112  075172  170104                          LDFPS   R4                      ;
15113  075174  012737  075230  000244          MOV     #10$,@#FPVEC            ;SETUP WILD TRAP
15114  075202  177027                          LDCLD   (R7)+,AC0               ;*TEST INSTRUCTION
15115  075204  005201                          INC     R1                      ;
15116  075206  005201                          INC     R1                      ;
15117  075210  005201                          INC     R1                      ;
15118  075212  005201                          INC     R1                      ;
15119  075214  020127  000003                  CMP     R1,#3                   ;VERIFY
15120  075220  001406                          BEQ     1$                      ;BRANCH IF GOOD
15121  075222  104003                          ERROR   .3              ;FPP ERROR
15122                                                                  ;INSTRUCTION FAILED
15123  075224  000167  000006                  JMP     1$                      ;JUMP OVER WILD TRAP
15124  075230  012600                  10$:    MOV     (SP)+,R0                ;SAVE PC
15125  075232  012605                          MOV     (SP)+,R5                        ;SAVE PS
15126  075234  104003                          ERROR   .3              ;FPP ERROR
15127                                                                  ;WILD TRAP ON INSTRUCTION
15128  075236  012704  003264          1$:     MOV     #TAB6A,R4               ;POINT TO EXPECTED DATA
15129  075242  012701  003102                  MOV     #RECDST,R1              ;POINT TO DATA BUFFER
15130  075246  174011                          STD     AC0,(R1)                ;VERIFY DATA
15131  075250  004767  036206                  JSR     R7,DATVER               ;
15132  075254  005767  105600                  TST     COUNT
15133  075260  001401                          BEQ     2$                      ;BRANCH IF GOOD DATA
15134  075262  104003                          ERROR   .3              ;FPP ERROR
15135                                                                  ;BAD DATA
```

```
15136  075264                          2$:
15137
15138
15139
15140                                  ;
15141                                  ;
15142                                  ;-----------------------------------
15143                                  ;TEST LDCIF, LDCLF
15144                                  ;
15145                                          ;
15146                                          ;
15147                                          ;
15148  075264                          MLCF:
15149
15150                                  ;
15151                                  ;1/INT=0
15152  075264  004767  000500                  JSR     R7,LCFSUB               ;DO TEST
15153  075270  000000  000000                  .WORD   0,0             ;FSRC
15154  075274  000000  000000                  .WORD   0,0             ;RESULT
15155  075300  000000                          .WORD   0               ; TEST FPS
15156  075302  000004                          .WORD   4               ;RESULT FPS
15157                                  ;2/INT=0,-1
15158  075304  004767  000460                  JSR     R7,LCFSUB               ;DO TEST
15159  075310  000000  177777                  .WORD   0,-1            ;FSRC
15160  075314  000000  000000                  .WORD   0,0             ;RESULT
15161  075320  007440                          .WORD   7440            ; TEST FPS
15162  075322  007444                          .WORD   7444            ;RESULT FPS
15163                                  ;3/LONG=0
15164  075324  004767  000440                  JSR     R7,LCFSUB               ;DO TEST
15165  075330  000000  000000                  .WORD   0,0             ;FSRC
15166  075334  000000  000000                  .WORD   0,0             ;RESULT
15167  075340  000100                          .WORD   100             ; TEST FPS
15168  075342  000104                          .WORD   104             ;RESULT FPS
15169                                  ;4/INT=40000
15170  075344  004767  000420                  JSR     R7,LCFSUB               ;DO TEST
15171  075350  040000  000000                  .WORD   40000,0         ;FSRC
15172  075354  043600  000000                  .WORD   43600,0         ;RESULT
15173  075360  000017                          .WORD   17              ; TEST FPS
15174  075362  000000                          .WORD   0               ;RESULT FPS
15175                                  ;5/LONG=1
15176  075364  004767  000400                  JSR     R7,LCFSUB               ;DO TEST
15177  075370  000000  000001                  .WORD   0,1             ;FSRC
15178  075374  040200  000000                  .WORD   40200,0         ;RESULT
15179  075400  000117                          .WORD   117             ; TEST FPS
15180  075402  000100                          .WORD   100             ;RESULT FPS
15181                                  ;6/INT=PATTERN
15182  075404  004767  000360                  JSR     R7,LCFSUB               ;DO TEST
15183  075410  000252  025252                  .WORD   252,25252       ;FSRC
15184  075414  042052  000000                  .WORD   42052,0         ;RESULT
15185  075420  000000                          .WORD   0               ; TEST FPS
15186  075422  000000                          .WORD   0               ;RESULT FPS
15187                                  ;7/INT=-40000
15188  075424  004767  000340                  JSR     R7,LCFSUB               ;DO TEST
15189  075430  140000  000000                  .WORD   -40000,0        ;FSRC
15190  075434  143600  000000                  .WORD   143600,0                ;RESULT
15191  075440  000007                          .WORD   7               ; TEST FPS
```

```
075442  000010                          .WORD   10              ;RESULT FPS
                                ;8/INT=-1
075444  004767  000320                  JSR     R7,LCFSUB               ;DO TEST
075450  177777  000000                  .WORD   -1,0            ;FSRC
075454  140200  000000                  .WORD   140200,0                ;RESULT
075460  000007                          .WORD   7               ; TEST FPS
075462  000010                          .WORD   10              ;RESULT FPS
                                ;9/INT=PATTERN
075464  004767  000300                  JSR     R7,LCFSUB               ;DO TEST
075470  125252  125252                  .WORD   125252,125252           ;FSRC
075474  143652  126000                  .WORD   143652,126000           ;RESULT
075500  000007                          .WORD   7               ; TEST FPS
075502  000010                          .WORD   10              ;RESULT FPS
                                ;10/LONG=40000
075504  004767  000260                  JSR     R7,LCFSUB               ;DO TEST
075510  040000  000000                  .WORD   40000,0         ;FSRC
075514  047600  000000                  .WORD   47600,0         ;RESULT
075520  000117                          .WORD   117             ; TEST FPS
075522  000100                          .WORD   100             ;RESULT FPS
                                ;11/LONG=1
075524  004767  000240                  JSR     R7,LCFSUB               ;DO TEST
075530  000000  000001                  .WORD   0,1             ;FSRC
075534  040200  000000                  .WORD   40200,0         ;RESULT
075540  007557                          .WORD   7557            ; TEST FPS
075542  007540                          .WORD   7540            ;RESULT FPS
                                ;12/LONG=PATTERN
075544  004767  000220                  JSR     R7,LCFSUB               ;DO TEST
075550  000000  000252                  .WORD   0,252           ;FSRC
075554  042052  000000                  .WORD   42052,0         ;RESULT
075560  007557                          .WORD   7557            ; TEST FPS
075562  007540                          .WORD   7540            ;RESULT FPS
                                ;13/LONG = -40000
075564  004767  000200                  JSR     R7,LCFSUB               ;DO TEST
075570  140000  000000                  .WORD   -40000,0                ;FSRC
075574  147600  000000                  .WORD   147600,0                ;RESULT
075600  000107                          .WORD   107             ; TEST FPS
075602  000110                          .WORD   110             ;RESULT FPS
                                ;14/LONG=-1
075604  004767  000160                  JSR     R7,LCFSUB               ;DO TEST
075610  177777  177777                  .WORD   -1,-1           ;FSRC
075614  140200  000000                  .WORD   140200,0                ;RESULT
075620  007500                          .WORD   7500            ; TEST FPS
075622  007510                          .WORD   7510            ;RESULT FPS
                                ;15/LONG=PATTERN
075624  004767  000140                  JSR     R7,LCFSUB               ;DJ TEST
075630  125252  125252                  .WORD   125252,125252           ;FSRC
075634  147652  125253                  .WORD   147652,125253           ;RESULT
075640  000105                          .WORD   105             ; TEST FPS
075642  000110                          .WORD   110             ;RESULT FPS
                                ;16/LONG=77777,177500
075644  004767  000120                  JSR     R7,LCFSUB               ;DO TEST
075650  077777  177500                  .WORD   77777,177500            ;FSRC
075654  047777  177777                  .WORD   47777,177777            ;RESULT
075660  000117                          .WORD   117             ; TEST FPS
075662  000100                          .WORD   100             ;RESULT FPS
                                ;17/LONG=40000,100
```

```
15248  075664  004767  000100              JSR     R7,LCFSUB               ;DO TEST
15249  075670  040000  000100              .WORD   40000,100               ;FSRC
15250  075674  047600  000001              .WORD   47600,1         ;RESULT
15251  075700  007502                      .WORD   7502            ; TEST FPS
15252  075702  007500                      .WORD   7500            ;RESULT FPS
15253                              ;18/LONG=40000,100 - TRUNCATE
15254  075704  004767  000060              JSR     R7,LCFSUB               ;DO TEST
15255  075710  040000  000100              .WORD   40000,100               ;FSRC
15256  075714  047600  000000              .WORD   47600,0         ;RESULT
15257  075720  007557                      .WORD   7557            ; TEST FPS
15258  075722  007540                      .WORD   7540            ;RESULT FPS
15259                              ;19/INT= MOST NEGATIVE
15260  075724  004767  000040              JSR     R7,LCFSUB               ;DO TEST
15261  075730  100000  000000              .WORD   100000,0                ;FSRC
15262  075734  144000  000000              .WORD   144000,0                ;RESULT
15263  075740  000007                      .WORD   7               ; TEST FPS
15264  075742  000010                      .WORD   10              ;RESULT FPS
15265                              ;20/LONG= MOST NEGATIVE
15266  075744  004767  000020              JSR     R7,LCFSUB               ;DO TEST
15267  075750  100000  000000              .WORD   100000,0                ;FSRC
15268  075754  150000  000000              .WORD   150000,0                ;RESULT
15269  075760  000107                      .WORD   107             ; TEST FPS
15270  075762  000110                      .WORD   110             ;RESULT FPS
15271                              ;
15272
15273  075764  000167  000112              JMP     HOP18           ;GET OVER SUBROUTINE
15274                              ;
15275                              ;
15276                              ;*X*X*X
```

```
15304  076034  020003                  CMP     R0,R3                   ;VERIFY STATUS
15305  076036  001401                  BEQ     3$                      ;BRANCH IF GOOD
15306  076040  104003                  ERROR   .3              ;FPP ERROR
15307                                                          ;BAD FPS
15308  076042  010204          3$:     MOV     R2,R4                   ;POINT TO EXPECTED DATA
15309  076044  062704  000004          ADD     #4,R4
15310  076050  004767  035370  4$:     JSR     R7,DATVFR                       ;VERIFY DATA
15311  076054  005767  105000          TST     COUNT
15312  076060  001401                  BEQ     5$                      ;BRANCH IF GOOD
15313  076062  104003                  ERROR   .3              ;FPP ERROR
15314                                                          ;BAD ACO
15315  076064  000162  000014  5$:     JMP     14(R2)                  ;RETURN FROM TEST
15316                                  ;
15317                                  ;INSTRUCTION TRAPPED
15318  076070  012600                  50$:    MOV     (SP)+,R0                ;SAVE PC
15319  076072  012605                  MOV     (SP)+,R5                ;SAVE PS
15320  076074  104003                  ERROR   .3              ;FPP ERROR
15321                                                          ;INSTRUCTION WASNT SUPPOSE TO TRAP
15322  076076  000167  177762          JMP     5$                      ;CONTINUE
15323  076102                  HOP18:
15324
15325                                  ;
15326                                  ;
15327                                  ;-----------------------------------
15328                                  ;TEST LDCID, LDCLD
15329                                  ;
15330                                          ;
15331                                          ;
15332                                          ;
15333  076102                  MLCD:
15334
15335                                  ;
15336                                  ;1/LONG=0
15337  076102  004767  000264                  JSR     R7,LCDSUB               ;DO TEST
15338  076106  000000  000000                  .WORD   0,0             ;FSRC
15339  076112  000000  000000  000000          .WORD   0,0,0,0         ;RESULT
15340  076120  000000
15341  076122  007313                  .WORD   7313            ; TEST FPS
15342  076124  007304                  .WORD   7304            ;RESULT FPS
15343                                  ;2/INT=0
15344  076126  004767  000240                  JSR     R7,LCDSUB               ;DO TEST
15345  076132  000000  000001                  .WORD   0,1             ;FSRC
15346  076136  040200  000000  000000          .WORD   40200,0,0,0             ;RESULT
15347  076144  000000
15348  076146  007757                  .WORD   7757            ; TEST FPS
15349  076150  007740                  .WORD   7740            ;RESULT FPS
15350                                  ;3/INT=40000
15351  076152  004767  000214                  JSR     R7,LCDSUB               ;DO TEST
15352  076156  040000  177777                  .WORD   40000,-1                ;FSRC
15353  076162  043600  000000  000000          .WORD   43600,0,0,0             ;RESULT
15354  076170  000000
15355  076172  007617                  .WORD   7617            ; TEST FPS
15356  076174  007600                  .WORD   7600            ;RESULT FPS
15357                                  ;4/INT=-40000
15358  076176  004767  000170                  JSR     R7,LCDSUB               ;DO TEST
15359  076202  140000  177777                  .WORD   -40000,1                ;FSRC
```

```
15360  076206  143600  000000  000000          .WORD   143600,0,0,0            ;RESULT
15361  076214  000000
15362  076216  007600                          .WORD   7600            ; TEST FPS
15363  076220  007610                          .WORD   7610            ;RESULT FPS
15364                          ;5/LONG=40000
15365  076222  004767  000144                  JSR     R7,LCDSUB               ;DO TEST
15366  076226  040000  000000                  .WORD   40000,0         ;FSRC
15367  076232  047600  000000  000000          .WORD   47600,0,0,0             ;RESULT
15368  076240  000000
15369  076242  007757                          .WORD   7757            ; TEST FPS
15370  076244  007740                          .WORD   7740            ;RESULT FPS
15371                          ;6/LONG=1
15372  076246  004767  000120                  JSR     R7,LCDSUB               ;DO TEST
15373  076252  000000  000001                  .WORD   0,1             ;FSRC
15374  076256  040200  000000  000000          .WORD   40200,0,0,0             ;RESULT
15375  076264  000000
15376  076266  000300                          .WORD   300             ; TEST FPS
15377  076270  000300                          .WORD   300             ;RESULT FPS
15378                          ;7/LONG=-2
15379  076272  004767  000074                  JSR     R7,LCDSUB               ;DO TEST
15380  076276  177777  177776                  .WORD   -1,-2           ;FSRC
15381  076302  140400  000000  000000          .WORD   140400,0,0,0            ;RESULT
15382  076310  000000
15383  076312  007300                          .WORD   7300            ; TEST FPS
15384  076314  007310                          .WORD   7310            ;RESULT FPS
15385                          ;8/INT=PATTERN
15386  076316  004767  000050                  JSR     R7,LCDSUB               ;DO TEST
15387  076322  123456  176543                  .WORD   123456,176543           ;FSRC
15388  076326  143661  122000  000000          .WORD   143661,122000,0,0               ;RESULT
15389  076334  000000
15390  076336  000200                          .WORD   200             ; TEST FPS
15391  076340  000210                          .WORD   210             ;RESULT FPS
15392                          ;9/LONG=PATTERN
15393  076342  004767  000024                  JSR     R7,LCDSUB               ;DO TEST
15394  076346  125252  125252                  .WORD   125252,125252           ;FSRC
15395  076352  147652  125252  126000          .WORD   147652,125252,126000,0          ;RESULT
15396  076360  000000
15397  076362  000300                          .WORD   300             ; TEST FPS
15398  076364  000310                          .WORD   310             ;RESULT FPS
15399                          ;
15400
15401                          ;
15402  076366  000167  000112                  JMP     HOP19           ;GET OVER SUBROUTINE
15403                          ;
15404                          ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
15405                          ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
15406                          ;LDCID, LDCLD
15407                          ;
15408                          ;               FSRC
15409                          ;               RESULT
15410                          ;               FPS BEFORE EXECUTION
15411                          ;               FPS AFTER EXECUTION
15412                          ;
15413                          ;NO TRAP CAN OCCUR
15414                          ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
15415                          ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
```

```
15416
15417  076372  012602                  LCDSUB: MOV     (SP)+,R2        ; RETURN ADDRESS TO USE AS POINTER
15418  076374  012737  076472  000244          MOV     #50$,@#FPVEC    ;REDIRECT TRAP VECTOR
15419  076402  012701  003102                  MOV     #RECDST,R1      ;POINT TO RESULT AREA
15420  076406  016200  000014                  MOV     14(R2),R0       ;GET TEST FPS
15421  076412  170100                          LDFPS   R0              ;LOAD TEST FPS
15422  076414  010204                          MOV     R2,R4           ;POINT TO TEST DATA
15423                                  ;
15424  076416  177014                  40$:    LDCID   (R4),AC0        ;*TEST INSTRUCTION (ACCORDING TO MODE)
15425                                  ;
15426                                  ;VERIFY STATUS
15427  076420  170203                  2$:     STFPS   R3              ;SAVE FPS
15428  076422  012700  000200                  MOV     #200,R0         ;SET FPP STATUS TO DOUBLE
15429  076426  170100                          LDFPS   R0              ;
15430  076430  174011                          STD     AC0,(R1)        ;SAVE TEST RESULT INTO RECDST
15431  076432  016200  000016                  MOV     16(R2),R0               ;GET EXPECTED STATUS
15432  076436  020003                          CMP     R0,R3           ;VERIFY STATUS
15433  076440  001401                          BEQ     3$              ;BRANCH IF GOOD
15434  076442  104003                          ERROR   +3      ;FPP ERROR
15435                                  ;BAD FPS
15436  076444  010204                  3$:     MOV     R2,R4           ;POINT TO EXPECTED DATA
15437  076446  062704  000004                  ADD     #4,R4
15438  076452  004767  035004          4$:     JSR     R7,DATVER               ;VERIFY DATA
15439  076456  005767  104376                  TST     COUNT
15440  076462  001401                          BEQ     5$              ;BRANCH IF GOOD
15441  076464  104003                          ERROR   +3      ;FPP ERROR
15442                                  ;BAD AC0
15443  076466  000162  000020          5$:     JMP     20(R2)          ;RETURN FROM TEST
15444                                  ;
15445                                  ;INSTRUCTION TRAPPED
15446  076472  012600                  50$:    MOV     (SP)+,R0                ;SAVE PC
15447  076474  012605                          MOV     (SP)+,R5                ;SAVE PS
15448  076476  104003                          ERROR   +3              ;FPP ERROR
15449                                                                  ;INSTRUCTION WASNT SUPPOSE TO TRAP
15450  076500  000167  177762                  JMP     5$                      ;CONTINUE
15451                                  ;
15452  076504                          HOP19:
15453
15454                                  ;
15455                                  ;
15456                                  ;-----------------------------------
15457                                  ;TEST LDEXP
15458                                  ;DOUBLE
15459                                  ;
15460                                          ;
15461                                          ;
15462                                          ;
15463  076504                          MLXP:
15464
15465                                  ;
15466                                  ;1/EXP=10 - AC=NEG
15467  076504  005037  002776                  CLR     @#FLAG                  ;NO INTERRUPTS
15468  076510  004767  001140                  JSR     R7,LXPSUB               ;DO TEST
15469  076514  123456  067012  025252          .WORD   123456,67012,25252,171717       ;AC0
15470  076522  171717
15471  076524  000010                          .WORD   10                      ;EXP
```

```
15472  076526  142056  067012  025252          .WORD   142056,67012,25252,171717               ;RESULT
15473  076534  171717
15474  076536  007757                          .WORD   7757            ; TEST FPS
15475  076540  007750                          .WORD   7750            ;RESULT FPS
15476                          ;2/EXP=177 - ACO=POS
15477  076542  005037  002776                  CLR     @#FLAG                  ;NO INTERRUPTS
15478  076546  004767  001102                  JSR     R7,LXPSUB               ;DO TEST
15479  076552  023456  070123  100000          .WORD   23456,70123,100000,1            ;ACO
15480  076560  000001
15481  076562  000177                          .WORD   177             ;EXP
15482  076564  077656  070123  100000          .WORD   77656,70123,100000,1            ;RESULT
15483  076572  000001
15484  076574  007700                          .WORD   7700            ; TEST FPS
15485  076576  007700                          .WORD   7700            ;RESULT FPS
15486                          ;3/EXP=56
15487  076600  005037  002776                  CLR     @#FLAG                  ;NO INTERRUPTS
15488  076604  004767  001044                  JSR     R7,LXPSUB               ;DO TEST
15489  076610  055555  044444  033333          .WORD   55555,44444,33333,22222         ;ACO
15490  076616  022222
15491  076620  000056                          .WORD   56              ;EXP
15492  076622  053555  044444  033333          .WORD   53555,44444,33333,22222 ;RESULT
15493  076630  022222
15494  076632  007757                          .WORD   7757            ; TEST FPS
15495  076634  007740                          .WORD   7740            ;RESULT FPS
15496                          ;4/EXP=-151, ACO=UV
15497  076636  005037  002776                  CLR     @#FLAG                  ;NO INTERRUPTS
15498  076642  004767  001006                  JSR     R7,LXPSUB               ;DO TEST
15499  076646  100077  177777  177777          .WORD   100077,-1,-1,-2         ;ACO
15500  076654  177776
15501  076656  177623                          .WORD   -155            ;EXP
15502  076660  104677  177777  177777          .WORD   104677,-1,-1,-2         ;RESULT
15503  076666  177776
15504  076670  007757                          .WORD   7757            ; TEST FPS
15505  076672  007750                          .WORD   7750            ;RESULT FPS
15506                          ;5/EXP=-177
15507  076674  005037  002776                  CLR     @#FLAG                  ;NO INTERRUPTS
15508  076700  004767  000750                  JSR     R7,LXPSUB               ;DO TEST
15509  076704  000177  177777  177777          .WORD   177,-1,-1,-2            ;ACO
15510  076712  177776
15511  076714  177601                          .WORD   -177            ;EXP
15512  076716  000377  177777  177777          .WORD   377,-1,-1,-2            ;RESULT
15513  076724  177776
15514  076726  007700                          .WORD   7700            ; TEST FPS
15515  076730  007700                          .WORD   7700            ;RESULT FPS
15516                          ;6/EXP=-200, UNDERFLOW
15517  076732  012737  000001  002776          MOV     #1,@#FLAG               ; INTERRUPTS
15518  076740  004767  000710                  JSR     R7,LXPSUB               ;DO TEST
15519  076744  030131  032334  035363          .WORD   30131,32334,35363,73031         ;ACO
15520  076752  073031
15521  076754  177600                          .WORD   -200            ;EXP
15522  076756  000131  032334  035363          .WORD   131,32334,35363,73031           ;RESULT
15523  076764  073031
15524  076766  007740                          .WORD   7740            ; TEST FPS
15525  076770  107744                          .WORD   107744          ;RESULT FPS
15526  076772  000012                          .WORD   12              ;FEC
15527                          ;7/EXP=LARGEST NEGATIVE
```

```
15528  076774  012737  000001  002776          MOV     #1,@#FLAG               ;EXPECT  INTERRUPTS
15529  077002  004767  000646                  JSR     R7,LXPSUB               ;DO TEST
15530  077006  000000  000123  000456          .WORD   0,123,456,1             ;ACO
15531  077014  000001
15532  077016  100000                          .WORD   100000          ;EXP
15533  077020  040000  000123  000456          .WORD   40000,123,456,1         ;RESULT
15534  077026  000001
15535  077030  002200                          .WORD   2200            ; TEST FPS
15536  077032  102200                          .WORD   102200          ;RESULT FPS
15537  077034  000012                          .WORD   12              ;FEC
15538                          ;8/EXP=-200, NEG. ACO
15539  077036  012737  000001  002776          MOV     #1,@#FLAG               ; INTERRUPTS
15540  077044  004767  000604                  JSR     R7,LXPSUB               ;DO TEST
15541  077050  111111  100000  100000          .WORD   111111,100000,100000,-1         ;ACO
15542  077056  177777
15543  077060  177600                          .WORD   -200            ;EXP
15544  077062  100111  100000  100000          .WORD   100111,100000,100000,-1         ;RESULT
15545  077070  177777
15546  077072  002217                          .WORD   2217            ; TEST FPS
15547  077074  102214                          .WORD   102214          ;RESULT FPS
15548  077076  000012                          .WORD   12              ;FEC
15549                          ;9/EXP=-1743, FIU=0
15550  077100  012737  000002  002776          MOV     #2,@#FLAG               ;NO INTERRUPTS
15551  077106  004767  000542                  JSR     R7,LXPSUB               ;DO TEST
15552  077112  123456  012346  012346          .WORD   123456,12346,12346,123          ;ACO
15553  077120  000123
15554  077122  176035                          .WORD   -1743           ;EXP
15555  077124  000000  000000  000000          .WORD   0,0,0,0         ;RESULT
15556  077132  000000
15557  077134  005700                          .WORD   5700            ; TEST FPS
15558  077136  005704                          .WORD   5704            ;RESULT FPS
15559  077140  000012                          .WORD   12              ;FEC
15560                          ;10/EXP =-16616, FID=1
15561  077142  012737  000002  002776          MOV     #2,@#FLAG               ;NO INTERRUPTS
15562  077150  004767  000500                  JSR     R7,LXPSUB               ;DO TEST
15563  077154  000377  123456  065432          .WORD   377,123456,65432,1              ;ACO
15564  077162  000001
15565  077164  161162                          .WORD   -16616          ;EXP
15566  077166  074577  123456  065432          .WORD   74577,123456,65432,1            ;RESULT
15567  077174  000001
15568  077176  047700                          .WORD   47700           ; TEST FPS
15569  077200  147700                          .WORD   147700          ;RESULT FPS
15570  077202  000012                          .WORD   12              ;FEC
15571                          ;11/EXP=177, ACO=UNDEFINED VARIABLE
15572  077204  005037  002776                  CLR     @#FLAG          ;NO INTERRUPTS
15573  077210  004767  000440                  JSR     R7,LXPSUB               ;DO TEST
15574  077214  100177  177777  177777          .WORD   100177,-1,-1,-1         ;ACO
15575  077222  177777
15576  077224  000177                          .WORD   177             ;EXP
15577  077226  177777  177777  177777          .WORD   -1,-1,-1,-1             ;RESULT
15578  077234  177777
15579  077236  007700                          .WORD   7700            ; TEST FPS
15580  077240  007710                          .WORD   7710            ;RESULT FPS
15581                          ;12/EXP=150 ACO=POS
15582  077242  005037  002776                  CLR     @#FLAG          ;NO INTERRUPT
15583  077246  004767  000402                  JSR     R7,LXPSUB               ;DO TEST
```

K6


```
15584  077252  000200  000100  000200          .WORD   200,100,200,300         ;ACO
15585  077260  000300
15586  077262  000150                          .WORD   150                 ;EXP
15587  077264  072000  000100  000200          .WORD   72000,100,200,300               ;RESULT
15588  077272  000300
15589  077274  007717                          .WORD   7717                ; TEST FPS
15590  077276  007700                          .WORD   7700                ;RESULT FPS
15591                          ;13/EXP=200, ACO=NEG
15592  077300  012737  000001  002776          MOV     #1,@#FLAG
15593  077306  004767  000342                  JSR     R7,LXPSUB                   ;DO TEST
15594  077312  177777  177777  177777          .WORD   -1,-1,-1,-1                 ;ACO
15595  077320  177777
15596  077322  000200                          .WORD   200                 ;EXP
15597  077324  100177  177777  177777          .WORD   100177,-1,-1,-1             ;RESULT
15598  077332  177777
15599  077334  007705                          .WORD   7705                ; TEST FPS
15600  077336  107716                          .WORD   107716              ;RESULT FPS
15601  077340  000010                          .WORD   10                  ;FEC
15602                          ;14/EXP=400, FID
15603  077342  012737  000002  002776          MOV     #2,@#FLAG                   ;INTERRUPT
15604  077350  004767  000300                  JSR     R7,LXPSUB                   ;DO TEST
15605  077354  000555  177777  177776          .WORD   555,-1,-2,-3                ;ACO
15606  077362  177775
15607  077364  000400                          .WORD   400                 ;EXP
15608  077366  040155  177777  177776          .WORD   40155,-1,-2,-3              ;RESULT
15609  077374  177775
15610  077376  047700                          .WORD   47700               ; TEST FPS
15611  077400  147702                          .WORD   147702              ;RESULT FPS
15612  077402  000010                          .WORD   10                  ;FEC
15613                          ;15/EXP=11011 FIU=0
15614  077404  012737  000000  002776          MOV     #0,@#FLAG                   ;NO INTERRUPT
15615  077412  004767  000236                  JSR     R7,LXPSUB                   ;DO TEST
15616  077416  177773  177777  177776          .WORD   177773,-1,-2,-3             ;ACO
15617  077424  177775
15618  077426  011011                          .WORD   11011               ;EXP
15619  077430  000000  000000  000000          .WORD   0,0,0,0             ;RESULT
15620  077436  000000
15621  077440  006700                          .WORD   6700                ; TEST FPS
15622  077442  006706                          .WORD   6706                ;RESULT FPS
15623                          ;16/EXP=LARGEST POSITIVE
15624  077444  012737  000001  002776          MOV     #1,@#FLAG                   ;INTERRUPT
15625  077452  004767  000176                  JSR     R7,LXPSUB                   ;DO TEST
15626  077456  123456  000100  000100          .WORD   123456,100,100,200                  ;ACO
15627  077464  000200
15628  077466  077777                          .WORD   77777               ;EXP
15629  077470  137656  000100  000100          .WORD   137656,100,100,200                  ;RESULT
15630  077476  000200
15631  077500  007740                          .WORD   7740                ; TEST FPS
15632  077502  107752                          .WORD   107752              ;RESULT FPS
15633  077504  000010                          .WORD   10                  ;FEC
15634                          ;17/FLOATING
15635  077506  005037  002776                  CLR     @#FLAG              ;NO INTERRUPT
15636  077512  004767  000136                  JSR     R7,LXPSUB                   ;DO TEST
15637  077516  123456  023465  000555          .WORD   123456,23465,555,444                ;ACO
15638  077524  000444
15639  077526  000050                          .WORD   50                  ;EXP
```

```
15640  077530  152056  023465  000555          .WORD   152056,23465,555,444           ;RESULT
15641  077536  000444
15642  077540  007500                          .WORD   7500            ; TEST FPS
15643  077542  007510                          .WORD   7510            ;RESULT FPS
15644                          ;18/FLOATING UNDERFLOW
15645  077544  012737  000001  002776          MOV     #1,@#FLAG               ;INTERRUPT
15646  077552  004767  000076                  JSR     R7,LXPSUB               ;DO TEST
15647  077556  000333  000444  000555          .WORD   333,444,555,666         ;ACO
15648  077564  000666
15649  077566  177600                          .WORD   -200            ;EXP
15650  077570  000133  000444  000555          .WORD   133,444,555,666         ;RESULT
15651  077576  000666
15652  077600  007500                          .WORD   7500            ; TEST FPS
15653  077602  107504                          .WORD   107504          ;RESULT FPS
15654  077604  000012                          .WORD   12              ;FEC
15655                          ;19/FLOATING OVERFLOW
15656  077606  012737  000001  002776          MOV     #1,@#FLAG               ;INTERRUPT
15657  077614  004767  000034                  JSR     R7,LXPSUB               ;DO TEST
15658  077620  012346  000123  000345          .WORD   12346,123,345,456               ;ACO
15659  077626  000456
15660  077630  000400                          .WORD   400             ;EXP
15661  077632  040146  000123  000345          .WORD   40146,123,345,456               ;RESULT
15662  077640  000456
15663  077642  007400                          .WORD   7400            ; TEST FPS
15664  077644  107402                          .WORD   107402          ;RESULT FPS
15665  077646  000010                          .WORD   10              ;FEC
15666                          ;
15667
15668  077650  000167  000220                  JMP     HOP20                   ;GET OVER SUBROUTINE
15669                          ;*X*X*X
```

M6


```
15696                                   ;
15697                                   ;
15698                                   ;INSTRUCTION DIDNT TRAP
15699  077722  032737  000001  002776          BIT     #1,@#FLAG               ;VERIFY A NO TRAP CONDITION
15700  077730  001420                          BEQ     2$                      ;BRANCH IF GOOD
15701  077732  104003                          ERROR   .3              ;FPP ERROR
15702                                                                 ;INSTRUCTION SHOULD HAVE TRAPPED
15703  077734  000167  000032                  JMP     2$                      ;REJOIN CODE
15704                                   ;
15705                                   ;INSTRUCTION TRAPPED
15706  077740  032737  000001  002776  50$:    BIT     #1,@#FLAG               ;SEE IF EXPECTING A TRAP
15707  077746  001003                          BNE     51$                     ;BRANCH IF EXPECTING A TRAP
15708  077750  104003                          ERROR   .3              ;FPP ERROR
15709                                                                 ;INSTRUCTION WASNT SUPPOSE TO TRAP
15710  077752  000167  000014                  JMP     2$                      ;REJOIN CODE
15711  077756  012604                  51$:    MOV     (SP)+,R4                ;SEE IF PC = INSTRUCTION
15712  077760  005726                          TST     (SP)+                   ;CLEAN UP STACK
15713  077762  022704  077716                  CMP     #1$,R4                  ;
15714  077766  001401                          BEQ     2$                      ;BRANCH IF GOOD COMPARE
15715  077770  104003                          ERROR   .3              ;FPP ERROR
15716                                                                 ;PC WAS INCORRECT
15717                                   ;
15718                                   ;COMMON CODE FOR TRAP AND NO TRAP
15719                                   ;VERIFY STATUS
15720  077772  170203                  2$:     STFPS   R3                      ;SAVE FPS
15721  077774  012700  000200                  MOV     #200,R0                 ;SETUP FPS
15722  100000  170100                          LDFPS   R0                      ;FPS=200
15723  100002  174011                          STD     AC0,(R1)                ;GET RESULT
15724  100004  016200  000024                  MOV     24(R2),R0                       ;GET EXPECTED STATUS
15725  100010  020003                          CMP     R0,R3                   ;VERIFY STATUS
15726  100012  001401                          BEQ     3$                      ;BRANCH IF GOOD
15727  100014  104003                          ERROR   .3              ;FPP ERROR
15728                                                                 ;BAD FPS
15729  100016  010204                  3$:     MOV     R2,R4                   ;POINT TO EXPECTED DATA
15730  100020  062704  000012                  ADD     #12,R4
15731  100024  004767  033432          4$:     JSR     R7,DATVER                       ;VERIFY DATA
15732  100030  005767  103024                  TST     COUNT
15733  100034  001401                          BEQ     5$                      ;BRANCH IF GOOD
15734  100036  104003                          ERROR   .3              ;FPP ERROR
15735                                                                 ;BAD AC0
15736  100040  005737  002776          5$:     TST     @#FLAG                  ;SEE IF NEED TO CHECK FEC
15737  100044  001002                          BNE     7$                      ;BRANCH IF NEED TO CHECK
15738  100046  000162  000026                  JMP     26(R2)                  ;RETURN FROM TEST
15739                                   ;VERIFY FEC
15740  100052  012704  003062          7$:     MOV     #RECFEC,R4              ;POINT TO FEC AREA
15741  100056  170314                          STST    (R4)                    ;SAVE FEC
15742  100060  021462  000026                  CMP     (R4),26(R2)                     ;VERIFY FEC FOR OVERFLOW
15743  100064  001401                          BEQ     8$                      ;BRANCH IF GOOD
15744  100066  104003                          ERROR   .3              ;FPP ERROR
15745                                                                 ;BAD FEC
15746  100070  000162  000030          8$:     JMP     30(R2)          ;RETURN FROM TEST
15747                                   ;
15748  100074                          HOP20:
15749
15750                                   ;
15751                                   ;
```

```
15752                                   ;-----------------------------------
15753                                   ;TEST STCDI, STCDL
15754                                   ;
15755                                           ;
15756                                           ;
15757                                           ;
15758  100074                           MSCD:
15759
15760                                   ;
15761                                   ;1/AC0=0, INT
15762  100074  005037  002776                  CLR     @#FLAG          ;NO INTERRUPTS
15763  100100  004767  000610                  JSR     R7,SCDSUB               ;DO TEST
15764  100104  000177  000000  000000          .WORD   0177,0,0,0              ;AC0
15765  100112  000000
15766  100114  000000  177777                  .WORD   0,-1            ;RESULT
15767  100120  007640                          .WORD   7640            ; TEST FPS
15768  100122  007644                          .WORD   7644            ;RESULT FPS
15769                                   ;2/AC0=-0, LONG
15770  100124  005037  002776                  CLR     @#FLAG          ;INTERRUPT
15771  100130  004767  000560                  JSR     R7,SCDSUB               ;DO TEST
15772  100134  100177  177777  177777          .WORD   100177,-1,-1,-1         ;AC0
15773  100142  177777
15774  100144  000000  000000                  .WORD   0,0             ;RESULT
15775  100150  007700                          .WORD   7700            ; TEST FPS
15776  100152  007704                          .WORD   7704            ;RESULT FPS
15777                                   ;3/EXP=100, LONG
15778  100154  005037  002776                  CLR     @#FLAG                  ;NO INTERRUPT
15779  100160  004767  000530                  JSR     R7,SCDSUB               ;DO TEST
15780  100164  020000  000000  000000          .WORD   20000,0,0,0             ;AC0
15781  100172  000000
15782  100174  000000  000000                  .WORD   0,0             ;RESULT
15783  100200  000300                          .WORD   300             ; TEST FPS
15784  100202  000304                          .WORD   304             ;RESULT FPS
15785                                   ;4/EXP=200, BAISED 0, INT, ROUND
15786  100204  005037  002776                  CLR     @#FLAG          ;INTERRUPT
15787  100210  004767  000500                  JSR     R7,SCDSUB                       ;DO TEST
15788  100214  140177  177777  000001          .WORD   140177,177777,1,1                       ;AC0
15789  100222  000001
15790  100224  000000  000000                  .WORD   0,0             ;RESULT
15791  100230  007700                          .WORD   7700            ; TEST FPS
15792  100232  007704                          .WORD   7704            ;RESULT FPS
15793                                   ;5/LONG
15794  100234  005037  002776                  CLR     @#FLAG          ;INTERRUPT
15795  100240  004767  000450                  JSR     R7,SCDSUB               ;DO TEST
15796  100244  047667  075757  157737          .WORD   47667,75757,157737,167773                       ;AC0
15797  100252  167773
15798  100254  055675  173757                  .WORD   55675,173757            ;RESULT
15799  100260  007717                          .WORD   7717            ; TEST FPS
15800  100262  007700                          .WORD   7700            ;RESULT FPS
15801                                   ;6/LONG, EXP=2**32
15802  100264  005037  002776                  CLR     @#FLAG                  ;NO INTERRUPT
15803  100270  004767  000420                  JSR     R7,SCDSUB               ;DO TEST
15804  100274  046400  000000  000000          .WORD   46400,0,0,0             ;AC0
15805  100302  000000
15806  100304  001000  000000                  .WORD   1000,0          ;RESULT
15807  100310  007700                          .WORD   7700            ; TEST FPS
```

```
15808  100312  007700                          .WORD   7700            ;RESULT FPS
15809                                  ;7/LONG, EXP>2**32
15810  100314  012737  000001  002776          MOV     #1,@#FLAG               ;INTERRUPT
15811  100322  004767  000366                  JSR     R7,SCDSUB               ;DO TEST
15812  100326  077607  000000  000000          .WORD   77607,0,0,0             ;ACO
15813  100334  000000
15814  100336  000000  000000                  .WORD   0,0             ;RESULT
15815  100342  007700                          .WORD   7700            ; TEST FPS
15816  100344  107705                          .WORD   107705          ;RESULT FPS
15817                                  ;8/INT, EXP=2**15
15818  100346  005037  002776                  CLR     @#FLAG          ;NO INTERRUPTS
15819  100352  004767  000336                  JSR     R7,SCDSUB               ;DO TEST
15820  100356  043200  000000  000000          .WORD   43200,0,0,0             ;ACO
15821  100364  000000
15822  100366  010000  177777                  .WORD   10000,-1                ;RESULT
15823  100372  007600                          .WORD   7600            ; TEST FPS
15824  100374  007600                          .WORD   7600            ;RESULT FPS
15825                                  ;9/INT, EXP>2**15
15826  100376  012737  000001  002776          MOV     #1,@#FLAG               ;INTERRUPT
15827  100404  004767  000304                  JSR     R7,SCDSUB               ;DO TEST
15828  100410  077777  177777  177777          .WORD   77777,-1,-1,-1          ;ACO
15829  100416  177777
15830  100420  000000  177777                  .WORD   0,-1            ;RESULT
15831  100424  007600                          .WORD   7600            ; TEST FPS
15832  100426  107605                          .WORD   107605          ;RESULT FPS
15833                                  ;10/INT, EXP>2**15, FID
15834  100430  012737  000000  002776          MOV     #0,@#FLAG               ;NO INTERRUPT
15835  100436  004767  000252                  JSR     R7,SCDSUB               ;DO TEST
15836  100442  043300  000000  000000          .WORD   43300,0,0,0             ;ACO
15837  100450  000000
15838  100452  000000  014000                  .WORD   0,14000         ;RESULT
15839  100456  047700                          .WORD   47700           ; TEST FPS
15840  100460  047700                          .WORD   47700           ;RESULT FPS
15841                                  ;11/INT, EXP>2**15, FIC=0
15842  100462  012737  000000  002776          MOV     #0,@#FLAG               ;NO INTERRUPT
15843  100470  004767  000220                  JSR     R7,SCDSUB               ;DO TEST
15844  100474  143300  177777  177777          .WORD   143300,-1,-1,-1         ;ACO
15845  100502  177777
15846  100504  177777  163741                  .WORD   -1,163741               ;RESULT
15847  100510  007300                          .WORD   7300            ; TEST FPS
15848  100512  007310                          .WORD   7310            ;RESULT FPS
15849                                  ;12/LONG, EXP>2**32, FID
15850  100514  012737  000002  002776          MOV     #2,@#FLAG               ;INTERRUPT
15851  100522  004767  000166                  JSR     R7,SCDSUB               ;DO TEST
15852  100526  050100  000000  000000          .WORD   50100,0,0,0             ;ACO
15853  100534  000000
15854  100536  000000  000000                  .WORD   0,0             ;RESULT
15855  100542  047700                          .WORD   47700           ; TEST FPS
15856  100544  147705                          .WORD   147705          ;RESULT FPS
15857                                  ;13/LONG, EXP>2**32, FIC=0
15858  100546  012737  000000  002776          MOV     #0,@#FLAG               ;NO INTERRUPT
15859  100554  004767  000134                  JSR     R7,SCDSUB               ;DO TEST
15860  100560  050377  177777  177777          .WORD   50377,-1,-1,-1          ;ACO
15861  100566  177777
15862  100570  000000  000000                  .WORD   0,0             ;RESULT
15863  100574  007300                          .WORD   7300            ; TEST FPS
```

```
15864  100576  007305                          .WORD   7305            ;RESULT FPS
15865                                  ;14/LONG, EXP<0
15866  100600  005037  002776                  CLR     @#FLAG          ;NO INTERRUPTS
15867  100604  004767  000104                  JSR     R7,SCDSUB               ;DO TEST
15868  100610  100200  177777  177777          .WORD   100200,-1,-1,-1         ;AC0
15869  100616  177777
15870  100620  000000  000000                  .WORD   0,0             ;RESULT
15871  100624  007757                          .WORD   7757            ; TEST FPS
15872  100626  007744                          .WORD   7744            ;RESULT FPS
15873                                  ;15/INT, EXP<0
15874  100630  005037  002776                  CLR     @#FLAG          ;NO INTERRUPTS
15875  100634  004767  000054                  JSR     R7,SCDSUB               ;DO TEST
15876  100640  037700  177777  177777          .WORD   37700,-1, 1,-2          ;AC0
15877  100646  177776
15878  100650  000000  177777                  .WORD   0,-1            ;RESULT
15879  100654  007600                          .WORD   7600            ; TEST FPS
15880  100656  007604                          .WORD   7604            ;RESULT FPS
15881                                  ;16/INT, EXP=10
15882  100660  005037  002776                  CLR     @#FLAG          ;NO INTERRUPTS
15883  100664  004767  000024                  JSR     R7,SCDSUB               ;DO TEST
15884  100670  004377  177777  177777          .WORD   4377,-1,-1,-1           ;AC0
15885  100676  177777
15886  100700  000000  177777                  .WORD   0,-1            ;RESULT
15887  100704  007600                          .WORD   7600            ; TEST FPS
15888  100706  007604                          .WORD   7604            ;RESULT FPS
15889                                  ;
15890
15891  100710  000167  000214                  JMP     HOP21           ;GET OVER SUBROUTINE
15892                                  ;*X*X*X
```

```
15920  100764  000000                          .WORD   0                       ;STORE STATUS HERE.
15921                                  ;
15922                                  ;
15923                                  ;INSTRUCTION DIDNT TRAP
15924  100766  032737  000001  002776          BIT     #1,@#FLAG                       ;VERIFY A NO TRAP CONDITION
15925  100774  001420                          BEQ     2$              ;BRANCH IF GOOD
15926  100776  104003                          ERROR   .3              ;FPP ERROR
15927                                                                  ;INSTRUCTION SHOULD HAVE TRAPPED
15928  101000  000167  000032                  JMP     2$                      ;REJOIN CODE
15929                                  ;
15930                                  ;INSTRUCTION TRAPPED
15931  101004  032737  000001  002776  50$:    BIT     #1,@#FLAG               ;SEE IF EXPECTING A TRAP
15932  101012  001003                          BNE     51$             ;BRANCH IF EXPECTING A TRAP
15933  101014  104003                          ERROR   .3              ;FPP ERROR
15934                                                                  ;INSTRUCTION WASNT SUPPOSE TO TRAP
15935  101016  000167  000014                  JMP     2$                      ;REJOIN CODE
15936  101022  012604                  51$:    MOV     (SP)+,R4                ;SEE IF PC = INSTRUCTION
15937  101024  005726                          TST     (SP)+                   ;CLEAN UP STACK
15938  101026  022704  100762                  CMP     #1$,R4                  ;
15939  101032  001401                          BEQ     2$                      ;BRANCH IF GOOD COMPARE
15940  101034  104003                          ERROR   .3              ;FPP ERROR
15941                                                                  ;PC WAS INCORRECT
15942                                  ;
15943                                  ;COMMON CODE FOR TRAP AND NO TRAP
15944                                  ;VERIFY STATUS
15945  101036  170203                  2$:     STFPS   R3                      ;SAVE FPS
15946  101040  016200  000016                  MOV     16(R2),R0                       ;GET EXPECTED STATUS
15947  101044  020003                          CMP     R0,R3                   ;VERIFY STATUS
15948  101046  001401                          BEQ     3$                      ;BRANCH IF GOOD
15949  101050  104003                          ERROR   .3              ;FPP ERROR
15950                                                                  ;BAD FPS
15951  101052  010204                  3$:     MOV     R2,R4                           ;POINT TO EXPECTED DATA
15952  101054  062704  000010                  ADD     #10,R4
15953  101060  004767  032360          4$:     JSR     R7,DATVFR                               ;VERIFY DATA
15954  101064  005767  101770                  TST     COUNT
15955  101070  001401                          BEQ     5$                      ;BRANCH IF GOOD
15956  101072  104003                          ERROR   .3              ;FPP ERROR
15957                                                                  ;BAD ACO
15958  101074  005737  002776          5$:     TST     @#FLAG                  ;SEE IF NEED TO CHECK FEC
15959  101100  001002                          BNE     7$                      ;BRANCH IF NEED TO CHECK
15960  101102  000162  000020                  JMP     20(R2)                  ;RETURN FROM TEST
15961                                  ;VERIFY FEC
15962  101106  012704  003062          7$:     MOV     #RECFEC,R4              ;POINT TO FEC AREA
15963  101112  170314                          STST    (R4)                    ;SAVE FEC
15964  101114  021427  000006                  CMP     (R4),#6                 ;VERIFY FEC FOR OVERFLOW
15965  101120  001401                          BEQ     8$                      ;BRANCH IF GOOD
15966  101122  104003                          ERROR   .3              ;FPP ERROR
15967                                                                  ;BAD FEC
15968  101124  000162  000020          8$:     JMP     20(R2)          ;RETURN FROM TEST
15969                                  ;
15970  101130                          MOP21:
15971                                  ;
15972                                  ;
15973                                  ;
15974                                  ;-------------------------------
15975                                  ;TEST STCFI, STCFL
```

```
                                        ;
                                        ;
                                        ;
                                        ;
101130                                  MSCF:
                                        ;
                                        ;1/LONG EXP =30
101130  005037  002776                  CLR     @#FLAG          ;NO INTERRUPTS
101134  004767  177554                  JSR     R7,SCDSUB               ;DO TEST
101140  044541  052525  177777          .WORD   44541,52525,-1,-1               ;ACO
101146  177777
101150  000003  102525                  .WORD   3,102525                ;RESULT
101154  007517                          .WORD   7517            ; TEST FPS
101156  007500                          .WORD   7500            ;RESULT FPS
                                        ;2/INT, EXP<0
101160  005037  002776                  CLR     @#FLAG          ;NO INTERRUPTS
101164  004767  177524                  JSR     R7,SCDSUB               ;DO TEST
101170  002300  177777  177777          .WORD   2300,-1,-1,-1           ;ACO
101176  177777
101200  000000  177777                  .WORD   0,-1            ;RESULT
101204  007400                          .WORD   7400            ; TEST FPS
101206  007404                          .WORD   7404            ;RESULT FPS
                                        ;3/LONG, EXP
101210  012737  000001  002776          MOV     #1,@#FLAG               ;INTERRUPT
101216  004767  177472                  JSR     R7,SCDSUB               ;DO TEST
101222  070000  177777  177777          .WORD   70000,-1,-1,-1          ;ACO
101230  177777
101232  000000  000000                  .WORD   0,0             ;RESULT
101236  007540                          .WORD   7540            ; TEST FPS
101240  107545                          .WORD   107545          ;RESULT FPS
                                        ;4/INT,EXP=5, FIC=0, FID=1
101242  005037  002776                  CLR     @#FLAG          ;NO INTERRUPTS
101246  004767  177442                  JSR     R7,SCDSUB               ;DO TEST
101252  052000  000000  177777          .WORD   52000,0,-1,-1           ;ACO
101260  177777
101262  000000  177777                  .WORD   0,-1            ;RESULT
101266  047000                          .WORD   47000           ; TEST FPS
101270  047005                          .WORD   47005           ;RESULT FPS
                                        ;
                                        ;
                                        ;
                                        ;
                                        ;
                                        ;----------------------------------
                                        ;TEST STEXP
                                        ;
                                        ;
                                        ;
                                        ;
101272                                  MSXP:
                                        ;
                                        ;1/EXP=100
101272  004767  000154                  JSR     R7,SXPSUB               ;DO TEST
101276  020000  000000  000000          .WORD   20000,0,0,0             ;ACO
```

```
101304  000000
101306  177700                          .WORD   -100            ;RESULT
101310  007740                          .WORD   7740            ; TEST FPS
101312  007750                          .WORD   7750            ;RESULT FPS
                                ;2/EXP=201 FLOAT, NEG
101314  004767  000132                  JSR     R7,SXPSUB               ;DO TEST
101320  140377  177777  177777          .WORD   140377,-1,-1,0          ;AC0
101326  000000
101330  000001                          .WORD   1               ;RESULT
101332  007500                          .WORD   7500            ; TEST FPS
101334  007500                          .WORD   7500            ;RESULT FPS
                                ;3/EXP=-177
101336  004767  000110                  JSR     R7,SXPSUB               ;DO TEST
101342  000177  177777  177777          .WORD   177,-1,-1,-1            ;AC0
101350  177777
101352  177600                          .WORD   177600          ;RESULT
101354  007700                          .WORD   7700            ; TEST FPS
101356  007710                          .WORD   7710            ;RESULT FPS
                                ;4/EXP=-100
101360  004767  000066                  JSR     R7,SXPSUB               ;DO TEST
101364  020000  000000  177777          .WORD   20000,0,-1,-1           ;AC0
101372  177777
101374  177700                          .WORD   -100            ;RESULT
101376  040200                          .WORD   40200           ; TEST FPS
101400  040210                          .WORD   40210           ;RESULT FPS
                                ;5/EXP=200
101402  004767  000044                  JSR     R7,SXPSUB               ;DO TEST
101406  040000  000000  000000          .WORD   40000,0,0,0             ;AC0
101414  000000
101416  000000                          .WORD   0               ;RESULT
101420  007700                          .WORD   7700            ; TEST FPS
101422  007704                          .WORD   7704    ;RESULT FPS
                                ;6/EXP=0
101424  004767  000022                  JSR     R7,SXPSUB               ;DO TEST
101430  000177  177777  177777          .WORD   177,-1,-1,-1            ;AC0
101436  177777
101440  177600                          .WORD   177600          ;RESULT
101442  000000                          .WORD   0               ; TEST FPS
101444  000010                          .WORD   10              ;RESULT FPS
                                ;


101446  000167  000104                  JMP     HOP22           ;GET OVER SUBROUTINE
                                ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
                                ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
                                ;STEXP
                                ;               AC0
                                ;               EXPONENT RESULT
                                ;               FPS BEFORE EXECUTION
                                ;               FPS AFTER EXECUTION
                                ;
                                ;NO TRAPS CAN OCCUR
                                ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
                                ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
                                ;
                                SXPSUB: MOV     (SP)+,R2                ; RETURN ADDRESS TO USE AS POINTER
101452  012602
101454  012737  101544  000244          MOV     #50$,@#FPVEC            ;REDIRECT TRAP VECTOR
```

```
16088   101462  012701  003102                  MOV     #RECDST,R1              ;POINT TO RESULT AREA
16089   101466  012700  000200                  MOV     #200,R0                 ;SET FPS TO DOUBLE
16090   101472  170100                          LDFPS   R0                      ;
16091   101474  010204                          MOV     R2,R4                   ;POINT TO ACO DATA
16092   101476  172414                          LDD     (R4),AC0                        ;LOAD ACO
16093   101500  016200  000012                  MOV     12(R2),R0               ;GET TEST FPS
16094   101504  170100                          LDFPS   R0                      ;LOAD TEST FPS
16095                                   ;
16096   101506  175011                  40$:    STEXP   AC0,(R1)                ;*TEST INSTRUCTION(ACCORDING TO MODE)
16097                                   ;
16098                                   ;VERIFY STATUS
16099   101510  170203                  2$:     STFPS   R3                      ;SAVE FPS
16100   101512  016200  000014                  MOV     14(R2),R0                       ;GET EXPECTED STATUS
16101   101516  020003                          CMP     R0,R3                   ;VERIFY STATUS
16102   101520  001401                          BEQ     3$                      ;BRANCH IF GOOD
16103   101522  104003                          ERROR   .3                      ;FPP ERROR
16104                                                                           ;BAD FPS
16105   101524  016204  000010          3$:     MOV     10(R2),R4               ;POINT TO EXPECTED EXPONENT
16106   101530  020437  003102                  CMP     R4,@#RECDST             ;VERIFY EXPONENT
16107   101534  001401                          BEQ     5$                      ;BRANCH IF GOOD
16108   101536  104003                          ERROR   .3                      ;FPP ERROR
16109                                                                           ;BAD ACO
16110   101540  000162  000016          5$:     JMP     16(R2)                  ;RETURN FROM TEST
16111                                   ;
16112                                   ;INSTRUCTION TRAPPED
16113   101544  012600                  50$:    MOV     (SP)+,R0                ;SAVE PC
16114   101546  012605                          MOV     (SP)+,R5                ;SAVE OLD PS
16115   101550  104003                          ERROR   .3                      ;FPP ERROR
16116                                                                           ;WILD TRAP DURING STEXP
16117   101552  000167  177762                  JMP     5$                      ;REJOIN CODE
16118                                   ;
16119
16120   101556                          HOP22:
16121
16122
16123                                   .ENABL AMA
16124                                   .SBTTL  TEST  - CHECK REGISTER ACCESS
16125                                   ;CHECK REGISTER ACCESS - THIS TEST WILL VERIFY EACH OF THE THREE
16126                                   ;CACHE MEMORY SYSTEM REGISTERS CAN BE ACCESSED WITHOUT A TRAP TO
16127                                   ;LOCATION 4(NON-EXISTANT ADDRESS TRAP) OCCURRING. THE REGISTERS TO
16128                                   ;BE TESTED ARE:
16129                                   ;       CACHE CONTROL           17777746
16130                                   ;       MEMORY SYSTEM ERROR     17777744
16131                                   ;       HIT/MISS                17777752
16132                                   ;EACH REGISTER WILL BE ACCESSED WITH A WRITE CYCLE.
16133                                   ;
16134                                   ;BGNTST
16135                                   ;SAVE CONTENT OF LOCATION 4
16136                                   ;LET LOCATION 4 POINT TO ERROR ROUTINE
16137                                   ;INITIALIZE R TO TOP OF ADDRESS TABLE
16138                                   ;DO UNTIL FOR ALL REGISTERS
16139                                   ;.      TEST REGISTER UNDER TEST
16140                                   ;.      IF TIMEOUT DID HAPPEN
16141                                   ;.              ERROR
16142                                   ;.      ENDIF
16143                                   ;ENDDO
```

```
                                        ;RESTORE CONTENTS OF LOCATION 4
                                        ;EXIT   TST
                                        ;
                                        ;ADDRESS TABLE: 17777746
                                        ;               17777744
                                        ;               17777752
                                        ;
                                        ;ENDTST
                                        ;ERROR ROUTINE: SET TIMEOUT FLAG
                                        ;               RETURN

                                        ;;*****************************************************************
101556  000004                          TST2:   SCOPE
101560  012737  133124  000004                  MOV     #TOUT,  @#4
101566  013737  000004  002762                  MOV     @#4,    SLOC00          ;SAVE CONTENTS OF VECTOR 4
101574  012737  101700  000004                  MOV     #ERROUT,@#4             ;LET VECTOR 4 POINT TO ERROR ROUTINE
101602  005001                                  CLR     R1                      ;CLEAR TIMEOUT FLAG
101604  005737  177746                  1$:     TST     CCR                     ;READ REGISTER UNDER TEST
101610  005701                                  TST     R1                      ;TIMEOUT?
101612  001404                                  BEQ     2$
101614  012737  177746  001122                  MOV     #CCR,$BDADR             ;STORE LOCATION
101622  104130                                  ERROR   +130                    ;TIMEOUT ACCESING CCR
101624  005001                          2$:     CLR     R1                      ;CLEAR TIMOUT FLAG
101626  005737  177744                          TST     MSER                    ;READ MSER
101632  005701                                  TST     R1                      ;TIMEOUT ?
101634  001404                                  BEQ     3$
101636  012737  177744  001122                  MOV     #MSER,$BDADR            ;STORE LOCATION
101644  104130                                  ERROR   +130                    ;TIMEOUT ACCESSING MSER
101646  005001                          3$:     CLR     R1                      ;CLEAR TIMEOUT FLAG
101650  005737  177752                          TST     HITMIS                  ;ACCESS HIT/MISS
101654  005701                                  TST     R1                      ;TIMEOUT?
101656  001404                                  BEQ     4$
101660  012737  177752  001122                  MOV     #HITMIS,$BDADR          ;STORE LOCATION
101666  104130                                  ERROR   +130                    ;TIMEOUT ACCESSING HIT/MISS
101670  013737  002762  000004          4$:     MOV     SLOC00, @#4             ;RESTORE VECTOR 4
101676  000402                                  BR      TST3            ;;GO TO NEXT TEST

101700                                  ERROUT:
101700  005201                                  INC     R1                              ;FLAG TIMEOUT
101702  000002                                  RTI
```

```
                                        .SBTTL  TEST  - CCR REGISTER BIT TEST
                                        ;CCR REGISTER BIT TEST - THIS TEST WILL VERIFY THAT EACH READ/WRITE BIT OF
                                        ;THE CACHE CONTROL REGISTER CAN BE SET AND CLEARED INDIVIDUALLY AND THAT
                                        ;BITS 15 - 11 AND BIT 8 ARE ALWAYS READ AS ZEROS.
                                        ;
                                        ;BGNTST
                                        ;INITIALIZE GOOD DATA TO #1
                                        ;CLEAR CCR
                                        ;DO UNTIL ALL BITS TESTED
                                        ;.      WRITE GOOD DATA TO CCR
                                        ;.      READ CCR
                                        ;.      IF CCR NOT EQUAL TO GOOD DATA THEN
                                        ;.      .       IF CCR EQUAL TO ZERO THEN
                                        ;.      .       .       IF GOOD DATA NOT EQUAL TO BIT 15-11 OR BIT 8 THEN
                                        ;.      .       .       .       ERROR IN READ/WRITE BITS OF CCR
                                        ;.      .       .       ENDIF
                                        ;.      ENDIF
                                        ;.      UPDATE TO NEXT BIT
                                        ;ENDDO
                                        ;ENDTST

                                        ;;*****************************************************************
101704  000004                  TST3:   SCOPE
101706  012701  000001                  MOV     #1,     R1              ;INITIALIZE GOOD DATA TO #1
101712  005037  177746                  CLR     CCR                     ;CLEAR CCR
101716  010137  177746          1$:     MOV     R1,     CCR             ;WRITE GOOD DATA TO CCR
101722  023701  177746                  CMP     CCR,    R1              ;IF CCR NOT EQUAL GOOD DATA
101726  001407                          BEQ     3$                      ;THEN
101730  005737  177746                  TST     CCR                     ;IF CCR EQUAL TO ZERO
101734  001003                          BNE     2$                      ;THEN
101736  032701  174400                  BIT     #174400,R1              ;IF GOOD DATA NE TO BIT 15-11 OR BIT 8
101742  001001                          BNE     3$                      ;THEN
101744  104004                  2$:     ERROR   .4                      ;ERROR IN READ/WRITE BITS OF CCR
101746  006101                  3$:     ROL     R1                      ;UPDATE TO NEXT BIT
101750  103362                          BCC     1$                      ;DO UNTIL ALL BITS TESTED
```

J7


```
16223                                   .SBTTL  TEST  - FORCE MISS TEST
16224                                   ;FORCE MISS TEST - THIS TEST WILL VERIFY THAT ALL REFERENCES MADE
16225                                   ;WITH EITHER BIT<3> OR BIT<2> OF THE CCR SET CAUSE A CACHE MISS AND
16226                                   ;LEAVE THE CACHE ENTRY UNCHANGED. FIRST WRITE A TEST ADDRESS WITH
16227                                   ;BITS<3:2> CLEARED TO ALLOCATE CACHE AND SET A KNOWN DATA PATTERN
16228                                   ;INTO THE CACHE. THEN SET BIT<2> AND REWRITE THE SAME ADDRESS WITH
16229                                   ;NEW DATA. NEXT CLEAR BITS<3:2> AND READ THE TEST ADDRESS. THE DATA
16230                                   ;SHOULD EQUAL TO PATTERN 1. FINALLY READ THE TEST ADDRESS WITH BIT<3>
16231                                   ;SET. THE DATA SHOULD EQUAL PATTERN 2. A LAST WRITE MUST BE DONE WITH
16232                                   ;BOTH FORCE BITS CLEARED BECAUSE THE CACHE AND MAIN MEMORY HAVE
16233                                   ;DIFFERENT DATA.
16234                                   ;
16235                                   ;
16236                                   ;BGNTST
16237                                   ;WRITE TEST ADDRESS WITH PATTERN 1
16238                                   ;SET CCR BITS<3:2> = 0,1
16239                                   ;WRITE TEST ADDRESS WITH PATTERN 2
16240                                   ;CLEAR CCR BIT<3>
16241                                   ;READ TEST ADDRESS
16242                                   ;SAVE HIT/MISS REGISTER DATA
16243                                   ;COMPARE RECEIVED DATA TO PATTERN 1
16244                                   ;IF DATA NOT EQUAL THEN
16245                                   ;.      IF DATA EQUAL TO PATTERN 2 THEN
16246                                   ;.      .       IF HIT THEN
16247                                   ;.      .       .       ERROR FORCE MISS WRITES TO CACHE
16248                                   ;.      .       ELSE
16249                                   ;.      .       .       IF PARITY ERROR INDICATORS EQUAL 0 THEN
16250                                   ;.      .       .       .       ERROR FROCE MISS INVALIDATES CACHE
16251                                   ;.      .       .       ELSE
16252                                   ;.      .       .       .       IF ALL PARITY INDICATORS SET THEN
16253                                   ;.      .       .       .       .       ERROR PARITY ABORT AFTER FORCE MISS
16254                                   ;.      .       .       .       ENDIF
16255                                   ;.      .       .       .       IF TAG PARITY ERROR THEN
16256                                   ;.      .       .       .       .       ERROR  TAG PARITY ERROR AFTER FORCE MISS
16257                                   ;.      .       .       .       ELSE
16258                                   ;.      .       .       .       .       IF B0 AND  B1 ERROR THEN
16259                                   ;.      .       .       .       .       .       ERROR DATA PARITY ERROR AFTER FORCE MISS
16260                                   ;.      .       .       .       .       ENDIF
16261                                   ;.      .       .       .       .       IF B0 PARITY ERROR THEN
16262                                   ;.      .       .       .       .       .       ERROR LOW BYTE PARITY ERROR
16263                                   ;.      .       .       .       .       ENDIF
16264                                   ;.      .       .       .       .       IF B1 PARITY ERROR THEN
16265                                   ;.      .       .       .       .       .       ERROR HIGH BYTE PARITY ERROR
16266                                   ;.      .       .       .       .       ENDIF
16267                                   ;.      .       .       .       ENDIF
16268                                   ;.      .       .       ENDIF
16269                                   ;.      .       ENDIF
16270                                   ;.      ELSE
16271                                   ;.      .       ERROR IN DATA PATH
16272                                   ;.      ENDIF
16273                                   ;ENDIF
16274                                   ;SET CCR<3:2> = 1,0
16275                                   ;READ TEST ADDRESS
16276                                   ;SAVE HIT/MISS REGISTER DATA
16277                                   ;COMPARE RECEIVED DATA TO PATTERN 2
16278                                   ;IF DATA NOT EQUAL THEN
```

```
                                        ;.      IF RECIEVED DATA EQUAL TO PATTERN 1 THEN
                                        ;.      .       IF HIT THEN
                                        ;.      .       .       ERROR FORCE MISS READS FROM CACHE
                                        ;.      .       ELSE
                                        ;.      .       .       ERROR FORCE MISS READS FROM CACHE AND MISS
                                        ;.      .       ENDIF
                                        ;.      ELSE
                                        ;.      .       ERROR IN DATA PATH
                                        ;.      ENDIF
                                        ;ENDIF
                                        ;CLEAR CCR<3:2>
                                        ;WRITE TEST ADDRESS
                                        ;ENDTST
                                        ;
                                        ;;**************************************************************
101752  000004                  TST4:   SCOPE
101754  013737  000114  002762          MOV     @#114,  SLOC00          ;SAVE CONTENTS OF VECTOR 114
101762  012737  102306  000114          MOV     #FMPARR,@#114           ;SETUP PARITY VECTOR TO POINT TO HANDLER
101770  005003                          CLR     R3                      ;CLEAR MSER SAVE LOCATION
101772  005037  003122                  CLR     @#TSTLOC                ;WRITE TEST LOCATION WITH PATTERN 1
101776  012737  000004  177746          MOV     #BIT02, CCR             ;SET CCR BITS<3:2> = 0,1
102004  012737  177777  003122          MOV     #177777,@#TSTLOC        ;WRITE TEST LOCATION WITH PATTERN 2
102012  012737  000200  177746          MOV     #200,   CCR             ;CLEAR CCR BIT 3 AND SET PARITY ABORTS
102020  012737  177777  001124          MOV     #177777,$GDDAT          ;SAVE DATA IN MEMORY
102026  042737  001000  177520          BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
102034  013701  003122                  MOV     @#TSTLOC,R1             ;READ TEST ADDRESS
102040  013702  177752                  MOV     @#HITMIS,R2             ;SAVE HIT/MISS DATA
102044  052737  001000  177520          BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
102052  005701                          TST     R1                      ;COMPARE RECEIVED DATA TO PATTERN 1
102054  001451                          BEQ     1$                      ;IF DATAS NOT EQUAL THEN
102056  022701  177777                  CMP     #177777,R1              ;IF DATA EQUAL TO PATTERN 2
102062  001045                          BNE     106$                    ;THEN
102064  032702  000002                  BIT     #BIT01, R2              ;IF HIT
102070  001402                          BEQ     100$                    ;THEN
102072  104005                          ERROR   .5                      ;FORCE MISS WRITES TO CACHE
102074  000441                          BR      1$                      ;ELSE
102076  032703  100340          100$:   BIT     #100340,R3              ;IF PARITY INDICATORS EQUAL 0
102102  001002                          BNE     101$                    ;THEN
102104  104006                          ERROR   .6                      ;FORCE MISS WRITE INVALIDATES CACHE
102106  000434                          BR      1$                      ;ELSE
102110  022703  100340          101$:   CMP     #100340,R3              ;IF ALL PARITY INDICATORS SET
102114  001002                          BNE     102$                    ;THEN
102116  104007                          ERROR   .7                      ;PARITY ABORT AFTER FORCE MISS
102120  000427                          BR      1$                      ;ELSE
102122  032703  000040          102$:   BIT     #BIT05, R3              ;IF TAG PARITY ERROR
102126  001402                          BEQ     103$                    ;THEN
102130  104010                          ERROR   .10                     ;TAG PARITY ERROR AFTER FORCE MISS
102132  000422                          BR      1$                      ;ELSE
102134  010304                  103$:   MOV     R3,     R4              ;COPY MSER INTO REG 4
102136  042704  177477                  BIC     #177477,R4              ;MASK FOR B0 AND B1 DATA
102142  022704  000300                  CMP     #300,   R4              ;IF B0 AND B1 DATA
102146  001002                          BNE     104$                    ;THEN
102150  104011                          ERROR   .11                     ;DATA PARITY ERROR AFTER FORCE MISS
102152  000412                          BR      1$                      ;ELSE
102154  032703  000100          104$:   BIT     #100,   R3              ;IF B0 ERROR
102160  001401                          BEQ     105$                    ;THEN
```

```
16335  102162  104012                          ERROR   +12                     ;LOW BYTE PARITY ERROR AFTER FORCE MISS
16336  102164  032703  000200          105$:   BIT     #200,   R3              ;IF B1 ERROR
16337  102170  001403                          BEQ     1$                      ;THEN
16338  102172  104013                          ERROR   +13                     ;HIGH BYTE PARITY ERROR AFTER FORCE MISS
16339  102174  000401                          BR      1$                      ;ENDIF
16340  102176  104014                  106$:   ERROR   +14                     ;ERROR DATA PATH
16341  102200  005003                  1$:     CLR     R3                      ;CLEAR MSER SAVE REGISTER
16342  102202  052737  000010  177746          BIS     #BIT03, CCR             ;SET CCR BITS <3:2> = 1,0
16343  102210  042737  001000  177520          BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
16344  102216  013701  003122                  MOV     @#TSTLOC,R1             ;READ TEST LOCATION
16345  102222  013702  177752                  MOV     @#HITMIS,R2             ;SAVE HIT/MISS REGISTER DATA
16346  102226  052737  001000  177520          BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
16347  102234  020127  177777                  CMP     R1,     #177777         ;COMPARE RECEIVED DATA TO PATTERN 2
16348  102240  001412                          BEQ     2$                      ;IF DATAS NOT EQUAL THEN
16349  102242  005701                          TST     R1                      ;IF RECIEVED DATA EQUAL TO PATTERN 1
16350  102244  001007                          BNE     108$                    ;THEN
16351  102246  032702  000002                  BIT     #BIT01, R2              ;IF HIT
16352  102252  001402                          BEQ     107$                    ;THEN
16353  102254  104015                          ERROR   +15                     ;FORCE MISS READS FROM CACHE
16354  102256  000403                          BR      2$                      ;ELSE
16355  102260  104016                  107$:   ERROR   +16                     ;FORCE MISS READS FROM CACHE AND MISS
16356  102262  000401                          BR      2$                      ;ENDIF
16357  102264  104014                  108$:   ERROR   +14                     ;ERROR IN DATA PATH
16358  102266  013737  002762  000114  2$:     MOV     SLOC00, @#114           ;RESTORE VECTOR 114
16359  102274  005037  177746                  CLR     CCR                     ;CLEAR CCR BITS<3:2>
16360  102300  005037  003122                  CLR     @#TSTLOC                ;WRITE TEST ADDRESS
16361  102304  000405                          BR      TST5            ;;GO TO NEXT TEST
16362
16363
16364  102306  011637  001122          FMPARR: MOV     (SP),$BDADR             ;SAVE ADDRESS THAT CAUSED ABORT
16365  102312  013703  177744                  MOV     MSER,   R3              ;SAVE CONTENTS OF MEMORY SYSTEM ERROR
16366  102316  000002                          RTI                             ;REGISTER AND RETURN
16367
```

```
16368                                   .SBTTL  TEST  - HIT/MISS REGISTER TEST PART 1
16369                                   ;HIT/MISS REGISTER TEST PART 1 - THIS TEST WILL VERIFY THAT THE HIT/MISS
16370                                   ;REGISTER CORRECTLY LOGS HITS AND MISSES.FIRST WRITE A TEST ADDRESS WITH
16371                                   ;BITS<3:2> CLEARED TO ALLOCATE CACHE AND SET A KNOWN DATA PATTERN INTO
16372                                   ;THE CACHE. THEN SET BIT<2> AND REWRITE THE SAME ADDRESS WITH NEW DATA.
16373                                   ;NEXT CLEAR CCR BITS<3:2> AND READ THE TEST ADDRESS, THE HIT/MISS REGISTER
16374                                   ;SHOULD LOGGED A HIT. FINALLY READ THE TEST ADDRESS PLUS 20000(8) THE
16375                                   ;HIT/MISS REGISTER SHOULD HAVE LOGGED A MISS.
16376                                   ;
16377                                   ;
16378                                   ;BGNTST
16379                                   ;WRITE TEST ADDRESS WITH PATTERN 1
16380                                   ;SET CCR BITS<3:2> = 0,1
16381                                   ;WRITE TEST ADDRESS WITH PATTERN 2
16382                                   ;CLEAR CCR BIT<3>
16383                                   ;READ TEST ADDRESS
16384                                   ;IF HIT/MISS REGISTER BIT 1 NOT SET THEN
16385                                   ;.      ERROR IN RECORDING HITS IN HIT/MISS
16386                                   ;ENDIF
16387                                   ;READ TEST ADDRESS + 20000(8)
16388                                   ;IF HIT/MISS REGISTER BIT 1 SET THEN
16389                                   ;.      ERROR IN RECORDING HITS IN HIT/MISS
16390                                   ;ENDIF
16391                                   ;ENDTST
16392                                   ;
16393                                   ;;*****************************************************************
16394  102320  000004                  TST5:   SCOPE
16395  102322  013737  000114  002762          MOV     @#114,  SLOC00          ;SAVE CONTENTS OF VECTOR 114
16396  102330  012737  102460  000114          MOV     #HMPARR,@#114           ;SETUP PARITY VECTOR TO POINT TO HANDLER
16397  102336  005037  003122                  CLR     @#TSTLOC                ;WRITE TEST LOCATION WITH PATTERN 1
16398  102342  012737  000004  177746          MOV     #BIT02, CCR             ;SET CCR BITS<3:2> = 0,1
16399  102350  012737  177777  003122          MOV     #177777,@#TSTLOC        ;WRITE TEST LOCATION WITH PATTERN 2
16400  102356  012737  000200  177746          MOV     #200,   CCR             ;CLEAR CCR BIT 3 AND SET PARITY ABORTS
16401  102364  042737  001000  177520          BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
16402  102372  013701  003122                  MOV     @#TSTLOC,R1             ;READ TEST ADDRESS
16403  102376  013737  177752  110504          MOV     HITMIS,RECDAT           ;STORE REGISTER
16404  102404  032737  000004  110504          BIT     #BIT02,RECDAT           ;IF HIT/MISS REGISTER BIT 1 NOT SET
16405  102412  001001                          BNE     1$                      ;THEN
16406  102414  104045                          ERROR   +45                     ;ERROR IN RECORDING HITS IN HIT/MISS
16407  102416  013701  023122          1$:     MOV     @#TSTLOC+8192.,R1       ;READ TEST LOCATION + 20000(8)
16408  102422  013737  177752  110504          MOV     HITMIS,RECDAT           ;STORE REGISTER
16409  102430  032737  000004  110504          BIT     #BIT02,RECDAT           ;IF HIT/MISS REGISTER BIT 1 SET
16410  102436  001401                          BEQ     2$                      ;THEN
16411  102440  104045                          ERROR   +45                     ;ERROR IN RECORDING HITS IN HIT/MISS
16412  102442  013737  002762  000114  2$:     MOV     SLOC00, @#114           ;RESTORE VECTOR 114
16413  102450  052737  001000  177520          BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
16414  102456  000403                          BR      TST6            ;;GO TO NEXT TEST
16415
16416  102460  013703  177744          HMPARR: MOV     MSER,   R3              ;SAVE CONTENTS OF MEMORY SYSTEM ERROR
16417  102464  000002                          RTI                             ;REGISTER AND RETURN
16418
```

```
                                        .SBTTL  TEST  - HIT/MISS REGISTER TEST
                                        ;HIT/MISS REGISTER TEST - THIS TEST WILL VERIFY THAT THE HIT/MISS
                                        ;REGISTER CORRECTLY LOGS CACHE HITS AND MISSES. IT WILL ALSO VERIFY
                                        ;THAT EACH BIT OF THE REGISTER IS UNIQUE. THIS WILL BE DONE BY
                                        ;FLOATING A ZERO THROUGH A FIELD OF ONES. THE ROTATING WILL BE DONE
                                        ;BY EXECUTING A TST @#HM (WHERE HM IS THE ADDRESS OF THE HIT/MISS
                                        ;REGISTER) AT SUCCESSIVE POSITIONS IN A SET OF EIGHT NOPS. THE READ
                                        ;OF THE I/O PAGE ADDRESS OF THE HIT/MISS REGISTER SHOULD CAUSE A
                                        ;MISS TO BE RECORDED.
                                        ;
                                        ;BGNTST
                                        ;INITIALIZE LOOP INDICATOR
                                        ;INITIALIZE EXPECTED DATA
                                        ;DO UNTIL LOOP INDICATOR = SEVEN
                                        ;.      PUT BACKGROUND DATA INTO EXECUTION BUFFER
                                        ;.      PUT TST INSTRUCTION INTO TEST LOCATION
                                        ;.      JUMP TO EXECUTE BUFFER
                                        ;.      IF EXPECTED DATA NE RECEIVED DATA THEN
                                        ;.      .       ERROR IN RECORDING HITS IN HIT/MISS
                                        ;.      ENDIF
                                        ;.      INCREMENT LOOP INDICATOR
                                        ;.      UPDATE EXPECTED DATA
                                        ;ENDDO
                                        ;EXIT   TST
                                        ;
                                        ;BACKGROUND DATA:NOP
                                        ;               NOP
                                        ;               NOP
                                        ;               NOP
                                        ;               NOP
                                        ;               NOP
                                        ;               NOP
                                        ;               NOP
                                        ;               NOP
                                        ;               MOV     @#HM,R2
                                        ;               RTS     PC
                                        ;ENDTST
                                        ;;*****************************************************************
102466  000004                          TST6:   SCOPE
102470  005037  003114                          CLR     LOOPIN                  ;INITIALIZE LOOP INDICATOR
102474  013737  023144  001162                  MOV     @#TSTLOC+20022,$TMP1    ;SAVE FOR LATER
102502  012703  003124                          MOV     #TSTLOC+2,R3            ;GET ADDRESS OF EXECUTION BUFFER
102506  012704  102672                          MOV     #EXPTBL,R4              ;GET ADDRESS OF EXPECTED DATA TABLE
102512  012705  177752                          MOV     #HITMIS,R5              ;PUT ADDRESS OF HIT/MISS REGISTER IN GPR
102516  042737  001000  177520                  BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
102524  023727  003114  000007          1$:     CMP     LOOPIN, #7              ;DO UNTIL LOOP INDICATOR EQUALS 7
102532  001440                                  BEQ     ENDHRT                  ;THEN EXIT
102534  012702  000013                          MOV     #13,    R2
102540  012700  102644                          MOV     #BACDAT,R0              ;PUT BACKGROUND DATA INTO
102544  012701  003122                          MOV     #TSTLOC,R1              ;EXECUTION BUFFER
102550  012021                          2$:     MOV     (R0)+,  (R1)+
102552  077202                                  SOB     R2,     2$              ;LOOP FOR TEN WORDS
102554  023727  003114  000006                  CMP     LOOPIN, #6              ;IS THIS LAST LOOP
102562  001004                                  BNE     3$                      ;IF YES THEN
102564  013737  001162  023144                  MOV     $TMP1,@#TSTLOC+20022    ;INVALIDATE FETCH OF "RECDAT"
102572  000404                                  BR      4$                      ;ELSE
102574  012723  005737                  3$:     MOV     #5737,  (R3)+           ;MOVE "TST" INSTRUCTION TO BUFFER
```

```
COKDAB0 KDJ11-B CLUSTER MACY11 30(1046)  05-APR-84  16:45  PAGE 300
COKDAB.P11      05-APR-84 16:45          TEST - HIT/MISS REGISTER TEST                                    SEQ 0299

 16475  102600  012713  177752          MOV     #HITMIS,(R3)            ;MOVE HIT.MISS REG. ADD. TO BUFFER
 16476  102604  004737  003122  4$:     JSR     PC,     TSTLOC          ;GO EXECUTE TEST CODE
 16477  102610  021437  110504          CMP     (R4),   RECDAT          ;IF EXPECTED DATA NOT EQUAL TO
 16478  102614  001403                  BEQ     5$                      ;RECEIVED DATA THEN
 16479  102616  011437  001124          MOV     (R4),$GDDAT             ;SAVE EXPECTED PATTERN
 16480  102622  104017                  ERROR   .17                     ;ERROR IN RECORDING HITS IN HIT/MISS
 16481  102624  005724          5$:     TST     (R4)+                   ;INCREMENT POINTER TO EXPECTED PATTERN
 16482  102626  005237  003114          INC     LOOPIN                  ;INCREMENT LOOP COUNTER
 16483  102632  000734                  BR      1$
 16484  102634                  ENDHRT:
 16485  102634  052737  001000  177520  BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
 16486  102642  000422                  BR      TST7            ;;GO TO NEXT TEST
 16487
 16488  102644  000240          BACDAT: .WORD   NOP
 16489  102646  000240                  .WORD   NOP
 16490  102650  000240                  .WORD   NOP
 16491  102652  000240                  .WORD   NOP
 16492  102654  000240                  .WORD   NOP
 16493  102656  000240                  .WORD   NOP
 16494  102660  000240                  .WORD   NOP
 16495  102662  000240                  .WORD   NOP
 16496  102664  011537  110504          MOV     (R5),RECDAT             ;SAVE CONTENTS OF HIT/MISS REGISTER
 16497  102670  000207                  RTS     PC
 16498
 16499  102672  000077          EXPTBL: .WORD   77
 16500  102674  000037                  .WORD   37
 16501  102676  000057                  .WORD   57
 16502  102700  000067                  .WORD   67
 16503  102702  000073                  .WORD   73
 16504  102704  000075                  .WORD   75
 16505  102706  000076                  .WORD   76
 16506
```

```
                        .SBTTL  TEST  - BYTE ALLOCATION TEST
                        ;BYTE ALLOCATION TEST - THIS TEST WILL VERIFY THAT THE CACHE SYSTEM CORRECTLY
                        ;HANDLES BYTE ACCESSES. THE TEST WILL FIRST CHECK THAT A WRITE ACCESS WHICH
                        ;IS A MISS DOES NOT UPDATE THE CACHE. THIS WILL BE DONE BY FIRST ASSURRING
                        ;A MISS AT A TEST LOCATION. THEN A WRITE BYTE ACCESS WILL BE DONE. FINALLY
                        ;THE LOCATION WILL BE READ. THE WRITE BYTE SHOULD NOT HAVE ALLOCATED THE
                        ;ADDRESS. NEXT THE TEST WILL VERIFY THAT IF A WRITE BYTE ACCESS IS A HIT
                        ;THAT THE CACHE LOCATION IS UPDATED. THE TEST WILL FIRST WRITE A TEST LOCATION
                        ;WITH A PATTERN TO ALLOCATE THE ADDRESS. THEN A SECOND PATTERN WILL BE WRITTEN
                        ;TO THE HIGH BYTE OF THE TEST LOCATION. THE TEST LOCATION WILL THEN BE READ.
                        ;IF THE HIGH BYTE OF THE TEST LOCATION IS EQUAL TO THE SECOND PATTERN THEN THE
                        ;LOCATION WAS UPDATED. THE SECOND TEST WILL BE REPEATED TO TEST LOW BYTE
                        ;ACCESSES.
                        ;
                        ;BGNTST
                        ;SAVE VECTOR 114
                        ;LET VECTOR 114 POINT TO BYTE PARITY ROUTINE
                        ;SET PARITY ABORT BIT IN CACHE CONTROL REGISTER
                        ;READ TEST LOCATION + 4K
                        ;WRITE LOW BYTE TO TEST ADDRESS
                        ;READ LOW BYTE OF TEST ADDRESS
                        ;IF HIT/MISS REGISTER BIT 1 SET THEN
                        ;.      ERROR WRITE BYTE ALLOCATES THE CACHE
                        ;ENDIF
                        ;WRITE PATTERN 1 TO TEST LOCATION
                        ;WRITE BYTE PATTERN 2 TO TEST LOCATION+1
                        ;IF BIT1 NOTSETIN HIT/MISS REGISTER THEN
                        ;.      IF B0 PARITY ERROR THEN
                        ;.      .       ERROR LOW BYTE PARITY ERROR ON WRITE BYTE HIT
                        ;.      ELSE
                        ;.      .       IF B1 PARITY ERROR THEN
                        ;.      .       .       ERROR HIGH BYTE PARITY ERROR ON WRITE BYTE HIT
                        ;.      .       ELSE
                        ;.      .       .       WRITE BYTE HIT DOES NOT RECORD A HIT
                        ;.      .       ENDIF
                        ;.      ENDIF
                        ;ELSE
                        ;.      READ TEST LOCATION
                        ;.      IF RECIEVED DATA NOT EQUAL TO EXPECTED DATA THEN
                        ;.      .       IF BYTE DATA REVERSED THEN
                        ;.      .       .       ERROR BYTES REVERSED ON WRITE CYCLES
                        ;.      .       ELSE
                        ;.      .       .       IF HIGH BYTE DATA ZERO THEN
                        ;.      .       .       .       ERROR IN WRITING TO HIGH BYTE
                        ;.      .       .       ELSE
                        ;.      .       .       .       ERROR IN WRITING TO HIGH BYTE
                        ;.      .       .       ENDIF
                        ;.      .       ENDIF
                        ;.      ENDIF
                        ;ENDIF
                        ;WRITE PATTERN 1 TO TEST LOCATION
                        ;WRITE BYTE PATTERN 2 TO TEST LOCATION LOW BYTE
                        ;IF BIT1 NOTSETIN HIT/MISS REGISTER THEN
                        ;.      IF B0 PARITY ERROR THEN
                        ;.      .       ERROR LOW BYTE PARITY
                        ;.      ELSE
```

```
16563                                   ;.      .       IF B1 PARITY ERROR THEN
16564                                   ;.      .       .       ERROR HIGH BYTE PRITY
16565                                   ;.      .       ELSE
16566                                   ;.      .       .       WRITE BYTE HIT DOES NOT RECORD A HIT
16567                                   ;.      .       ENDIF
16568                                   ;.      ENDIF
16569                                   ;ELSE
16570                                   ;.      READ TEST LOCATION
16571                                   ;.      IF RECIEVED DATA NOT EQUAL TO EXPECTED DATA THEN
16572                                   ;.      .       IF BYTE DATA REVERSED THEN
16573                                   ;.      .       .       ERROR BYTES REVERSED
16574                                   ;.      .       ELSE
16575                                   ;.      .       .       IF LOW BYTE DATA ZERO THEN
16576                                   ;.      .       .       .       ERROR IN DATA PATH
16577                                   ;.      .       .       ELSE
16578                                   ;.      .       .       .       ERROR IN DATA PATH
16579                                   ;.      .       .       ENDIF
16580                                   ;.      .       ENDIF
16581                                   ;.      ENDIF
16582                                   ;ENDIF
16583                                   ;CLEAR CACHE CONTROL REGISTER
16584                                   ;RESTORE VECTOR 114
16585                                   ;EXIT   TST
16586                                   ;
16587                                   ;BYTE PARITY ROUTINE:   SAVE MSER
16588                                   ;                       RETURN
16589                                   ;ENDTST
16590
16591                                   ;;*****************************************************************
16592  102710  000004                  TST7:   SCOPE
16593  102712  013737  000114  002762          MOV     @#114,  SLOC00          ;SAVE VECTOR 114
16594  102720  012737  103134  000114          MOV     #BYPARR,@#114           ;LET VECTOR 114 POINT TO ABORT ROUTINE
16595  102726  005003                          CLR     R3                      ;CLEAR MSER SAVE REGISTER
16596  102730  012737  000200  177746          MOV     #BIT07, CCR             ;SET PARITY ABORT BIT IN CCR
16597  102736  005737  023122                  TST     TSTLOC+8192.            ;READ TEST LOCATION + 4K TO CAUSE MISS
16598  102742  042737  001000  177520          BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
16599  102750  105037  003122                  CLRB    TSTLOC                  ;WRITE LOW BYTE OF TEST LOCATION
16600  102754  105737  003122                  TSTB    TSTLOC                  ;READ LOW BYTE OF TEST LOCATION
16601  102760  013737  177752  110504          MOV     HITMIS,RECDAT           ;STORE REGISTER
16602  102766  032737  000004  110504          BIT     #BIT02,RECDAT           ;IF BIT1 OF HIT/MISS REGISTER SET
16603  102774  001401                          BEQ     1$                      ;THEN
16604  102776  104020                          ERROR   +20                     ;WRITE BYTE ALLOCATES CACHE
16605  103000  005037  003122          1$:     CLR     TSTLOC                  ;WRITE PATTERN 1 TO TEST LOCATION
16606  103004  152737  000377  003123          BISB    #377,   @#TSTLOC+1      ;WRITE PATTERN 2 TO HIGH BYTE
16607  103012  013737  177752  110504          MOV     HITMIS,RECDAT           ;STORE REGISTER
16608  103020  032737  000004  110504          BIT     #BIT02,RECDAT           ;IF BIT1 NOTSETIN HIT/MISS REGISTER
16609  103026  001002                          BNE     2$                      ;THEN
16610  103030  104021                  101$:   ERROR   +21                     ;WRITE BYTE HIT DOES NOT RECORD HIT
16611  103032  000405                          BR      3$                      ;ELSE
16612  103034  022737  177400  003122  2$:     CMP     #177400,@#TSTLOC        ;IF TEST LOCATION NOT EQUAL TO
16613  103042  001401                          BEQ     3$                      ;PATTERN 2 THEN
16614  103044  104022                          ERROR   +22                     ;BYTES REVERSED ON WRITE CYCLES
16615  103046  005037  003122          3$:     CLR     TSTLOC                  ;WRITE PATTERN 1 TO TEST LOCATION
16616  103052  152737  000377  003122          BISB    #377,   @#TSTLOC        ;WRITE PATTERN 2 TO LOW BYTE
16617  103060  013737  177752  110504          MOV     HITMIS,RECDAT           ;STORE REGISTER
16618  103066  032737  000004  110504          BIT     #BIT02,RECDAT           ;IF BIT1 NOTSETIN HIT/MISS REGISTER
```

```
16619  103074  001001                          BNE     4$                      ;THEN
16620  103076  104012                          ERROR   .12                     ;LOW BYTE PARITY ERROR ON WRITE BYTE HIT
16621  103100  022737  000377  003122  4$:     CMP     #377,   @#TSTLOC        ;IF TEST LOCATION NOT EQUAL TO
16622  103106  001401                          BEQ     5$                      ;PATTERN 2 THEN
16623  103110  104022                          ERROR   .22                     ;BYTES REVERSED ON WRITE CYCLES
16624  103112  005037  177746          5$:     CLR     CCR                     ;CLEAR CCR BEFORE EXIT
16625  103116  052737  001000  177520          BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
16626  103124  013737  002762  000114          MOV     SLOC00, @#114           ;RESTORE VECTOR 114
16627  103132  000405                          BR      TST10           ;;GO TO NEXT TEST
16628
16629
16630  103134  011637  001122          BYPARR: MOV     (SP),   $BDADR          ;SAVE ADDRESS THAT CAUSE ABORT
16631  103140  013703  177744                  MOV     MSER,   R3              ;SAVE CONTENTS OF MSER
16632  103144  000002                          RTI                             ;RETURN
16633
```

```
                                .SBTTL  TEST  - PDR BIT15 (BYPASS) TEST
                                ;PDR BIT15 (BYPASS) TEST - THIS TEST WILL VERIFY THAT WHEN BIT<15> IS SET
                                ;IN A PDR AND AN ACCESS IS MADE TO A LOCATION MAPPED BY THE SELECTED PDR
                                ;THAT WOULD NORMALLY CAUSE A CACHE ACCESS, THE CACHE IS BYPASSED (I.E. THE
                                ;CACHE LOCATION IS INVALIDATED AND A MAIN MEMORY IS READ. THIS WILL BE DONE
                                ;BY FIRST WRITING A TEST LOCATION WITH A KNOWN PATTERN TO ALLOCATE THE ADDRESS.
                                ;THEN THE LOCATION WILL BE WRITTEN AGAIN WITH THE MMU ENABLED AND BIT <15> SET
                                ;IN THE CONTROLING PDR WITH A NEW PATTERN. THE LOCATION WILL THEN BE READ
                                ;AND A MISS SHOULD BE LOGGED IN THE HIT/MISS REGISTER. THIS READ WILL ALSO
                                ;BRING VALID DATA INTO CACHE FROM MEMORY. THE MMU WILL AGAIN BE ENABLED
                                ;WITH BIT <15> SET AND A READ CYCLE DONE. THIS SHOULD INVALIDATE THE CACHE.
                                ;NEXT THE MMU WILL BE DISABLED AND THE LOCATION READ FOR A THIRD TIME. THIS
                                ;WILL BE DONE WITH BIT<15> STILL SET. A MISS SHOULD ALSO BE LOGGED. LASTLY
                                ;BIT<15> WILL BE CLEARED AND THE MMU ENABLED AND THE LOCATION AGAIN READ.
                                ;THIS TIME A CACHE HIT SHOULD BE LOGGED.
                                ;
                                ;
                                ;BGNTST
                                ;SETUP MMU REGISTERS
                                ;WRITE TEST LOCATION WITH PATTERN 1
                                ;SET BIT<15> IN PDR FOR TEST LOCATION
                                ;ENABLE MMU
                                ;WRITE TEST LOCATION WITH PATTERN 2
                                ;DISABLE MMU AND CLEAR BIT<15> IN PDR
                                ;READ TEST LOCATION
                                ;IF HIT/MISS REGISTER BIT<1> SET THEN
                                ;.       ERROR CONDITIONAL BYPASS DOESN'T INVALIDATE CACHE
                                ;ENDIF
                                ;SET BIT<15> IN PDR FOR TEST LOCATION
                                ;ENABLE MMU
                                ;READ TEST LOCATION (SHOULD INVALIDATE CACHE)
                                ;DISABLE MMU
                                ;READ TEST LOCATION
                                ;IF HIT/MISS REGISTER BIT<1> SET THEN
                                ;.       ERROR CONDITIONAL BYPASS DOESN'T INVALIDATE CACHE
                                ;ENDIF
                                ;CLEAR BIT<15> IN PDR
                                ;TURN ON MMU
                                ;READ TEST LOCATION
                                ;IF HIT/MISS REGISTER BIT<1> NOT SET THEN
                                ;.       ERROR IN RECORDING HITS IN HIT/MISS
                                ;ENDIF
                                ;TURN OFF MMU
                                ;ENDTST
                                ;
                                ;;*****************************************************************
103146  000004                  TST10:  SCOPE
103150  004737  132156                  JSR     PC,     INITMM          ;SETUP MEMORY MANAGEMEMT REGISTERS
103154  005037  003122                  CLR     @#TSTLOC                ;WRITE TEST LOCATION WITH PATTERN 1
103160  012737  177777  001124          MOV     #177777,$GDDAT          ;SAVE DATA IN MEMORY
103166  052737  100000  172300          BIS     #BIT15, KIPDR0          ;SET BIT15 IN PDR FOR TEST LOCATION
103174  005237  177572                  INC     SR0                     ;TURN ON MEMORY MANAGEMENT
103200  012737  177777  003122          MOV     #177777,@#TSTLOC        ;WRITE TEST LOCATION WITH PATTERN 2
103206  005037  177572                  CLR     SR0                     ;TURN OFF MEMORY MANAGEMENT
103212  042737  100000  172300          BIC     #BIT15, KIPDR0          ;CLEAR BIT15 IN PDR FOR TEST LOCATION
103220  042737  001000  177520          BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
```

```
16690  103226  013701  003122                  MOV     @#TSTLOC,R1     ;READ TEST LOCATION
16691  103232  013737  177752  110504          MOV     HITMIS,RECDAT   ;SAVE HIT/MISS REGISTER
16692  103240  032737  000004  110504          BIT     #BIT02,RECDAT   ;IF HIT/MISS REGISTER BIT 1 SET
16693  103246  001401                          BEQ     2$              ;THEN
16694  103250  104023                          ERROR   .23             ;CONDITIONAL BYPASS DOESN'T INVALIDATE CACHE
16695  103252  052737  100000  172300  2$:     BIS     #BIT15, KIPDR0  ;SET BIT15 IN PDR FOR TEST LOCATION
16696  103260  005237  177572                  INC     SR0             ;TURN ON MEMORY MANAGEMENT
16697  103264  005737  003122                  TST     TSTLOC          ;READ TEST LOCATION
16698  103270  042737  100000  172300          BIC     #BIT15, KIPDR0  ;CLEAR BIT 15 IN PDR
16699  103276  005037  177572                  CLR     SR0             ;TURN OFF MMU
16700  103302  013701  003122                  MOV     @#TSTLOC,R1     ;READ TEST LOCATION
16701  103306  013737  177752  110504          MOV     HITMIS,RECDAT   ;STORE HIT/MISS TO REGISTER 2
16702  103314  032737  000004  110504          BIT     #BIT02,RECDAT   ;IF HIT/MISS REGISTER BIT 1 SET
16703  103322  001401                          BEQ     3$              ;THEN
16704  103324  104023                          ERROR   .23             ;CONDITIONAL BYPASS DOESN'T INVALIDATE 0 CACHE
16705  103326  005237  177572          3$:     INC     SR0             ;TURN ON MEMORY MANAGEMENT
16706  103332  005737  003122                  TST     @#TSTLOC        ;READ TEST LOCATION ONE MORE TIME
16707  103336  013737  177752  110504          MOV     HITMIS,RECDAT   ;STORE HIT/MISS TO REGISTER 2
16708  103344  032737  000004  110504          BIT     #BIT02,RECDAT   ;IF HIT/MISS REGISTER BIT 1 NOT SET
16709  103352  001001                          BNE     4$              ;THEN
16710  103354  104045                          ERROR   .45             ;ERROR IN RECORDING HITS IN HIT/MISS
16711  103356  042737  000001  177572  4$:     BIC     #BIT00, SR0     ;TURN OFF MMU
16712  103364  052737  001000  177520          BIS     #1000,BCSR      ;ENABLE HALT ON BREAK
16713
```

```
                                        .SBTTL  TEST  - FLUSH CACHE TEST
                                        ;FLUSH CACHE TEST - THIS TEST WILL VERIFY THAT WHEN CCR BIT<8> IS
                                        ;SET, THE ENTIRE CACHE IS INVALIDATED. FIRST 8K BYTES OF ADDRESSES
                                        ;WILL BE READ TO ALLOCATE CACHE. THEN THE CACHE WILL BE FLUSHED.
                                        ;THE SAME SET OF ADDRESS WILL THEN BE READ AND THE HIT/MISS REGISTER
                                        ;CHECKED AFTER EACH READ FOR A MISS TO BE LOGGED.
                                        ;
                                        ;BGNTST
                                        ;GET FIRST ADDRESS OF 8KB BUFFER
                                        ;INITIALIZE LOOP COUNTER TO 4KWORD COUNT
                                        ;DO UNTIL LOOP COUNTER = 0
                                        ;.      READ @ADDRESS+
                                        ;ENDDO
                                        ;GET FIRST ADDRESS OF 8KB BUFFER
                                        ;INITIALIZE LOOP COUNTER TO 2KWORD COUNT
                                        ;INITIALIZE MISS COUNT TO 4KWORD COUNT
                                        ;FLUSH CACHE
                                        ;DO UNTIL LOOP COUNTER = 0
                                        ;.      READ @ADDRESS+
                                        ;.      ICREMENT ADDRESS TO READ EVERY 2WORDS
                                        ;.      IF HIT/MISS REGISTER BIT<1> SET THEN
                                        ;.      .       DECREMENT MISS COUNT
                                        ;.      ENDIF
                                        ;ENDDO
                                        ;GET FIRST ADDRESS OF 8KB BUFFER
                                        ;INITIALIZE LOOP COUNTER TO 4KWORD COUNT
                                        ;DO UNTIL LOOP COUNTER = 0
                                        ;.      READ @ADDRESS+
                                        ;ENDDO
                                        ;GET LAST ADDRESS OF 8KB BUFFER
                                        ;INITIALIZE LOOP COUNTER TO 2KWORD COUNT
                                        ;FLUSH CACHE
                                        ;DO UNTIL LOOP COUNTER = 0
                                        ;.      READ @ADDRESS-
                                        ;.      DECREMENT ADDRESS TO READ EVERY 2WORDS
                                        ;.      IF HIT/MISS REGISTER BIT<1> SET THEN
                                        ;.      .       DECREMENT MISS COUNT
                                        ;.      ENDIF
                                        ;ENDDO
                                        ;IF MISS COUNT NOT EQUAL TO 4096. THEN
                                        ;.      ERROR HITS RECORDED AFTER FLUSHING CACHE
                                        ;ENDIF
                                        ;ENDTST
                                        ;
                                        ;;*****************************************************************
103372  000004                          TST11:  SCOPE
103374  004737  132156                          JSR     PC,     INITMM          ;INITIALIZE MMU
103400  012737  001600  172354                  MOV     #1600,KIPAR6            ;LET PAR6 POINT TO LOW 28K
103406  012701  140000                          MOV     #140000,R1              ;GET ADDRESS OF 8K BYTE BUFFER
103412  005237  177572                          INC     SR0                     ;ENABLE MMU
103416  012702  010000                          MOV     #4096., R2              ;INIT LOOP COUNTER TO 4K WORD COUNT
103422  005721                          1$:     TST     (R1)+                   ;READ ADDRESS TO ALLOCATE CACHE
103424  077202                                  SOB     R2,     1$              ;DO UNTIL ALL LOCATIONS ALLOCATED
103426  012701  140000                          MOV     #140000,R1              ;GET ADDRESS OF 8K BYTE BUFFER
103432  012702  004000                          MOV     #2048., R2              ;INIT LOOP COUNTER TO 2K WORD COUNT
103436  012703  010000                          MOV     #4096., R3              ;INITIALIZE MISS COUNT TO 4K
```

```
16770  103442  012737  000400  177746          MOV     #400,   CCR             ;FLUSH CACHE
16771  103450  042737  001000  177520          BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
16772  103456  005721                  2$:     TST     (R1)+                   ;READ ADDRESS
16773  103460  013737  177752  110504          MOV     @#HITMIS,RECDAT         ;STORE REGISTER
16774  103466  032737  000004  110504          BIT     #BIT02,RECDAT           ;IF HIT/MISS REGISTER NOT EQUAL
16775  103474  001401                          BEQ     3$                      ;TO ZERO THEN
16776  103476  005303                          DEC     R3                      ;DECREMENT MISS COUNT
16777  103500  062701  000002          3$:     ADD     #2,R1                   ;DO IN 2 WORD INCREMENTS (PMI MEMORY)
16778  103504  077214                          SOB     R2,     2$              ;LOOP UNTIL ENTIRE CACHE CHECKED
16779  103506  052737  001000  177520          BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
16780  103514  012702  010000                  MOV     #4096.,R2               ;DO FOR 4K
16781  103520  012701  140000                  MOV     #140000,R1              ;STARTING WITH 0
16782  103524  005721                  4$:     TST     (R1)+                   ;ALLOCATE IN CACHE
16783  103526  077202                          SOB     R2,4$                   ;DO FOR 4K
16784  103530  012702  004000                  MOV     #2048.,R2               ;DO FOR THE 2ND HALF
16785  103534  012701  160000                  MOV     #160000,R1              ;START WITH LAST LOCATION
16786  103540  012737  000400  177746          MOV     #400,CCR                ;FLUSH CACHE
16787  103546  042737  001000  177520          BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
16788  103554  005741                  5$:     TST     -(R1)                   ;READ ADDRESS
16789  103556  013737  177752  110504          MOV     @#HITMIS,RECDAT         ;STORE REGISTER
16790  103564  032737  000004  110504          BIT     #BIT02,RECDAT           ;IF HIT/MISS REGISTER NOT EQUAL
16791  103572  001401                          BEQ     6$                      ;TO ZERO THEN
16792  103574  005303                          DEC     R3                      ;DECREMENT MISS COUNT
16793  103576  162701  000002          6$:     SUB     #2,R1                   ;DO IN 2 WORD DECREMENTS
16794  103602  077214                          SOB     R2,5$                   ;DO FOR 2K WORDS
16795  103604  052737  001000  177520          BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
16796  103612  005037  177572                  CLR     SR0                     ;TURN OFF MMU
16797  103616  022703  010000                  CMP     #4096., R3              ;IF MISS COUNT NOT EQUAL TO 4K WORDS
16798  103622  001401                          BEQ     8$                      ;THEN
16799  103624  104024                          ERROR   .24                     ;HITS RECORDED AFTER FLUSHING CACHE
16800  103626                          8$:
16801  103626  000400                          BR      TST12                   ;;GO TO NEXT TEST
16802
```

```
                                        .SBTTL  TEST  - UNCONDITIONAL BYPASS TEST
                                        ;UNCONDITIONAL BYPASS TEST - THIS TEST WILL VERIFY THAT WHEN CCR
                                        ;BIT<9> IS SET, A MEMORY REFERENCE IS FORCED TO MAIN MEMORY. THIS
                                        ;WILL ALSO VERIFY THAT READ AND WRITE HIT INVALIDATE THE CACHE
                                        ;LOCATIONS WHEN BYPASS IS SET.
                                        ;
                                        ;BGNTST
                                        ;READ TEST LOCATIONS TO SETUP POSSIBILITY OF HIT
                                        ;SET CCR BIT<9>
                                        ;READ FIRST TEST LOCATION
                                        ;IF HIT/MISS REGISTER BIT<1> NOT SET THEN
                                        ;.      ERROR IN RECORDING HITS IN HIT/MISS
                                        ;ENDIF
                                        ;WRITE SECOND TEST LOCATION
                                        ;IF HIT/MISS REGISTER BIT<1> NOT SET THEN
                                        ;.      ERROR IN RECORDING HITS IN HIT/MISS
                                        ;ENDIF
                                        ;CLEAR CCR BIT<9>
                                        ;READ FIRST LOCATION
                                        ;IF HIT/MISS REGISTER BIT<1> SET THEN
                                        ;.      ERROR BYPASS DOESN'T INVALIDATE CACHE
                                        ;ENDIF
                                        ;READ SECOND LOCATION
                                        ;IF HIT/MISS REGISTER BIT<1> SET THEN
                                        ;.      ERROR BYPASS DOESN'T INVALIDATE CACHE
                                        ;ENDIF
                                        ;ENDTST
                                        ;
                                        ;;*****************************************************************
103630  000004                          TST12:  SCOPE
103632  005737  003122                          TST     @#TSTLOC                ;ALLOCATE FIRST TEST LOCATION
103636  005737  003124                          TST     @#TSTLOC+2              ;ALLOCATE SECOND TEST LOCATION
103642  052737  001000  177746                  BIS     #BIT09, CCR             ;SET CCR BIT 9 (CACHE BYPASS)
103650  042737  001000  177520          1$:     BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
103656  005737  003122                          TST     @#TSTLOC                ;READ FIRST TEST LOCATION
103662  013737  177752  110504                  MOV     HITMIS,RECDAT           ;STORE HIT/MISS TO REGISTER 2
103670  032737  000004  110504                  BIT     #BIT02,RECDAT           ;IF HIT/MISS REG. BIT 1 NOT SET
103676  001001                                  BNE     2$                      ;THEN
103700  104045                                  ERROR   +45                     ;ERROR IN RECORDING HITS IN HIT/MISS WITH CCR<9>
103702  005037  003124                  2$:     CLR     @#TSTLOC+2              ;WRITE SECOND LOCATION
103706  013737  177752  110504                  MOV     HITMIS,RECDAT           ;STORE HIT/MISS TO REGISTER 2
103714  032737  000004  110504                  BIT     #BIT02,RECDAT           ;IF HIT/MISS REG. BIT 1 NOT SET
103722  001001                                  BNE     3$                      ;THEN
103724  104045                                  ERROR   +45                     ;ERROR IN RECORDING HITS IN HIT/MISS WITH CCCR<9
103726  005737  003122                  3$:     TST     @#TSTLOC                ;READ FIRST TEST LOCATION
103732  013737  177752  110504                  MOV     HITMIS,RECDAT           ;STORE HIT/MISS TO REGISTER 2
103740  032737  000004  110504                  BIT     #BIT02,RECDAT           ;IF HIT/MISS REG. BIT 1 SET
103746  001401                                  BEQ     4$                      ;THEN
103750  104025                                  ERROR   +25                     ;BYPASS DOESN'T INVALIDATE CACHE
103752  005037  003124                  4$:     CLR     @#TSTLOC+2              ;WRITE SECOND LOCATION
103756  013737  177752  110504                  MOV     HITMIS,RECDAT           ;STORE HIT/MISS TO REGISTER 2
103764  032737  000004  110504                  BIT     #BIT02,RECDAT           ;IF HIT/MISS REG. BIT 1 SET
103772  001401                                  BEQ     5$                      ;THEN
```

```
16856  103774  104025                          ERROR   #25
16857  103776  042737  001000  177746  5$:     BIC     #BIT09,CCR      ;BYPASS DOESN'T INVALIDATE CACHE
16858  104004  052737  001000  177520          BIS     #1000,BCSR      ;RESTORE CCR
16859                                                                   ;ENABLE HALT ON BREAK
```

```
16860                                   .SBTTL  TEST  - WRITE WRONG DATA PARITY TEST
16861                                   ;WRITE WRONG DATA PARITY TEST - THIS TEST WILL VERIFY THAT WHEN CCR
16862                                   ;BIT<6> = 1 AND CCR BITS<7,0> = 0,1, A READ MISS OCCURS AFTER A
16863                                   ;WRITE. THE WRITE WITH CCR BIT<6> = 1 TO A LOCATION WILL CAUSE A
16864                                   ;CACHE UPDATE AND WRONG PARITY TO BE WRITTEN SO WHEN THE LOCATION
16865                                   ;IS READ INSTEAD OF A HIT BEING RECORDED THE CACHE PARITY ERROR
16866                                   ;WILL CAUSE A CACHE MISS. THIS TEST WILL BE DONE A BYTE AT A TIME.
16867                                   ;
16868                                   ;BGNTST
16869                                   ;SAVE CONTENTS OF VECTOR 114
16870                                   ;LET VECTOR 114 POINT TO ABORT ROUTINE
16871                                   ;CLEAR MSER
16872                                   ;IF MSER NOT CLEAR THEN
16873                                   ;.      ERROR MSER DOESN'T CLEAR ON WRITE REFERENCE
16874                                   ;ENDIF
16875                                   ;WRITE TEST LOCATION
16876                                   ;SET BITS<6,0> IN CCR
16877                                   ;WRITE TEST LOCATION LOW BYTE
16878                                   ;CLEAR CCR BIT<6> (WRITE WRONG DATA PARITY)
16879                                   ;INITIALIZE ERROR INDICATORS
16880                                   ;READ TEST LOCATION LOW BYTE
16881                                   ;IF BIT<1> SET IN HIT/MISS REGISTER THEN
16882                                   ;.      PARITY ERROR DOESN'T CAUSE MISS
16883                                   ;ELSE
16884                                   ;.      IF MSER BITS <7:5> ZERO THEN
16885                                   ;.      .       PARITY ERROR DOESN'T SET MSER PROPERLY
16886                                   ;.      ELSE
16887                                   ;.      .       SET ERROR IN LOW BYTE INDICATOR
16888                                   ;.      ENDIF
16889                                   ;.      ENDIF
16890                                   ;ENDIF
16891                                   ;CLEAR MSER
16892                                   ;IF MSER NOT CLEAR THEN
16893                                   ;.      .       ERROR MSER DOESN'T CLEAR ON WRITE REFERENCE
16894                                   ;ENDIF
16895                                   ;SET CCR BIT<6>
16896                                   ;WRITE TEST LOCATION HIGH BYTE
16897                                   ;CLEAR CCR BIT<6>
16898                                   ;READ TEST LOCATION HIGH BYTE
16899                                   ;IF BIT<1> SET IN HIT/MISS REGISTER THEN
16900                                   ;.      IF MSER BITS<7:5> NOT SET THEN
16901                                   ;.      .       PARITY ERROR DOESN'T CAUSE A MISS
16902                                   ;.      ELSE
16903                                   ;.      .       SET ERROR IN HIGH BYTE INDICATOR
16904                                   ;.      ENDIF
16905                                   ;ELSE
16906                                   ;.      IF MSER BITS <7:5> ZERO THEN
16907                                   ;.      .       PARITY ERROR DOESN'T SET MSER
16908                                   ;.      ENDIF
16909                                   ;ENDIF
16910                                   ;RESTORE CONTENTS OF VECTOR 114
16911                                   ;IF ERROR INDICATORS SET THEN
16912                                   ;.      IF ERROR IN BOTH BYTES THEN
16913                                   ;.      .       PARITY ERROR IGNORED
16914                                   ;.      ELSE
16915                                   ;.      .       IF ERROR IN LOW BYTE THEN
```

```
16916                                   ;.      .       .       LOW BYTE PARITY ERROR IGNORED
16917                                   ;.      .       ELSE
16918                                   ;.      .       .       HIGH BYTE PARITY ERROR IGNORED
16919                                   ;.      .       ENDIF
16920                                   ;.      ENDIF
16921                                   ;ENDIF
16922                                   ;EXIT   TST
16923                                   ;
16924                                   ;DATA PARITY ABORT ROUTINE:
16925                                   ;                               ILLEGAL PARITY INTERRUPT
16926                                   ;                               RETURN
16927                                   ;
16928                                   ;ENDTST
16929                                   ;;*****************************************************************************
16930  104012  000004                   TST13:  SCOPE
16931  104014  013737  000114  002762           MOV     @#114,  SLOC00          ;SAVE CONTENTS OF VECTOR 114
16932  104022  012737  104300  000114           MOV     #DAPAB0,@#114           ;LET VECTOR POINT TO ABORT ROUTINE
16933  104030  005037  177744                   CLR     MSER                    ;CLEAR MSER
16934  104034  005737  177744                   TST     MSER                    ;IF MSER NOT CLEAR
16935  104040  001401                           BEQ     1$                      ;THEN
16936  104042  104026                           ERROR   .+26                    ;MSER DOES NOT CLEAR ON WRITE REFERENCE
16937  104044  005037  003122           1$:     CLR     @#TSTLOC                ;WRITE TEST LOCATION TO ALLOCATE CACHE
16938  104050  012737  000101  177746           MOV     #101,   CCR             ;SET BITS<6,0> IN CCR
16939  104056  112737  000377  003122           MOVB    #377,   TSTLOC          ;WRITE LOW BYTE WITH BAD PARITY
16940  104064  042737  000100  177746           BIC     #BIT06, CCR             ;CLEAR WRITE WRONG DATA PARITY BIT
16941  104072  005002                           CLR     R2                      ;CLEAR ERROR INDICATORS
16942  104074  042737  001000  177520           BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
16943  104102  105737  003122                   TSTB    @#TSTLOC                ;READ LOW BYTE OF TEST LOCATION
16944  104106  013703  177752                   MOV     HITMIS, R3              ;SAVE HIT/MISS
16945  104112  032703  000004                   BIT     #BIT02, R3              ;IF BIT 1 SET IN HIT/MISS REGISTER
16946  104116  001402                           BEQ     2$                      ;THEN
16947  104120  104027                           ERROR   .+27                    ;PARITY ERROR DON'T CAUSE A MISS
16948  104122  000405                           BR      3$                      ;ELSE
16949  104124  032737  000340  177744  2$:      BIT     #340,   MSER            ;IF MSER BIT<7:5> NOT ZERO
16950  104132  001001                           BNE     3$                      ;THEN
16951  104134  104030                           ERROR   .+30                    ;PARITY ERROR DON'T SET MSER WITH CCR<7>=0
16952  104136  005037  177744           3$:     CLR     MSER                    ;CLEAR MSER
16953  104142  005737  177744                   TST     MSER                    ;IF MSER NOT CLEAR
16954  104146  001401                           BEQ     4$                      ;THEN
16955  104150  104026                           ERROR   .+26                    ;MSER DOES NOT CLEAR ON WRITE REFERENCE
16956  104152  052737  000100  177746  4$:      BIS     #BIT06, CCR             ;SET CCR BIT 6 (WRITE WRONG PARITY)
16957  104160  112737  000377  003123           MOVB    #377,   @#TSTLOC+1      ;WRITE HIGH BYTE OF TEST LOCATION
16958  104166  042737  000100  177746           BIC     #BIT06, CCR             ;CLEAR WRITE WRONG PARITY BIT
16959  104174  105737  003123                   TSTB    @#TSTLOC+1              ;READ HIGH BYTE OF TEST LOCATION
16960  104200  013737  177752  110504           MOV     HITMIS,RECDAT           ;SAVE HIT/MISS
16961  104206  032737  000004  110504           BIT     #BIT02,RECDAT           ;IF BIT 1 SET IN HIT/MISS REGISTER
16962  104214  001402                           BEQ     5$                      ;THEN
16963  104216  104027                           ERROR   .+27                    ;PARITY ERROR DON'T CAUSE A MISS
16964  104220  000405                           BR      6$                      ;ELSE
16965  104222  032737  000340  177744  5$:      BIT     #340,   MSER            ;IF BITS <7:5> ZERO
16966  104230  001001                           BNE     6$                      ;THEN
16967  104232  104030                           ERROR   .+30                    ;PARITY ERROR DON'T SET MSER WITH CCR<7>=0
16968  104234  005037  177746           6$:     CLR     CCR                     ;CLEAR CCR BEFORE EXIT
16969  104240  013737  002762  000114           MOV     SLOC00, @#114           ;RESTORE CONTENTS OF VECTOR 114
16970  104246  032702  000003                   BIT     #3,     R2              ;IF ERROR INDICATORS SET
16971  104252  001401                           BEQ     7$                      ;THEN
```

```
16972  104254  104031                 ERROR   +31                     ;PARITY ERROR IGNORED
16973  104256  005037  177746         7$:     CLR     CCR             ;CLEAR CCR
16974  104262  013737  002762  000114         MOV     SLOC00,@#114    ;RESTORE PARITY TRAP
16975  104270  052737  001000  177520         BIS     #1000,@CSR      ;ENABLE HALT ON BREAK
16976  104276  000404                 BR      TST14           ;;GO TO NEXT TEST
16977
16978
16979  104300  011637  001122         DAPABO: MOV     (SP),   $BDADR  ;SAVE ADDRESS THAT CAUSED ABORT
16980  104304  104007                 ERROR   +7              ;ILLEGAL PARITY INTERRUPT
16981  104306  000002                 RTI
16982
```

```
                                .SBTTL  TEST  - WRITE WRONG TAG PARITY
                                ;WRITE WRONG TAG PARITY - THIS TEST WILL VERIFY THAT A READ MISS
                                ;OCCURS AFTER A WRITE WILL CCR<10> = 1 AND CCR<7,0> = 0,1. THE WRITE
                                ;TO THE LOCATION WILL CAUSE A CACHE UPDATE BUT THE TAG WILL BE
                                ;WRITTEN WITH THE WRONG PARITY. WHEN THE LOCATION IS READ INSTEAD
                                ;OF A CACHE HIT OCCURRING THE PARITY ERROR SHOULD CAUSE A CACHE
                                ;MISS.
                                ;
                                ;BGNTST
                                ;SAVE CONTENTS OF VECTOR 114
                                ;LET VECTOR 114 POINT TO ABORT ROUTINE
                                ;CLEAR MSER
                                ;SET WRITE WRONG TAG PARITY BIT<10>
                                ;WRITE TEST LOCATION
                                ;CLEAR CCR BIT<10> AND SET ABORT DISABLE BIT<0>
                                ;READ TEST LOCATION
                                ;IF BIT<1> SET IN HIT/MISS REGISTER THEN
                                ;.      IF MSER BITS <7:5> SET THEN
                                ;.              .       PARITY ERROR DOESN'T CAUSE MISS
                                ;.      ELSE
                                ;.              .       PARITY ERROR DOESN'T SET MSER WITH CCR<7>=0
                                ;.      ENDIF
                                ;ENDIF
                                ;SET WRITE WRONG TAG PARITY BIT<10>
                                ;WRITE TO A BYTE OF A TEST LOCATION
                                ;SAVE HIT/MISS REGISTER
                                ;CLEAR WRITE WRONG TAG PARITY BIT
                                ;IF BIT<1> NOT SET IN HIT/MISS REGISTER THEN
                                ;.      ERROR
                                ;ENDIF
                                ;RESTORE VECTOR 114
                                ;EXIT   TST
                                ;
                                ;TAG PARITY ABORT ROUTINE:
                                ;                               ILLEGAL PARITY INTERRUPT
                                ;                               RETURN
                                ;ENDTST
                                ;;*****************************************************************
 104310  000004                 TST14:  SCOPE
 104312  013737  000114  002762         MOV     @#114,  SLOC00          ;SAVE CONTENTS OF VECTOR 114
 104320  012737  104426  000114         MOV     #TAPAB0,@#114           ;LET VECTOR POINT TO ABORT ROUTINE
 104326  042737  001000  177520         BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
 104334  005037  177744                 CLR     MSER                    ;CLEAR MSER
 104340  012737  002000  177746         MOV     #BIT10, CCR             ;SET WRITE WRONG TAG PARITY
 104346  005037  003122                 CLR     @#TSTLOC                ;WRITE LOCATION WITH BAD TAG PARITY
 104352  012737  000001  177746         MOV     #BIT00, CCR             ;CLEAR BIT 10 AND SET BIT 0
 104360  005737  003122                 TST     @#TSTLOC                ;READ TEST LOCATION
 104364  013737  177752  110504         MOV     HITMIS,RECDAT           ;SAVE HIT/MISS REGISTER
 104372  032737  000004  110504         BIT     #BIT02,RECDAT           ;IF BIT 1 SET IN HIT/MISS REGISTER
 104400  001401                         BEQ     2$                      ;THEN
 104402  104027                         ERROR   .27                     ;PARITY ERROR DON'T CAUSE A MISS
 104404  013737  002762  000114  2$:    MOV     SLOC00, @#114           ;RESTORE CONTENTS OF VECTOR 114
 104412  005037  177744                 CLR     MSER                    ;CLEAR ERRORS
 104416  052737  001000  177520         BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
 104424  000404                         BR      TST15           ;;GO TO NEXT TEST
```

```
17039
17040   104426  011637  001122          TAPABO: MOV     (SP),$BDADR             ;SAVE ADDRESS THAT CAUSE ABORT
17041   104432  104007                          ERROR   .7                      ;ILLEGAL PARITY INTERRUPT
17042   104434  000002                          RTI
17043
```

```
                                .SBTTL  TEST  - PARITY ABORT TEST
                                ;PARITY ABORT TEST - THIS TEST WILL VERIFY THAT WHEN CCR<7,0> =
                                ;1,0, AN ABORT OCCURS ON THE EXECUTION OF AN INSTRUCTION THAT HAS
                                ;BEEN WRITTEN WITH WRONG PARITY.
                                ;
                                ;
                                ;BGNTST
                                ;SAVE VECTOR 114
                                ;SAVE VECTOR 4
                                ;LET VECTOR 114 POINT TO ABORT ROUTINE
                                ;LET VECTOR 4 POINT TO ERROR ROUTINE
                                ;CLEAR EXPECTING ABORT FLAG
                                ;SET CCR<10> WRITE WRONG TAG PARITY BIT
                                ;WRITE TEST ADDRESS
                                ;CLEAR CCR<10>
                                ;SET CCR TO @200 ENABLE PARITY ABORTS
                                ;SET EXPECTING ABORT FLAG
                                ;READ TEST ADDRESS
                                ;IF ABORT FLAG NE 0 THEN
                                ;.      ERROR IN PARITY ABORT LOGIC
                                ;ENDIF
                                ;RESTORE VECTOR 114
                                ;RESTORE VECTOR 4
                                ;EXIT   TST
                                ;
                                ;ABORT ROUTINE: IF EXPECTING ABORT FLAG NOT SET THEN
                                ;                       .       ERROR NO ABORT SHOULD HAVE OCCURRED
                                ;               ELSE
                                ;                       .       CLEAR (EXPECTING) ABORT FLAG
                                ;               ENDIF
                                ;               IF MSER NOT EQUAL TO 100040 THEN
                                ;                       .       PARITY ABORT LOGIC DOESN'T SET MSER PROPERLY
                                ;               ENDIF
                                ;               IF PC = UPDATED PC THEN
                                ;                       .       ILLEGAL PARITY ABORT
                                ;               ENDIF
                                ;               RETURN
                                ;
                                ;ENDTST
                                ;;*****************************************************************
104436  000004                  TST15:  SCOPE
104440  013737  000114  002762          MOV     @#114,  SLOC00          ;SAVE VECTOR 114
104446  013737  000004  002764          MOV     @#4,    SLOC01          ;SAVE 4
104454  012737  104556  000114          MOV     #ABORTR,@#114           ;LET VECTOR 114 POINT TO ABORT ROUTINE
                                ;
                                ; TO AVOID CONFUSION ALLOCATE IN CACHE FOLLOWING INSTRUCIONS
                                ;
104462  012704  104462          1$:     MOV     #.,R4                   ;START WITH CURRENT
104466  005724                  2$:     TST     (R4)+                   ;READ A WORD
104470  022704  104556                  CMP     #ABORTR,R4              ;GOT TO ABORT ROUTINE?
104474  001374                          BNE     1$                      ;IF NOT, KEEP ON ALLOCATING
104476  005000                          CLR     R0                      ;CLEAR EXPECTING ABORT FLAG
104500  012737  002000  177746          MOV     #BIT10, CCR             ;SET WRITE WRONG PARITY BIT
104506  005037  003122                  CLR     @#TSTLOC                ;WRITE TEST LOCATION WITH BAD PARITY
104512  012737  000200  177746          MOV     #BIT07, CCR             ;ENABLE ABORTS, CLEAR WWTP BIT
104520  005100                          COM     R0                      ;SET EXPECTING ABORT FLAG
```

```
17100  104522  005737  003122          ABORTI: TST     @#TSTLOC                ;READ TEST LOCATION (SHOULD CAUSE ABORT)
17101  104526  005700                          TST     R0                      ;IF ABORT FLAG NOT EQUAL ZERO
17102  104530  001401                          BEQ     1$                      ;THEN
17103  104532  104034                          ERROR   +34                     ;PARITY ABORT LOGIC DOESN'T WORK
17104  104534  013737  002762  000114  1$:     MOV     SLOC00, @#114           ;RESTORE VECTOR 114
17105  104542  013737  002764  000004          MOV     SLOC01, @#4             ;RESTORE VECTORE 4
17106  104550  005037  177744                  CLR     MSER
17107  104554  000426                          BR      TST16           ;;GO TO NEXT TEST
17108
17109
17110  104556  013703  177744          ABORTR: MOV     MSER,R3                 ;SAVE MSER
17111  104562  005700                          TST     R0                      ;IF EXPECTING ABORT FLAG NOT SET
17112  104564  001004                          BNE     1$                      ;THEN
17113  104566  011637  001122                  MOV     (SP),   $BDADR          ;SAVE ABORT ADDRESS
17114  104572  104007                          ERROR   +7                      ;ILLEGAL PARITY INTERRUPT
17115  104574  000401                          BR      2$                      ;ELSE
17116  104576  005000                  1$:     CLR     R0                      ;CLEAR (EXPECTING) ABORT FLAG
17117  104600  022737  100040  177744  2$:     CMP     #100040,MSER            ;IF MSER NOT EQUAL TO 100040
17118  104606  001404                          BEQ     3$                      ;THEN
17119  104610  012737  100040  001124          MOV     #100040,$GDDAT          ;SAVE PROPER MSER SETTING
17120  104616  104035                          ERROR   +35                     ;PARITY ABORT DON'T SET MSER PROPERLY
17121  104620  021627  104526          3$:     CMP     (SP),   #ABORTI+4       ;IF PC EQUAL TO UPDATE PC
17122  104624  001401                          BEQ     4$                      ;THEN
17123  104626  104007                          ERROR   +7                      ;ILLEGAL PARITY INTERRUPT
17124  104630  000002                  4$:     RTI                             ;RETURN
17125
```

```
17126                                   .SBTTL  TEST  - PARITY INTERRUPT TEST
17127                                   ;PARITY INTERRUPT TEST - THIS TEST WILL VERIFY THAT WHEN CCR<7,0> =
17128                                   ;0,0, A PARITY INTERRRUPT OCCURS AFTER EXECUTION OF AN INSTRUCTION
17129                                   ;THAT HAS BEEN WRITTEN WITH WRONG PARITY.
17130                                   ;
17131                                   ;BGNTST
17132                                   ;SAVE CONTENTS OF 114
17133                                   ;SETUP VECTOR 114 TO POINT TO INTERRUPT ROUTINE
17134                                   ;CLEAR EXPECTING INTERRUPT FLAG
17135                                   ;WRITE TEST ADDRESS WITH BAD PARITY
17136                                   ;SET EXPECTING INTERRUPT FLAG
17137                                   ;READ TEST ADDRESS
17138                                   ;IF INTERRUPT FLAG NE 0 THEN
17139                                   ;.      PARITY INTERRUPT LOGIC DOESN'T WORK
17140                                   ;ENDIF
17141                                   ;RESTORE CONTENTS OF VECTOR 114
17142                                   ;EXIT   TST
17143                                   ;
17144                                   ;INTERRUPT ROUTINE:     IF EXPECTING INTERRUPT FLAG NE 1 THEN
17145                                   ;                       .       ERROR NO INTERRUPT SHOULD HAVE OCCURRED
17146                                   ;                       ELSE
17147                                   ;                       .       CLEAR (EXPECTING) INTERRUPT FLAG
17148                                   ;                       ENDIF
17149                                   ;                       IF SAVED PC NE TO UPDATED PC THEN
17150                                   ;                       .       IF PC = TEST INSTRUCTION PC THEN
17151                                   ;                       .       .       ERROR INSTRUCTION ABORTED
17152                                   ;                       .       ELSE
17153                                   ;                       .       .       ILLEGAL PARITY ABORT
17154                                   ;                       .       ENDIF
17155                                   ;                       ENDIF
17156                                   ;                       IF MSER NE #340 THEN
17157                                   ;                       .       PARITY INTERRUPT DOESN'T SET MSER PROPERLY
17158                                   ;                       ENDIF
17159                                   ;                       RETURN
17160                                   ;
17161                                   ;ENDTST
17162                                   ;;*****************************************************************
17163   104632  000004                  TST16:  SCOPE
17164   104634  013737  000114  002762          MOV     @#114,  SLOC00          ;SAVE CONTENTS OF VECTOR 114
17165   104642  012737  104730  000114          MOV     #INTERR,@#114           ;LET VECTOR POINT TO INTERRUPT ROUTINE
17166                                   ;
17167                                   ; TO AVOID CONFUSION ALLOCATE IN CACHE FOLLOWING INSTRUCIONS
17168                                   ;
17169   104650  012704  104650          1$:     MOV     #.,R4                   ;START WITH CURRENT
17170   104654  005724                          TST     (R4)+                   ;READ A WORD
17171   104656  022704  104730                  CMP     #INTERR,R4              ;GOT TO INTERRUPT ROUTINE?
17172   104662  001374                          BNE     1$                      ;IF NOT, KEEP ON ALLOCATING
17173   104664  005001                          CLR     R1                      ;CLEAR EXPECTING INTERRUPT FLAG
17174   104666  052737  000100  177746          BIS     #BIT06, CCR             ;SET WRITE WRONG DATA PARITY
17175   104674  005037  003122                  CLR     @#TSTLOC                ;WRITE LOCATION WITH BAD DATA PARITY
17176   104700  005037  177746                  CLR     CCR                     ;CLEAR WRITE WRONG DATA PARITY
17177   104704  005101                          COM     R1                      ;SET EXPECTING INTERRUPT FLAG
17178   104706  005737  003122          INTRPC: TST     @#TSTLOC                ;READ TEST LOCATION
17179   104712  005701                          TST     R1                      ;IF INTERRUPT FLAG NOT EQUAL ZERO
17180   104714  001401                          BEQ     1$                      ;THEN
17181   104716  104036                          ERROR   .36                     ;PARITY INTERRUPT LOGIC DOESN'T WORK
```

```
17182  104720  013737  002762  000114  1$:     MOV     SLOC00, @#114           ;RESTORE VECTOR 114
17183  104726  000424                          BR      TST17           ;;GO TO NEXT TEST
17184
17185
17186  104730  005701                  INTERR: TST     R1                      ;IF EXPECTING INTERRUPT FLAG NOT SET
17187  104732  001004                          BNE     1$                      ;THEN
17188  104734  011637  001122                  MOV     (SP),   $BDADR          ;SAVE INTERRUPT ADDRESS
17189  104740  104007                          ERROR   ;7                      ;ILLEGAL PARITY INTERRUPT
17190  104742  000401                          BR      2$                      ;ELSE
17191  104744  005001                  1$:     CLR     R1                      ;CLEAR (EXPECTING) INTERRUPT FLAG
17192  104746  021627  104712          2$:     CMP     (SP),   #INTRPC+4       ;IF SAVED PC NOT EQUAL TO UPDATED PC
17193  104752  001401                          BEQ     4$                      ;THEN
17194  104754  104007                          ERROR   ;7                      ;ILLEGAL PARITY INTERRUPT
17195  104756  022737  000340  177744  4$:     CMP     #340,   MSER            ;IF MSER NOT EQUAL TO EXPECTED VALUE
17196  104764  001404                          BEQ     5$                      ;THEN
17197  104766  012737  000340  001124          MOV     #340,   $GDDAT          ;SAVE PROPER MSER
17198  104774  104035                          ERROR   ;35                     ;PARITY INTERRUPT DON'T SET MSER PROPERLY
17199  104776  000002                  5$:     RTI                             ;RETURN
17200
```

```
                                        .SBTTL  TEST    - MISCELLANEOUS PARITY TEST
                                        ;MISCELLANEOUS PARITY TEST - THIS TEST CHECKS THAT BYPASS CYCLES WITH
                                        ;PARITY ERRORS CAUSE CACHE HIT RESPONSE AND THAT FORCE MISS CYCLES
                                        ;IGNORE PARITY ERRORS.
                                        ;
                                        ;BGNTST
                                        ;WRITE A LOCATION WITH BAD PARITY
                                        ;SET BYPASS IN CCR
                                        ;READ THE LOCATION BACK
                                        ;IF NO HIT OR MSER NOT SET THEN
                                        ;.      ERROR
                                        ;ENDIF
                                        ;WRITE A LOCATION WITH BAD PARITY AND SET BYPASS
                                        ;IF MSER SET WHILE READING IT BACK
                                        ;.      ERROR
                                        ;ENDIF
                                        ;ENDTST
                                        ;;*****************************************************************
105000  000004                          TST17:  SCOPE
105002  012703  105002                          MOV     #.,R3                           ;START WITH CURRENT INSTRUCTION
105006  005723                          10$:    TST     (R3)+                           ;READ A WORD
105010  022703  105152                          CMP     #4$,R3                          ;LAST WORD?
105014  001374                                  BNE     10$                             ;IF NOT, CONTINUE
                                        ;
                                        ; CHECK BYPASS AND BAD PARITY
                                        ;
105016  013737  000114  002762                  MOV     @#114,  SLOC00                  ;SAVE PARITY VECTOR
105024  012737  105060  000114                  MOV     #1$,@#114                       ;POINT NEW VECTOR
105032  052737  002101  177746                  BIS     #BIT10!BIT06!BIT00,CCR          ;DATA AND TAG PAR., NO INT.
105040  005037  003122                          CLR     @#TSTLOC                        ;WRITE CYCLE
105044  012737  001000  177746                  MOV     #BIT09,CCR                      ;SET BYPASS
105052  005737  003122                          TST     @#TSTLOC                        ;BYPASS WITH WRONG PARITY
105056  000410                                  BR      2$                              ;
105060  062706  000004                  1$:     ADD     #4,SP                           ;ADJUST STACK
105064  104045                                  ERROR   +45                             ;ERROR
105066  032737  000340  177744                  BIT     #340,MSER                       ;MSER OK?
105074  001001                                  BNE     2$                              ;IF YES, BRANCH
105076  104042                                  ERROR   +42                             ;BYPASS WRONG
                                        ;
                                        ; CHECK FORCE MISS AND BAD PARITY
                                        ;
105100  005037  177744                  2$:     CLR     MSER                            ;CLEAR MSER
105104  005037  177746                          CLR     CCR                             ;CLEAR CCR
105110  012737  105144  000114                  MOV     #3$,@#114                       ;POINT NEW VECTOR
105116  052737  002101  177746                  BIS     #BIT10!BIT06!BIT00,CCR          ;DATA AND TAG PAR., NO INTER
105124  005037  003122                          CLR     @#TSTLOC                        ;FORCE MISS WITH PARITY
105130  012737  000014  177746                  MOV     #14,CCR                         ;FORCE MISS
105136  005737  003122                          TST     @#TSTLOC                        ;ALLOCATE CACHE
105142  000403                                  BR      4$
105144  104042                          3$:     ERROR   +42
105146  062706  000004                          ADD     #4,SP                           ;ADJUST STACK
105152  005037  177746                  4$:     CLR     CCR                             ;CLEAR CCR
105156  013737  002762  000114                  MOV     SLOC00,@#114                    ;RESTORE PARITY VECTOR
105164  012737  000400  177746                  MOV     #BIT08,CCR                      ;FLUSH CACHE
```

```
17256                                   .SBTTL  TEST  - MEMORY SYSTEM ERROR REGISTER TEST
17257                                   ;MEMORY SYSTEM ERROR REGISTER TEST - THIS TEST WILL VERIFY THE
17258                                   ;FUNCTIONALITY OF BITS <15> AND <7:5> OF THE MEMORY SYSTEM ERROR
17259                                   ;REGISTER. THIS TEST WILL USE THE WRITE WRONG PARITY BITS (BITS<10>
17260                                   ;AND <6> OF THE CCR) TO WRITE BAD PARITY INTO A LOCATION. THE
17261                                   ;LOCATION WILL THEN BE READ WITH CACHE TRAPS ENABLED AND THE MSER
17262                                   ;WILL BE CHECKED AFTER THE ABORT FOR THE CORRECT BIT(S) BEING SET.
17263                                   ;THIS WILL BE DONE FOR ALL COMBINATIONS OF BITS<10> AND <6> FOR WORD
17264                                   ;ACCESSES THEN REPEATED FOR ALL COMBINATIONS OF BITS<10> AND <6> FOR
17265                                   ;BYTE ACCESSES. THE TEST WILL THEN BE REPEATED A THIRD TIME FOR BYTE
17266                                   ;ACCESSES AND ABORTS DISABLED. THE MSER SHOULD CONTAIN BITS <7:5>
17267                                   ;SET TO 1'S AND BIT <15> A ZERO FOR ALL COMBINATIONS.
17268                                   ;
17269                                   ;BGNTST
17270                                   ;SAVE CONTENTS OF VECTOR 114
17271                                   ;SETUP VECTOR TO POINT TO ABORT ROUTINE
17272                                   ;INITIALIZE LOOP COUNTER
17273                                   ;DO UNTIL ALL WORD COMBINATIONS CHECKED
17274                                   ;.      INITIALIZE MSER
17275                                   ;.      SETUP CCR FROM CCR TABLE
17276                                   ;.      READ TEST LOCATION TO ALLOCATE LOCATION WITH DESIRED PARITY
17277                                   ;.      CLEAR <10,6> FROM CCR
17278                                   ;.      SETUP CCR FOR ABORT
17279                                   ;.      READ TEST LOCATION                      ;THIS COULD CAUSE TRAP
17280                                   ;.      IF EXPECTED DATA NE RECEIVED DATA THEN
17281                                   ;.      .       ERROR IN SETTING MSER
17282                                   ;.      ENDIF
17283                                   ;.      UPDATE LOOP COUNTER
17284                                   ;ENDDO
17285                                   ;INITIALIZE LOOP COUNTER
17286                                   ;DO UNTIL ALL BYTE COMBINATIONS CHECKED
17287                                   ;.      INITIALIZE MSER
17288                                   ;.      SETUP CCR FROM CCR TABLE
17289                                   ;.      READ TEST LOCATION TO ALLOCATE LOCATION WITH DESIRED PARITY
17290                                   ;.      CLEAR <10,6> FROM CCR
17291                                   ;.      SETUP CCR FOR ABORT
17292                                   ;.      READ LOW BYTE TEST LOCATION
17293                                   ;.      GET EXPECTED BYTE DATA FROM TABLE
17294                                   ;.      IF EXPECTED DATA NE RECEIVED DATA THEN
17295                                   ;.      .       ERROR IN SETTING MSER
17296                                   ;.      ENDIF
17297                                   ;.      INITIALIZE MSER
17298                                   ;.      READ HIGH BYTE TEST LOCATION
17299                                   ;.      GET EXPECTED BYTE DATA FROM TABLE
17300                                   ;.      IF EXPECTED DATA NE RECEIVED DATA THEN
17301                                   ;.      .       ERROR IN SETTING MSER
17302                                   ;.      ENDIF
17303                                   ;.      INCREMENT LOOP COUNTER
17304                                   ;ENDDO
17305                                   ;INITIALIZE LOOP COUNTER
17306                                   ;DO UNTIL ALL BYTE COMBINATIONS CHECKED
17307                                   ;.      INITIALIZE MSER
17308                                   ;.      SETUP CCR FROM CCR TABLE
17309                                   ;.      READ TEST LOCATION TO ALLOCATE LOCATION WITH DESIRED PARITY
17310                                   ;.      CLEAR <10,6> FROM CCR
17311                                   ;.      SETUP CCR FOR NO ABORTS
```

```
17312                                   ;.      READ LOW BYTE TEST LOCATION
17313                                   ;.      IF RECEIVED DATA NE TO #340 THEN
17314                                   ;.      .       ERROR IN SETTING MSER
17315                                   ;.      ENDIF
17316                                   ;.      INITIALIZE MSER
17317                                   ;.      READ HIGH BYTE TEST LOCATION
17318                                   ;.      IF RECEIVED DATA NE #340 THEN
17319                                   ;.      .       ERROR IN SETTING MSER
17320                                   ;.      ENDIF
17321                                   ;.      INCREMENT LOOP COUNTER
17322                                   ;ENDDO
17323                                   ;EXIT   TST
17324                                   ;
17325                                   ;CCR TABLE:     0
17326                                   ;               100
17327                                   ;               2000
17328                                   ;               2100
17329                                   ;EXPECTED WORD DATA:    0
17330                                   ;                       100300
17331                                   ;                       100040
17332                                   ;                       100340
17333                                   ;EXPECTED BYTE DATA:    0
17334                                   ;                       0
17335                                   ;                       100100
17336                                   ;                       100200
17337                                   ;                       100040
17338                                   ;                       100040
17339                                   ;                       100140
17340                                   ;                       100240
17341                                   ;ABORT ROUTINE: RTI
17342                                   ;
17343                                   ;ENDTST
17344                                   ;;*****************************************************************************
17345  105172  000004                   TST20:  SCOPE
17346  105174  013737  000114  002762           MOV     @#114,  SLOC00          ;SAVE CONTENTS OF VECTOR 114
17347  105202  012737  105652  000114           MOV     #ABROUT,@#114           ;SETUP VECTOR TO POINT TO ABORT ROUTINE
17348  105210  012704  000004                   MOV     #4,     R4              ;INITIALIZE LOOP COUNTER
17349  105214  012700  105612                   MOV     #CCRTBL,R0              ;GET ADDRESS OF CCR TABLE
17350  105220  012701  105622                   MOV     #EXPWDT,R1              ;GET ADDRESS OF EXPECTED DATA TABLE
17351  105224  005037  177744           1$:     CLR     MSER                    ;INITIALIZE MSER DATA
17352  105230  012037  177746                   MOV     (R0)+,  CCR             ;SETUP CCR FROM CCR TABLE
17353  105234  005037  003122                   CLR     @#TSTLOC                ;ALLOCATE CACHE LOC. WITH DESIRED PARITY
17354  105240  012737  000200  177746           MOV     #BIT07, CCR             ;SETUP CCR TO ABORT POSSIBLE BAD PARITY
17355  105246  005737  003122                   TST     @#TSTLOC                ;READ TEST LOCATION
17356  105252  021137  177744                   CMP     (R1),   MSER            ;IF RECEIVED DATA NOT EQUAL TO EXPECTED
17357  105256  001403                           BEQ     2$                      ;DATA THEN
17358  105260  011137  001124                   MOV     (R1),   $GDDAT          ;SAVE PROPER MSER SETTING
17359  105264  104035                           ERROR   +35                     ;MSER NOT SET PROPERLY
17360  105266  005721                   2$:     TST     (R1)+                   ;INCREMENT POINTER THRU MSER TABLE
17361  105270  005737  023122                   TST     @#TSTLOC+8192.          ;TO INSURE MISS ON THE NEXT LOOP
17362  105274  077425                           SOB     R4,     1$              ;LOOP UNTIL ALL COMBINATIONS CHECKED
17363                                   ;CHECK BYTE OPERATIONS NOW
17364  105276  012704  000004                   MOV     #4,     R4              ;INITIALIZE LOOP COUNTER
17365  105302  012700  105612                   MOV     #CCRTBL,R0              ;GET ADDRESS OF CCR TABLE
17366  105306  012701  105632                   MOV     #EXPBDT,R1              ;GET ADDRESS OF EXPECTED BYTE DATA TABLE
17367  105312  005037  177744           3$:     CLR     MSER                    ;INITIALIZE MSER
```

```
17368  105316  005737  003122            TST     @#TSTLOC                ;ALLOCATE TO HAVE WRITE BYTE HIT
17369  105322  011037  177746            MOV     (R0),   CCR             ;SETUP CCR FROM TABLE
17370  105326  105037  003122            CLRB    @#TSTLOC                ;ALLOCATE CACHE LOC. WITH DESIRED PARITY
17371  105332  012737  000200  177746    MOV     #BIT07, CCR             ;SETUP CCR TO ABORT POSSIBLE BAD PARITY
17372  105340  105737  003122            TSTB    @#TSTLOC                ;READ LOW BYTE OF TEST LOCATION
17373  105344  021137  177744            CMP     (R1),   MSER            ;IF RECEIVED DATA NOT EQUAL TO EXPECTED
17374  105350  001403                    BEQ     4$                      ;THEN
17375  105352  011137  001124            MOV     (R1),   $GDDAT          ;SAVE PROPER MSER SETTING
17376  105356  104035                    ERROR   +35                     ;MSER NOT SET PROPERLY
17377  105360  005721            4$:     TST     (R1)+                   ;INCREMENT POINTER THRU MSER TABLE
17378  105362  005037  177744            CLR     MSER                    ;INITIALIZE MSER
17379  105366  005737  003122            TST     @#TSTLOC                ;ALLOCATE TO HAVE WRITE BYTE HIT
17380  105372  012037  177746            MOV     (R0)+,  CCR             ;SETUP CCR FROM TABLE
17381  105376  105037  003123            CLRB    @#TSTLOC+1              ;ALLOCATE CACHE LOC. WITH DESIRED PARITY
17382  105402  012737  000200  177746    MOV     #BIT07, CCR             ;SETUP CCR TO ABORT POSSIBLE BAD PARITY
17383  105410  105737  003123            TSTB    @#TSTLOC+1              ;READ HIGH BYTE OF TEST LOCATION
17384  105414  021137  177744            CMP     (R1),   MSER            ;IF RECEIVED DATA NOT EQUAL TO EXPECTED
17385  105420  001403                    BEQ     5$                      ;THEN
17386  105422  011137  001124            MOV     (R1),   $GDDAT          ;SAVE PROPER MSER SETTING
17387  105426  104035                    ERROR   +35                     ;MSER NOT SET PROPERLY
17388  105430  005721            5$:     TST     (R1)+                   ;INCREMENT POINTER THRU MSER TABLE
17389  105432  077451                    SOB     R4,     3$              ;DO UNTIL ALL BYTE COMBINATIONS CHECKED
17390                          ;REPEAT WITHOUT ABORT
17391  105434  012704  000003            MOV     #3,     R4              ;INITIALIZE LOOP COUNTER
17392  105440  012700  105614            MOV     #CCRTBL+2,R0            ;GET ADDRESS OF CCR TABLE
17393  105444  005037  177744    6$:     CLR     MSER                    ;INITIALIZE MSER
17394  105450  005737  003122            TST     @#TSTLOC                ;ALLOCATE CACHE LOC.
17395  105454  011037  177746            MOV     (R0),   CCR             ;SETUP CCR FROM TABLE
17396  105460  105037  003122            CLRB    @#TSTLOC                ;ALLOCATE CACHE LOC. WITH DESIRED PARITY
17397  105464  012737  000001  177746    MOV     #BIT00, CCR             ;SETUP CCR TO NOT ABORT
17398  105472  105737  003122            TSTB    @#TSTLOC                ;READ LOW BYTE OF TEST LOCATION
17399  105476  022737  000340  177744    CMP     #340,   MSER            ;IF RECEIVED DATA NOT EQUAL TO EXPECTED
17400  105504  001404                    BEQ     7$                      ;THEN
17401  105506  012737  000340  001124    MOV     #340,   $GDDAT          ;SAVE PROPER MSER SETTING
17402  105514  104035                    ERROR   +35                     ;MSER NOT SET PROPERLY
17403  105516  005037  177744    7$:     CLR     MSER                    ;INITIALIZE MSER
17404  105522  005737  003122            TST     @#TSTLOC                ;ALLOCATE CACHE LOC.
17405  105526  012037  177746            MOV     (R0)+,  CCR             ;SETUP CCR FROM TABLE
17406  105532  105037  003123            CLRB    @#TSTLOC+1              ;ALLOCATE CACHE LOC. WITH DESIRED PARITY
17407  105536  012737  000001  177746    MOV     #BIT00, CCR             ;SETUP CCR TO NOT ABORT
17408  105544  105737  003123            TSTB    @#TSTLOC+1              ;READ HIGH BYTE OF TEST LOCATION
17409  105550  022737  000340  177744    CMP     #340,   MSER            ;IF RECEIVED DATA NOT EQUAL TO EXPECTED
17410  105556  001404                    BEQ     8$                      ;THEN
17411  105560  012737  000340  001124    MOV     #340,   $GDDAT          ;SAVE PROPER MSER SETTING
17412  105566  104035                    ERROR   +35                     ;MSER NOT SET PROPERLY
17413  105570  077453            8$:     SOB     R4,     6$              ;DO UNTIL ALL BYTE COMBINATIONS CHECKED
17414  105572  005037  177744            CLR     MSER                    ;CLEAR ERROR REGISTER
17415  105576  013737  002762  000114    MOV     SLOC00, @#114           ;RESTORE VECTOR 114
17416  105604  005037  177746            CLR     CCR                     ;CLEAR ALL BIT IN CCR
17417  105610  000421                    BR      TST21           ;;GO TO NEXT TEST
17418
17419
17420  105612  000000            CCRTBL: .WORD   0
17421  105614  000100                    .WORD   100
17422  105616  002000                    .WORD   2000
17423  105620  002100                    .WORD   2100
```

```
17424
17425  105622  000000                  EXPWDT: .WORD   0
17426  105624  100300                          .WORD   100300
17427  105626  100040                          .WORD   100040
17428  105630  100340                          .WORD   100340
17429
17430  105632  000000                  EXPBDT: .WORD   0
17431  105634  000000                          .WORD   0
17432  105636  100100                          .WORD   100100
17433  105640  100200                          .WORD   100200
17434  105642  100040                          .WORD   100040
17435  105644  100040                          .WORD   100040
17436  105646  100140                          .WORD   100140
17437  105650  100240                          .WORD   100240
17438
17439  105652  000002                  ABROUT: RTI
17440
17441
```

```
17442                                   .SBTTL  TEST  - CHECK PARITY ABORTS BLOCKED BY NON-EXISTENT MEMORY ABORT
17443                                   ;CHECK PARITY ABORTS BLOCKED BY NON-EXISTENT MEMORY ABORT - THIS TEST WILL
17444                                   ;VERIFY THAT IF A PARITY ERROR OCCURS ON THE SAME ADDRESS REFERENCE AS A
17445                                   ;NON-EXISTENT MEMORY ERROR THAT THE CACHE DATA PATH GATE ARRAY BLOCKS THE
17446                                   ;PARITY ERROR TO THE J-11 CHIP SET. THIS WILL BE DONE BY USING THE DIAGNOSTIC
17447                                   ;BIT TO CAUSE A PARITY ERROR IN A CACHE REFERENCE THAT DOES NOT HAVE A
17448                                   ;CORRESPONDING ADDRESS IN MAIN MEMORY. THE ADDRESS WILL THEN BE READ CAUSING
17449                                   ;BOTH A CACHE PARITY ERROR AND A NON-EXISTENT MEMORY ERROR. TO AVOID HAVING
17450                                   ;TO SIZE THE ENTIRE MEMORY TO FIND A NON-EXISTENT MEMORY ADDRESS THIS TEST
17451                                   ;WILL USE THE LARGEST NON-I/O ADDRESS (17757776). THE TEST WILL FIRST READ
17452                                   ;THIS ADDRESS AND IF A NXM TRAP OCCURS THE TEST WILL BE DONE. IF THE ACCESS
17453                                   ;TO THIS ADDRESS DOES NOT TRAP THEN THE TEST WILL BE SKIPPED.
17454                                   ;
17455                                   ;BGNTST
17456                                   ;SAVE CONTENTS OF VECTOR 4
17457                                   ;SAVE CONTENTS OF VECTOR 114
17458                                   ;LET VECTOR 4 POINT TO CONTINUE TESTING (A:)
17459                                   ;LET PAR6 = #177400
17460                                   ;ACCESS ADDRESS 157776 (PHYSICAL 17757776)
17461                                   ;IF NO TRAP THEN
17462                                   ;.      GOTO ENDTST
17463                                   ;ENDIF
17464                                   ;A:
17465                                   ;SET DIAGNOSTIC AND WRITE WRONG PARITY BITS IN CCR
17466                                   ;WRITE ADDRESS 157776
17467                                   ;CLEAR CCR
17468                                   ;SET PARITY ERROR ABORT BIT IN CCR
17469                                   ;LET VECTOR 4 POINT TO CONTINUE TESTING (B:)
17470                                   ;LET VECTOR 114 POINT TO NXM-PARITY ERROR ROUTINE
17471                                   ;READ ADDRESS 157776
17472                                   ;IF NO TRAP THEN
17473                                   ;.      ERROR IN ABORT LOGIC
17474                                   ;ENDIF
17475                                   ;B:
17476                                   ;CLEAR CCR
17477                                   ;RESTORE CONTENTS OF VECTOR 114
17478                                   ;RESTORE CONTENTS OF VECTOR 4
17479                                   ;EXIT   TST
17480                                   ;
17481                                   ;
17482                                   ;NXM-PARITY ERROR ROUTINE:      RESET STACK AFTER TRAP
17483                                   ;                                 PARITY ABORT NOT BLOCKED BY NXM
17484                                   ;                               GOTO B:
17485                                   ;ENDTST
17486                                   ;;*****************************************************************************
17487  105654  000004                  TST21:  SCOPE
17488  105656  013737  000004  002762          MOV     @#4,    SLOC00          ;SAVE CONTENTS OF VECTOR 4
17489  105664  013737  000114  002764          MOV     @#114,  SLOC01          ;SAVE CONTENTS OF VECTOR 114
17490  105672  012737  105720  000004          MOV     #1$,    @#4             ;LET VECTOR 4 POINT TO CONTINUE TESTING
17491  105700  012737  177400  172354          MOV     #177400,KIPAR6          ;LET PAR6 = OFFSET TO HIGHEST MEMORY
17492  105706  005237  177572                  INC     SR0                     ;TURN ON MMU
17493  105712  005737  157776                  TST     @#157776                ;ACCESS ADDRESS 17757776
17494  105716  000426                          BR      ABOEXT                  ;IF NO TRAP SKIP TEST
17495  105720  062706  000004          1$:     ADD     #4,     SP              ;RESET STACK AFTER TRAP
17496  105724  012737  000102  177746          MOV     #102,   CCR             ;SET DIAG. AND WRITE WRONG PARITY BITS
17497  105732  005037  157776                  CLR     @#157776                ;WRITE TO ADDRESS 17757776
```

```
COKDABO KDJ11-B CLUSTER MACY11 30(1046)  05-APR-84  16:45  PAGE 325
COKDAB.P11    05-APR-84 16:45           TEST  - CHECK PARITY ABORTS BLOCKED BY NON-EXISTENT MEMORY ABORT                    SEQ 0324

 17498  105736  012737  000200  177746          MOV     #200,   CCR             ;CLEAR CCR AND SET PARITY ABORT BIT
 17499  105744  012737  105770  000004          MOV     #2$,    @#4             ;LET VECTOR 4 POINT TO CONTINUE TESTING
 17500  105752  012737  106022  000114          MOV     #NXMPAR,@#114           ;LET VECTOR 114 POINT TO ERROR ROUTINE
 17501  105760  005737  157776                  TST     @#157776                ;READ ADDRESS 17757776 (SHOULD TRAP)
 17502  105764  104037                          ERROR   +37                     ;NXM AND PARITY ABORT DIN'T HAPPEN
 17503  105766  000402                          BR      ABOEXT                  ;GO EXIT TEST
 17504  105770  062706  000004          2$:     ADD     #4,     SP              ;RESET STACK AFTER TRAP
 17505  105774  005037  177746          ABOEXT: CLR     CCR                     ;CLEAR CCR FOR EXIT
 17506  106000  005037  177572                  CLR     SR0                     ;DISABLE MMU
 17507  106004  013737  002762  000004          MOV     SLOC00, @#4             ;RESTORE VECTOR 4
 17508  106012  013737  002764  000114          MOV     SLOC01, @#114           ;RESTORE VECTOR 114
 17509  106020  000404                          BR      TST22           ;;GO TO NEXT TEST
 17510
 17511
 17512
 17513  106022  062706  000004          NXMPAR: ADD     #4,     SP              ;RESET STACK AFTER TRAP
 17514  106026  104040                          ERROR   +40                     ;PARITY ABORT NOT BLOCKED BY NXM TRAP
 17515  106030  000002                          RTI
 17516
```

```
                                .SBTTL  TEST  - MULTIPROCCESSING INSTRUCTION TESTS
                                ;MULTIPROCCESSING INSTRUCTION TESTS - THIS TEST WILL VERIFY THAT THE MULTI-
                                ;PROCCESSING INSTRUCTIONS DO A BYPASS OF THE CACHE. THIS TEST WILL NOT VERIFY
                                ;THE REST OF THE FUNCTIONALITY OF THESE INSTRUCTIONS BECAUSE THAT WILL ALREADY
                                ;HAVE BEEN CHECKED IN THE BASE INSTRUCTION TESTS. THE TEST WILL FIRST ALLOCATE
                                ;AN ADDRESS IN CACHE THEN A TSTSET INSTRUCTION WILL BE DONE AT THAT ADDRESS.
                                ;A HIT SHOULD BE RECORDED ON THE ACCESS. NEXT, THE ADDRESS WILL BE READ AND
                                ;A MISS SHOULD BE RECORDED BECAUSE THE FORCED BYPASS ON THE TSTSET INSTRUCTION
                                ;SHOULD HAVE INVALIDATED THE CACHE ENTRY. THE SAME SEQUENCE WILL THEN BE
                                ;REPEATED FOR THE WRTLCK INSTRUCTION.
                                ;
                                ;BGNTST
                                ;READ TEST LOCATION TO ALLOCATE CACHE
                                ;DO TSTSET INSTRUCTION
                                ;IF HIT/MISS REGISTER BIT 3 NOT SET OR BIT 2 SET THEN
                                ;.       ERROR IN MULTI-PROCESSOR HOOKS
                                ;ENDIF
                                ;READ TEST LOCATION (ALSO ALLOCATES CACHE FOR WRTLCK)
                                ;DO WRTLCK INSTRUCTION
                                ;IF HIT/MISS REGISTER BIT 3 NOT SET OR BIT 2 SET THEN
                                ;.       ERROR IN MULTI-PROCESSOR HOOKS
                                ;ENDIF
                                ;READ TEST LOCATION
                                ;DO ASRB INSTRUCTION
                                ;IF HIT/MISS REGISTER BIT 3 NOT SET OR BIT 2 SET THEN
                                ;.       ERROR IN MULTI-PROCESSOR HOOKS
                                ;ENDIF
                                ;ENDTST
                                ;NOTE: THE CODE IS POSITION DEPENDENT

                                ;;*****************************************************************
106032  000004                  TST22:  SCOPE
106034  042737  001000  177520          BIC     #1000,BCSR      ;DISABLE HALT ON BREAK
106042  005737  003122                  TST     TSTLOC          ;READ TEST LOCATION TO ALLOCATE CACHE
                                        TSTSET  TSTLOC          ;DO TSTSET INSTRUCTION
106046  007237                  7$:     .WORD   7237            ;THESE NEXT TWO LOCATIONS ARE THE TSTSET
106050  003122                          .WORD   TSTLOC          ;INSTR. BECAUSE THE ASSEMBLER WAS NOT READY
106052  013737  177752  110504          MOV     HITMIS,RECDAT   ;STORE REGISTER
106060  032737  000010  110504          BIT     #BIT3,RECDAT    ;IF HIT/MISS REGISTER BIT 3 NOT SET
106066  001001                          BNE     1$              ;THEN
106070  104045                          ERROR   .45             ;ERROR IN RECORDING HITS IN HIT/MISS
106072  032737  000004  110504  1$:     BIT     #BIT2,RECDAT    ;IF HIT/MISS REGISTER BIT 2 SET
106100  001404                          BEQ     2$              ;THEN
106102  013737  106046  001126          MOV     @#7$,   $BDDAT  ;SAVE OPCODE
106110  104041                          ERROR   .41             ;MULTI-PROCESSOR HOOK INSTRUCTION DOESN'T CAUSE
                                ;2$:    WRTLCK  TSTLOC          ;DO WRTLCK INSTRUCTION
106112  007337                  2$:     .WORD   7337            ;THESE NEXT TWO WORDS ARE THE WRTLCK
106114  003122                          .WORD   TSTLOC          ;INSTR. BECAUSE THE ASSEMBLER WAS NOT READY
106116  013737  177752  110504          MOV     HITMIS,RECDAT   ;STORE REGISTER
106124  032737  000010  110504          BIT     #BIT3,RECDAT    ;IF HIT/MISS REGISTER BIT 3 NOT SET
106132  001001                          BNE     3$              ;THEN
106134  104045                          ERROR   .45             ;ERROR IN RECORDING HITS IN HIT/MISS
106136  032737  000004  110504  3$:     BIT     #BIT2,RECDAT    ;IF HIT/MISS REGISTER BIT 2 SET
106144  001404                          BEQ     4$              ;THEN
106146  013737  106112  001126          MOV     @#2$,   $BDDAT  ;SAVE OPCODE
106154  104041                          ERROR   .41             ;MULTI-PROCESSOR HOOK INSTRUCTION DOESN'T CAUSE
```

```
17573  106156  106237  003122          4$:     ASRB    TSTLOC                  ;THE THIRD BYPASS INSTRUCTION
17574  106162  013737  177752  110504          MOV     HITMIS,RECDAT           ;STORE REGISTER
17575  106170  032737  000010  110504          BIT     #BIT3,RECDAT            ;IF HIT/MISS REGISTER BIT 3 NOT SET
17576  106176  001001                          BNE     5$                      ;THEN
17577  106200  104045                          ERROR   .45                     ;ERROR IN RECORDING HITS IN HIT/MISS
17578  106202  052737  001000  177520  5$:     BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
17579  106210  032737  000004  110504          BIT     #BIT2,RECDAT            ;IF HIT/MISS REGISTER BIT 2 SET
17580  106216  001404                          BEQ     TST23           ;;EXIT TEST
17581  106220  013737  106156  001126          MOV     @#4$,   $BDDAT          ;SAVE OPCODE
17582  106226  104041                          ERROR   .41                     ;MULTI-PROCESSOR HOOK INSTRUCTION DOESN'TT CAUSE
17583
17584
17585
```

```
                                .SBTTL  TEST  - DATA STORE RAM TESTS
                                ;DATA STORE RAM TESTS - THERE ARE TWO TESTS FOR THE DATA STORE RAM.
                                ;THE FIRST TEST WILL BE A NO DUAL ADDRESSING TEST AND THE SECOND
                                ;TEST WILL BE A DATA RELIABILITY TEST. THE NO DUAL ADDRESSING TEST
                                ;WILL FIRST WRITE EACH WORD OF THE CACHE RAM WITH ITS WORD ADDRESS
                                ;AND VERIFY THE CONTENTS WITH A READ OF EACH ADDRESS. THEN EACH
                                ;BYTE WILL BE WRITTEN WITH ITS ADDRESS AND CHECKED. THE DATA
                                ;RELIABILITY TEST THAT WILL BE USED IS A TEST CALLED MOVING INVERSIONS.
                                ;
                                ;BGNTST 1
                                ;SETUP MMU REGISTERS TO HAVE BYPASS ON KERNAL SPACE AND NO
                                ;       BYPASS ON USER SPACE
                                ;LET PS EQUAL KERNAL FOR CURRENT MODE AND USER FOR PREVIOUS
                                ;       MODE
                                ;ENABLE MMU
                                ;GET FIRST ADDRESS OF 4K WORD DATA BUFFER
                                ;CLEAR DATA TO BE WRITTEN
                                ;SET DIAGNOSTIC BIT (BIT<1>) IN CCR
                                ;DO UNTIL DATA TO BE WRITTEN EQUALS 20000(8)
                                ;.      WRITE DATA TO ADDRESS
                                ;.      ADD 2 TO ADDRESS
                                ;.      ADD 2 TO DATA TO BE WRITTEN
                                ;ENDDO
                                ;GET FIRST ADDRESS OF 4K WORD DATA BUFFER
                                ;CLEAR EXPECTED DATA
                                ;DO UNTIL EXPECTED DATA EQUALS 20000(8)
                                ;.      READ ADDRESS
                                ;.      IF RECEIVED DATA NE EXPECTED DATA THEN
                                ;.      .       GOTO DATA STORE PARITY ERROR ROUTINE
                                ;.      ENDIF
                                ;.      ADD 2 TO EXPECTED DATA
                                ;.      ADD 2 TO ADDRESS
                                ;ENDDO
                                ;GET FIRST ADDRESS OF 8K BYTE DATA BUFFER
                                ;CLEAR DATA TO BE WRITTEN
                                ;DO UNTIL ALL BYTES CHECKED
                                ;.      WRITE DATA TO ADDRESS
                                ;.      ADD 1 TO ADDRESS
                                ;.      ADD 1 TO DATA TO BE WRITTEN
                                ;ENDDO
                                ;GET FIRST ADDRESS OF 8K BYTE DATA BUFFER
                                ;CLEAR EXPECTED DATA
                                ;DO UNTIL EXPECTED DATA EQUALS 20000(8)
                                ;.      READ ADDRESS
                                ;.      IF RECEIVED DATA NE EXPECTED DATA THEN
                                ;.      .       GOTO DATA STORE PARITY ERROR ROUTINE
                                ;.      ENDIF
                                ;.      ADD 1 TO EXPECTED DATA
                                ;.      ADD 1 TO ADDRESS
                                ;ENDDO
                                ;ENDTST
                                ;
                                ;;*****************************************************************
106230  000004                  TST23:  SCOPE
106232  004737  132156                  JSR     PC,     INITMM          ;SETUP MMU REGISTERS
106236  012737  002000  172354          MOV     #2000,  KIPAR6          ;LET PAR6 MAP TO ADDRESS 200000
```

```
17642  106244  005237  177572                  INC     SR0             ;TURN ON MEMORY MANAGEMENT
17643  106250  012702  140000                  MOV     #140000,R2      ;GET FIRST ADDRESS OF 4K WORD BUFFER
17644  106254  005001                          CLR     R1              ;INIT DATA TO BE WRITTEN
17645  106256  052737  000002  177746          BIS     #BIT01, CCR     ;SET DIAG BIT IN CASE NO MEMORY PRESENT
17646  106264  042737  001000  177520          BIC     #1000,BCSR      ;DISABLE HALT ON BREAK
17647  106272  020127  020000          1$:     CMP     R1,     #20000  ;DO UNTIL ALL ADDRESSES WRITTEN
17648  106276  001404                          BEQ     2$              ;IF DONE GO CHECK DATA
17649  106300  010122                          MOV     R1,     (R2)+   ;WRITE DATA TO ADDRESS
17650  106302  062701  000002                  ADD     #2,     R1      ;UPDATE DATA BY 2 (WORD BOUNDARY)
17651  106306  000771                          BR      1$              ;ENDDO
17652  106310  042737  000002  177746  2$:     BIC     #BIT01, CCR     ;CLEAR DIAGNOSTIC BIT
17653  106316  012702  140000                  MOV     #140000,R2      ;GET FIRST ADDRESS OF 4K WORD BUFFER
17654  106322  005001                          CLR     R1              ;INIT DATA TO BE CHECKED
17655  106324  020127  020000          3$:     CMP     R1,     #20000  ;DO UNTIL ALL ADDRESSES CHECKED
17656  106330  001426                          BEQ     5$              ;IF DONE GO DO BYTES
17657  106332  010137  001122                  MOV     R1,     $BDADR  ;STORE FOR ERRORS
17658  106336  052737  100000  001122          BIS     #BIT15, $BDADR  ;
17659  106344  012237  110504                  MOV     (R2)+,  RECDAT  ;READ TEST LOCATION
17660  106350  013737  177752  001160          MOV     HITMIS, $TMP0   ;STORE REGISTER
17661  106356  032737  000020  001160          BIT     #BIT04,$TMP0    ;HIT?
17662  106364  001001                          BNE     100$            ;IF YES, BRANCH
17663  106366  104045                          ERROR   45              ;DATA RAM ERROR
17664  106370  020137  110504          100$:   CMP     R1,     RECDAT  ;IF RECIEVED DATA NOT EQUAL TO EXPECTED
17665  106374  001401                          BEQ     4$              ;THEN
17666  106376  104043                          ERROR   43              ;DATA RAM ERROR
17667  106400  062701  000002          4$:     ADD     #2,     R1      ;UPDATE EXPECTED DATA BY 2
17668  106404  000747                          BR      3$              ;ENDDO
17669  106406  012702  140000          5$:     MOV     #140000,R2      ;GET FIRST ADDRESS OF 4K WORD BUFFER
17670  106412  005001                          CLR     R1              ;INIT DATA TO BE WRITTEN
17671  106414  052737  000002  177746          BIS     #BIT01, CCR     ;SET DIAG BIT IN CASE NO MEMORY PRESENT
17672  106422  020127  020000          6$:     CMP     R1,     #20000  ;DO UNTIL ALL ADDRESSES WRITTEN
17673  106426  001404                          BEQ     7$              ;IF DONE GO CHECK DATA
17674  106430  110122                          MOVB    R1,     (R2)+   ;WRITE DATA TO ADDRESS
17675  106432  062701  000001                  ADD     #1,     R1      ;UPDATE DATA BY 1 (BYTE BOUNDARY)
17676  106436  000771                          BR      6$              ;ENDDO
17677  106440  042737  000002  177746  7$:     BIC     #BIT01, CCR     ;CLEAR DIAGNOSTIC BIT
17678  106446  012702  140000                  MOV     #140000,R2      ;GET FIRST ADDRESS OF 4K WORD BUFFER
17679  106452  005001                          CLR     R1              ;INIT DATA TO BE CHECKED
17680  106454  020127  020000          8$:     CMP     R1,     #20000  ;DO UNTIL ALL ADDRESSES CHECKED
17681  106460  001426                          BEQ     10$             ;IF DONE EXIT TEST
17682  106462  010137  001122                  MOV     R1,     $BDADR  ;STORE FOR ERRORS
17683  106466  052737  100000  001122          BIS     #BIT15, $BDADR  ;
17684  106474  112237  110504                  MOVB    (R2)+,  RECDAT  ;READ TEST LOCATION
17685  106500  013737  177752  001160          MOV     HITMIS, $TMP0   ;STORE REGISTER
17686  106506  032737  000020  001160          BIT     #BIT04,$TMP0    ;HIT?
17687  106514  001001                          BNE     101$            ;IF YES, BRANCH
17688  106516  104045                          ERROR   45              ;DATA RAM ERROR
17689  106520  120137  110504          101$:   CMPB    R1,     RECDAT  ;IF RECIEVED DATA NOT EQUAL TO EXPECTED
17690  106524  001401                          BEQ     9$              ;THEN
17691  106526  104043                          ERROR   43              ;DATA RAM ERROR
17692  106530  062701  000001          9$:     ADD     #1,     R1      ;UPDATE EXPECTED DATA BY 1
17693  106534  000747                          BR      8$              ;ENDDO
17694  106536  052737  001000  177520  10$:    BIS     #1000,BCSR      ;ENABLE HALT ON BREAK
17695  106544  005037  177572                  CLR     SR0             ;TURN OFF MMU
17696
```

```
                        .SBTTL  TEST  - TAG STORE RAM TESTS
                        ;TAG STORE RAM TESTS - THERE ARE TWO TESTS FOR THE TAG STORE RAMS. THE FIRST
                        ;TEST IS AN ADDRESSING TEST TO ENSURE NO DUAL ADDRESSING OF THE TAG STORE.
                        ;THE SECOND TEST IS A "MOVING INVERSIONS" TEST THAT CHECKS THE DATA RELIABILITY
                        ;OF THE RAM STORE.
                        ;
                        ;DUAL ADDRESSING TEST - THIS WILL BE DONE BY FIRST FLUSHING THE CACHE, THEN
                        ;THE FIRST 512(10) LOCATIONS (BLOCK 0) WILL BE WRITTEN WITH ADDRESSES THAT
                        ;WILL CAUSE A INCREMENTING PATTERN TO BE STORED IN THOSE LOCATIONS OF THE TAG
                        ;STORE RAM. NEXT THE ENTIRE 4K ADDRESS RANGE WILL BE READ. THE HIT/MISS
                        ;REGISTER SHOULD ONLY REPORT HITS ON THE ADDRESSES THAT WERE ALLOCATED. THIS
                        ;PROCESS WILL BE REPEATED FOR EACH BLOCK.
                        ;
                        ;INITIALIZE LOW ADDRESS
                        ;SETUP AND ENABLE MMU
                        ;INITIALIZE BLOCK COUNTER
                        ;DO UNTIL ALL BLOCKS TESTED (8 TIMES)
                        ;.      FLUSH CACHE AND SET DIAGNOSTIC BIT
                        ;.      LET CURRENT ADDRESS = LOW ADDRESS
                        ;.      INITIALIZE ALLOCATION COUNTER
                        ;.      DO UNTIL ENTIRE BLOCK ALLOCATED (1000 TIMES)
                        ;.      .       .       READ CURRENT ADDRESS
                        ;.      .       ELSE
                        ;.      .       .       WRITE CURRENT ADDRESS
                        ;.      .       ENDIF
                        ;.      .       UPDATE CURRENT ADDRESS (ALSO ADD 200 TO PAR)
                        ;.      .       FOR I/O PAGE WRITE THE SAME ADDRESS AS BEFORE
                        ;.      ENDDO
                        ;.      INITIALIZE GOOD ADDRESS
                        ;.      INITIALIZE CURRENT ADDRESS
                        ;.      INITIALIZE LOCATION COUNTER
                        ;.      SAVE CONTENTS OF VECTOR 114
                        ;.      LET VECTOR 114 POINT TO TAG STORE PARITY ABORT ROUTINE
                        ;.      DO UNTIL ALL LOCATIONS CHECKED (1000 TIMES)
                        ;.      .       INITIALIZE CHECK COUNTER
                        ;.      .       DO UNTIL ADDRESS IN EACH BLOCK CHECKED
                        ;.      .       .       READ CURRENT ADDRESS
                        ;.      .       .       IF HIT THEN
                        ;.      .       .       .       IF CURRENT ADDRESS NE GOOD ADDRESS THEN
                        ;.      .       .       .       .       ERROR
                        ;.      .       .       .       ENDIF
                        ;.      .       .       ELSE
                        ;.      .       .       .       IF CURRENT ADDRESS EQUAL GOOD ADDRESS THEN
                        ;.      .       .       .       .       ERROR
                        ;.      .       .       .       ENDIF
                        ;.      .       .       ENDIF
                        ;.      .       .       UPDATE GOOD ADDRESS (ADD #2)
                        ;.      .       .       UPDATE CURRENT ADDRESS
                        ;.      .       ENDDO
                        ;.      .       UPDATE LOW ADDRESS (ADD #2000)
                        ;.      ENDDO
                        ;ENDDO
                        ;DISABLE MMU
                        ;RESTORE VECTOR 114
                        ;ENDTST
```

```
17753                                   ;;*****************************************************************
17754  106550  000004                   TST24:  SCOPE
17755  106552  013737  000004  001160           MOV     @#4,$TMP0               ;STORE TIMEOUT VECTOR
17756  106560  012737  107304  000004           MOV     #20$,@#4                ;POINT NEW
17757  106566  012737  140000  002766           MOV     #140000,LOWADD          ;INITIALIZE LOW ADDRESS (USE PAR6)
17758  106574  004737  132156                   JSR     PC,     INITMM          ;INITIALIZE MMU
17759  106600  005237  177572                   INC     SR0                     ;ENABLE MMU
17760  106604  012737  000020  172516           MOV     #BIT04,MMR3             ;ENABLE 22-BIT MAPPING
17761  106612  012737  177770  003114           MOV     #-10.,  LOOPIN          ;DO UNTIL ALL BLOCKS TESTED
17762  106620  012701  000000           1$:     MOV     #0,R1                   ;DO IN 2 WORDS TO AVOID PMI MEMORY
17763  106624  042737  001000  177520   9$:     BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
17764  106632  005037  172354                   CLR     KIPAR6                  ;SET UP PAR6 FOR THIS TEST
17765  106636  012737  000402  177746           MOV     #402,   CCR             ;FLUSH CACHE AND SET DIAG BIT
17766  106644  013737  002766  110524           MOV     LOWADD, CURADD          ;GET FIRST ADDRESS IN CURRENT BLOCK
17767  106652  012737  177400  003112           MOV     #-400,  ALLCTR          ;DO UNTIL ALL ADDRESSES ALLOCATED
17768  106660  022737  002000  172354   2$:     CMP     #2000,  KIPAR6          ;IF ADDRESS LESS THAN 32K
17769  106666  002413                           BLT     3$                      ;THEN
17770  106670  052737  100000  172314           BIS     #BIT15,KIPDR6           ;SET BYPASS
17771  106676  017702  001622                   MOV     @CURADD,R2              ;STORE CURRENT DATA
17772  106702  042737  100000  172314           BIC     #BIT15,KIPDR6           ;ALLOCATE NEXT ACCESS
17773  106710  010277  001610                   MOV     R2,@CURADD              ;WRITE ALLOCATE
17774  106714  000402                           BR      4$                      ;ELSE
17775  106716  005077  001602           3$:     CLR     @CURADD                 ;WRITE CURRENT ADDRESS
17776  106722  062737  000002  110524   4$:     ADD     #2,     CURADD          ;UPDATE CURRENT ADDRESS
17777  106730  062737  000200  172354           ADD     #200,   KIPAR6
17778  106736  022737  177600  172354           CMP     #177600,KIPAR6          ;REACHED I/O PAGE?
17779  106744  001003                           BNE     10$                     ;BRANCH IF NOT
17780  106746  162737  000200  172354           SUB     #200,KIPAR6             ;DON'T UPDATE PAR FOR I/O PAGE
17781  106754  005237  003112           10$:    INC     ALLCTR                  ;IF ALL ADDRESSES ALLOCATED
17782  106760  002737                           BLT     2$                      ;ENDDO
17783  106762  052737  000004  177746           BIS     #BIT02, CCR             ;RUN WITH FORCE MISS
17784  106770  013737  002766  002770           MOV     LOWADD, GOODAD          ;INITIALIZE GOOD ADDRESS
17785  106776  060137  002770                   ADD     R1,     GOODAD          ;
17786  107002  012737  140000  110524           MOV     #140000,CURADD          ;GET FIRST ADDRESS
17787  107010  060137  110524                   ADD     R1,     CURADD          ;START WITH 1 OR 2 LOCATION
17788  107014  005037  172354                   CLR     KIPAR6
17789  107020  072127  000006                   ASH     #6,R1                   ;
17790  107024  060137  172354                   ADD     R1,KIPAR6               ;MAKE SURE ON THE RIGHT BOUNDARY
17791  107030  012737  177600  003112           MOV     #-200.  ALLCTR          ;DO UNTIL ALL LOCATIONS CHECKED
17792  107036  012737  177770  110500   5$:     MOV     #-10.,  DCOUNT          ;DO UNTIL ADDRESS IN EACH BLOCK CHECKED
17793  107044  013737  110524  001122   6$:     MOV     CURADD, $BDADR          ;IN CASE OF ERRORS
17794  107052  042737  160000  001122           BIC     #160000,$BDADR          ;CLEAR PAR BITS
17795  107060  042737  000004  177746           BIC     #BIT02, CCR             ;CLEAR FORCE MISS
17796  107066  005777  001432                   TST     @CURADD                 ;READ CURRENT ADDRESS
17797  107072  013737  177752  110504           MOV     HITMIS, RECDAT          ;STORE REGISTER
17798  107100  032737  000004  110504           BIT     #BIT2,  RECDAT          ;IF ACCESS WAS A HIT
17799  107106  001406                           BEQ     7$                      ;THEN
17800  107110  023737  110524  002770           CMP     CURADD, GOODAD          ;IF CURRENT ADDRESS NOT EQUAL TO GOOD
17801  107116  001407                           BEQ     8$                      ;ADDRESS THEN
17802  107120  104046                           ERROR   .46                     ;ERROR IN TAG STORE
17803  107122  000405                           BR      8$                      ;
17804  107124  023737  110524  002770   7$:     CMP     CURADD, GOODAD          ;IF CURRENT ADDRESS EQUAL GOOD ADDRESS
17805  107132  001001                           BNE     8$                      ;THEN
17806  107134  104047                           ERROR   .47                     ;ERROR IN TAG STORE
17807  107136  062737  001000  110524   8$:     ADD     #1000,  CURADD          ;UPDATE TO NEXT BLOCK
17808  107144  052737  000004  177746           BIS     #BIT02, CCR             ;RUN WITH FORCE MISS
```

```
17809  107152  005237  110500                      INC     DCOUNT                  ;IF ADDRESS NOT CHECKED FOR EACH BLOCK
17810  107156  002732                              BLT     6$                      ;ENDDO
17811  107160  062737  000004  002770              ADD     #4,     GOODAD          ;UPDATE GOOD ADDRESS
17812  107166  162737  007774  110524              SUB     #7774,  CURADD          ;INCREMENT IN 2 WORDS
17813  107174  062737  000400  172354              ADD     #400,   KIPAR6
17814  107202  005237  003112                      INC     ALLCTR                  ;IF ALL ADDRESSES CHECKED
17815  107206  002713                              BLT     5$                      ;ENDDO
17816  107210  052737  001000  177520              BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
17817  107216  062701  000002                      ADD     #2.     R1              ;PREPARE FOR THE 2ND PASS THRU
17818  107222  022701  000002                      CMP     #2.     R1              ;DONE TWICE?
17819  107226  001002                              BNE     30$                     ;IF SO, EXIT
17820  107230  000137  106624                      JMP     9$
17821  107234  062737  002000  002766  30$:       ADD     #2000,  LOWADD          ;UPDATE TO NEXT BLOCK TO BE ALLOCATED
17822  107242  005237  003114                      INC     LOOPIN                  ;IF ALL BLOCKS WERE TESTED
17823  107246  002002                              BGE     15$                     ;
17824  107250  000137  106620                      JMP     1$                      ;ENDDO
17825  107254  012737  000400  177746  15$:       MOV     #400,   CCR             ;CLEAR DIAGNOSTIC BIT
17826  107262  005037  177572                      CLR     SR0                     ;DISABLE MMU
17827  107266  013737  001160  000004              MOV     $TMP0,@#4               ;RESTORE TIMEOUT VECTOR
17828  107274  042737  000020  172516              BIC     #BIT04,MMR3             ;ENABLE 22-BIT MAPPING
17829  107302  000401                              BR      TST25           ;;GO TO NEXT TEST
17830
17831  107304  000002                      20$:    RTI
17832
17833
```

```
                                        .SBTTL  TEST - STANDALONE MODE TEST
                                        ;THIS TEST VERIFIES THAT NMX CAN BE CREATED IN STANDALONE MODE.
                                        ;
                                        ;ALLOCATE INSTRUCTIONS IN CACHE
                                        ;GUARANTEE MISS ON TEST LOCATION
                                        ;IN STANALONE MODE ACCESS TEST LOCATION
                                        ;IF NO TIMEOUT THEN
                                        ;.      ERROR
                                        ;ENDIF
                                        ;
                                        ;;*****************************************************************
107306  000004                  TST25:  SCOPE
107310  012737  000400  177746          MOV     #400,CCR                ;FLUSH CACHE
107316  012701  107316                  MOV     #.,R1                   ;START WITH CURRENT INSTRUCTION
107322  005721                  1$:     TST     (R1)+                   ;READ A WORD
107324  022701  107422                  CMP     #10$,R1                 ;DONE?
107330  001374                          BNE     1$                      ;IF NOT, CONTINUE
107332  005737  020000                  TST     @#20000                 ;TO GUARANTY MISS ON 0
107336  013702  000004                  MOV     @#4,R2                  ;STORE TIMEOUT VECTOR
107342  012737  107376  000004          MOV     #5$,@#4                 ;POINT NEW TO THE TEST
107350  012737  000340  000006          MOV     #340,@#6                ;AT PRIORITY 7
107356  042737  001000  177520          BIC     #BIT09,BCSR             ;DISABLE HALT ON BREAK
107364  052737  000400  177520          BIS     #BIT08,BCSR             ;GO TO STANDALONE MODE
107372  005737  000000                  TST     @#0                     ;MISS, SHOULD TIMEOUT
107376  042737  000400  177520  5$:     BIC     #BIT08,BCSR             ;CLEAR STANDALONE BIT
107404  052737  001000  177520          BIS     #BIT09,BCSR             ;ENABLE HALT ON BREAK
107412  022716  107376                  CMP     #5$,(SP)                ;TIMEOUT?
107416  001401                          BEQ     10$                     ;IF YES, BRANCH
107420  104044                          ERROR   .44
107422  012706  001100          10$:    MOV     #1100,SP                ;RESTORE STACK
107426  010237  000004                  MOV     R2,@#4                  ;AND TIMEOUT VECTOR
```

```
17865                                   .SBTTL  TEST  - MOVING INVERSIONS TEST FOR DATA RAMS
17866                                   ;MOVING INVERSIONS TEST FOR DATA RAMS - THE TEST IS STARTED AFTER LOADING THE
17867                                   ;RAM STORE WITH 0'S. EACH ADDRESS IS READ AND VERIFIED TO BE ALL 0'S. THEN A
17868                                   ;1 IS SUBSTITUTED IN A BIT POSITION AND THE NEW WORD IS WRITTEN. NEXT THE
17869                                   ;ADDRESS IS READ TO VERIFY THE NEW CONTENTS. THIS IS REPEATED FOR EACH BIT OF
17870                                   ;THE WORD LEAVING THE ARRAY FILED WITH 1'S. THE WHOLE PROCESS IS REPEATED
17871                                   ;PLUGGING IN 0'S TO THE 1'S AND REPEATED TWICE MORE ADDRESSING IN THE DOWNWARD
17872                                   ;DIRECTION. FINALLY EVERYTHING IS REPEATED FOR EACH BIT POSITION BEING THE
17873                                   ;LSB. TO SAVE TIME AND KNOWING THE LAYOUT OF THE RAM CHIPS INSTEAD OF DOING
17874                                   ;ONLY A SINGLE BIT AT A TIME EVERY FOURTH BIT WILL BE DONE CONCURRENTLY.
17875                                   ;THIS TEST RUNS IN STANDALONE MODE.
17876                                   ;
17877                                   ;BGNTST
17878                                   ;SETUP AND ENABLE MMU
17879                                   ;SETUP CCR TO ABORT PARITY ERRORS
17880                                   ;CLEAR CACHE
17881                                   ;DO IN STANDALONE MODE FOR EACH HALF SEPARATELY
17882                                   ;LET FWDSEQ = #1
17883                                   ;LET ADDLSB = #1
17884                                   ;DO UNTIL ADDLSB EQ #20000
17885                                   ;.      LET CURDAT = 0
17886                                   ;.      LET RITEDA = #1
17887                                   ;.      LET NEWDAT = #1
17888                                   ;.      IF FWDSEQ = #1 THEN
17889                                   ;.      .       LET FSTADD EQUAL FIRST ADDRESS OF 4K BYTE BUFFER
17890                                   ;.      .       LET LASTAD EQUAL LAST ADDRESS OF 4K BYTE BUFFER
17891                                   ;.      ELSE
17892                                   ;.      .       LET FSTADD EQUAL LAST ADDRESS OF 4K BYTE BUFFER
17893                                   ;.      .       LET LASTAD EQUAL FIRST ADDRESS OF 4K BYTE BUFFER
17894                                   ;.      ENDIF
17895                                   ;.      LET CURADD = FSTADD
17896                                   ;.      LET DCOUNT = #0
17897                                   ;.      SAVE CONTENTS OF VECTOR 114
17898                                   ;.      LET VECTOR 114 POINT TO DATA STORE PARITY ABORT ROUTINE
17899                                   ;.      DO UNTIL DCOUNT EQ #10
17900                                   ;.      .       LET RECDAT = @CURADD
17901                                   ;.      .       IF RECDAT NE CURDAT THEN
17902                                   ;.      .       .       LET R1 EQUAL CURRENT DATA
17903                                   ;.      .       .       GOTO DATA STORE PARITY ERROR ROUTINE
17904                                   ;.      .       ENDIF
17905                                   ;.      .       IF DCOUNT GT #3 THEN
17906                                   ;.      .       .       LET @CURADD = @CURADD CLEARBY RITEDA
17907                                   ;.      .       ELSE
17908                                   ;.      .       .       LET @CURADD = @CURADD SETBY RITEDA
17909                                   ;.      .       ENDIF
17910                                   ;.      .       LET RECDAT = @CURADD
17911                                   ;.      .       IF RECDAT NE NEWDAT THEN
17912                                   ;.      .       .       LET R1 EQUAL NEW DATA
17913                                   ;.      .       .       GOTO DATA STORE PARITY ERROR ROUTINE
17914                                   ;.      .       ENDIF
17915                                   ;.      .       IF CURADD EQ LASTAD THEN
17916                                   ;.      .       .       IF RITEDA = #210 THEN
17917                                   ;.      .       .       .       IF DCOUNT NE #7 THEN
17918                                   ;.      .       .       .       .       LET CURDAT = #377
17919                                   ;.      .       .       .       .       LET RITEDA = #1
17920                                   ;.      .       .       .       .       LET NEWDAT = #376
```

```
17921                                   ;.      .       .       .       ENDIF
17922                                   ;.      .       .       ELSE
17923                                   ;.      .       .       .       LET CURDAT = NEWDAT
17924                                   ;.      .       .       .       ROTATE RITEDA
17925                                   ;.      .       .       .       IF DCOUNT GT #3 THEN
17926                                   ;.      .       .       .       .       LET NEWDAT = NEWDAT CLEAREDBY RITEDA
17927                                   ;.      .       .       .       ELSE
17928                                   ;.      .       .       .       .       LET NEWDAT = NEWDAT SETBY RITEDA
17929                                   ;.      .       .       .       ENDIF
17930                                   ;.      .       .       ENDIF
17931                                   ;.      .       .       INCREMENT DCOUNT
17932                                   ;.      .       .       LET CURADD = FSTADD
17933                                   ;.      .       ELSE
17934                                   ;.      .       .       IF FWDSEQ = #1 THEN
17935                                   ;.      .       .       .       LET CURADD = CURADD + ADDLSB
17936                                   ;.      .       .       .       IF CARRY THEN
17937                                   ;.      .       .       .       .       LET CURADD = CURADD + #1
17938                                   ;.      .       .       .       ENDIF
17939                                   ;.      .       .       ELSE
17940                                   ;.      .       .       .       LET CURADD = CURADD - ADDLSB
17941                                   ;.      .       .       .       IF CARRY THEN
17942                                   ;.      .       .       .       .       LET CURADD = LASTAD - ADDLSB
17943                                   ;.      .       .       .       .       LET CURADD = CURADD - #1
17944                                   ;.      .       .       .       ENDIF
17945                                   ;.      .       .       ENDIF
17946                                   ;.      .       ENDIF
17947                                   ;.      ENDDO
17948                                   ;.      IF FWDSEQ EQ #1 THEN
17949                                   ;.      .       LET FWDSEQ = #0
17950                                   ;.      ELSE
17951                                   ;.      .       ROTATE ADDLSB
17952                                   ;.      .       LET FWDSEQ = #1
17953                                   ;.      ENDIF
17954                                   ;ENDDO
17955                                   ;RESTORE VECTOR 114
17956                                   ;ENDTST
17957
17958                                   ;;**********************************************************************
17959   107432  000004                  TST26:  SCOPE
17960   107434  032777  000200  071476          BIT     #BIT07,@SWR             ;RUN THIS TEST?
17961   107442  001002                          BNE     100$                    ;IF SET, GO DO IT
17962   107444  000137  110526                  JMP     ENDMOV                  ;OTHERWISE, GO TO NEXT TEST
17963   107450  042737  001000  177520  100$:   BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
17964   107456  004737  132156                  JSR     PC,     INITMM          ;SETUP MEMORY MANAGEMENT
17965   107462  012737  002000  172354          MOV     #2000,KIPAR6            ;START ON 32K BOUNDARY
17966   107470  005237  177572                  INC     SR0                     ;TURN ON MMU
17967   107474  052737  000002  177746          BIS     #2,     CCR             ;SET DIAG. BIT
17968   107502  013737  000004  001160          MOV     @#4,    $TMP0           ;SAVE 4
17969                                   ;
17970                                   ;STORE TEST IN THE FIRST 2K AND THEN IN THE SECOND 2K
17971                                   ;
17972   107510  012703  140000                  MOV     #140000,R3              ;START FOR THE TEST
17973   107514  012704  150000                  MOV     #150000,R4              ;LOWER BOUNDARY TEST AREA
17974   107520  012705  157777                  MOV     #157777,R5              ;HIGH BOUNDARY TEST AREA
17975   107524  000406                          BR      2$
17976   107526  012703  150000          1$:     MOV     #150000,R3              ;START OF THE TEST
```

```
17977  107532  012704  140000          MOV     #140000,R4              ;LOWER BOUNDARY TEST AREA
17978  107536  012705  147777          MOV     #147777,R5              ;HIGH BOUNDARY
17979  107542  012702  107662  2$:     MOV     #STMOVI,R2              ;START WITH CURRENT
17980  107546  010300                  MOV     R3,R0                   ;MOVE TO UPPER 4K
17981  107550  012220          3$:     MOV     (R2)+,(R0)+             ;WORD BY WORD
17982  107552  022702  110526          CMP     #ENDMOV,R2              ;ALL DONE
17983  107556  001374                  BNE     3$                      ;
17984  107560  010400                  MOV     R4,R0                   ;CLEAR CACHE UNDER TEST
17985  107562  012701  004000          MOV     #4000.  R1
17986  107566  005020          4$:     CLR     (R0)+
17987  107570  077102                  SOB     R1,     4$
17988  107572  004713                  JSR     PC,(R3)                 ;GO DO THE ROUTINE
17989  107574  005702                  TST     R2                      ;ANY ERRORS?
17990  107576  001411                  BEQ     5$                      ;IF NOT, CONTINUE
17991  107600  010037  110504          MOV     R0,RECDAT               ;DATA RECEIVED
17992  107604  010237  001122          MOV     R2,$BDADR               ;ADDRESS RECIEVED
17993  107610  042737  140000  001122  BIC     #140000,$BDADR          ;STRIP OF PAR BITS
17994  107616  104043                  ERROR   +43                     ;
17995  107620  000403                  BR      6$                      ;EXIT
17996  107622  022703  150000  5$:     CMP     #150000,R3              ;DONE FOR BOTH HALVES?
17997  107626  001337                  BNE     1$                      ;IF NOT, DO AGAIN
17998  107630  052737  001000  177520  6$:     BIS     #BIT09,BCSR             ;ENABLE HALT ON BREAK
17999  107636  013737  001160  000004          MOV     $TMPO,@#4               ;RESTORE TIMEOUT VECTOR
18000  107644  012737  000400  177746          MOV     #400,CCR                ;INIT CCR FOR EXIT
18001  107652  005037  177572                  CLR     SR0                     ;TURN OFF MMU
18002  107656  000137  110526                  JMP     ENDMOV                  ;GO TO NEXT TEST
18003
18004                                  .DSABL AMA
18005  107662  052737  000400  177520  STMOVI: BIS     #BIT08,@#BCSR           ;STANDALONE MODE
18006  107670  005002                  CLR     R2                      ;ERROR INDICATOR
18007  107672  012767  000001  000606          MOV     #1,     FWDSEQ          ;INIT UPWARD ADDRESSING INDICATOR
18008  107700  012767  000001  000602          MOV     #1,     ADDLSB          ;INIT LSB AND DO LOOP UNTIL SHIFTED OUT
18009  107706  042737  100000  172300          BIC     #100000,@#KIPDR0        ;NO BYPASS
18010  107714  012737  172360  000004          MOV     #KDPAR0,@#4             ;ALLOCATE TIMEOUT VECTOR
18011  107722  012737  000340  000006          MOV     #340,   @#6             ;AT PRIORITY 7
18012  107730  012737  000006  172360          MOV     #6,     @#KDPAR0        ;PUT RETURN
18013  107736  052737  100000  172300          BIS     #100000,@#KIPDR0        ;BYPASS
18014  107744  005067  000546  TSTLUP: CLR     CURDAT                  ;INIT CURRENT DATA
18015  107750  012767  000021  000534          MOV     #21.    RITEDA          ;INIT DATA TO BE WRITTEN
18016  107756  012767  000021  000530          MOV     #21.    NEWDAT          ;INIT EXPECTED DATA
18017  107764  005767  000516                  TST     FWDSEQ                  ;IF ADDRESSING UPWARD
18018  107770  001405                  BEQ     1$                      ;THEN
18019  107772  010467  000522                  MOV     R4,FSTADD               ;LET FIRST ADDRESS EQUAL LOWEST VALUE
18020  107776  010567  000520                  MOV     R5,LSTADD               ;LET LAST ADDRESS EQUAL HIGHEST VALUE
18021  110002  000404                  BR      2$                      ;ELSE
18022  110004  010567  000510  1$:     MOV     R5,FSTADD               ;LET FIRST ADDRESS EQUAL HIGHEST VALUE
18023  110010  010467  000506          MOV     R4,LSTADD               ;LET LAST ADDRESS EQUAL LOWEST VALUE
18024  110014  016767  000500  000502  2$:     MOV     FSTADD, CURADD          ;LET CURRENT ADDRESS EQUAL FIRST ADDRESS
18025  110022  005067  000452          CLR     DCOUNT                  ;INIT LOOP COUNTER
18026  110026  022767  000010  000444  BGNTLP: CMP     #10.    DCOUNT          ;DO LOOP 8 TIMES (4 TIMES TO WRITE 1'S
18027  110034  001567                  BEQ     ENDTLP                  ;AND 4 TIMES TO WRITE BACK 0'S)
18028                                  ;
18029                                  ; DON'T REWRITE TIMEOUT VECTOR
18030                                  ;
18031  110036  016700  000462          MOV     CURADD, R0              ;STORE CURRENT ADDRESS
18032  110042  042700  170000          BIC     #170000,R0              ;LEAVE ONLY LOW 4K BITS
```

```
18033  110046  022700  000004              CMP     #4,     R0              ;TIMOUT VECTOR?
18034  110052  001531                      BEQ     8$                      ;IF SO, DON'T REWRITE IT
18035  110054  022700  000005              CMP     #5,     R0              ;
18036  110060  001526                      BEQ     8$                      ;
18037  110062  022700  000006              CMP     #6,     R0              ;
18038  110066  001523                      BEQ     8$                      ;
18039  110070  022700  000007              CMP     #7,     R0              ;
18040  110074  001520                      BEQ     8$                      ;
18041                                  ;
18042                                  ;CHECK PATTERNS NOW
18043                                  ;
18044  110076  016701  000414              MOV     CURDAT, R1              ;MOVE GOOD DATA TO R1
18045  110102  117767  000416  000374      MOVB    @CURADD,RECDAT          ;FIRST READ LOCATION
18046  110110  126767  000370  000400      CMPB    RECDAT, CURDAT          ;IF RECIEVED DATA NOT EQUAL TO EXPECTED
18047  110116  001402                      BEQ     1$                      ;DATA THEN
18048  110120  000167  000334              JMP     EXBAD                   ;EXIT TEST
18049  110124  016701  000364      1$:     MOV     NEWDAT, R1              ;MOVE GOOD DATA TO R1
18050  110130  022767  000003  000342      CMP     #3,     DCOUNT          ;IF LOOP COUNT GREATER THAN 3
18051  110136  002004                      BGE     2$                      ;THEN
18052  110140  146777  000346  000356      BICB    RITEDA, @CURADD         ;CLEAR TEST DATA BY WRITE DATA
18053  110146  000403                      BR      3$                      ;ELSE
18054  110150  156777  000336  000346  2$: BISB    RITEDA, @CURADD         ;SET TEST DATA BY WRITE DATA
18055  110156  117767  000342  000320  3$: MOVB    @CURADD,RECDAT          ;DO READ AFTER WRITE
18056  110164  126767  000314  000322      CMPB    RECDAT, NEWDAT          ;IF RECIEVED DATA NOT EQUAL TO EXPECTED
18057  110172  001402                      BEQ     4$                      ;DATA THEN
18058  110174  000167  000260              JMP     EXBAD                   ;EXIT TEST
18059  110200  026767  000320  000314  4$: CMP     CURADD, LSTADD          ;IF CURRENT ADDRESS EQUALS LAST ADDRESS
18060  110206  001053                      BNE     8$                      ;THEN
18061  110210  022767  000210  000274      CMP     #210,   RITEDA          ;AND IF WRITE PATTERN EQUALS LAST
18062  110216  001016                      BNE     5$                      ;PATTERN THEN
18063  110220  022767  000007  000252      CMP     #7,     DCOUNT          ;AND TEST IS NOT ON LAST LOOP
18064  110226  001435                      BEQ     7$                      ;THEN
18065  110230  012767  000377  000260      MOV     #377,   CURDAT          ;SET UP TO WRITE 0'S
18066  110236  012767  000021  000246      MOV     #21,    RITEDA
18067  110244  012767  000356  000242      MOV     #356,   NEWDAT
18068  110252  000423                      BR      7$                      ;ELSE
18069  110254  016767  000234  000234  5$: MOV     NEWDAT, CURDAT          ;SET UP FOR NEXT DATA PATTERN
18070  110262  000241                      CLC
18071  110264  106167  000222              ROLB    RITEDA
18072  110270  005567  000216              ADC     RITEDA                  ;GET A PATTERN
18073  110274  022767  000003  000176      CMP     #3,     DCOUNT          ;AND IF DOWNWARD ADDRESSING
18074  110302  002004                      BGE     6$                      ;THEN
18075  110304  046767  000202  000202      BIC     RITEDA, NEWDAT          ;LET NEWDAT BE CLEARED BY WRITE DATA
18076  110312  000403                      BR      7$                      ;ELSE
18077  110314  056767  000172  000172  6$: BIS     RITEDA, NEWDAT          ;LET NEWDAT BE SET BY WRITE DATA
18078  110322  005267  000152      7$:     INC     DCOUNT                  ;UPDATE LOOP COUNTER
18079  110326  016767  000166  000170      MOV     FSTADD, CURADD          ;REINIT FIRST ADDRESS FOR THIS PASS
18080  110334  000426                      BR      10$                     ;ELSE
18081  110336  005767  000144      8$:     TST     FWDSEQ                  ;IF ADDRESSING UPWARD
18082  110342  001412                      BEQ     9$                      ;THEN
18083  110344  066767  000140  000152      ADD     ADDLSB, CURADD          ;CALCULATE NEXT HIGHER ADDRESS
18084  110352  020567  000146              CMP     R5,CURADD               ;IF CURRENT ADDRESS HAS BEEN INCREASED
18085  110356  002015                      BGE     10$                     ;ABOVE HIGHEST ADDRESS THEN
18086  110360  162767  007777  000136      SUB     #7777,  CURADD          ;ROLL ADDRESS BACK
18087  110366  000411                      BR      10$                     ;ELSE
18088  110370  166767  000114  000126  9$: SUB     ADDLSB, CURADD          ;CALCULATE NEXT LOWER ADDRESS
```

```
18089  110376  020467  000122                CMP     R4,CURADD               ;IF CURRENT ADDRESS HAS BEEN DECREASED
18090  110402  003403                        BLE     10$                     ;BELOW LOWEST ADDRESS THEN
18091  110404  062767  007777  000112        ADD     #7777,  CURADD          ;ROLL ADDRESS BACK
18092  110412  000605                10$:    BR      BGNTLP                  ;ENDDO
18093  110414  005767  000066        ENDTLP: TST     FWDSEQ                  ;IF ADDRESSING UPWARD FINISHED
18094  110420  001404                        BEQ     1$                      ;THEN
18095  110422  005067  000060                CLR     FWDSEQ                  ;DO ADDRESSING DOWNWARD
18096  110426  000167  177312                JMP     TSTLUP
18097  110432  012767  000001  000046  1$:   MOV     #1,     FWDSEQ          ;SET ADDRESSING UPWARD INDICATOR
18098  110440  006167  000044                ROL     ADDLSB                  ;UPDATE LSB TO NEXT POSITION
18099  110444  022767  020000  000036  ENDLUP: CMP    #20000,ADDLSB           ;ALL DONE?
18100  110452  001406                        BEQ     EXITST                  ;ENDDO
18101  110454  000167  177264                JMP     TSTLUP
18102  110460  016700  000020        EXBAD:  MOV     RECDAT,R0               ;STORE RECEIVED DATA
18103  110464  016702  000034                MOV     CURADD,R2               ;STORE ADDRESS
18104  110470  042737  000400  177520  EXITST: BIC   #BIT08,@#BCSR           ;OUT OF STANDALONE
18105  110476  000207                        RTS     PC
18106                                  .ENABL AMA
18107
18108  110500  000000                DCOUNT: .WORD   0
18109  110502  000000                EXPDAT: .WORD   0               ;STORES EXPECTED (GOOD) DATA FOR COMPARISONS
18110  110504  000000                RECDAT: .WORD   0               ;STORES RECIEVED DATA TO BE VERIFIED
18111  110506  000000                FWDSEQ: .WORD   0               ;USED TO INDICATE DIRECTION OF ADDRESSING
18112  110510  000000                ADDLSB: .WORD   0               ;STORES LEAST SIGNIFICANT BIT FOR RAM TESTS
18113  110512  000000                RITEDA: .WORD   0               ;STORES WRITE DATA FOR RAM TESTS
18114  110514  000000                NEWDAT: .WORD   0               ;DATA STORE FOR RAM TESTS
18115  110516  000000                CURDAT: .WORD   0               ;DATA STORE FOR RAM TESTS
18116  110520  000000                FSTADD: .WORD   0               ;STORES FIRST ADDRESS IN ADDRESSING SEQUENCE
18117  110522  000000                LSTADD: .WORD   0               ;STORES LAST ADDRESS IN ADDRESSING SEQUENCE
18118  110524  000000                CURADD: .WORD   0               ;STORES CURRENT ADDRESS FOR RAM TESTS
18119
18120  110526                        ENDMOV:
```

```
18121                                   .SBTTL  TEST  - MOVING INVERSIONS TEST FOR TAG STORE
18122                                   ;MOVING INVERSIONS TEST - THE TEST IS STARTED AFTER LOADING THE RAM STORE
18123                                   ;WITH 0'S. EACH ADDRESS IS READ AND VERIFIED TO BE ALL 0'S. THEN A 1 IS
18124                                   ;SUBSTITUTED IN A BIT POSITION AND THE NEW WORD IS WRITTEN. NEXT THE ADDRESS
18125                                   ;IS READ TO VERIFY THE NEW CONTENTS. THIS IS REPEATED FOR EACH BIT OF THE
18126                                   ;WORD LEAVING THE ARRAY FILED WITH 1'S. THE WHOLE PROCESS IS REPEATED PLUGGING
18127                                   ;IN 0'S TO THE 1'S AND REPEATED TWICE MORE ADDRESS IN THE DOWARD DIRECTION.
18128                                   ;FINALLY EVERYTHING IS REPEATED FOR EACH BIT POSITION BEING THE LSB. TO SAVE
18129                                   ;TIME AND KNOWING THE LAYOUT OF THE RAM CHIPS INSTEAD OF DOING ONLY A SINGLE
18130                                   ;BIT AT A TIME EVERY THIRD BIT WILL BE DONE CONCURRENTLY. ALSO NOTE THAT
18131                                   ;SINCE THE TAG STORE CANNOT BE DIRECTLY ACCESSED THE ENTIRE PATTERN MUST BE
18132                                   ;DONE BY DOING MEMORY CYCLES TO THE CORRECT BUS ADDRESSES. TO DO THIS THE
18133                                   ;TEST WILL BE DONE WITH THE DIAGNOSTIC BIT IN THE CCR (BIT 1) SET TO A 1.
18134                                   ;THIS TEST RUNS IN STANDALONE MODE.
18135                                   ;
18136                                   ;
18137                                   ;BGNTST
18138                                   ;SETUP AND ENABLE MMU
18139                                   ;SETUP CCR TO ABORT PARITY ERRORS
18140                                   ;DO IN STANDALONE MODE FOR EACH HALF SEPARATELY
18141                                   ;LET FWDSEQ = #1
18142                                   ;LET ADDLSB = #1
18143                                   ;DO UNTIL ADDLSB EQ #20000
18144                                   ;.      LET NEWDAT = 22200
18145                                   ;.      LET CURDAT = 0
18146                                   ;.      IF FWDSEQ = #1 THEN
18147                                   ;.      .       LET FSTADD EQUAL FIRST ADDRESS OF 4K BYTE BUFFER
18148                                   ;.      .       LET LASTAD EQUAL LAST ADDRESS OF 4K BYTE BUFFER
18149                                   ;.      ELSE
18150                                   ;.      .       LET FSTADD EQUAL LAST ADDRESS OF 4K BYTE BUFFER
18151                                   ;.      .       LET LASTAD EQUAL FIRST ADDRESS OF 4K BYTE BUFFER
18152                                   ;.      ENDIF
18153                                   ;.      LET DCOUNT = #0
18154                                   ;.      DO UNTIL DCOUNT EQ #10
18155                                   ;.      .       READ CURADD USING CURDAT AS PAR
18156                                   ;.      .       IF MISS THEN
18157                                   ;.      .       .       ERROR
18158                                   ;.      .       ENDIF
18159                                   ;.      .       WRITE CURADD USING NEWDAT AS PAR
18160                                   ;.      .       ENDIF
18161                                   ;.      .       IF HIT THEN
18162                                   ;.      .       .       ERROR
18163                                   ;.      .       ENDIF
18164                                   ;.      .       READ CURADD USING NEWDAT AS PAR
18165                                   ;.      .       IF MISS THEN
18166                                   ;.      .       .       ERROR
18167                                   ;.      .       ENDIF
18168                                   ;.      .       IF CURADD EQ LASTAD THEN
18169                                   ;.      .       .       LET CURDAT = NEWDAT
18170                                   ;.      .       .       IF DCOUNT < #1 THEN
18171                                   ;.      .       .       .       LET NEWDAT = NEWDAT SETBY RITEDA ROTATED LEFT
18172                                   ;.      .       .       ENDIF
18173                                   ;.      .       .       IF DCOUNT > #1 THEN
18174                                   ;.      .       .       .       LET NEWDAT = NEWDAT CLR BY RITEDA ROTATED LEFT
18175                                   ;.      .       .       ELSE
18176                                   ;.      .       .       .       LET NEWDAT = #155400 (NO ALL 1'S)
```

```
                                        ;.      .       .       .       LET RITEDA = #22200
                                        ;.      .       .       ENDIF
                                        ;.      .       .       INCREMENT DCOUNT
                                        ;.      .       ELSE
                                        ;.      .       .       IF FWDSEQ = #1 THEN
                                        ;.      .       .       .       LET CURADD = CURADD + ADDLSB
                                        ;.      .       .       .       IF CURADD GE HIGH ADDRESS THEN
                                        ;.      .       .       .       .       LET CURADD = CURADD + #7776
                                        ;.      .       .       .       ENDIF
                                        ;.      .       .       ELSE
                                        ;.      .       .       .       LET CURADD = CURADD - ADDLSB
                                        ;.      .       .       .       IF CURADD LE LOW ADDRESS THEN
                                        ;.      .       .       .       .       LET CURADD = CURADD - #7776
                                        ;.      .       .       .       ENDIF
                                        ;.      .       .       ENDIF
                                        ;.      .       ENDIF
                                        ;.      ENDDO
                                        ;.      IF FWDSEQ EQ #1 THEN
                                        ;.      .       LET FWDSEQ = #0
                                        ;.      ELSE
                                        ;.      .       ROTATE ADDLSB
                                        ;.      .       LET FWDSEQ = #1
                                        ;.      ENDIF
                                        ;ENDDO
                                        ;RESTORE PARITY ABORT VECTOR
                                        ;ENDTST

                                        ;;*****************************************************************
110526  000004                          TST27:  SCOPE
110530  032777  000100  070402                  BIT     #BIT06,@SWR             ;RUN THIS TEST?
110536  001002                          100$:   BNE     100$                    ;IF SET, GO DO IT
110540  000137  111630                          JMP     ENDTAG                  ;OTHERWISE, GO TO NEXT TEST
110544  042737  001000  177520                  BIC     #1000,@CSR              ;DISABLE HALT ON BREAK
110552  004737  132156                          JSR     PC,     INITMM          ;SETUP MEMORY MANAGEMENT
110556  005237  177572                          INC     SR0                     ;TURN ON MMU
110562  012737  000020  172516                  MOV     #BIT04,MMR3             ;ENABLE 22-BIT MAPPING
110570  052737  000002  177746                  BIS     #2,     CCR             ;SET DIAG. BIT
110576  013737  000004  001160                  MOV     @#4,    $TMPO           ;STORE TIMEOUT VECTOR
                                        ;
                                        ;STORE TEST IN THE FIRST 2K AND THEN IN THE SECOND 2K
                                        ;
110604  012703  140000                  2$:     MOV     #140000,R3              ;START FOR THE TEST
110610  012704  130000                          MOV     #130000,R4              ;LOWER BOUNDARY TEST AREA
110614  012705  137776                          MOV     #137776,R5              ;HIGH BOUNDARY TEST AREA
110620  000406                          1$:     BR      2$
110622  012703  150000                          MOV     #150000,R3              ;START OF THE TEST
110626  012704  120000                          MOV     #120000,R4              ;LOWER BOUNDARY TEST AREA
110632  012705  127776                          MOV     #127776,R5              ;HIGH BOUNDARY
110636  012737  002000  172354                  MOV     #2000,KIPAR6            ;ABOVE 32K
110644  012702  110470                          MOV     #EXITST,R2              ;START WITH CURRENT
110650  010300                          3$:     MOV     R3,R0                   ;MOVE TO UPPER 4K
110652  012220                                  MOV     (R2)+,(R0)+             ;WORD BY WORD
110654  022702  111630                          CMP     #ENDTAG,R2              ;ALL DONE?
110660  001374                                  BNE     3$                      ;
110662  012700  110764                          MOV     #STMOVT,R0              ;ADDRESS OF ROUTINE
110666  162700  110470                          SUB     #EXITST,R0              ;PROPER OFSET
```

```
18253  110672  050003                          BIS     R0,R3                   ;
18254  110674  004713                          JSR     PC,(R3)                 ;GO DO THE ROUTINE
18255  110676  005700                          TST     R0                      ;ANY ERRORS?
18256  110700  001407                          BEQ     4$                      ;CONTINUE
18257  110702  010037  001122                  MOV     R0,$BDADR               ;STORE FAILED ADDRESS
18258  110706  042737  160000  001122          BIC     #160000,$BDADR          ;CLEAR PAR
18259  110714  104050                          ERROR   .50
18260  110716  000403                          BR      5$                      ;EXIT TEST
18261  110720  022703  150000          4$:     CMP     #150000,R3              ;DONE FOR BOTH HALVES?
18262  110724  103736                          BLO     1$                      ;IF NOT, DO AGAIN
18263  110726  013737  001160  000004  5$:     MOV     $TMP0,  @#4             ;RESTORE TIMEOUT VECTOR
18264  110734  012737  000400  177746          MOV     #400,   CCR             ;INIT CCR FOR EXIT
18265  110742  005037  177572                  CLR     SR0                     ;TURN OFF MMU
18266  110746  005037  172516                  CLR     MMR3                    ;
18267  110752  052737  001000  177520          BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
18268  110760  000137  111630                  JMP     ENDTAG                  ;EXIT TEST
18269
18270                                  .DSABL AMA
18271  110764  052737  000400  177520  STMOVT: BIS     #BIT08,@#BCSR           ;STANDALONE MODE
18272  110772  042737  100000  172312          BIC     #BIT15, @#KIPDR5        ;NO BYPASS
18273  111000  042737  100000  172300          BIC     #100000,@#KIPDR0        ;NO BYPASS
18274  111006  012737  172360  000004          MOV     #KDPAR0,@#4             ;ALLOCATE TIMEOUT VECTOR
18275  111014  012737  000340  000006          MOV     #340,   @#6             ;AT 7
18276  111022  012737  000006  172360          MOV     #6,     @#KDPAR0        ;PUT RETURN
18277  111030  052737  100000  172300          BIS     #100000,@#KIPDR0        ;BYPASS
18278  111036  005037  172352                  CLR     @#KIPAR5                ;
18279  111042  010400                          MOV     R4,R0                   ;CLEAR CACHE UNDER TEST
18280  111044  012701  004000                  MOV     #4000,  R1              ;
18281  111050  005020                  4$:     CLR     (R0)+                   ;
18282  111052  077102                          SOB     R1,     4$              ;
18283  111054  005000                          CLR     R0                      ;CLEAR ERROR FLAG
18284  111056  012767  000001  177422          MOV     #1,     FWDSEQ          ;SET UPWARD ADDRESSING INDICATOR
18285  111064  012767  000002  177416          MOV     #2,     ADDLSB          ;INITIALIZE LEAST SIGNIFIGANT BIT
18286  111072  005067  177420          TSLOOP: CLR     CURDAT                  ;OLD DATA
18287  111076  012767  022200  177410          MOV     #22200,NEWDAT           ;SET ADDRESS BITS
18288  111104  012767  022200  177400          MOV     #22200,RITEDA           ;SET ADDRESS BITS
18289  111112  005767  177370                  TST     FWDSEQ                  ;IF ADDRESSING UPWARD
18290  111116  001405                          BEQ     1$                      ;THEN
18291  111120  010467  177374                  MOV     R4,     FSTADD          ;FIRST ADDRESS WILL BE LOWEST ADDRESS
18292  111124  010567  177372                  MOV     R5,     LSTADD          ;AND LAST ADDRESS WILL BE HIGHEST
18293  111130  000404                          BR      2$                      ;ELSE
18294  111132  010567  177362          1$:     MOV     R5,     FSTADD          ;FIRST ADDRESS WILL BE HIGHEST ADDRESS
18295  111136  010467  177360                  MOV     R4,     LSTADD          ;AND LAST ADDRESS WILL BE LOWEST
18296  111142  005067  177332          2$:     CLR     DCOUNT                  ;INITIALIZE LOOP COUNTER
18297  111146  016767  177346  177350          MOV     FSTADD, CURADD
18298                                  ;
18299                                  ; DON'T REWRITE TIMEOUT VECTOR
18300                                  ;
18301  111154  016700  177344          3$:     MOV     CURADD, R0              ;STORE CURRENT ADDRESS
18302  111160  042700  170000                  BIC     #170000,R0              ;LEAVE ONLY LOW 4K BITS
18303  111164  022700  000004                  CMP     #4,     R0              ;TIMOUT VECTOR?
18304  111170  001526                          BEQ     16$                     ;IF SO, DON'T REWRITE IT
18305  111172  022700  000006                  CMP     #6,     R0              ;OR PRIORITY
18306  111176  001523                          BEQ     16$                     ;
18307  111200  016737  177312  172352          MOV     CURDAT, @#KIPAR5        ;GET OLD PATTERN
18308  111206  005777  177312                  TST     @CURADD                 ;OLD DATA OK?
```

```
18289  111212  013767  177752  177264         MOV     @#HITMIS,RECDAT
18290  111220  032767  000004  177256  5$:    BIT     #BIT02, RECDAT          ;IF ACCESS WAS A MISS
18291  111226  001002                         BNE     6$                      ;THEN
18292  111230  000167  000346                 JMP     EXBAD2                  ;ERROR, EXIT
18293  111234  016737  177254  172352  6$:    MOV     NEWDAT, @#KIPAR5        ;GET NEW PATTERN
18294  111242  005077  177256                 CLR     @CURADD                 ;REGISTER AND WRITE LOCATION
18295  111246  013767  177752  177230         MOV     @#HITMIS,RECDAT
18296  111254  032767  000004  177222  8$:    BIT     #BIT02, RECDAT          ;IF ACCESS WAS A HIT
18297  111262  001406                         BEQ     9$                      ;THEN
18298  111264  032767  000010  177212  10$:   BIT     #BIT03, RECDAT          ;MISS?
18299  111272  001402                         BEQ     9$                      ;IF SO, CONTINUE
18300  111274  000167  000302                 JMP     EXBAD2                  ;ERROR
18301  111300  005777  177220          9$:    TST     @CURADD                 ;REGISTER AND READ LOCATION
18302  111304  013767  177752  177172         MOV     @#HITMIS,RECDAT
18303  111312  032767  000004  177164  11$:   BIT     #BIT02, RECDAT          ;IF ACCESS WAS A MISS
18304  111320  001002                         BNE     12$                     ;THEN
18305  111322  000167  000254                 JMP     EXBAD2                  ;ERROR, EXIT
18306  111326  026767  177170  177170  12$:   CMP     LSTADD, CURADD          ;IF CURRENT ADDRESS IS LAST ADDRESS
18307  111334  001044                         BNE     16$                     ;THEN
18308  111336  016767  177152  177152         MOV     NEWDAT,CURDAT           ;NEW KIPAR5
18309  111344  022767  000001  177126         CMP     #1,     DCOUNT          ;IF LOOP COUNTER LESS THAN 1
18310  111352  003406                         BLE     13$                     ;THEN
18311  111354  006367  177132                 ASL     RITEDA                  ;UPDATE OF NEW DATA....
18312  111360  056767  177126  177126         BIS     RITEDA, NEWDAT          ;BY SETTING BITS
18313  111366  000421                         BR      15$                     ;ENDIF
18314  111370  022767  000001  177102  13$:   CMP     #1,     DCOUNT          ;CHANGE PATTERN?
18315  111376  001007                         BNE     14$                     ;IF SO, BRANCH
18316  111400  012767  022200  177104         MOV     #22200, RITEDA          ;START WITH THE SAME
18317  111406  012767  155400  177100         MOV     #155400,NEWDAT          ;DON'T DO ALL 1'S
18318  111414  000406                         BR      15$                     ;
18319  111416  006367  177070          14$:   ASL     RITEDA                  ;UPDATE OF NEW DATA....
18320  111422  046767  177064  177064         BIC     RITEDA, NEWDAT          ;BY CLEARING BITS
18321  111430  000400                         BR      15$                     ;ELSE
18322  111432  005267  177042          15$:   INC     DCOUNT                  ;INCREMENT THE LOOP COUNTER
18323  111436  016767  177056  177060         MOV     FSTADD, CURADD          ;FINISH FIRST CURRENT ADDRESS
18324  111444  000426                         BR      18$                     ;ELSE
18325  111446  005767  177034          16$:   TST     FWDSEQ                  ;IF ADDRESSING UPWARD
18326  111452  001412                         BEQ     17$                     ;THEN
18327  111454  066767  177030  177042         ADD     ADDLSB, CURADD          ;ADD THE VALUE OF THE CURRENT LSB
18328  111462  020567  177036                 CMP     R5,     CURADD          ;IF CURRENT ADDRESS GREATER THAN HIGHEST
18329  111466  002015                         BGE     18$                     ;VIRTUAL ADDRESS THEN
18330  111470  162767  007776  177026         SUB     #7776,  CURADD          ;ROLL IT BACK
18331  111476  000411                         BR      18$                     ;ELSE
18332  111500  166767  177004  177016  17$:   SUB     ADDLSB, CURADD          ;IF ADDRESSING DOWNWARD THEN SUBTRACT LSB
18333  111506  020467  177012                 CMP     R4,     CURADD          ;IF CURRENT ADDRESS LESS THEN LOWEST
18334  111512  003403                         BLE     18$                     ;VIRTUAL ADDRESS THEN
18335  111514  062767  007776  177002         ADD     #7776,  CURADD          ;ROLL IT BACK
18336  111522  022767  000005  176750  18$:   CMP     #5,     DCOUNT          ;IF LOOP COUNTER LESS THAN 7
18337  111530  003402                         BLE     181$                    ;THEN LOOP BACK FOR NEXT BIT POSITION
18338  111532  000167  177416                 JMP     3$
18339  111536  005767  176744          181$:  TST     FWDSEQ                  ;IF ADDRESSING UPWARD
18340  111542  001404                         BEQ     19$                     ;THEN
18341  111544  005067  176736                 CLR     FWDSEQ                  ;START ADDRESSING DOWNWARD
18342  111550  000167  177316                 JMP     TSLOOP                  ;ELSE
18343  111554  012767  000001  176724  19$:   MOV     #1,     FWDSEQ          ;START ADDRESSING UPWARD
18344  111562  006167  176722                 ROL     ADDLSB                  ;ROTATE LSB TO NEXT POSITION
```

```
 18345  111566  022767  020000  176714          CMP     #20000, ADDLSB          ;ALL DONE?
 18346  111574  001406                          BEQ     TSEND                   ;EXIT
 18347  111576  000167  177270                  JMP     TSLOOP                  ;ENDDO
 18348  111602  013701  172352          EXBAD2: MOV     @#KIPAR5,R1             ;STORE PAR
 18349  111606  016700  176712                  MOV     CURADD, R0              ;AND ADDRESS
 18350  111612  012737  001200  172352  TSEND:  MOV     #1200,  @#KIPAR5        ;RESTORE PAR
 18351  111620  042737  000400  177520          BIC     #BIT08, @#BCSR          ;OUT OF STANDALONE MODE
 18352  111626  000207                          RTS     PC                      ;RETURN
 18353  111630                          ENDTAG:
 18354                                  .ENABL AMA
 18355
```

```
18356                                  .SBTTL  TEST  - PCR READ/WRITE BITS
18357                                  ;PCR AND BCSR READ/WRITE BITS
18358                                  ;THE FIRST TEST WILL CHECK THAT PCR REGISTER IS BOTH WORD AND
18359                                  ;BYTE ADDRESSABLE. BITS 14-09 AND 06-01 WILL BE WRITTEN AND READ
18360                                  ;AS ZEROES AND ONES. THE REST OF THE BITS HAVE TO BE ALL 0'S.
18361                                  ;ROUTINE TEST
18362                                  ;.      SAVE PCR
18363                                  ;.      LET PCR=0
18364                                  ;.      DO FOR PATTERN=001111,110011,101010,010101
18365                                  ;.      .       WRITE PCR<14-09>=PATTERN
18366                                  ;.      .       WRITE PCR<06-01>=PATTERN
18367                                  ;.      .       IF PCR<14-09> NE PATTERN OR PCR<06-01> NE PREVIOUS
18368                                  ;.      .                                               PATTERN
18369                                  ;.      .       .       THEN ERROR
18370                                  ;.      .       ENDIF
18371                                  ;.      .       WRITE PCR<06-01>=PATTERN
18372                                  ;.      .       IF PCR<14-09> NE PATTERN OR PCR<06-01> NE PATTERN
18373                                  ;.      .       .       THEN ERROR
18374                                  ;.      .       ENDIF
18375                                  ;.      ENDDO
18376                                  ;.      WRITE PCR=0101010101010101
18377                                  ;.      IF PCR NE 0101010001010100 THEN
18378                                  ;.      .       ERROR
18379                                  ;.      ENDIF
18380                                  ;ENDROUTINE
18381
18382                                  ;;*****************************************************************
18383  111630  000004                 TST30:  SCOPE
18384  111632  005037  177522                 CLR     PCR                     ;INITIALIZE PCR TO 0
18385  111636  012702  111775                 MOV     #SIXBIT+1,R2            ;R2->TABLE OF PATTERNS FOR 6 R/W BITS IN
18386  111642  012704  111774                 MOV     #SIXBIT,R4              ;R3 POINTER TO PREVIOUS PATTERN
18387  111646  012703  000004                 MOV     #4,R3                   ;DO 4 TIMES
18388                                  ;
18389                                  ; WRITE TO HIGH BYTE FIRST
18390                                  ;
18391  111652  111237  177523         1$:     MOVB    (R2),PCR+1              ;WRITE TO HIGH BYTE
18392  111656  121237  177523                 CMPB    (R2),PCR+1              ;BYTE WRITTEN OK?
18393  111662  001003                         BNE     2$                      ;IF NOT, BRANCH
18394  111664  121437  177522                 CMPB    (R4),PCR                ;LOW BYTE CHANGED?
18395  111670  001405                         BEQ     3$                      ;IF NOT, BRANCH
18396  111672  111237  001125         2$:     MOVB    (R2),$GDDAT+1           ;EXPECTED PATTERN HIGH BYTE
18397  111676  111437  001124                 MOVB    (R4),$GDDAT             ;LOW BYTE
18398  111702  104051                         ERROR   +51                     ;ERROR PCR READ/WRITE BITS
18399  111704  105724                 3$:     TSTB    (R4)+                   ;INCREMENT POINTER FOR OLD
18400                                  ;
18401                                  ; WRITE TO LOW BYTE
18402                                  ;
18403  111706  111237  177522                 MOVB    (R2),PCR                ;WRITE TO LOW BYTE
18404  111712  121237  177522                 CMPB    (R2),PCR                ;BYTE WRITTEN OK?
18405  111716  001003                         BNE     4$                      ;IF NOT,BRANCH
18406  111720  121237  177522                 CMPB    (R2),PCR                ;HIGH BYTE CHANGED?
18407  111724  001405                         BEQ     5$                      ;IF NOT, BRANCH
18408  111726  111237  001124         4$:     MOVB    (R2),$GDDAT             ;EXPECTED PATTERN HIGH BYTE
18409  111732  111237  001124                 MOVB    (R2),$GDDAT             ;LOW BYTE
18410  111736  104051                         ERROR   +51                     ;ERROR PCR READ/WRITE BITS
18411  111740  105722                 5$:     TSTB    (R2)+                   ;INCREMENT POINTER FOR NEW PATTERN
```

H11


```
111742  077335                          SOB     R3,1$                           ;DO FOR ALL 4 PATTERNS
                                ;
                                ; NOW TRY WORD ADDRESSING
                                ;
111744  012737  052525  177522          MOV     #52525,PCR                      ;WRITE A PATERN
111752  022737  052124  177522          CMP     #52124,PCR                      ;ALL BUT BITS <8,7,0> OK?
111760  001404                          BEQ     6$                              ;IF SO, BRANCH
111762  112737  052124  001124          MOVB    #52124,$GDDAT                   ;EXPECTED PATTERN
111770  104051                          ERROR   +51                             ;ERROR PCR READ/WRITE BITS
111772                          6$:
111772  000403                          BR      TST31           ;;GO TO NEXT TEST

111774     000     036     146  SIXBIT: .BYTE 0,36,146,124,52                   ;001111,110011,101010,010101
111777     124     052
                                                                                ;BINARY DIVIDE FOR 6 BITS
        112002                          .EVEN
```

```
                                .SBTTL  TEST  - BCSR READ/WRITE BITS
                                ;THE SECOND TEST WILL CHECK THAT BCSR<7-5;2-0> CAN BE WRITTEN AND
                                ;READ AS ZEROES AND ONES.  BCSR<14,03> SHOULD BE 0'S.  BCSR<04>
                                ;SHOULD BE CLEARED BY RESET INSTRUCTION.
                                ;ROUTINE TEST
                                ;.      FOR PATTERN=011,010,101 DO
                                ;.      .       WRITE BCSR<7-5>=PATTERN
                                ;.      .       IF BCSR<7-5> NE PATTERN THEN
                                ;.      .       .       ERROR
                                ;.      .       ENDIF
                                ;.      .       WRITE BCSR<2-0>=PATTERN
                                ;.      .       IF BCSR<2-0> NE PATTERN THEN
                                ;.      .       .       ERROR
                                ;.      .       ENDIF
                                ;.      ENDDO
                                ;.      IF BCSR<14,03> NE <0,0> THEN
                                ;.      .       ERROR
                                ;.      ENDIF
                                ;.      LET BCSR<04>=1
                                ;.      IF BCSR<04> NE #1 THEN
                                ;.      .       ERROR
                                ;.      ENDIF
                                ;.      EXECUTE "RESET"
                                ;.      IF BCSR<04> NE 0 THEN
                                ;.      .       ERROR
                                ;.      ENDIF
                                ;.      LET BCSR<04>=0 (THIS BIT IS WRITE ENABLE FOR EAROM)
                                ;ENDROUTINE

                                ;;**********************************************************************
112002  000004                  TST31:  SCOPE
112004  013737  177520  002710          MOV     BCSR,SAVBR              ;SAVE BCSR
112012  005037  177520                  CLR     BCSR                    ;CLEAR BCSR
                                ;
                                ; WRITE TO BITS <7-5> AND <2-0>
                                ;
112016  012703  112252                  MOV     #THRBIT,R3              ;POINTER FOR PATTERN TABLE
112022  005037  001124                  CLR     $GDDAT                  ;CLEAR A LOCATION
112026  012702  000003                  MOV     #3,R2                   ;DO FOR ALL PATTERNS
112032  111337  177520          1$:     MOVB    (R3),BCSR               ;WRITE TO BITS <7-5>
112036  111337  001124                  MOVB    (R3),$GDDAT             ;EXPECTED PATTERN
112042  122337  177520                  CMPB    (R3)+,BCSR              ;BITS WRITTEN OK?
112046  001401                          BEQ     2$                      ;IF SO, BRANCH
112050  104052                          ERROR   +52                     ;ERROR IN BCSR READ/WRITE BITS
112052  111337  177520          2$:     MOVB    (R3),BCSR               ;WRITE TO BITS <2-0>
112056  111337  001124                  MOVB    (R3),$GDDAT             ;EXPECTED PATTERN FOR ERRORS
112062  122337  177520                  CMPB    (R3)+,BCSR              ;BITS WRITTEN OK?
112066  001401                          BEQ     3$                      ;IF SO, BRANCH
112070  104052                          ERROR   +52                     ;ERROR IN BCSR READ/WRITE BITS
112072  077221          3$:     SOB     R2,1$                           ;CONTINUE TILL ALL PATTERNS DONE
                                ;
                                ; CHECK UNUSED BITS <3>
                                ;
112074  012737  000010  177520          MOV     #10,BCSR                ;WRITE TO BIT <3>
112102  032737  000010  177520          BIT     #BIT03,BCSR             ;ALL ZEROES?
112110  001403                          BEQ     4$                      ;IF YES, BRANCH
```

```
18485  112112  005037  001124              CLR     $GDDAT                          ;EXPECTED PATTERN
18486  112116  104052                      ERROR   +52                            ;ERROR IN BCSR READ/WRITE BITS
18487                                  ;
18488                                  ; CHECK THAT BIT <4> CLEARS BY RESET
18489                                  ;
18490  112120  052737  000020  177520  4$:     BIS     #BIT04,BCSR                 ;SET BIT 4
18491  112126  032737  000020  177520          BIT     #BIT04,BCSR                 ;WRITTEN OK?
18492  112134  001005                      BNE     5$                              ;IF SO BRANCH
18493  112136  012737  000020  001124          MOV     #BIT04,$GDDAT               ;EXPECTED PATTERN
18494  112144  104052                      ERROR   +52                            ;ERROR IN BCSR READ/WRITE BITS
18495  112146  000421                      BR      7$                              ;EXIT TEST
18496  112150  042737  000020  177520  5$:     BIC     #BIT04,BCSR                 ;TRY TO CLEAR BIT 4
18497  112156  032737  000020  177520          BIT     #BIT04,BCSR                 ;CLEARED OK?
18498  112164  001403                      BEQ     6$                              ;IF SO BRANCH
18499  112166  005037  001124              CLR     $GDDAT                          ;EXPECTED PATTERN
18500  112172  104052                      ERROR   +52                            ;ERROR IN BCSR READ/WRITE BITS
18501  112174  122737  000001  001220  6$:     CMPB    #APTENV,$ENV                ;RUNNING IN APT MODE?
18502  112202  001003                      BNE     7$                              ;NO, GO DO TEST
18503  112204  005737  001206              TST     $PASS                           ;FIRST PASS?
18504  112210  001014                      BNE     8$                              ;IF APT AND NOT FIRST PASS,EXIT
18505  112212  052737  000020  177520  7$:     BIS     #BIT04,BCSR                 ;SET BIT 4 AGAIN
18506  112220  000005                      RESET                                   ;EXECUTE RESET
18507  112222  032737  000020  177520          BIT     #BIT04,BCSR                 ;BIT 4 CLEARED?
18508  112230  001404                      BEQ     8$                              ;IF YES, BRANCH
18509  112232  104053                      ERROR   +53                            ;RESET DOESN'T CLEAR BCSR<4>
18510  112234  042737  000020  177520          BIC     #BIT04,BCSR                 ;CLEAR BIT 4
18511  112242  013737  002710  177520  8$:     MOV     SAVBR,BCSR                  ;RESTORE BCSR
18512  112250  000403                      BR      TST32           ;;GO TO NEXT TEST
18513
18514  112252      140     003     100  THRBIT: .BYTE   140,3,100,2,240,5           ;011,010,101 FOR BITS <7-5,2-0>
18515  112255      002     240     005
18516
```

```
18517                                  .SBTTL  TEST  - 16 BIT ROM CHECKSUM TEST
18518                                  ;ROM'S CHECKSUMS
18519                                  ;
18520                                  ;16 BIT ROM TEST
18521                                  ;THE FIRST TEST WILL CLEAR BCSR<07>, LOAD PCR<14-09> WITH
18522                                  ;ROM ADDRESS BITS <14-09>, AND CHECK CHECKSUMS OF 16-BIT ROM
18523                                  ;BY ACCESSING IT THRU BUS ADDRESSES 173000-173776.  THEN WITH
18524                                  ;BCSR<06;05> BOTH CLEAR, PCR<06-01> USED AS ADDRESS BITS 14-09,
18525                                  ;THE SAME THING WILL BE DONE BY ADDRESSING 16-BIT ROM THRU BUS ADDRESSES
18526                                  ;165000-165776. THE RESULTS SHOULD BE THE SAME AND SHOULD COMPARE
18527                                  ;WITH THAT STORED IN THE BOOT AND DIAGNOSTIC ROM.
18528                                  ;
18529                                  ;BCSR   <07>    DISABLE 17773000
18530                                  ;       <06>    DISABLE 17765000
18531                                  ;       <05>    ROM SOCKET 3 AT 17765000
18532                                  ;
18533                                  ;ROUTINE TEST
18534                                  ;.      LET BCSR<7,6,5>=0,0,0
18535                                  ;.      DO FOR R1 FROM #0 TO #31. BY #1 DO
18536                                  ;.      .       LET PCR<14-09>=R1
18537                                  ;.      .       DO FOR R2 FROM #0 TO #776 BY 2
18538                                  ;.      .       .       CALCULATE CHECKSUM THRU 173000
18539                                  ;.      .       ENDDO
18540                                  ;.      ENDDO
18541                                  ;.      IF CHECKSUM NE #0 THEN
18542                                  ;.      .       ERROR
18543                                  ;.      ENDIF
18544                                  ;.      DO FOR R1 FROM #0 TO #31. BY #1
18545                                  ;.      .       LET PCR<06-01>=R1
18546                                  ;.      .       DO FOR R2 FROM #0 TO #776 BY 2
18547                                  ;.      .       .       CALCULATE CHECKSUM THRU 165000
18548                                  ;.      .       ENDDO
18549                                  ;.      ENDDO
18550                                  ;.      IF CHECKSUM NE #0 THEN
18551                                  ;.      .       ERROR
18552                                  ;.      ENDIF
18553                                  ;ENDROUTINE
18554
18555                                  ;;*****************************************************************************
18556  112260  000004                 TST32:  SCOPE
18557  112262  013737  177520  002710         MOV     BCSR,SAVBR                      ;SAVE BCSR
18558  112270  042737  000340  177520         BIC     #BIT07!BIT06!BIT05,BCSR         ;READ 16 BIT ROM
18559                                  ;
18560                                  ; CALCULATE LOW BYTE CHECKSUM'S THRU 173000 AND 165000
18561                                  ;
18562  112276  005001                         CLR     R1                              ;PAGE COUNT FOR ALL 8K
18563  112300  005037  177522                 CLR     PCR                             ;CLEAR PAGE CONTROL REGISTER
18564  112304  005037  002704         1$:     CLR     ACTCHS                          ;CLEAR CHECKSUM AT 173000
18565  112310  005037  001160                 CLR     $TMP0                           ;AT 165000
18566  112314  005002                         CLR     R2                              ;CLEAR COUNTER THRU A PAGE
18567  112316  016203  173000         2$:     MOV     173000(R2),R3                   ;GET LOW BYTE THRU 173000
18568  112322  016204  165000                 MOV     165000(R2),R4                   ;THRU 165000
18569  112326  060337  002704                 ADD     R3,ACTCHS                       ;CALCULATE CHECKSUM THRU 173000
18570  112332  060437  001160                 ADD     R4,$TMP0                        ;CALCULATE CHEKCSUM THRU 165000
18571  112336  005722                         TST     (R2)+                           ;GET NEXT WORD
18572  112340  022702  000776                 CMP     #776,R2                         ;WORD BEFORE LAST?
```

```
18573  112344  001004                        BNE     3$              ;IF NOT, BRANCH
18574  112346  020337  177522                CMP     R3,PCR          ;PAGE NUMBER STORED OK?
18575  112352  001401                        BEQ     3$              ;IF YES, BRANCH
18576  112354  104054                        ERROR   .54             ;PAGE NUMBER STORED WRONG
18577  112356  022702  001000          3$:   CMP     #1000,R2        ;LAST WORD IN A PAGE?
18578  112362  003355                        BGT     2$              ;IF NOT, BRANCH
18579  112364  005737  002704                TST     ACTCHS          ;CHECKSUM 0?
18580  112370  001401                        BEQ     4$              ;IF YES, BRANCH
18581  112372  104054                        ERROR   .54             ;IN CHECKSUM AT 173000
18582  112374  005737  001160          4$:   TST     $TMP0           ;CHECKSUM 0 AT 165000?
18583  112400  001401                        BEQ     5$              ;IF YES, BRANCH
18584  112402  104054                        ERROR   .54             ;IN CHECKSUM AT 165000
18585  112404  062701  000002          5$:   ADD     #2,R1           ;GET NEW PAGE
18586  112410  110137  177522                MOVB    R1,PCR          ;STORE IN PCR<6-1>
18587  112414  110137  177523                MOVB    R1,PCR+1        ;STORE IN PCR<14-9>
18588  112420  022701  000160                CMP     #160,R1         ;ALL  PAGES IN R2 FROM BIT1?
18589  112424  003327                        BGT     1$              ;IF NOT BRANCH
18590                                  ;
18591                                  ; CHECK THE LAST 2K OF ASCII TEXT
18592                                  ;
18593  112426  005037  002704          LASTCH: CLR     ACTCHS          ;CLEAR CHECKSUM AT 173000
18594  112432  005037  001160                  CLR     $TMP0           ;AT 165000
18595  112436  005002                  1$:     CLR     R2              ;CLEAR COUNTER THRU A PAGE
18596  112440  016203  173000          2$:     MOV     173000(R2),R3   ;GET LOW BYTE THRU 173000
18597  112444  016204  165000                  MOV     165000(R2),R4   ;THRU 165000
18598  112450  060337  002704                  ADD     R3,ACTCHS       ;CALCULATE CHECKSUM THRU 173000
18599  112454  060437  001160                  ADD     R4,$TMP0        ;CALCULATE CHEKCSUM THRU 165000
18600  112460  005722                          TST     (R2)+           ;GET NEXT WORD
18601  112462  022702  001000          3$:     CMP     #1000,R2        ;LAST WORD IN A PAGE?
18602  112466  003364                          BGT     2$              ;IF NOT, BRANCH
18603  112470  022701  000176                  CMP     #176,R1         ;LAST PAGE?
18604  112474  001005                          BNE     4$              ;IF NOT, BRANCH
18605  112476  023737  173774  177522          CMP     173774,PCR      ;PROPER PAGE NUMBER?
18606  112504  001401                          BEQ     4$              ;IF YES, BRANCH
18607  112506  104054                          ERROR   .54             ;ERROR IN ROM
18608  112510  062701  000002          4$:     ADD     #2,R1           ;GET NEW PAGE
18609  112514  110137  177522                  MOVB    R1,PCR          ;STORE IN PCR<6-1>
18610  112520  110137  177523                  MOVB    R1,PCR+1        ;STORE IN PCR<14-9>
18611  112524  022701  000200                  CMP     #200,R1         ;ALL  PAGES IN R2 FROM BIT1?
18612  112530  003342                          BGT     1$              ;IF NOT BRANCH
18613  112532  005737  002704                  TST     ACTCHS          ;CHECKSUM 0?
18614  112536  001401                          BEQ     5$              ;IF YES, BRANCH
18615  112540  104054                          ERROR   .54             ;IN CHECKSUM AT 173000
18616  112542  005737  001160          5$:     TST     $TMP0           ;CHECKSUM 0 AT 165000?
18617  112546  001401                          BEQ     6$              ;IF YES, BRANCH
18618  112550  104054                          ERROR   .54             ;IN CHECKSUM AT 165000
18619  112552  005037  177522          6$:     CLR     PCR             ;CLEAR PCR
18620  112556  013737  177520  002710          MOV     BCSR,SAVBR      ;RESTORE BCSR
18621
18622
18623
```

```
18624                                   .SBTTL  TEST  - 8 BIT ROM CRC TEST
18625                                   ;THIS TEST WILL CLEAR BCSR<6>, SET BCSR<5>, LOAD PCR<5-1>
18626                                   ;WITH ADDRESS BITS 13-09, AND CHECK CRC PATTERNS OF 8-BIT ROM BY
18627                                   ;ACCESSING IT THRU ADDRESSES 165000-165776. THIS TEST VERIFIES THE
18628                                   ;RESPONSE ONLY OF THE BASE AREA.
18629                                   ;ROUTINE TEST
18630                                   ;.      LET BCSR<5>=1
18631                                   ;.      LET OLDCRC=#0
18632                                   ;.      LET PCR<05-01>=#0
18633                                   ;.      CALCULATE CHECKSUM FOR THE FIRST 320 LOCATIONS
18634                                   ;.      IF RESULTING CHECKSUM NOT ZERO THEN
18635                                   ;.          .       ERROR
18636                                   ;.      ENDIF
18637                                   ;ENDROUITNE
18638
18639                                   ;;*****************************************************************
18640  112564  000004                  TST33:  SCOPE
18641  112566  013737  177520  002710          MOV     BCSR,SAVBR                      ;SAVE BCSR
18642  112574  042737  000100  177520          BIC     #BIT06,BCSR                     ;ENABLE INTERNAL RESPONSE
18643  112602  052737  000040  177520          BIS     #BIT05,BCSR                     ;SELECT 8-BIT ROM
18644  112610  005037  001160                  CLR     $TMP0                           ;CLEAR SUM
18645                                  ;
18646                                  ; CALCULATE LOW BYTE CHECKSUM'S THRU 165000
18647                                  ;
18648  112614  005001                  1$:     CLR     R1                              ;PAGE COUNT FOR ALL 8K
18649  112616  005037  177522                  CLR     PCR                             ;CLEAR PAGE CONTROL REGISTER
18650  112622  005037  002704          2$:     CLR     ACTCHS                          ;CLEAR CHECKSUM AT 165000
18651  112626  005002                          CLR     R2                              ;CLEAR COUNTER THRU A PAGE
18652  112630  116204  165000          3$:     MOVB    165000(R2),R4                   ;GET A BYTE THRU 165000
18653  112634  060437  001160                  ADD     R4,$TMP0                        ;CALCULATE CHEKCSUM THRU 165000
18654  112640  005722                          TST     (R2)+                           ;GET NEXT WORD
18655  112642  022702  000316                  CMP     #316,R2                         ;WORD BEFORE LAST?
18656  112646  001004                          BNE     4$                              ;IF NOT, BRANCH
18657  112650  122704  000252                  CMPB    #252,R4                         ;314 SHOULD HAVE 252
18658  112654  001765                          BEQ     3$                              ;IF YES, BRANCH
18659  112656  104055                          ERROR   .55                             ;PAGE NUMBER STORED WRONG
18660  112660  022702  000322          4$:     CMP     #322,R2                         ;LAST WORD IN A PAGE?
18661  112664  003361                          BGT     3$                              ;IF NOT, BRANCH
18662  112666  113737  165010  001162          MOVB    165010,$TMP1                    ;STORE SIZE,<3>=1 2K
18663  112674  105737  001160                  TSTB    $TMP0                           ;CHECKSUM 0 AT 165000?
18664  112700  001401                          BEQ     5$                              ;IF YES, BRANCH
18665  112702  104055                          ERROR   .55                             ;IN CHECKSUM AT 165000
18666  112704  005037  177522          5$:     CLR     PCR
18667  112710  013737  002710  177520          MOV     SAVBR,BCSR                      ;RESTORE BCSR
18668
18669
```

```
                                .SBTTL  TEST  - LKS BIT 7
                                ;THESE TESTS WILL CHECK FUNCTIONALITY OF LKS<7-6>
                                ;
                                ;ACCESS LKS, CHECK THAT READY LKS<07> CAN BE 0 AND 1.
                                ;
                                ;LKS     <07>    LINE CLOCK MONITOR
                                ;        <06>    LINE CLOCK INTERRUPT ENABLE
                                ;ROUTINE TEST
                                ;IF UFD AND LKS IS DISABLED THEN
                                ;.       EXIT TEST
                                ;ENDIF
                                ;.       READ LKS TO SEE IF IT TIMEOUTS
                                ;.(CHECK LKS<07>)
                                ;.       LET R1=#77777(WORST CASE COUNTER FOR SLOW CLOCK)
                                ;.       LET 100=#ADDRESS OF LINE_CLOCK_INTERRUPT
                                ;.       CLEAR INTERRUPT_FLAG
                                ;.       DO 3 TIMES
                                ;.       .       REPEAT
                                ;.       .       .       DECREMENT R1
                                ;.       .       UNTIL R1 NE #0 OR LOW BYTE OF LKS LT #0
                                ;.       .       IF LOW BYTE OF LKS GE #0 THEN (READY DIDN'T COME UP)
                                ;.       .       .       ERROR
                                ;.       .       ENDIF
                                ;.       ENDDO
                                ;.       IF INTERRUPT_FLAG NE #0 THEN (INTERUPT W/O LKS<6>=1)
                                ;.       .       ERROR
                                ;.       ENDIF
                                ;ENDROUTINE

                                ;;*****************************************************************
112700  112716  000004          TST34:  SCOPE
112701  112720  032737  000100  000052          BIT     #BIT06,@#52                     ;UFD MODE?
112702  112726  001404                          BEQ     1$                              ;IF NOT, GO DO TEST
112703  112730  032737  010000  177520          BIT     #BIT12,BCSR                     ;LKS DISABLED?
112704  112736  001053                          BNE     TST35           ;;IF DISABLED, EXIT TEST
112705                                  ;
112706                                  ; TRY TO ACCESS LKS WITHOUT TIMEOUT
112707                                  ;
112708  112740  013737  000004  001160  1$:     MOV     ERRVEC,$TMP0                    ;SAVE TIMEOUT VECTOR
112709  112746  012737  112770  000004          MOV     #2$,ERRVEC                      ;POINT NEW VECTOR TO PROGRAM
112710  112754  012737  000340  000006          MOV     #340,ERRVEC+2                   ;AT PRIORITY 7
112711  112762  005737  177546                  TST     LKS                             ;ACCESS LKS
112712  112766  000403                          BR      3$                              ;IF NO TIMEOUT, CONTINUE
112713  112770  104056                  2$:     ERROR   +56                             ;TIMEOUT READING LKS
112714  112772  005726                          TST     (SP)+                           ;ADJUST STACK POINTER
112715  112774  005726                          TST     (SP)+
112716  112776  013737  001160  000004  3$:     MOV     $TMP0,ERRVEC                    ;RESTORE TIMEOUT VECTOR
112717                                  ;
112718                                  ; CHECK THAT READY BIT LKS<7> CAN BE 1
112719                                  ;
112720  113004  005037  002702                  CLR     LKSFL                           ;CLEAR INTERRUPT FLAG
112721  113010  012737  132444  000100          MOV     #LKSINT,100                     ;POINT VECTOR TO INERRUPT ROUTINE
112722  113016  012737  000340  000102          MOV     #340,102                        ;AT PRIORITY 7
112723  113024  012702  000003                  MOV     #3,R2                           ;DO 3 TIMES TO SYNCHRONISE
112724  113030  012701  077777          4$:     MOV     #77777,R1                       ;COUNTER FOR SLOW CLOCKS
112725  113034  105737  177546          5$:     TSTB    LKS                             ;READY LKS<7>=1?
```

```
18726  113040  100401                      BMI     6$                      ;IF SO, DO NEXT LOOP
18727  113042  077104                      SOB     R1,5$                   ;OTHERWISE, GO THRU COUNT
18728  113044  105737  177546      6$:     TSTB    LKS                     ;WAS READY 1?
18729  113050  100401                      BMI     7$                      ;IF YES, BRANCH
18730  113052  104057                      ERROR   •57                     ;LKS<07> DOES NOT BECOME 1
18731  113054  077213              7$:     SOB     R2,4$                   ;DO ALL 3 TIMES
18732  113056  005737  002702      8$:     TST     LKSFL                   ;ANY INTERRUPTS W/O LKS<6>=1?
18733  113062  001401                      BEQ     TST35           ;;IF NONE, EXIT TEST
18734  113064  104061                      ERROR   •61                     ;ILLEGAL CLOCK INTERRUPTS
18735
```

```
18736                                   .SBTTL  TEST  - LKS INTERRUPT PRIORITY
18737                                   ;CHECK THAT LKS INTERRUPTS HAPPEN AT PRIORITY 5 CLEARING LKS<07>
18738                                   ;AND DON'T HAPPEN AT PRIORITY 6.
18739                                   ;ROUTINE TEST
18740                                   ;IF UFD AND LKS IS DISABLED THEN
18741                                   ;.      EXIT TEST
18742                                   ;ENDIF
18743                                   ;.      SET PRIORITY TO 5
18744                                   ;.      CLEAR INTERRUPT_FLAG
18745                                   ;.      LET LKS<06>=#1 (ENABLE INTERRUPTS)
18746                                   ;.      SET COUNTER TO WAIT FOR 3 INTERRUPTS
18747                                   ;.      REPEAT
18748                                   ;.      .       DECREMENT COUNTER
18749                                   ;.      UNTIL INTERRUPT_FLAG EQ #3 OR COUNTER EQ #0
18750                                   ;.      CLEAR LKS<06>
18751                                   ;.      IF LKS<07> EQ #1 THEN
18752                                   ;.      .       ERROR (WAS NOT CLEARED ON INTERRUPT)
18753                                   ;.      ENDIF
18754                                   ;.      IF COUNTER LT TIME_REQUIRED_FOR_3_INTERRUPTS_FOR_800HZ
18755                                   ;.      .       ERROR (INTERRUPTS NEVER GO LOW)
18756                                   ;.      ENDIF
18757                                   ;.      IF INTERRUPT_FLAG LT #3 THEN
18758                                   ;.      .       ERROR (INTERRUPTS DON'T HAPPEN)
18759                                   ;.      ENDIF
18760                                   ;.      CLEAR INTERUPT_FLAG
18761                                   ;.      WAIT FOR LKS<7>=1
18762                                   ;.      LET LKS<7>=0
18763                                   ;.      IF LKS<7> NE #0 THEN
18764                                   ;.      .       ERROR (LKS<7> NOT CLEARED
18765                                   ;.      ENDIF
18766                                   ;.      SET PRIORITY TO 6
18767                                   ;.      SET COUNTER TO 1 SLOW CLOCK INTERRUPT
18768                                   ;.      SET LKS<06>
18769                                   ;.      REPEAT
18770                                   ;.      .       DECREMENT COUNTER
18771                                   ;.      UNTIL COUNTER EQ #0 OR INTERRUPT_FLAG NE #0
18772                                   ;.      IF INTERRUPT_FLAG NE #0 THEN
18773                                   ;.      .       ERROR (INTERRUPT WAS AT WRONG PRIORITY)
18774                                   ;.      ENDIF
18775                                   ;.      RESTORE ORIGINAL PRIORITY
18776                                   ;ENDROUTINE
18777                                   ;
18778                                   ;ROUTINE LINE_CLOCK_INTERUPT
18779                                   ;.      INCREMENT INTERRUPT_FLAG
18780                                   ;ENDROUTINE
18781
18782                                   ;;*****************************************************************
18783  113066  000004                  TST35:  SCOPE
18784  113070  032737  000100  000052          BIT     #BIT06,@#52                     ;UFD MODE?
18785  113076  001404                          BEQ     1$                              ;IF NOT, GO DO TEST
18786  113100  032737  010000  177520          BIT     #BIT12,BCSR                     ;LKS DISABLED?
18787  113106  001121                          BNE     TST36           ;;IF DISABLED, EXIT TEST
18788                                  ;
18789                                  ; WAIT FOR 3 INTERRUPTS AND CHECK LKS<7> TO BE 0 AFTER INTERRUPT
18790                                  ;
18791  113110  005037  002702          1$:     CLR     LKSFL                           ;CLEAR INTERRUPT FLAG
```

```
18792  113114  012737  132444  000100          MOV     #LKSINT,100             ;POINT VECTOR TO ROUTINE
18793  113122  012701  077777                  MOV     #77777,R1               ;COUNTER FOR SLOW CLOCK
18794  113126  052737  000100  177546          BIS     #BIT06,LKS              ;SET INTERRUPT ENABLE BIT
18795  113134  032737  000100  177546          BIT     #BIT06,LKS              ;BIT SET OK?
18796  113142  001001                          BNE     2$                      ;IF YES, BRANCH
18797  113144  104131                          ERROR   •131                    ;ERROR WRITING 1 TO LKS<6>
18798  113146  106427  000240          2$:     MTPS    #240                    ;SET PRIORITY TO 5
18799  113152  022737  000003  002702  3$:     CMP     #3,LKSFL                ;3 INTERRUPTS HAPPENED?
18800  113160  001401                          BEQ     4$                      ;IF YES, BRANCH
18801  113162  077105                          SOB     R1,3$                   ;STAY IN A LOOP
18802                                  ;
18803                                  ; DISABLE INTERRUPTS AND CHECK THAT PROPER CONDITIONS ARE MET
18804                                  ;
18805  113164  106427  000340          4$:     MTPS    #340                    ;RAISE PRIORITY
18806  113170  042737  000100  177546          BIC     #BIT06,LKS              ;DISABLE INTERRUPTS
18807  113176  032737  000200  177546          BIT     #BIT07,LKS              ;LKS<7> CLEARED AFTER INTERRUPTS?
18808  113204  001401                          BEQ     5$                      ;IF 0, BRANCH
18809  113206  104062                          ERROR   •62                     ;INTERRUPTS DON'T CLEAR LKS<7>
18810  113210  105737  177546          5$:     TSTB    LKS                     ;LKS<7>=1?
18811  113214  100375                          BPL     5$                      ;IF NOT, WAIT
18812  113216  005037  177546                  CLR     LKS                     ;CLEAR LKS<7>
18813  113222  032737  000200  177546          BIT     #BIT07,LKS              ;LKS<7> CLEARED?
18814  113230  001401                          BEQ     6$                      ;IF YES, BRANCH
18815  113232  104060                          ERROR   •60                     ;LKS<7> NOT CLEARED ON WRITE
18816  113234  032737  000100  177546  6$:     BIT     #BIT06,LKS              ;LKS<6>=0?
18817  113242  001401                          BEQ     7$                      ;IF YES, BRANCH
18818  113244  104131                          ERROR   •131                    ;ERROR WRITING 0 TO LKS<6>
18819  113246  022701  077737          7$:     CMP     #77737,R1               ;COUNTER AT LESS THAN 800HZ?
18820  113252  002001                          BGE     8$                      ;IF NOT, BRANCH
18821  113254  104063                          ERROR   •63                     ;READY LINE DOES NOT GO LOW
18822  113256  022737  000003  002702  8$:     CMP     #3,LKSFL                ;DID 3 INTERRUPTS HAPPEN?
18823  113264  001404                          BEQ     9$                      ;IF YES, BRANCH
18824  113266  012737  000003  001124          MOV     #3,$GDDAT               ;3 INTERRUPTS EXPECTED
18825  113274  104064                          ERROR   •64                     ;INTERRUPTS DON'T HAPPEN
18826                                  ;
18827                                  ; CHECK WHETHER INTERRUPTS HAPPEN AT PRIORITY 6
18828                                  ;
18829  113276  005037  002702          9$:     CLR     LKSFL                   ;CLEAR INTERRUPT FLAG
18830  113302  106427  000300                  MTPS    #300                    ;RAISE PRIORITY TO 6
18831  113306  012701  077777                  MOV     #77777,R1               ;COUNTER FOR SLOW CLOCK
18832  113312  052737  000100  177546          BIS     #BIT06,LKS              ;SET INTERRUPT ENABLE BIT
18833  113320  005737  002702          10$:    TST     LKSFL                   ;ANY INTERRUPTS?
18834  113324  001001                          BNE     11$                     ;IF YES, EXIT LOOP
18835  113326  077104                          SOB     R1,10$                  ;CONTINUE WITH COUNT
18836  113330  005737  002702          11$:    TST     LKSFL                   ;ANY INTERRUPTS?
18837  113334  001404                          BEQ     12$                     ;IF NO, BRANCH
18838  113336  012737  000005  001124          MOV     #5,$GDDAT               ;STORE PRIORITY FOR TYPE OUT
18839  113344  104065                          ERROR   •65                     ;INTERUPTS HAPPEN AT WRONG PRIORITY
18840  113346  106427  000340          12$:    MTPS    #340                    ;RESTORE PRIORITY
18841
```

```
                                        .SBTTL  TEST  - LINE CLOCK DISABLE
                                        ;LINE CLOCK DISABLE(*)
                                        ;THIS TEST WILL CHECK THAT BCSR<12> DISABLES RESPONSE
                                        ;OF LKS REGISTER.
                                        ;
                                        ;BCSR   <12>    LINE CLOCK STATUS REGISTER DISABLE
                                        ;ROUTINE TEST
                                        ;IF UFD AND LKS IS NOT DISABLED THEN
                                        ;.      EXIT TEST
                                        ;ENDIF
                                        ;.      WRITE BCSR<12>=1
                                        ;.      LET 4=ADDRESS OF LKS_TRAP
                                        ;.      LET TRAP_LKS=0
                                        ;.      READ LKS
                                        ;.      IF TRAP_LKS NE 1 THEN
                                        ;.      .       ERROR
                                        ;.      ENDIF
                                        ;ENDROUTINE
                                        ;
                                        ;ROUTINE LKS_TRAP
                                        ;.      LET TRAP_LKS=1
                                        ;.      LET BCSR<12>=0
                                        ;RTI

                                        ;;*****************************************************************
113352  000004                          TST36:  SCOPE
113354  032737  000100  000052                  BIT     #BIT06,@#52                     ;UFD MODE?
113362  001404                                  BEQ     1$                              ;IF NOT, BRANCH
113364  032737  010000  177520                  BIT     #BIT12,BCSR                     ;LKS DISABLED?
113372  001052                                  BNE     TST37           ;;IF DISABLED, EXIT TEST
                                        ;
                                        ; CHECK BCSR<12> TO BE 0 AND 1
                                        ;
113374  013737  177520  002710          1$:     MOV     BCSR,SAVBR                      ;SAVE BCSR REGISTER
113402  042737  010000  177520                  BIC     #BIT12,BCSR                             ;CLEAR BCSR
113410  032737  010000  177520                  BIT     #BIT12,BCSR                     ;<12>=0?
113416  001403                                  BEQ     2$                              ;IF OK, BRANCH
113420  005037  001124                          CLR     $GDDAT                          ;CLEAR EXPECTED PATTERN
113424  104052                                  ERROR   +52                             ;ERROR BCSR READ/WRITE BITS
113426  052737  010000  177520          2$:     BIS     #BIT12,BCSR                     ;SET BIT 12
113434  032737  010000  177520                  BIT     #BIT12,BCSR                     ;GOT SET OK?
113442  001004                                  BNE     3$                              ;IF OK, BRANCH
113444  012737  010000  001124                  MOV     #BIT12,$GDDAT                   ;EXPECTED PATTERN
113452  104052                                  ERROR   +52                             ;ERROR BCSR READ/WRITE BITS
                                        ;
                                        ; TRY TO ACCESS LKS TO GET A TIMEOUT WITH BCSR<12>=1
                                        ;
113454  013701  000004                  3$:     MOV     ERRVEC,R1                       ;SAVE TIMEOUT VECTOR
113460  012737  113502  000004                  MOV     #4$,ERRVEC                      ;POINT TO PROGRAM AREA
113466  012737  000340  000006                  MOV     #340,ERRVEC+2                   ;AT PRIORITY 7
113474  005737  177546                          TST     LKS                             ;ACCESS LKS REGISTER
113500  104066                                  ERROR   +66                             ;BCSR<12> DOES NOT DISABLE LKS
```

F12


```
113502  005726                  4$:     TST     (SP)+                   ;RESTORE STACK POINTER
113504  005726                          TST     (SP)+
113506  010137  000004                  MOV     R1,ERRVEC               ;RESTORE TIMEOUT VECTOR
113512  013737  002710  177520          MOV     SAVBR,BCSR              ;RESTORE BCSR
```

```
18900                                  .SBTTL  TEST  - UNCONDITIONAL CLOCK LINE INTERRUPTS
18901                                  ;UNCONDITIONAL CLOCK LINE INTERRUPTS(*)
18902                                  ;THIS TEST WILL CHECK THAT SETTING BCSR<13> TO 1 WILL
18903                                  ;REQUEST INTERRUPTS WHENEVER A CLOCK LINE IS ASSERTED.  THIS
18904                                  ;SHOULD HAPPEN WITHOUT ACCESSING LKS, THEREFORE, EVEN WITH BCSR<12>=1
18905                                  ;INTERRUPTS SHOULD HAPPEN.
18906                                  ;
18907                                  ;BCSR   <13>    FORCE LINE CLOCK INTERRUPT ENABLE
18908                                  ;
18909
18910                                  ;ROUTINE TEST
18911                                  ;IF UFD AND FORCE LKS NOT DISABLED THEN
18912                                  ;.      EXIT TEST
18913                                  ;ENDIF
18914                                  ;.      LET 100=ADDRESS OF UNCONDITIONAL_INTERRUPT_ROUTINE
18915                                  ;.      DO FOR BCSR<12> FROM #0 TO #1(LKS DISABLED AND ENABLED)
18916                                  ;.      (IF UFD DO ONLY FOR SELECTED LINE CLOCK)
18917                                  ;.      .       CLEAR UNCONDITIONAL_INTERRUPT
18918                                  ;.      .       SET COUNTER TO WAIT  FOR 3 INTERRUPTS
18919                                  ;.      .       LET BCSR<13>=1
18920                                  ;.      .       REPEAT
18921                                  ;.      .       .       DECREMENT COUNTER
18922                                  ;.      .       UNTIL UNCONDITIONAL_INTERUPT EQ #3 OR COUNTER EQ #0
18923                                  ;.      .       IF COUNTER GT TIME_REQUIRED_FOR_3_INTERRUPTS_FOR_800HZ
18924                                  ;.      .       .       ERROR (INTERRUPTS NEVER GO LOW)
18925                                  ;.      .       ENDIF
18926                                  ;.      .       IF UNCONDITIONAL_INTERRUPT LT #3 THEN
18927                                  ;.      .       .       ERROR (INTERRUPTS DON'T HAPPEN)
18928                                  ;.      .       ENDIF
18929                                  ;.      LET BCSR<13>=#0
18930                                  ;.      ENDDO
18931                                  ;ENDROUTINE
18932                                  ;
18933                                  ;ROUTINE UNCONDITIONAL_INTERRUPT_ROUTINE
18934                                  ;.      INCREMENT UNCONDITIONAL_INTERRUPT
18935                                  ;RETURN
18936
18937                                  ;;**************************************************************************
18938  113520  000004                 TST37:  SCOPE
18939  113522  032737  000100  000052         BIT     #BIT06,@#52                     ;UFD MODE?
18940  113530  001404                         BEQ     1$                              ;IF NOT, BRANCH
18941  113532  032737  020000  177520         BIT     #BIT13,BCSR                     ;FORCE INTERRUPT SET?
18942  113540  001525                         BEQ     TST40           ;;IF SET, EXIT TEST
18943                                  ;
18944                                  ; CHECK BSCR<13> TO BE 0 AND 1
18945                                  ;
18946  113542  013737  177520  002710 1$:     MOV     BCSR,SAVBR                      ;SAVE BCSR REGISTER
18947  113550  042737  020000  177520         BIC     #BIT13,BCSR                     ;CLEAR BCSR
18948  113556  032737  020000  177520         BIT     #BIT13,BCSR                     ;<13>=0?
18949  113564  001403                         BEQ     2$                              ;IF OK, BRANCH
18950  113566  005037  001124                 CLR     $GDDAT                          ;CLEAR EXPECTED PATTERN
18951  113572  104052                         ERROR   +52                             ;ERROR BCSR READ/WRITE BITS
18952  113574  052737  020000  177520 2$:     BIS     #BIT13,BCSR                     ;SET BIT 13
18953  113602  032737  020000  177520         BIT     #BIT13,BCSR                     ;GOT SET OK?
18954  113610  001004                         BNE     3$                              ;IF OK, BRANCH
18955  113612  012737  020000  001124         MOV     #BIT13,$GDDAT                   ;EXPECTED PATTERN
```

```
113620  104052                          ERROR   +52                     ;ERROR BCSR READ/WRITE BITS
                                ;
                                ; SET UP  TO DO UNCONDITIONAL INTERRUPTS
                                ;
113622  012737  132444  000100  3$:     MOV     #LKSINT,@#100           ;SET UP INTERRUPT VECTOR
113630  012737  000340  000102          MOV     #340,@#102              ;AT PRIORITY 7
113636  052737  010000  177520          BIS     #BIT12,BCSR             ;FOR 1ST TIME DISABLE LKS
113644  000403                          BR      5$                      ;GO DO IT
113646  042737  010000  177520  4$:     BIC     #BIT12,BCSR             ;FOR THE 2ND TIME, ENABLE LKS
113654  005037  002702          5$:     CLR     LKSFL                   ;CLEAR INTERRUPTS FLAG
113660  012702  077777                  MOV     #77777,R2               ;COUNTER TO WAIT FOR INTERRUPTS
113664  106427  000240                  MTPS    #240                    ;LOWER PRIORITY TO 5
113670  022737  000003  002702  6$:     CMP     #3,LKSFL                ;3 INTERRUPTS HAPPENED?
113676  001401                          BEQ     7$                      ;EXIT LOOP, IF SO
113700  077205                          SOB     R2,6$                   ;OTHERWISE, KEEP WAITING
113702  106427  000340          7$:     MTPS    #340                    ;RAISE PRIORITY TO 7
113706  022702  077700                  CMP     #77700,R2               ;INTERRUPTS HAPPEN TOO OFTEN?
113712  002001                          BGE     8$                      ;IF NOT, BRANCH
113714  104063                          ERROR   +63                     ;READY LINE DOESN'T GO LOW
113716  022737  000003  002702  8$:     CMP     #3,LKSFL                ;AT LEAST 3 INTERRUPTS HAPPENED?
113724  002004                          BGE     9$                      ;IF SO, BRANCH
113726  012737  000003  001124          MOV     #3,$GDDAT               ;EXPECTED DATA
113734  104064                          ERROR   +64                     ;INTERRUPTS DON'T HAPPEN
113736  032737  010000  177520  9$:     BIT     #BIT12,BCSR             ;SECOND TIME THRU THE LOOP?
113744  001340                          BNE     4$                      ;IF NOT, DO IT AGAIN
113746  032737  000100  000052          BIT     #BIT06,@#52             ;UFD MODE?
113754  001404                          BEQ     10$                     ;IF NOT, BRANCH
113756  032737  010000  177520          BIT     #BIT12,BCSR             ;IF UFD AND LKS DISABLED?
113764  001010                          BNE     12$                     ;DON'T CHECK LKS
113766  032737  000100  177546  10$:    BIT     #BIT06,LKS              ;INTERRUPT ENABLE LINE HOLD 1?
113774  001001                          BNE     11$                     ;IF SO, BRANCH
113776  104067                          ERROR   +67                     ;BCSR<13> DOESN'T SET LKS<6>
114000  042737  000100  177546  11$:    BIC     #BIT06,LKS              ;DISABALE LKS INTERRUPTS
114006  013737  002710  177520  12$:    MOV     SAVBR,BCSR              ;RESTORE BCSR
```

```
                                        .SBTTL  TEST  - RESETTING LKS
                                        ;RESETTING LKS(*)
                                        ;THIS TEST WILL PROVE THAT RESET INSTRUCTION SETS LKS<07> AND
                                        ;CLEARS LKS<06>.
                                        ;ROUTINE TEST
                                        ;IF UFD AND LKS IS DISABLED THEN
                                        ;.      EXIT TEST
                                        ;ENDIF
                                        ;.      POINT LKS VECTOR 100 TO ERROR_LKS_ILLEGAL_INTERRUPT
                                        ;.      SYNCHRONIZE LKS BY WAITING FOR 3 PULSES
                                        ;.      LET LKS<06>=#1
                                        ;.      CLEAR LKS (CLEARS LKS<07>
                                        ;.      EXECUTE "RESET"
                                        ;.      IF LKS<7> NE #1 OR LKS<6> NE #0 THEN
                                        ;.      .       ERROR
                                        ;.      ENDIF
                                        ;.      IF ILLEGAL_LINE_CLOCK_INTERRUPT NE 0 THEN
                                        ;.      .       ERROR
                                        ;.      ENDIF
                                        ;ENDROUTINE
                                        ;
                                        ;ROUTINE ERROR_LKS_ILLEGAL_INTERRUPT
                                        ;.      FLAG ILLEGAL_LINE_CLOCK_INTERRUPT
                                        ;RETURN

                                        ;;*****************************************************************
114014  000004                          TST40:  SCOPE
114016  122737  000001  001220                  CMPB    #APTENV,$ENV                    ;RUNNING IN APT MODE?
114024  001013                                  BNE     1$                              ;NO, GO DO TEST
114026  005737  001206                          TST     $PASS                           ;FIRST PASS?
114032  001057                                  BNE     TST41           ;;IF APT AND NOT FIRST PASS, EXIT TEST
114034  032737  000100  000052                  BIT     #BIT06,@#52                     ;UFD MODE?
114042  001404                                  BEQ     1$                              ;IF NOT, BRANCH
114044  032737  010000  177520                  BIT     #BIT12,BCSR                     ;LKS DISABLED?
114052  001447                                  BEQ     TST41           ;;IF DISABLED, EXIT TEST
                                        ;
                                        ; SYNCHRONISE WITH LINE TIME CLOCK BY WAITING FOR 3 INTERRUPTS
                                        ;
114054  013737  177520  002710          1$:     MOV     BCSR,SAVBR                      ;SAVE BCSR
114062  042737  010000  177520                  BIC     #BIT12,BCSR                     ;ENABLE LKS RESPONSE
114070  012737  132444  000100                  MOV     #LKSINT,@#100                   ;SET UP INTERRUPT VECTOR
114076  012737  000340  000102                  MOV     #340,@#102                      ;AT PROIRITY 7
114104  052737  000100  177546                  BIS     #BIT06,LKS                      ;SET INTERRUPT ENABLE BIT
114112  005037  002702                          CLR     LKSFL                           ;CLEAR INTERRUPTS FLAG
114116  012702  077777                          MOV     #77777,R2                       ;COUNTER TO WAIT FOR INTERRUPTS
114122  106427  000240                          MTPS    #240                            ;LOWER PRIORITY TO 5
114126  022737  000003  002702          2$:     CMP     #3,LKSFL                        ;3 INTERRUPTS HAPPENED?
114134  001401                                  BEQ     3$                              ;EXIT LOOP, IF SO
114136  077205                                  SOB     R2,2$                           ;OTHERWISE, KEEP WAITING
114140  106427  000340                  3$:     MTPS    #340                            ;RAISE PRIORITY TO 7
114144  000005                                  RESET                                   ;EXECUTE RESET
114146  032737  000200  177546                  BIT     #BIT07,LKS                      ;READY BIT SET?
114154  001001                                  BNE     4$                              ;IF SO, BRANCH
114156  104070                                  ERROR   +70                             ;RESET DOESN'T SET LKS<07.
114160  032737  000100  177546          4$:     BIT     #BIT06,LKS                      ;INTERRUPT ENABLE BIT CLEARED?
114166  001401                                  BEQ     TST41           ;;IF SO, EXIT TEST
```

```
19048  114170  104071                    ERROR   +71                    ;RESET DOESN'T CLEAR LKS
19049
19050
```

```
19051                                   .SBTTL  TEST  - LINE CLOCK INTERRUPTS
19052                                   ;LINE CLOCK INTERRUPTS(*)
19053                                   ;BY SETTING TO 1 LKS<06>, THIS TEST WILL CHECK FOR INTERRUPTS
19054                                   ;FROM BEVENT LINE AND FROM KDJ11-B 50HZ, 60HZ, 800HZ ON BOARD SIGNALS
19055                                   ;(THE LATTER SIGNALS WILL BE ACCESSED BY SETTING BCSR<11-10>).
19056                                   ;
19057                                   ;BCSR   <11>    <10>            CLOCK SELECT BITS 1 AND 0
19058                                   ;
19059                                   ;        0       0              EXTERNAL BEVENT LINE
19060                                   ;        0       1              ON-BOARD 50 HZ
19061                                   ;        1       0              ON-BOARD 60 HZ
19062                                   ;        1       1              ON-BOARD 800 HZ
19063                                   ;
19064                                   ;ROUTINE TEST
19065                                   ;IF UFD THEN
19066                                   ;.      IF LKS DISABLED THEN
19067                                   ;.      .       EXIT TEST
19068                                   ;.      ENDIF
19069                                   ;.      SET FLAGS TO RUN ONLY WHAT SPECIFIED IN EAPROM
19070                                   ;ENDIF
19071                                   ;.      LET 100=ADDRESS OF LKS_INTERRUPT
19072                                   ;.      DO FOR BCSR<11:10> FROM #0 TO #3
19073                                   ;.      .       LET LKS<06>=#1
19074                                   ;.      .       WAIT FOR 10 INTERRUPTS FOR EACH CLOCK
19075                                   ;.      .       STORE ACTUAL NUMBER OF INTERRUPTS FOR EACH CLOCK
19076                                   ;.      .       LET INTERRUPT_FLAG=0
19077                                   ;.      ENDDO
19078                                   ;.      COMPARE NUMBER OF INTERRUPTS FOR EACH CLOCK
19079                                   ;ENDROUTINE
19080                                   ;
19081                                   ;ROUTINE LKS_INTERRUPT
19082                                   ;.      INCREMENT INTERRUPT_FLAG
19083                                   ;RETURN
19084
19085                                   ;;*****************************************************************
19086  114172  000004                 TST41:  SCOPE
19087  114174  032737  000100  000052         BIT     #BIT06,@#52                     ;UFD MODE?
19088  114202  001411                         BEQ     1$                              ;IF NOT, GO DO THE TEST
19089  114204  032737  010000  177520         BIT     #BIT12,BCSR                     ;LKS IS DISABLED?
19090  114212  001123                         BNE     TST42           ;;IF DISABLED, EXIT TESTS
19091  114214  005002                         CLR     R2                              ;CLEAR R2 TO SET FLAGS
19092  114216  053702  177520                 BIS     BCSR,R2                         ;SET R2 ACCORDING TO BCSR
19093  114222  042702  171777                 BIC     #171777,R2                      ;LEAVE ONLY BITS <11-10>
19094                                  ;
19095                                  ; SETUP FOR INTERRUPTS, IN UFD MODE ONLY FROM THE CLOCK SPECIFIED IN BCSR<11-10>
19096                                  ;
19097  114226  013737  177520  002710  1$:     MOV     BCSR,SAVBR                      ;STORE BCSR
19098  114234  012737  132444  000100         MOV     #LKSINT,100                     ;SET UP LKS VECTOR
19099  114242  012737  000340  000102         MOV     #340,102                        ;AT PRIORITY 7
19100  114250  012705  114452                 MOV     #TIMDEL,R5                      ;POINTER TO DEL
19101  114254  012704  000004                 MOV     #4,R4                           ;R4 IS THE COUNTER FOR ALL CLOCKS
19102  114260  005037  177520                 CLR     BCSR                            ;DO FOR BEVENT LINE INTERRUPTS
19103  114264  000403                         BR      3$                              ;GO DO THE LOOP
19104  114266  062737  002000  177520  2$:     ADD     #2000,BCSR                      ;SET UP FOR THE NEXT CLOCK LINE
19105  114274  032737  000100  000052  3$:     BIT     #BIT06,@#52                     ;UFD MODE?
19106  114302  001402                         BEQ     4$                              ;IF NOT, BRANCH
```

```
19107   114304  010237  177520                  MOV     R2,BCSR                 ;IN UFD, DO ONLY FOR SPECIFIED
19108   114310  005037  002702          4$:     CLR     LKSFL                   ;CLEAR INTERRUPT FLAG
19109   114314  052737  000100  177546          BIS     #BIT06,LKS              ;SET INTERRUPT ENABLE BIT
19110   114322  012703  000010                  MOV     #10,R3                  ;START COUNTER FOR 10 INTERURRUPTS
19111   114326  012701  177777          5$:     MOV     #177777,R1              ;START COUNTER TO WAIT FOR INTERRUPT
19112   114332  106427  000240                  MTPS    #240                    ;LOWER PRIORITY TO 5
19113   114336  023703  002702          6$:     CMP     LKSFL,R3                ;NEW INTERRUPT HAPPENED?
19114   114342  001401                          BEQ     7$                      ;IF SO, EXIT WAIT LOOP
19115   114344  077104                          SOB     R1,6$                   ;OTHERWISE, KEEP WAITING
19116   114346  010125          7$:     MOV     R1,(R5)+                ;STORE DELAY FOR EACH CLOCK
19117   114350  032737  000100  000052          BIT     #BIT06,@#52             ;UFD MODE?
19118   114356  001026                          BNE     10$                     ;IF UFD, DON'T DO FOR ANY OTHER
19119   114360  077436                          SOB     R4,2$                   ;ALL LINE CLOCKS DONE?
19120                                   ;
19121                                   ; CHECK THE DELAY VALUES FOR ALL CLOCKS
19122                                   ;
19123   114362  106427  000340                  MTPS    #340                    ;RAISE PRIORITY
19124   114366  042737  000100  177546          BIC     #BIT06,LKS              ;DISABLE INTERRUPTS
19125   114374  012705  114452                  MOV     #TIMDEL,R5              ;POINTER TO DELAY TABLE
19126   114400  021565  000006                  CMP     (R5),6(R5)              ;DELAY FOR BEVENT AND 800HZ?
19127   114404  103401                          BLO     8$                      ;BEVENT IS NOT 800HZ
19128   114406  104064                          ERROR   +64                     ;WRONG # OF INTERRUPTS
19129   114410  005725          8$:     TST     (R5)+                   ;INCREMENT POINTER
19130   114412  021565  000002                  CMP     (R5),2(R5)              ;DELAY FOR 50HZ AND 60HZ?
19131   114416  103401                          BLO     9$                      ;IF FIRST BIGGER, BRANCH
19132   114420  104064                          ERROR   +64                     ;WRONG # OF INTERRUPTS
19133   114422  005725          9$:     TST     (R5)+                   ;INCREMENT POINTER
19134   114424  021565  000002                  CMP     (R5),2(R5)              ;DELAY FOR 50HZ AND 800HZ
19135   114430  103401                          BLO     10$                     ;IF FIRST BIGGER, BRANCH
19136   114432  104064                          ERROR   +64                     ;WRONG # OF INTERRUPTS
19137   114434  013737  002710  177520  10$:    MOV     SAVBR,BCSR              ;RESTORE BCSR
19138   114442  042737  000100  177546          BIC     #BIT06,LKS              ;DISABLE INTERRUPTS
19139   114450  000404                          BR      TST42           ;;EXIT TEST
19140
19141   114452  000004                  TIMDEL: .BLKW 4
19142
```

```
                                       .SBTTL  TEST  - MAINTENANCE REGISTER TEST
                                       ;MAINTENANCE REGISTER TEST
                                       ;THIS TEST WILL ADDRESS MAINTENANCE REGISTER AND CHECK BITS
                                       ;7-4 TO BE 0010, 2-1 TO BE 10, AND READ BITS 10-08, 03, 00
                                       ;FOR FUTURE USE.  THOSE BITS REPRESENT THE FOLLOWING SIGNALS:
                                       ;MULTIPROCESSOR SLAVE, UNIBUS SYSTEM, FPA AVAILABLE, HALT/TRAP
                                       ;OPTION, AND AC POWER OKAY.
                                       ;ROUTINE TEST
                                       ;.      IF MAINT. REG. BITS <7-4> NE 0010 OR <2-1> NE 10 THEN
                                       ;.      .       ERROR
                                       ;.      ENDIF
                                       ;.      READ MAINT.REG. BITS <10-08,03,00>
                                       ;ENDROUTINE

                                       ;;*****************************************************************
114462  000004                         TST42:  SCOPE
114464  032737  174000  177750                 BIT     #174000,MAIREG                  ;UNUSED BITS ALL ZEROS?
114472  001401                                 BEQ     1$                              ;IF OK, BRANCH
114474  104132                                 ERROR   +132                            ;MAINTENANCE REGISTER ERROR
114476  032737  000044  177750  1$:            BIT     #44,MAIREG                      ;<5,2> SET ?
114504  001001                                 BNE     2$                              ;IF SO, BRANCH
114506  104132                                 ERROR   +132                            ;MAINTENANCE REGISTER ERROR
114510  032737  000322  177750  2$:            BIT     #322,MAIREG                     ;<7,6,4,1> CLEAR?
114516  001401                                 BEQ     TST43           ;;IF YES, BRANCH
114520  104132                                 ERROR   +132
```

```
                                .SBTTL  TEST  - SERIAL LINE UNIT REGISTERS
                                ;SERIAL LINE UNIT TEST(4)
                                ;BCR<2-0> WILL BE READ TO FIND OUT BAUD RATE. SLU WILL BE PROG-
                                ;RAMMED TO CHECK THE INTERRUPT LEVELS BY SETTING BIT<06> IN
                                ;RCSR AND XMIT. LOOP BACK CAPABILITIES WILL BE TESTED BY SETTING
                                ;TO 1 XCSR<02>.   THE LINE CLOCK INTERRUPT SUBROUTINE WILL BE
                                ;USED TO RETURN TO THE EXECUTION OF THE DIAGNOSTICS, IF THE
                                ;PROGRAM HANGS IN THE LOOP BACK MODE.
                                ;ROUTINE TEST
                                ;IF UFD AND CONSOLE NOT PRESENT
                                ;.      GO TO TEST_22
                                ;ENDIF
                                ;.      IF BCR<07> EQ #0 THEN
                                ;.      .       READ BCR<2-0> TO GET BAUD RATE
                                ;.      ENDIF
                                ;.      LET 4=ADDRESS_OF_TIMEOUT _ROUTINE
                                ;.      DO FOR RCSR,XCSR,RBUF,XBUF
                                ;.      .       READ XRCSR,XCSR,RBUF,XBUF
                                ;.      .       IF TIMEOUT_FLAG NE #0 THEN
                                ;.      .               ERROR
                                ;.      .       ENDIF
                                ;.      ENDDO
                                ;ENDROUTINE
                                ;
                                ;ROUTINE TIMEOUT
                                ;.      LET TIMEOUT_FLAG=#1
                                ;ENDROUTINE
                                ;
                                ;;*****************************************************************
114522  000004                  TST43:  SCOPE
114524  032737  000100  000052          BIT     #BIT06,@#52             ;UFD MODE?
114532  001406                  1$      BEQ     1$                      ;IF NOT, GO DO THE TEST
114534  032737  000200  177524          BIT     #BIT07,BCR              ;IF UFD AND CONSOLE NOT PRESENT
114542  001402                          BEQ     1$                      ;NOT TRUE, DO THE TEST
114544  000137  116510                  JMP     SLEND                   ;IF TRUE, SKIP ALL SLU TESTS
                                ;
                                ; TRY TO ACCESS SLU REGISTERS
                                ;
114550  013701  000004          1$:     MOV     ERRVEC,R1               ;SAVE TIMEOUT VECTOR
114554  012737  114600  000004          MOV     #3$,ERRVEC              ;POINT NEW ONE TO PROGRAM AREA
114562  012737  000340  000006          MOV     #340,ERRVEC+2           ;AT PRIORITY 7
114570  012702  177560                  MOV     #RCSR,R2                ;START ACCESSING WITH RCSR
114574  005712                  2$:     TST     (R2)                    ;ACCESS SLU REGISTER
114576  000403                          BR      4$                      ;IF NO TIMEOUT, CONTINUE
114600  010237  001126          3$:     MOV     R2,$BDDAT               ;STORE ADDRESS THAT TIMED OUT
114604  104072                          ERROR   +72                     ;TIMEOUT ACCESSING SLU REGISTER
114606  022722  177566          4$:     CMP     #XBUF,(R2)+             ;LAST REGISTER ACCESSED?
114612  103770                          BLO     2$                      ;IF NOT, BRANCH
114614  010137  000004                  MOV     R1,ERRVEC               ;RESTORE TIMEOUT VECTOR
```

```
19220                                   .SBTTL  TEST  - XCSR BIT 7
19221                                   ;CHECK THAT XCSR<07> CAN BE 0 AND 1.
19222                                   ;
19223                                   ;XCSR   <07>            TRANSMITTER READY
19224                                   ;
19225                                   ;ROUTINE TEST
19226                                   ;.      WAIT FOR XCSR<07>=#1 NO MORE THAN 200MSEC
19227                                   ;.      IF XCSR<07> NE #1 THEN
19228                                   ;.          .       ERROR
19229                                   ;.      ENDIF
19230                                   ;.      LET XBUF=#NULL
19231                                   ;.      WAIT FOR XCSR<07>=#1
19232                                   ;.      LET XBUF=#NULL
19233                                   ;.      IF XCSR<07> NE 0 THEN
19234                                   ;.          .       ERROR (READY DIDN'T GO LOW)
19235                                   ;.      ENDIF
19236                                   ;ENDROUTINE
19237
19238                                   ;;*****************************************************************
19239  114620  000004                  TST44:  SCOPE
19240  114622  012701  001000                  MOV     #1000,R1                ;COUNTER FOR ABOUT 200MICROSEC.
19241  114626  122737  000001  001220          CMPB    #APTENV,$ENV            ;RUNNING IN APT MODE?
19242  114634  001003                          BNE     1$                      ;NO, GO DO TEST
19243  114636  005737  001206                  TST     $PASS                   ;FIRST PASS?
19244  114642  001017                          BNE     TST45           ;;IF APT AND NOT FIRST PASS, EXIT TEST
19245  114644  105737  177564          1$:     TSTB    XCSR                    ;XCSR<7> READY 1?
19246  114650  100401                          BMI     2$                      ;IF SO, EXIT WAIT LOOP
19247  114652  077104                          SOB     R1,1$                   ;IF NOT 1, CONTINUE WAITING
19248  114654  105737  177564          2$:     TSTB    XCSR                    ;XCSR<7>=1?
19249  114660  100401                          BMI     3$                      ;IF YES, BRANCH
19250  114662  104073                          ERROR   .73                     ;XCSR<7> DOES NOT BECOME 1
19251  114664  012737  000000  177566  3$:     MOV     #NULL,XBUF              ;TRY TO TRANSMIT NULL CHARACTER
19252  114672  105737  177564                  TSTB    XCSR                    ;XCSR<7>=0
19253  114676  100001                          BPL     TST45           ;;IF YES, EXIT TEST
19254  114700  104073                          ERROR   .73                     ;XMIT READY DIDN'T GO LOW
19255
19256
```

```
                                        .SBTTL  TEST  - RCSR BIT 7 AND XCSR BIT 2
                                        ;CHECK THAT RCSR<07> CAN BE 0 AND 1 AND THAT XCSR<02> WORKS PROPERLY.
                                        ;
                                        ;RCSR   <07>    RECEIVER DONE
                                        ;XCSR   <02>    MAINTENANCE
                                        ;
                                        ;ROUTINE TEST
                                        ;.(CHECK RCSR<07> AND XCSR<07>)
                                        ;.      WAIT FOR XCSR<07>=#1
                                        ;.      LET XCSR<02>=#1 (LOOP BACK MODE)
                                        ;.      LET XBUF=#125
                                        ;.      WAIT FOR RCSR<07>=#1 NO MORE THAN 200MSEC
                                        ;.      IF RCSR<07> NE #1 THEN
                                        ;.      .       ERROR (RCSR<07> DOES NOT BECOME 1 OR XCSR<02>DOES NOT
                                        ;.      .               WORK)
                                        ;.      ENDIF
                                        ;.      IF RBUF NE #125 THEN
                                        ;.      .       ERROR
                                        ;.      ENDIF
                                        ;.      IF RCSR<07> NE #0 THEN
                                        ;.      .       ERROR (RCSR<07>DOES NOT GO LOW)
                                        ;.      ENDIF
                                        ;.      LET XCSR<02>=#0
                                        ;ENDROUITNE
                                        ;
                                        ;;*****************************************************************
114702  000004                  TST45:  SCOPE
114704  012701  000013                  MOV     #13,R1                  ;COUNTER FOR ABOUT 200MICROSEC.
114710  122737  000001  001220          CMPB    @APTENV,$ENV            ;RUNNING IN APT MODE?
114716  001003                          BNE     1$                      ;NO, GO DO TEST
114720  005737  001206                  TST     $PASS                   ;FIRST PASS?
114724  001074                          BNE     TST46           ;;IF APT AND NOT FIRST PASS, EXIT TEST
114726  105737  177564          1$:     TSTB    XCSR                    ;XCSR<7> READY 1?
114732  100375                          BPL     1$                      ;IF NOT 1, CONTINUE WAITING
114734  052737  000004  177564  2$:     BIS     #BIT02,XCSR             ;SET LOOP BACK MODE
114742  032737  000004  177564          BIT     #BIT02,XCSR             ;GOT SET OK?
114750  001004                          BNE     3$                      ;IF YES, BRANCH
114752  005037  177564                  CLR     XCSR                    ;RESET TO PRINT ERROR
114756  104114                          ERROR   +114                    ;XCSR<2> DOES NOT BECOME 1
114760  000456                          BR      TST46           ;;EXIT TEST
                                        ;
                                        ; STALL FOR A WHILE IN CASE XCSR<2> CAUSES RCSR<7> TO BE 1
                                        ;
114762  012701  002000          3$:     MOV     #2000,R1                ;STALL IN CASE XCSR<2> SETS READY
114766  105737  177560          4$:     TSTB    RCSR                    ;IF RECEIVER READY SET?
114772  100402                          BMI     5$                      ;IF SET, BRANCH
114774  077104                          SOB     R1,4$                   ;OTHERWISE, STAY FOR A WHILE
114776  000402                          BR      6$                      ;IF NOT READY, BRANCH
115000  005737  177562          5$:     TST     RBUF                    ;READ RBUF
                                        ;
                                        ; TRANSMIT XON AND CHECK RCSR<7>
                                        ;
115004  012737  000021  177566  6$:     MOV     #21,XBUF                ;TRANSMIT A CHARACTER
115012  012701  002000                  MOV     #2000,R1                ;COUNTER TO WAIT
115016  105737  177560          7$:     TSTB    RCSR                    ;RCSR<7> READY 1?
115022  100401                          BMI     8$                      ;IF YES, EXIT WAIT LOOP
```

```
19313  115024  077104                          SOB     R1,7$           ;OTHERWISE, CONTINUE WAITING
19314  115026  105737  177560          8$:     TSTB    RCSR            ;RCSR<7>=1?
19315  115032  100403                          BMI     9$              ;IF YES, BRANCH
19316  115034  005037  177564                  CLR     XCSR            ;RESET XCSR<2>
19317  115040  104074                          ERROR   .74             ;RECEIVER READY DIDN'T COME UP
19318  115042  013737  177562  001126  9$:     MOV     RBUF,$BDDAT     ;STORE RECEIVED DATA
19319  115050  022737  000021  001126          CMP     #21,$BDDAT      ;DATA RECEIVED OK?
19320  115056  001406                          BEQ     10$             ;IF YES, BRANCH
19321  115060  012737  000021  001124          MOV     #21,$GDDAT
19322  115066  005037  177564                  CLR     XCSR            ;RESET TO ENABLE SLU
19323  115072  104075                          ERROR   .75             ;WRONG CHARACTER RECEIVED
19324  115074  105737  177560          10$:    TSTB    RCSR            ;RCSR<7>=0?
19325  115100  100003                          BPL     11$             ;IF ZERO, BRANCH
19326  115102  005037  177564                  CLR     XCSR            ;RESET TO ENABLE SLU
19327  115106  104076                          ERROR   .76             ;RCSR<07><>0 AFTER READING RBUF
19328  115110  042737  000004  177564  11$:    BIC     #BIT02,XCSR     ;DISABLE LOOP BACK MODE
19329
19330
```

```
                                        .SBTTL  TEST  - RESET AND XCSR<2!0>
                                        ;CHECK THAT RESET CLEARS XCSR<0!2>.
                                        ;ROUTINE TEST
                                        ;.(CHECK RCSR<07> AND XCSR<07> AND RESET)
                                        ;.      LET XCSR<02,00>=#1 (LOOP BACK MODE)
                                        ;.      EXECUTE "RESET"
                                        ;.      IF XCSR<02!00> NE #0 THEN
                                        ;.      .       ERROR
                                        ;.      ENDIF
                                        ;.      LET XCSR<02>=#0
                                        ;ENDROUTINE

                                        ;;*****************************************************************
115116  000004                          TST46:  SCOPE
115120  122737  000001  001220                  CMPB    #APTENV,$ENV                    ;RUNNING IN APT MODE?
115126  001003                                  BNE     1$                              ;NO, GO DO TEST
115130  005737  001206                          TST     $PASS                           ;FIRST PASS?
115134  001011                                  BNE     TST47           ;;IF APT AND NOT FIRST PASS, EXIT TEST
115136  052737  000005  177564          1$:     BIS     #BIT02!BIT00,XCSR               ;LOOP BACK MODE
                                        ;
                                        ; EXECUTE RESET AND VALIDATE THAT XCSR<7,2> BECOMES <1,0>
                                        ;
115144  000005                                  RESET                                   ;EXECUTE RESET
115146  032737  000005  177564                  BIT     #BIT02!BIT00,XCSR               ;XCSR<2,0> CLEAR?
115154  001401                                  BEQ     TST47           ;;IF YES, BRANCH
115156  104102                                  ERROR   +102                            ;XCSR<2,0> NOT CLEARED ON RESET
```

```
                                .SBTTL  TEST  - RESET AND INTERRUPT ENABLE BITS
                                ;CHECK THAT INTERRUPTS DON'T HAPPEN AT PRIORITY 4 AND THAT RESET
                                ;CLEARS XCSR<06> AND RCSR<06>.
                                ;
                                ;RCSR   <06>    RECEIVER INTERRUPT ENABLE
                                ;XCSR   <06>    TRANSMITTER INTERRUPT ENABLE
                                ;
                                ;ROUTINE TEST
                                ;.      LET 60=@ADDRESS_OF_ILLEGAL_INTERRUPT_XRCSR
                                ;.      LET 64=@ADDRESS_OF_ILLEGAL_INTERRUPT_XRCSR
                                ;.      SET PRIORITY TO 4
                                ;.      LET XCSR<02>=#1 (LOOPBACK MODE)
                                ;.      LET XCSR<06>=#1 (ENABLE TRANSMIT INTERRUPTS)
                                ;.      LET RCSR<06>=#1 (ENABLE RECEIVE INTERRUPTS)
                                ;.      WAIT FOR XCSR<07>=#1 (READY TO TRANSMIT)
                                ;.      LET XBUF=@NULL (SEND A CHARACTER)
                                ;.      WAIT FOR ILLEGAL INTERRUPTS (ABOUT 200MSEC)
                                ;.      EXECUTE "RESET"
                                ;.      IF XCSR<06> NE #0 OR RCSR<06> NE #0 OR XRCSR NE #0 THEN
                                ;.      .       ERROR
                                ;.      ENDIF
                                ;.      RESTORE PRIORITY TO NORMAL
                                ;ENDROUTINE
                                ;
                                ;ROUTINE ILLEGAL_INTERRUPT_XRCSR
                                ;.      INCREMENT XRCSR
                                ;ENDROUTINE

                                ;;*****************************************************************
115160  000004                  TST47:  SCOPE
115162  122737  000001  001220          CMPB    @APTENV,$ENV                    ;RUNNING IN APT MODE?
115170  001003                        BNE     1$                              ;NO, GO DO TEST
115172  005737  001206                TST     $PASS                           ;FIRST PASS?
115176  001033                        BNE     5$                              ;SKIP RESET PART OF THE TEST
                                ;
                                ; CHECK THAT INTERRUPTS ENABLE BITS FOR RECEIVER AND TRASMITTER OF SLU
                                ; ARE CLEARED BY RESET
                                ;
115200  052737  000100  177564  1$:     BIS     @#BIT06,XCSR                    ;SET INTERRUPT ENABLE BIT IN XCSR
115206  032737  000100  177564          BIT     @#BIT06,XCSR                    ;GOT SET OK?
115214  001001                        BNE     2$                              ;IF YES, BRANCH
115216  104110                        ERROR   .110                            ;IN BIT 6 OF XCSR
115220  052737  000100  177560  2$:     BIS     @#BIT06,RCSR                    ;SET INTERRUPT ENABLE BIT IN RCSR
115226  032737  000100  177560          BIT     @#BIT06,RCSR                    ;GOT SET OK?
115234  001001                        BNE     3$                              ;IF YES, BRANCH
115236  104110                        ERROR   .110                            ;IN BIT 6 OF RSCR
115240  000005                  3$:     RESET                                   ;INLINE BUS RESET
115242  032737  000100  177564          BIT     @#BIT06,XCSR                    ;XMIT INTERRUPT ENABLE BIT CLEARED?
115250  001401                        BEQ     4$                              ;IF CLEARED, BRANCH
115252  104102                        ERROR   .102                            ;INTERRUPT ENABLE NOT CLEARED ON RESET
115254  032737  000100  177560  4$:     BIT     @#BIT06,RCSR                    ;RECEIVE INTERRUPT ENBLE CLEARED?
115262  001401                        BEQ     5$                              ;IF CLEARED, BRANCH
115264  104102                        ERROR   .102                            ;INTERRUPT ENABLE NOT CLEARED ON RESET
                                ;
                                ; CHECK THAT TRANSMIT INTERRUPTS DON'T HAPPEN AT PRIORITY HIGHER THAN 3
                                ;
```

```
19415  115266  012737  115334  000064  5$:     MOV     #9$,@#64                ;POINT XMIT VECTOR TO PROGRAM AREA
19416  115274  012737  000340  000066          MOV     #340,@#66               ;AT PRIORITY 7
19417  115302  052737  000100  177564          BIS     #BIT06,XCSR             ;SET INTERRUPT ENABLE BIT IN XCSR
19418  115310  012702  000340                  MOV     #340,R2                 ;SET PRIORITY TO 7
19419  115314  000402                          BR      7$                      ;GO WAIT IN CASE OF INTERRUPTS
19420  115316  162702  000040          6$:     SUB     #40,R2                  ;LOWER PRIORITY LEVEL
19421  115322  106402                  7$:     MTPS    R2                      ;SET PRIORITY
19422  115324  012703  000026                  MOV     #26,R3                  ;TIME DELAY
19423  115330  077301                  8$:     SOB     R3,8$                   ;WAIT FOR INTERRUPTS
19424  115332  000403                          BR      10$                     ;IF INTERRUPTS DIDN'T HAPPENED, BRANCH
19425  115334  104101                  9$:     ERROR   +101                    ;INTERRUPTS HAPPEN AT WRONG PRIORITY
19426  115336  005726                          TST     (SP)+                   ;CLEAN UP THE STACK
19427  115340  005726                          TST     (SP)+
19428  115342  022702  000200          10$:    CMP     #200,R2                 ;AT PRIORITY 4?
19429  115346  001363                          BNE     6$                      ;IF NOT LAST ONE, CONTINUE
19430  115350  106427  000340                  MTPS    #340                    ;RESTORE PRIORITY 7
19431  115354  042737  000100  177564          BIC     #BIT06,XCSR             ;CLEAR INTERRUPT ENABLE BIT
19432                                          ;
19433                                          ; CHECK THAT RECEIVE INTERRUPTS DON'T HAPPEN AT PRIORITY HIGHER THAN 3
19434                                          ;
19435  115362  012737  115452  000060          MOV     #15$,@#60               ;POINT RECEIVE VECTOR TO PROGRAM AREA
19436  115370  012737  000340  000062          MOV     #340,@#62               ;AT PRIORITY 7
19437  115376  105737  177564          11$:    TSTB    XCSR                    ;TRANSMITTER READY
19438  115402  100375                          BPL     11$                     ;IF NOT, WAIT
19439  115404  012737  000000  177566          MOV     #NULL,XBUF              ;TRY TO TRANSMIT NULL
19440  115412  052737  000100  177560          BIS     #BIT06,RCSR             ;SET INTERRUPT ENABLE BIT IN RCSR
19441  115420  012702  000340                  MOV     #340,R2                 ;SET PRIORITY TO 7
19442  115424  000402                          BR      13$                     ;GO WAIT IN CASE OF INTERRUPTS
19443  115426  162702  000040          12$:    SUB     #40,R2                  ;LOWER PRIORITY LEVEL
19444  115432  052737  000004  177564  13$:    BIS     #BIT02,XCSR             ;SET LOOP BACK MODE
19445  115440  106402                          MTPS    R2                      ;SET PRIORITY
19446  115442  012703  000100                  MOV     #100,R3                 ;TIME DELAY
19447  115446  077301                  14$:    SOB     R3,14$                  ;WAIT FOR INTERRUPTS
19448  115450  000406                          BR      16$                     ;IF INTERRUPTS DIDN'T HAPPENED, BRANCH
19449  115452  042737  000004  177564  15$:    BIC     #BIT02,XCSR             ;CLEAR LOOP BACK MODE
19450  115460  104101                          ERROR   +101                    ;INTERRUPTS HAPPEN AT WRONG PRIORITY
19451  115462  005726                          TST     (SP)+                   ;CLEAN UP THE STACK
19452  115464  005726                          TST     (SP)+
19453  115466  022702  000200          16$:    CMP     #200,R2                 ;AT PRIORITY 4?
19454  115472  001355                          BNE     12$                     ;IF NOT LAST ONE, CONTINUE
19455                                          ;
19456                                          ; CLEAN UP BEFORE NEXT TEST
19457                                          ;
19458  115474  106427  000340                  MTPS    #340                    ;RESTORE PRIORITY 7
19459  115500  042737  000100  177560          BIC     #BIT06,RCSR             ;CLEAR INTERRUPT ENABLE BIT
19460  115506  012702  000300                  MOV     #300,R2                 ;STALL DELAY
19461  115512  105737  177560          17$:    TSTB    RCSR                    ;RECEIVE READY?
19462  115516  100401                          BMI     18$                     ;STOP WAITING, IF SO
19463  115520  077204                          SOB     R2,17$                  ;OTHERWISE, STAY IN THE LOOP
19464  115522  005737  177562          18$:    TST     RBUF                    ;READ CHARACTER TRANSMITTED
19465  115526  042737  000004  177564          BIC     #BIT02,XCSR             ;CLEAR LOOP BACK MODE
19466
```

H13


```
                                        .SBTTL  TEST  - INTERRUPT PRIORITY FOR SLU
                                        ;CHECK THAT INTERRUPTS HAPPEN AT PRIORITY 3 AND THAT THEY CLEAR
                                        ;RCSR<06> AND XCSR<06>.
                                        ;
                                        ;ROUTINE TEST
                                        ;.      LET 60=#ADDRESS_OF_LEGAL_RINTERRUPT
                                        ;.      LET 64=#ADDRESS_OF_LEGAL_XINTERRUPT
                                        ;.      LET XCSR<02>=#1
                                        ;.      SET PRIORITY TO #3
                                        ;.      WAIT FOR XINTERRUPT=#3
                                        ;.      IF XCSR<07> EQ #1 THEN
                                        ;.      .       ERROR
                                        ;.      ENDIF
                                        ;.      WAIT FOR RINTERRUPT=#3
                                        ;.      IF RCSR<07> EQ #0 THEN
                                        ;.      .       ERROR
                                        ;.      ENDIF
                                        ;.      LET XCSR<02>=#0
                                        ;.      SET PRIORITY TO NORMAL
                                        ;ENDROUTINE
                                        ;
                                        ;ROUTINE LEGAL_XINTERRUPT
                                        ;.      LET XBUF=#CHARACTER
                                        ;.      INCREMENT XINTERRUPT
                                        ;ENDROUTINE
                                        ;
                                        ;ROUTINE LEGAL_RINTERRUPT
                                        ;.      READ RCSR
                                        ;.      INCREMENT RINTERRUPT
                                        ;ENDROUTINE

                                        ;;*****************************************************************
115534  000004                          TST50:  SCOPE
115536  122737  000001  001220                  CMPB    #APTENV,$ENV                    ;RUNNING IN APT MODE?
115544  001003                                  BNE     100$                            ;NO, GO DO TEST
115546  005737  001206                          TST     $PASS                           ;FIRST PASS?
115552  001113                                  BNE     TST51           ;;IF APT AND NOT FIRST PASS, EXIT TEST
                                        ;
                                        ; GET READY FOR INTERRUPTS
                                        ;
115554  012737  115750  000060          100$:   MOV     #8$,@#60                        ;STORE RECEIVER VECTOR
115562  012737  115674  000064                  MOV     #6$,@#64                        ;STORE TRANSMITTER VECTOR
115570  012737  000340  000062                  MOV     #340,@#62                       ;AT PRIORITY 7
115576  012737  000340  000066                  MOV     #340,@#66                       ;FOR RECEIVER AND TRANSMITTER
115604  052737  000004  177564                  BIS     #BIT02,XCSR                     ;SET LOOP BACK MODE
115612  012701  000100                          MOV     #100,R1                         ;DELAY FOR UNEXPECTED CHARACTERS
115616  105737  177560                  1$:     TSTB    RCSR                            ;RECEIVER READY?
115622  100401                                  BMI     2$                              ;IF YES, BRANCH
115624  077104                                  SOB     R1,1$                           ;OTHERWISE, WAIT JUST IN CASE
115626  005737  177562                  2$:     TST     RBUF                            ;READ RECEIVER
                                        ;
                                        ; SET PRIORITIES AND XMIT INTERRUPTS
                                        ;
115632  012702  000140                          MOV     #140,R2                         ;START WITH PRIORITY 3
115636  000402                                  BR      4$                              ;TRY TO DO IT
115640  162702  000040                  3$:     SUB     #40,R2                          ;LOWER PRIORITY
```

```
19523  115644  106402                  4$:     MTPS    R2                      ;TRY TO DO AT LOWER PRIORITY
19524  115646  052737  000100  177564          BIS     #BIT06,XCSR             ;LOOP BACK & INTERRUPT ENABLE
19525  115654  012703  000100                  MOV     #100,R3                 ;WAIT DELAY FOR INTERRUPTS
19526  115660  077301                  5$:     SOB     R3,5$                   ;WAIT FOR XMIT INTERRUPTS
19527  115662  042737  000004  177564          BIC     #BIT02,XCSR             ;CLEAR LOOP BACK BIT
19528  115670  104107                          ERROR   +107                    ;NO XMIT INTERRUPTS
19529  115672  000443                          BR      TST51           ;;IF ERROR, EXIT TEST
19530                                  ;
19531                                  ; TRANSMITTER INTERRUPT HERE
19532                                  ;
19533  115674  005726                  6$:     TST     (SP)+                   ;CLEAN UP STACK
19534  115676  005726                          TST     (SP)+
19535  115700  042737  000100  177564          BIC     #BIT06,XCSR             ;CLEAR INTERRUPT ENABLE
19536  115706  012737  000000  177566          MOV     #NULL,XBUF              ;TRANSMIT NULL
19537  115714  052737  000100  177560          BIS     #BIT06,RCSR             ;SET RECEIVE INTERRUPT
19538  115722  106402                          MTPS    R2                      ;SET NEXT PRIORITY
19539  115724  012703  020000                  MOV     #20000,R3               ;WAIT DELAY FOR INTERRUPTS
19540  115730  077301                  7$:     SOB     R3,7$                   ;WAIT FOR RECEIVE INTERUPTS
19541  115732  042737  000004  177564          BIC     #BIT02,XCSR             ;CLEAR LOOP BACK MODE BIT
19542  115740  104107                          ERROR   +107                    ;NO RECEIVE INTERRUPTS
19543  115742  000406                          BR      9$                      ;DON'T TOUCH STACK
19544  115744  106427  000340                  MTPS    #340                    ;RAISE PRIORITY
19545                                  ;
19546                                  ; RECEIVER INTERRUPT HERE
19547                                  ;
19548  115750  005726                  8$:     TST     (SP)+                   ;CLEAN UP STACK
19549  115752  005726                          TST     (SP)+
19550  115754  005737  177562                  TST     RBUF                    ;READ RECEIVER BUFFER
19551  115760  005702                  9$:     TST     R2                      ;PRIORITY 0
19552  115762  001326                          BNE     3$                      ;IF NOT YET, CONTINUE
19553  115764  106427  000340                  MTPS    #340                    ;RAISE PRIORITY TO 7
19554  115770  042737  000100  177560          BIC     #BIT06,RCSR             ;CLEAR RECEIVE INTER. ENABLE
19555  115776  005037  177564                  CLR     XCSR                    ;CLEAR XCSR
19556
```

```
                                        .SBTTL  TEST  - BREAK CONDITION
                                        ;CHECK THAT SENDING BREAK CAUSES FRAMING ERROR.
                                        ;
                                        ;RCSR   <15>    ERROR
                                        ;       <13>    FRAMING ERROR
                                        ;       <11>    RECEIVED BREAK
                                        ;
                                        ;XCSR   <00>    TRANSMIT BREAK
                                        ;
                                        ;ROUTINE TEST
                                        ;.      LET XCSR<02>=#1
                                        ;.      LET XCSR<00>=#1
                                        ;.      WAIT FOR RCSR<07>=#1
                                        ;.      IF RBUF<15!13!11> NE #1 THEN
                                        ;.      .       ERROR (ERROR, FRAMING ERROR, RECEIVE BREAK NE 1)
                                        ;.      ENDIF
                                        ;.      LET XCSR<00>=#0
                                        ;.      IF XCSR<00> NE #0 THEN
                                        ;.      .       ERROR (XCSR<00> DOES NOT GO LOW)
                                        ;.      ENDIF
                                        ;.      WAIT FOR XCSR<07>=#1
                                        ;.      LET XBUF=#NULL (SEND NULL CHARACTER TO SEE ERROR CLEARED)
                                        ;.      WAIT FOR RCSR<07>=#1
                                        ;.      IF RBUF<15!13!11> NE #0 THEN
                                        ;.      .       ERROR
                                        ;.      ENDIF
                                        ;.      LET XCSR<00>=#1
                                        ;.      EXECUTE "RESET"
                                        ;.      IF XCSR<00> NE #0 THEN
                                        ;.      .       ERROR
                                        ;.      ENDIF
                                        ;.      LET XCSR<02>=#0
                                        ;ENDROUTINE


                                        ;;*****************************************************************
116002  000004                          TST51:  SCOPE
                                        ;
                                        ; DECIDE WHETHER TO RUN THIS TEST
                                        ;
116004  032737  000200  000052                  BIT     #BIT07,@#52                     ;UFD MODE?
116012  001127                                  BNE     TST52           ;;IN UFD MODE, EXIT TEST
116014  005737  001206                          TST     $PASS                           ;FIRST PASS?
116020  001124                                  BNE     TST52           ;;IF APT AND NOT FIRST PASS, EXIT TEST
                                        ;
                                        ; SEND BREAK AND CHECK ERROR BITS IN RBUF
                                        ;
116022  052737  000004  177564          1$:     BIS     #BIT02,XCSR                     ;TRANSMIT IN LOOP BACK
116030  013737  177520  002710                  MOV     BCSR,SAVBR                      ;SAVE BCSR
116036  042737  001000  177520                  BIC     #BIT09,BCSR                     ;DISABLE HALT ON BREAK
116044  052737  000001  177564                  BIS     #BIT00,XCSR                     ;SET SEND BREAK BIT
116052  032737  000001  177564                  BIT     #BIT00,XCSR                     ;GOT SET OK?
116060  001001                                  BNE     2$                              ;IF YES, BRANCH
116062  104110                                  ERROR   +110                            ;WRITING 1 TO XCSR<0>
116064  012701  000100                  2$:     MOV     #100,R1                         ;STALL DELAY
116070  105737  177560                  4$:     TSTB    RCSR                            ;RECEIVER READY?
116074  100401                                  BMI     5$                              ;IF YES, BRANCH
```

K13


```
116076  077104                          SOB     R1,4$                           ;WAIT JUST IN CASE OF A CHARACTER
116100  005737  177562          5$:     TST     RBUF                            ;READ A CHARACTER
116104  052737  000001  177564          BIS     #BIT00,XCSR                     ;TRANSMIT BREAK
116112  012701  001000                  MOV     #1000,R1                        ;ANOTHER DELAY TO GET BREAK
116116  077101                  6$:     SOB     R1,6$                           ;WAIT A WHILE
116120  105737  177560          7$:     TSTB    RCSR                            ;RECEIVER READY?
116124  100375                          BPL     7$                              ;IF NOT, WAIT
116126  013737  177562  001126          MOV     RBUF,$BDDAT                     ;STORE WHATEVER RECEIVED
116134  022737  124000  001126          CMP     #BIT15!BIT13!BIT11,$BDDAT       ;ALL ERROR BITS SET?
116142  001405                          BEQ     8$                              ;IF YES, BRANCH
116144  042737  000004  177564          BIC     #BIT02,XCSR                     ;RESET TO ENABLE SLU
116152  104105                          ERROR   +105                            ;BREAK DOES NOT CAUSE ERRORS
116154  000446                          BR      TST52           ;;EXIT
116156  042737  000001  177564  8$:     BIC     #BIT00,XCSR                     ;CLEAR TRANSMIT BREAK
116164  032737  000001  177564          BIT     #BIT00,XCSR                     ;GOT CLEARED OK?
116172  001405                          BEQ     9$                              ;IF YES, BRANCH
116174  042737  000004  177564          BIC     #BIT02,XCSR                     ;RESET TO ENABLE SLU
116202  104110                          ERROR   +110                            ;ERROR WRITING 0 TO XCSR<0>
116204  000432                          BR      TST52           ;;EXIT
                                ;
                                ; CHECK THAT BREAK CONDITION IS CLEARED
                                ;
116206  013737  002710  177520  9$:     MOV     SAVBR,BCSR                      ;RESTORE BCSR
116214  105737  177564          10$:    TSTB    XCSR                            ;XMIT READY?
116220  100375                          BPL     10$                             ;IF NOT, WAIT
116222  012737  000177  177566          MOV     #177,XBUF                       ;TRY TO TRANSMIT DELETE
116230  105737  177560          11$:    TSTB    RCSR                            ;RECEIVER READY
116234  100375                          BPL     11$                             ;IF NOT, WAIT
116236  013737  177562  001126          MOV     RBUF,$BDDAT                     ;STORE RECEIVE BUFFER
116244  032737  124000  001126          BIT     #BIT15!BIT13!BIT11,$BDDAT       ;ERRORS CLEARED?
116252  001404                          BEQ     12$                             ;IF YES, BRANCH
116254  042737  000004  177564          BIC     #BIT02,XCSR                     ;RESET TO ENABLE SLU
116262  104106                          ERROR   +106                            ;BREAK NOT CLEARED ON NEXT CHARACTER
116264  042737  000004  177564  12$:    BIC     #BIT02,XCSR                     ;CLEAR LOOP BACK MODE
```

```
 19649                                  .SBTTL  TEST  - OVERRUN CONDITION
 19650                                  ;CHECK OVERRUN CONDITION
 19651                                  ;
 19652                                  ;RCSR   <14>    OVERRUN ERROR
 19653                                  ;
 19654                                  ;ROUTINE TEST
 19655                                  ;.      LET XCSR<02>=#1 (LOOPBACK MODE)
 19656                                  ;.      WAIT FOR XCSR<07>=#1
 19657                                  ;.      LET XBUF=#252
 19658                                  ;.      WAIT FOR XCSR<07>=#1
 19659                                  ;.      LET XBUF=#125 (SEND THE 2ND W/O READING THE 1ST CHARACTER)
 19660                                  ;.      WAIT FOR RCSR<07>=#1
 19661                                  ;.      STALL FOR LOWEST BAUD RATE TO GET 2ND CHARACTER
 19662                                  ;.      IF LOW BYTE OF RBUF NE #125 THEN
 19663                                  ;.      .       ERROR (1ST CHARACTER WASN'T OVERRUN)
 19664                                  ;.      ENDIF
 19665                                  ;.      IF RBUF<15!14> NE #1 THEN
 19666                                  ;.      .       ERROR (NO OVERRUN BIT SET)
 19667                                  ;.      ENDIF
 19668                                  ;.      WAIT FOR XCSR<07>=#1
 19669                                  ;.      LET XBUF=#NULL
 19670                                  ;.      WAIT FOR RCSR<07>=#1
 19671                                  ;.      IF RBUF<15!14> NE #0 THEN
 19672                                  ;.      .       ERROR (WASN'T CLEARED ON THE NEXT CHARACTER RECEIVED)
 19673                                  ;.      ENDIF
 19674                                  ;.      LET XCSR<02>=#0
 19675                                  ;ENDROUTINE
 19676
 19677                                  ;;*****************************************************************
 19678  116272  000004                 TST52:  SCOPE
 19679  116274  122737  000001  001220         CMPB    #APTENV,$ENV                    ;RUNNING IN APT MODE?
 19680  116302  001003                         BNE     100$                            ;NO, GO DO TEST
 19681  116304  005737  001206                 TST     $PASS                           ;FIRST PASS?
 19682  116310  001077                         BNE     TST53           ;;IF APT AND NOT FIRST PASS, EXIT TEST
 19683  116312  052737  000004  177564 100$:   BIS     #BIT02,XCSR                     ;SET LOOP BACK MODE
 19684  116320  105737  177564         1$:     TSTB    XCSR                            ;READY TO TRANSMIT?
 19685  116324  100375                         BPL     1$                              ;IF NOT, WAIT
 19686  116326  012737  000021  177566         MOV     #21,XBUF                        ;TRANSMIT A CHARACTER
 19687  116334  105737  177560         2$:     TSTB    RCSR                            ;RECEIVE READY?
 19688  116340  100375                         BPL     2$                              ;IF NOT, WAIT
 19689  116342  105737  177564         3$:     TSTB    XCSR                            ;READY TO TRANSMIT?
 19690  116346  100375                         BPL     3$                              ;IF NOT, WAIT
 19691  116350  012737  000177  177566         MOV     #177,XBUF                       ;TRANSMIT THE 2ND CHARACTER
 19692  116356  012703  010000                 MOV     #10000,R3                       ;STALL FOR THE 2ND CHARACTER
 19693  116362  077301                 4$:     SOB     R3,4$                           ;WAIT A WHILE
 19694  116364  013737  177562  001126         MOV     RBUF,$BDDAT                     ;STORE RECEIVED DATA
 19695  116372  012737  140177  001124         MOV     #140177,$GDDAT                  ;EXPETED PATTERN
 19696  116400  122737  000177  001126         CMPB    #177,$BDDAT                     ;2ND CHARACTER RECEIVED?
 19697  116406  001404                         BEQ     5$                              ;IF YES, BRANCH
 19698  116410  042737  000004  177564         BIC     #BIT02,XCSR                     ;RESET TO ENABLE SLU
 19699  116416  104111                         ERROR   +111                            ;2ND CHARACTER DIDN'T OVERRUN 1ST
 19700  116420  122737  000300  001127 5$:     CMPB    #BIT7!BIT6,$BDDAT+1             ;OVERRUN ERROR BITS SET?
 19701  116426  001404                         BEQ     6$                              ;IF YES, BRANCH
 19702  116430  005037  177564                 CLR     XCSR                            ;RESET TO ENABLE SLU
 19703  116434  104112                         ERROR   +112                            ;OVERRUN DOES NOT SET ERRORS BITS
 19704  116436  000424                         BR      TST53           ;;EXIT
```

```
                                        ;
                                        ; SEND NEXT CHARACTER TO CLEAR OVERRUN CONDITIONS
                                        ;
116440  105737  177564                  6$:     TSTB    XCSR                            ;TRANSMITTER READY?
116444  100375                                  BPL     6$                              ;IF NOT, BRANCH AND WAIT
116446  012737  000000  177566                  MOV     #NULL,XBUF                      ;TRANSMIT NULL CHARACTER
116454  105737  177560                  7$:     TSTB    RCSR                            ;RECEIVER READY?
116460  100375                                  BPL     7$                              ;IF NOT, BRANCH AND WAIT
116462  032737  140000  177562                  BIT     #BIT15!BIT14,RBUF               ;ANY ERRORS SET?
116470  001404                                  BEQ     8$                              ;IF NOT, BRANCH
116472  042737  000004  177564                  BIC     #BIT02,XCSR                     ;RESET TO ENABLE SLU
116500  104113                                  ERROR   +113                            ;OVERRUN NOT CLEARED ON NEXT CHAR.
116502  042737  000004  177564          8$:     BIC     #BIT02,XCSR                     ;CLEAR LOOP BACK MODE BIT

116510                                  SLEND:                                          ;LAST SLU TEST
```

```
                                        .SBTTL  TEST  - LED'S ON
                                        ;LED'S ON
                                        ;THIS TEST WILL INITIALIZE BDR TO CONTAIN A ROTATING PATTERN
                                        ;DISPLAYED IN LED'S.
                                        ;
                                        ;ROUTINE TEST
                                        ;.      WHILE A KEY NOT RECEIVED FROM KEYBOARD DO
                                        ;.      .       STALL ALLOWING TIME TO SEE PATTERN
                                        ;.      .       ROTATE LEFT TO LIGHT UP NEXT LED'S
                                        ;.      ENDDO
                                        ;ENDROUTINE

                                        ;;*****************************************************************
116510  000004                  TST53:  SCOPE
116512  005005                          CLR     R5                      ;FLAG IN NO INTERRRUPT MODE
116514  032737  000001  000052          BIT     #BIT00,@#52             ;IF RUNNING IN CHAIN MODE
116522  001427                          BEQ     1$                      ;SKIP PRINTOUTS
116524  005737  001206                  TST     $PASS                   ;1ST PASS?
116530  001024                          BNE     1$                      ;IF NOT, SKIP PRINTOUTS
116532  122737  000001  001220          CMPB    #APTENV,$ENV            ;APT MODE?
116540  001420                          BEQ     1$                      ;YES, SKIP PRINTOUT'S
116542  005105                          COM     R5                      ;CLEAR FLAG IN INTERRUPT MODE
116544  104401  001175                  TYPE    ,$CRLF
116550  104401  116672                  TYPE    ,LEDS                   ;IDENTIFY THE TEST
116554  012737  116652  000060          MOV     #5$,@#60                ;RECEIVE SLU VECTOR
116562  012737  000340  000062          MOV     #340,@#62               ;AT PRIORITY 7
116570  052737  000100  177560          BIS     #BIT06,RCSR             ;ENABLE INTERRUPTS
116576  106427  000140                  MTPS    #140                    ;LOWER PRIORITY
116602  012704  000006          1$:     MOV     #6,R4                   ;FOR EACH LOOP
116606  012701  000076                  MOV     #76,R1                  ;START WITH 1
116612  110137  177524          2$:     MOVB    R1,BDR                  ;TURN OFF FIRST LED
116616  012703  000004                  MOV     #4,R3                   ;STALL DELAY
116622  012702  177777          3$:     MOV     #177777,R2              ;STALL DELAY
116626  077201                  4$:     SOB     R2,4$                   ;WAIT A WHILE
116630  077304                          SOB     R3,3$                   ;WAIT A WHILE
116632  000261                          SEC                             ;SET CARRY
116634  006101                          ROL     R1                      ;GET ANOTHER LED
116636  077413                          SOB     R4,2$                   ;DO A FEW TIMES
116640  005705                          TST     R5                      ;RUNNING IN INTERACTIVE MODE?
116642  001407                          BEQ     6$                      ;IF NOT, EXIT
116644  104401  001170                  TYPE    ,$BELL                  ;
116650  000754                          BR      1$                      ;REPEAT PATTERN
116652  005737  177562          5$:     TST     RBUF                    ;READ BUFFER
116656  062706  000004                  ADD     #4,SP                   ;ADJUST STACK
116662  112737  000377  177524  6$:     MOVB    #377,BDR                ;NO MORE
116670  000452                          BR      TST54           ;;EXIT TEST

116672  006412  044124  051511  LEDS:   .ASCII  <12><15>/THIS IS A TEST FOR ON-BOARD LED'S/<12><15>
116700  044440  020123  020101
116706  042524  052123  043040
116714  051117  047440  026516
116722  047502  051101  020104
116730  042514  023504  005123
116736     015
116737     124  050131  020105          .ASCIZ  /TYPE ANY CHARACTER ON A KEYBOARD TO CONTINUE/<12><15>
116744  047101  020131  044103
```

```
19776   116752  051101  041501  042524
19777   116760  020122  047117  040440
19778   116766  045440  054505  047502
19779   116774  051101  020104  047524
19780   117002  041440  047117  044524
19781   117010  052516  005105  000015
19782                           .EVEN
19783
```

```
                                        .SBTTL  TEST  - MEMORY MAPPING
                                        ;MEMORY MAPPING
                                        ;THIS TEST WILL AUTOSIZE MEMORY IN 2K BYTES. EVERY PAGE WILL BE
                                        ;CHECKED FOR WHAT TYPE IT IS: Q-BUS, UNIBUS OR PMI BUS MEMORY BY
                                        ;SEEING HOW IT CAN BE CACHED.
                                        ;ROUTINE TEST
                                        ;.  LET 4=ADDRESS OF NON-EXISTENT_MEMORY
                                        ;.  IF KMCR<05-00> NE #1 THEN (NOT ALL UNIBUS MEMORY)
                                        ;.      TURN ON MMU AND REMAP THE PROGRAM AREA
                                        ;.      REPEAT
                                        ;.      .  DO FOR KDPARO FROM #0 TO #177600
                                        ;.      .       DO FOR R1 FROM #0 TO #20000 BY #4000
                                        ;.      .       .       READ (R1)
                                        ;.      .       .       IF ABORT_NON-EXISTENT_MEMORY NE 1
                                        ;.      .       .       .       MEMORY EXISTS
                                        ;.      .       .       .       READ (R1)+
                                        ;.      .       .       .       READ (R1)
                                        ;.      .       .       .       IF HIT/MISS EQ 2_HITS THEN
                                        ;.      .       .       .       .       PMI MEMORY
                                        ;.      .       .       .       ELSE
                                        ;.      .       .       .       .       IF HIT/MISS EQ HIT THEN
                                        ;.      .       .       .       .       .       Q-BUS MEMORY
                                        ;.      .       .       .       .       ELSE
                                        ;.      .       .       .       .       .       ERROR
                                        ;.      .       .       .       .       ENDIF
                                        ;.      .       .       .       ENDIF
                                        ;.      .       .       ENDIF
                                        ;.      .       .       LET ABORT_NON-EXISTENT_MEMORY=#0
                                        ;.      .       ENDDO
                                        ;.      .  ENDDO
                                        ;.      UNTILL ABORT_NON-EXISTANT_MEMORY EQ #1
                                        ;.  ENDIF
                                        ;.
                                        ;ENDROUTINE
                                        ;
                                        ;ROUTINE NON-EXISTENT_MEMORY
                                        ;.      LET ABORT_NON-EXISTENT_MEMORY=#1
                                        ;.      RETURN
                                        ;ENDROUTINE


                                        ;;*****************************************************************
117016  000004                          TST54:  SCOPE
117020  032737  000001  000052                  BIT     #BIT00,@#52                     ;CHAIN MODE?
117026  001561                                  BEQ     TST55           ;;IF SO, EXIT TEST
117030  005737  001206                          TST     $PASS                           ;FIRST PASS?
117034  001156                                  BNE     TST55           ;;IF APT AND NOT FIRST PASS, EXIT TEST
117036  122737  000001  001220                  CMPB    #APTENV,$ENV                    ;APT MODE?
117044  001552                                  BEQ     TST55           ;;YES, SKIP PRINTOUT'S
                                        ;
                                        ; SETUP ALL REGISTERS FOR MAPPING
                                        ;
117046  012737  000400  177746                  MOV     #400,CCR                        ;FLUSH THE CACHE
117054  013704  000004                          MOV     ERRVEC,R4                       ;STORE TIMEOUT VECTOR
117060  012737  117230  000004                  MOV     #7$,@#ERRVEC                    ;POINT TO PROGRAM AREA
117066  012737  000340  000006                  MOV     #340,ERRVEC+2                   ;AT PRIORITY 4
117074  004737  132156                          JSR     PC,INITMM                       ;REMAP PROGRAM AREA
```

```
19840  117100  005037  172354            CLR     KIPAR6                  ;USED FOR MAPPING MEMORY
19841  117104  005002                    CLR     R2                      ;CLEAR PAGE COUNT FOR PMI
19842  117106  005003                    CLR     R3                      ;CLEAR PAGE COUNT FOR QBUS
19843  117110  005237  177572            INC     MMR0                    ;ENABLE MEMORY MANGEMENT
19844  117114  052737  000020  172516    BIS     #BIT04,MMR3             ;ENABLE 22 BITS
19845  117122  000403                    BR      2$                      ;GO ACCESS
19846                                    ;
19847                                    ; TRY TO MAP ALL PAGES
19848                                    ;
19849  117124  062737  000200  172354  1$: ADD     #200,KIPAR6             ;INCREMENT BY 4K WORDS
19850  117132  012701  140000          2$: MOV     #140000,R1              ;ACCESS THRU KIPAR6
19851  117136  000402                    BR      4$                      ;START DOING IT
19852  117140  062701  003776          3$: ADD     #3776,R1                ;INCREMENT BY 1K WORDS
19853  117144  005721                  4$: TST     (R1)+                   ;ACCESS 1ST LOCATION
19854  117146  042737  001000  177520    BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
19855  117154  005711                    TST     (R1)                    ;ACCESS 2ND LOCATION
19856  117156  013737  177752  110504    MOV     HITMIS,RECDAT           ;STORE REGISTER
19857  117164  052737  001000  177520    BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
19858  117172  032737  000004  110504    BIT     #BIT02,RECDAT           ;LAST (R1) HIT?
19859  117200  001402                    BEQ     5$                      ;IF NOT, QBUS MEMORY
19860  117202  005202                    INC     R2                      ;INCREMENT 1K COUNT FOR PMI
19861  117204  000401                    BR      6$                      ;GO CONITNUE
19862  117206  005203          5$: INC     R3                      ;INCREMENT COUNT FOR QBUS MEM.
19863  117210  022701  154000          6$: CMP     #154000,R1              ;LAST IN 4K PAGE?
19864  117214  101351                    BHI     3$                      ;IF NOT, BRANCH
19865  117216  022737  177600  172354    CMP     #177600,KIPAR6          ;2M BOUNDARY?
19866  117224  001337                    BNE     1$                      ;IF NOT, BRANCH
19867  117226  000402                    BR      8$                      ;IF 2M, DON'T TOUCH STACK
19868                                    ;
19869                                    ; MAPPING IS DONE, FIND OUT HOW MANY PAGES WERE THERE
19870                                    ;
19871  117230  005726          7$: TST     (SP)+                   ;ADJUST STACK
19872  117232  005726                    TST     (SP)+
19873  117234  010437  000004  8$: MOV     R4,ERRVEC               ;RESTORE ERROR VECTOR
19874  117240  005037  177572            CLR     MMR0                    ;DISABLE MEMORY MANGEMENT
19875  117244  042737  000020  172516    BIC     #BIT04,MMR3             ;DISABLE 22 BITS
19876  117252  005702                    TST     R2                      ;ANY PMI MEMORY?
19877  117254  001405                    BEQ     9$                      ;IF NOT, GO CHECK QBUS MEMORY
19878  117256  006302                    ASL     R2                      ;TRANSLATE TO BYTES
19879  117260  010246                    MOV     R2,-(SP)                ;STORE 1K # ON STACK
19880  117262  104405                    TYPDS                           ;TYPE # OF PAGES
19881  117264  104401  117310            TYPE    ,MEMK                   ;TYPE ASCII
19882  117270  005703          9$: TST     R3                      ;ANY Q-BUS MEMORY?
19883  117272  001405                    BEQ     10$                     ;IF NOT, BRANCH
19884  117274  006303                    ASL     R3                      ;TRANSLATE TO BYTES
19885  117276  010346                    MOV     R3,-(SP)                ;STORE 1K # ON STACK
19886  117300  104405                    TYPDS                           ;TYPE # OF PAGES
19887  117302  104401  117340            TYPE    ,MEMQ                   ;TYPE ASCII
19888  117306                        10$:
19889  117306  000431                    BR      TST55           ;;EXIT TEST
19890
19891  117310  020113  054502  042524  MEMK:   .ASCIZ  /K BYTES OF PMI MEMORY/<12><15>
19892  117316  020123  043117  050040
19893  117324  044515  046440  046505
19894  117332  051117  005131  000015
19895  117340  020113  054502  042524  MEMQ:   .ASCIZ  /K BYTES OF Q-BUS MEMORY/<12><15>
```

```
19896  117346  020123  043117  050440
19897  117354  041055  051525  046440
19898  117362  046505  051117  005131
19899  117370  000015
19900                                  .EVEN
19901
19902                                  .SBTTL WRONG PARITY ABORT TEST
19903                                  ;WRONG PARITY ABORT
19904                                  ;THIS TEST VERIFIES ABORT TO 114 USING PARITY OR ECC MEMORY CSR.
19905                                  ;IF MORE THEN 1 CSR PRESENT, ALL OF THEM WILL BE WRITTEN AT THE
19906                                  ;SAME TIME.
19907                                  ;
19908                                  ;ROUTINE TEST
19909                                  ;.      SIZE FOR ALL POSSIBLE MEMORY CSR'S
19910                                  ;.      STORE UP TO 16 CSR STARTING FROM TEMP
19911                                  ;.      WRITE ALL WITH WRONG PARITY
19912                                  ;.      WRITE TO 0
19913                                  ;.      READ IT BACK
19914                                  ;.      IF NO ABORT TO 114 THEN
19915                                  ;.              ERROR IN PARITY ABORT LOGIC
19916                                  ;.      ENDIF
19917                                  ;.      RESTORE CSR'S
19918                                  ;ENDROUTINE
19919
19920                                  ;;*****************************************************************
19921  117372  000004                  TST55:  SCOPE
19922                                  ;
19923                                  ; FIND OUT ALL POSSIBLE MEMORY CSR LOCATIONS UP TO 16
19924                                  ;
19925  117374  013701  000004                  MOV     ERRVEC,R1               ;STORE TIMEOUT VECTOR
19926  117400  012737  117436  000004          MOV     #2$,ERRVEC              ;POINT NEW TO PROGRAM
19927  117406  012737  000340  000006          MOV     #340,ERRVEC+2           ;AT PRIORITY 7
19928  117414  005004                          CLR     R4                      ;COUNT FOR CSR'S
19929  117416  012702  172100                  MOV     #172100,R2              ;FIRST POSSIBLE CSR
19930  117422  012703  002710                  MOV     #TEMP,R3                ;STORAGE LOCATION
19931  117426  005712                  1$:     TST     (R2)                    ;IS CSR THERE?
19932  117430  010223                          MOV     R2,(R3)+                ;IF THERE, STORE
19933  117432  005204                          INC     R4                      ;INCREMENT COUNT FOR CSR'S
19934  117434  000402                          BR      3$                      ;BRANCH AROUND
19935  117436  005726                  2$:     TST     (SP)+                   ;RESTORE STACK
19936  117440  005726                          TST     (SP)+
19937  117442  062702  000002          3$:     ADD     #2,R2                   ;POINT TO NEW CSR
19938  117446  022702  172136                  CMP     #172136,R2              ;ALL DONE?
19939  117452  101365                          BHI     1$                      ;IF NOT, BRANCH
19940  117454  010137  000004                  MOV     R1,ERRVEC               ;RESTORE ERROR VECTOR
19941  117460  052737  001000  177746          BIS     #BIT09,CCR              ;SET CACHE BYPASS
19942                                  ;
19943                                  ; WRITE ALL CSR'S WITH WRONG ECC CODE
19944                                  ; NOTE: IN PARITY MEMORY THOSE BITS ARE READ ONLY AND
19945                                  ;       DIAGNOSTIC MODE BIT FOR ECC IS THE SAME AS WRONG PARITY
19946                                  ;
19947  117466  013701  000114                  MOV     @#114,R1                ;STORE PARITY ABORT VECTOR
19948  117472  012737  117560  000114          MOV     #6$,@#114               ;POINT NEW TO PROGRAM
19949  117500  012737  000340  000116          MOV     #340,@#116              ;AT PRIORITY 7
19950  117506  010402                          MOV     R4,R2                   ;STORE CSR COUNT
19951  117510  012703  002710                  MOV     #TEMP,R3                ;START WITH 1ST CSR
```

F14


```
117514  052773  000005  000000  4$:     BIS     #BIT02!BIT00,@0(R3)          ;DIAGNOSTIC OR WRONG PARITY
117522  042733  003740                  BIC     #3740,@(R3)+                 ;CLEAR <10-5> CHECK BITS
117526  077406                          SOB     R4,4$                        ;DO FOR ALL CSR'S
117530  005037  000000                  CLR     @#0                          ;CLEAR TEST LOCATION WITH WRONG PR
117534  010204                          MOV     R2,R4                        ;RESTORE COUNTER
117536  012703  002710                  MOV     #TEMP,R3                     ;START WITH 1ST CSR
117542  042733  000004          5$:     BIC     #BIT02,@(R3)+                ;CLEAR ALL WRONG PARITY
117546  077203                          SOB     R2,5$                        ;IN ALL CSR'S
117550  005737  000000                  TST     @#0                          ;READ BACK WRONG PARITY
117554  104034                          ERROR   +34                          ;NO WRONG PARITY ABORT
117556  000402                          BR      7$
117560  005726                  6$:     TST     (SP)+                        ;ADJUST STACK
117562  005726                          TST     (SP)+
117564  012703  002710          7$:     MOV     #TEMP,R3                     ;START WITH 1ST CSR
117570  042733  000001          8$:     BIC     #BIT00,@(R3)+                ;CLEAR ALL WRONG PARITY
117574  077403                          SOB     R4,8$                        ;IN ALL CSR'S
117576  032737  100000  177744          BIT     #BIT15,MSER                  ;ABORT REFLECTED IN MSER?
117604  001004                          BNE     9$                           ;IF YES, BRANCH
117606  012737  100000  001124          MOV     #100000,$GDDAT               ;
117614  104035                          ERROR   +35                          ;MSER<15> NOT SET ON PARITY ABORT
117616  042737  001000  177746  9$:     BIC     #BIT09,CCR                   ;CLEAR CACHE BYPASS
117624  005037  000000                  CLR     @#0                          ;CLEAR WRONG PARITY
117630  005037  177744                  CLR     MSER                         ;CLEAR MSER
117634  010137  000114                  MOV     R1,@#114                     ;RESTORE PARITY ABORT
```

```
                                       .SBTTL  TEST  - DMA TAG PARITY IN STANDALONE MODE
                                       ;CHECK DMA TAG STORE PARITY BIT.
                                       ;ROUTINE TEST
                                       ;.      CACHE DMA_PARITY
                                       ;.      LET BCSR<08>=#1
                                       ;.      REPORT ALL ERRORS
                                       ;ENDROUTINE
                                       ;
                                       ;ROUTINE DMA_PARITY
                                       ;.      GENERATE PARITY ERRORS
                                       ;.      CHECK MSER<13>
                                       ;.      LET BCSR<08>=#0
                                       ;ENDROUTINE

                                       ;;*****************************************************************
117640  000004                         TST56:  SCOPE
                                       ;
                                       ; ALLOCATE CODE IN CACHE
                                       ;
117642  012702  117670                         MOV     #DMAPAR,R2              ;POINT TO STANDALONE CODE
117646  005722                       1$:       TST     (R2)+                   ;ALLOCATE IN CACHE
117650  022702  117730                         CMP     #DPAREN,R2              ;LAST ADDRESS?
117654  001374                               BNE     1$                      ;IF NOT, BRANCH
117656  005737  002710                       TST     TEMP                    ;ALLOCATE TEST LOCATION
117662  013737  177520  002710               MOV     BCSR,SAVBR              ;SAVE BCSR
                                       ;
                                       ; IN STANDALONE MODE TRY TO VERIFY RESPONSE TO PARITY ERRORS
                                       ;
117670  052737  000400  177520       DMAPAR: BIS     #BIT08,BCSR             ;SET STANDALONE MODE BIT
117676  052737  002001  177746               BIS     #BIT10!BIT00,CCR        ;WRITE WRONG TAG PARITY
117704  005037  002710                       CLR     TEMP                    ;WRITE HIT WITH WRONG PARITY
117710  013705  177744                       MOV     MSER,R5                 ;STORE MSER
117714  042737  002000  177746               BIC     #BIT10,CCR              ;CLEAR WRONG PARITY BIT
117722  042737  000400  177520       2$:     BIC     #BIT08,BCSR             ;CLEAR STANDALONE BIT
                                       ;
                                       ; RETURN FROM STANDALONE MODE
                                       ;
117730                                 DPAREN:
117730  022705  060020                         CMP     #60020,R5               ;WRONG DMA PARITY?
117734  001401                               BEQ     4$                      ;IF OK, BRANCH
117736  104117                               ERROR   +117                    ;MSER NOT SET IN STANDALONE MODE
117740  005037  177744               4$:     CLR     MSER
117744  012737  000400  177746               MOV     #400,CCR                ;FLUSH THE CACHE
117752  013737  002710  177520               MOV     SAVBR,BCSR              ;RESTORE BCSR
```

```
20023                                   .SBTTL  TEST  - DMA TAG PARITY W/O STANDALONE MODE
20024                                   ;CHECK DMA TAG PARITY BIT WITHOUT GOING INTO STANDALONE MODE.
20025                                   ;
20026                                   ;CCR    <10>    WRITE WRONG TAG PARITY
20027                                   ;
20028                                   ;ROUTINE TEST
20029                                   ;.      LET CCR<10>=#1
20030                                   ;.      ALLOCATE LOCATION IN CACHE
20031                                   ;.      INITIATE DMA WRITE TRANSFERS
20032                                   ;.      IF MSER<04> NE #1 THEN
20033                                   ;.          ERROR
20034                                   ;.      ENDIF
20035                                   ;ENDROUTINE
20036
20037                                   ;;*****************************************************************
20038  117760  000004                  TST57:  SCOPE
20039  117762  032737  000200  000052          BIT     #BIT07,@#52                     ;UFD MODE?
20040  117770  001402                          BEQ     110$                            ;IF NOT, BRANCH
20041  117772  000137  133514                  JMP     $EOP                            ;OTHERWISE, NEXT PASS
20042  117776  005737  002644          110$:   TST     CSR1                            ;AT LEAST ONE Q22BE FOUND?
20043  120002  001457                          BEQ     TST60           ;;IF NOT, EXIT TEST
20044                                   ;
20045                                   ; ALLOCATE TEST IN CACHE
20046                                   ;
20047  120004  012703  120004          11$:    MOV     #.,R3                           ;START WITH CURRENT INSTRUCTION
20048  120010  005723                  10$:    TST     (R3)+                           ;READ A WORD
20049  120012  022703  120122                  CMP     #4$,R3                          ;ALL DONE?
20050  120016  001374                          BNE     10$                             ;IF NOT, CONTINUE
20051                                   ;
20052                                   ; WRITE A WORD WITH WRONG PARITY
20053                                   ;
20054  120020  013737  000114  001160  1$:     MOV     @#114,$TMP0                     ;STORE PARITY VECTOR
20055  120026  012737  120074  000114          MOV     #2$,@#114                       ;POINT TO TEST AREA
20056  120034  052737  002000  177746          BIS     #BIT10,CCR                      ;WRITE WRONG TAG PARITY
20057  120042  005037  002710                  CLR     TEMP                            ;WRITE MISS WITH WRONG TAG PARITY
20058  120046  042737  002000  177746          BIC     #BIT10,CCR                      ;CLEAR WRONG TAG PARITY
20059  120054  052737  000200  177746          BIS     #BIT07,CCR                      ;PARITY ABORT
20060  120062  005000                          CLR     R0                              ;FLAG TO DO 1 TRANSFER
20061  120064  004737  132752                  JSR     PC,DMATRN                       ;DO DMA WRITE TO TEMP THRU Q22BE
20062  120070  104116                          ERROR   .116                            ;NO PARITY ABORT
20063  120072  000413                          BR      4$                              ;BRANCH TO TEST MSER
20064  120074  013704  177744          2$:     MOV     MSER,R4                         ;STORE REGISTER
20065  120100  005037  177744                  CLR     MSER                            ;CLEAR MSER
20066  120104  005726                          TST     (SP)+                           ;
20067  120106  005726                          TST     (SP)+                           ;RESTORE STACK
20068  120110  005726                          TST     (SP)+
20069  120112  032704  000020          3$:     BIT     #BIT04,R4                       ;MSER OK?
20070  120116  001001                          BNE     4$                              ;IF SET, BRANCH
20071  120120  104116                          ERROR   .116                            ;MSER<4> NOT SET
20072  120122  005037  177744          4$:     CLR     MSER                            ;CLEAR MSER
20073  120126  012737  000400  177746          MOV     #400,CCR                        ;FLUSH CACHE
20074  120134  013737  001160  000114          MOV     $TMP0,@#114                     ;RESTORE VECTOR
```

```
                                       .SBTTL  TEST  - DMA WRITE HIT CYCLES
                                       ;CHECK THAT DMA WRITE HITS INVALIDATE CACHE.
                                       ;ROUTINE TEST
                                       ;.      ALLOCATE A LOCATION IN CACHE
                                       ;.      INITIATE DMA WRITE TO THIS LOCATION
                                       ;.      READ THIS LOCATION BACK
                                       ;.      IF IT IS CHANGED OR HIT/MISS EQ HIT
                                       ;.          ERROR
                                       ;.      ENDIF
                                       ;.      WRITE TO THE FIRST 16 LOCATION THEIR ADDRESS
                                       ;.      DO BLOCK MODE TRANSFER TO THOSE LOCATIONS
                                       ;.      START READ WITH THE LAST LOCATION
                                       ;.      IF RECORD ANY HITS THEN
                                       ;.          ERROR
                                       ;.      ENDIF
                                       ;.      INITIATE READ DMA TO THE SAME TEST LOCATIONS
                                       ;.      READ THEM BACK
                                       ;.      IF HIT/MISS NE HIT THEN
                                       ;.          ERROR
                                       ;.      ENDIF
                                       ;ENDROUTINE

                                       ;;*****************************************************************
120142  000004                 TST60:  SCOPE
120144  032737  000200  000052         BIT     @#BIT07,@#52                    ;UFD MODE?
120152  001402                       BEQ     1$                              ;IF NOT, BRANCH
120154  000137  133514               JMP     $EOP                            ;OTHERWISE, NEXT PASS
120160  005737  002644         1$:   TST     CSR1                            ;AT LEAST 1 Q22BE?
120164  001002                       BNE     2$                              ;IF YES, BRANCH
120166  000137  133514               JMP     $EOP                            ;OTHERWISE, NEXT PASS
                                       ;
                                       ; TRY TO DO DMA WRITE AT ALL DIFFERENT PRIORITIES
                                       ;
120172  012702  000340         2$:   MOV     #340,R2                         ;START WITH 7
120176  000402                       BR      4$                              ;GO DO IT
120200  162702  000040         3$:   SUB     #40,R2                          ;LOWER PRIORITY
120204  005037  002710         4$:   CLR     TEMP                            ;CLEAR TEST LOCATION
120210  005000                       CLR     R0                              ;DO JUST 1 WORD
120212  106402                       MTPS    R2                              ;LOWER PRIORITY
120214  004737  132752               JSR     PC,DMATRN                       ;DO DATO
120220  042737  001000  177520       BIC     #1000,BCSR                      ;DISABLE HALT ON BREAK
120226  005737  002710               TST     TEMP                            ;STILL CACHED?
120232  013737  177752  110504       MOV     HITMIS,RECDAT                   ;STORE HIT/MISS
120240  052737  001000  177520       BIS     #1000,BCSR                      ;ENABLE HALT ON BREAK
120246  032737  000004  110504       BIT     @#BIT02,RECDAT                  ;LAST ACCESS HIT?
120254  001401                       BEQ     5$                              ;IF MISS, BRANCH
120256  104120                       ERROR   .120
120260  022737  012525  002710 5$:   CMP     #12525,TEMP                     ;DATO OK?
120266  001401                       BEQ     6$                              ;IF SO, BRANCH
120270  104123                       ERROR   .123                            ;DATO
120272  005702                 6$:   TST     R2                              ;LAST PRIORITY 0?
120274  001341                       BNE     3$                              ;IF NOT, BRANCH
120276  106427  000340               MTPS    @#340                           ;RESTORE PRIORITY
                                       ;
                                       ; VERIFY THAT DMA WITH FORCE MISS DOES NOT INVALIDATE CACHE
                                       ;
```

```
20131  120302  005037  002710                  CLR     TEMP                    ;ALLOCATE CACHE
20132  120306  012737  000004  177746          MOV     #BIT2,CCR               ;SET FORCE MISS
20133  120314  004737  132752                  JSR     PC,DMATRN               ;DO DMA DATI
20134  120320  005037  177746                  CLR     CCR                     ;CLEAR FORCE MISS
20135  120324  042737  001000  177520          BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
20136  120332  005737  002710                  TST     TEMP                    ;STILL IN CACHE
20137  120336  013737  177752  110504          MOV     HITMIS,RECDAT           ;STORE REGISTER
20138  120344  052737  001000  177520          BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
20139  120352  032737  000004  110504          BIT     #BIT02,RECDAT           ;HIT?
20140  120360  001001                          BNE     7$                      ;IF SO, BRANCH
20141  120362  104123                          ERROR   +123
20142  120364  005737  002710          7$:     TST     TEMP                    ;READ FROM CACHE?
20143  120370  001401                          BEQ     8$                      ;IF STILL 0, BRANCH
20144  120372  104123                          ERROR   +123
20145                                  ;
20146                                  ; VERIFY THAT BYPASS WITH DMA DATI INVALIDATES CACHE
20147                                  ;
20148  120374  012737  001000  177746  8$:     MOV     #BIT09,CCR              ;SET BYPASS
20149  120402  004737  132752                  JSR     PC,DMATRN               ;DO DMA DATI
20150  120406  005037  177746                  CLR     CCR                     ;CLEAR BYPASS
20151  120412  042737  001000  177520          BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
20152  120420  005737  002710                  TST     TEMP                    ;IN CACHE?
20153  120424  013737  177752  110504          MOV     HITMIS,RECDAT           ;STORE REGISTER
20154  120432  052737  001000  177520          BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
20155  120440  032737  000004  110504          BIT     #BIT02,RECDAT           ;TEMP WAS A HIT?
20156  120446  001401                          BEQ     DATBO                   ;IF NOT, BRANCH
20157  120450  104123                          ERROR   +123
20158                                  ;
20159                                  ; DO DATBO
20160                                  ;
20161  120452  012701  000010          DATBO:  MOV     #10,R1                  ;COUNTER FOR 8
20162  120456  012702  002710                  MOV     #TEMP,R2                ;START WITH TEMP
20163  120462  005022                  1$:     CLR     (R2)+                   ;CLEAR ALL 16
20164  120464  077102                          SOB     R1,1$                   ;DO ALL 16
20165  120466  005200                          INC     R0                      ;FLAG BLOCK MODE
20166  120470  012777  002710  062152          MOV     #TEMP,@BA               ;LOAD DMA ADDRESS
20167  120476  012777  177770  062146          MOV     #177770,@WC             ;DO 8 WORDS
20168  120504  012777  001701  062132          MOV     #1701,@CSR1             ;16 WORDS FROM 32K
20169  120512  004737  132752                  JSR     PC,DMATRN               ;DO DATBO
20170  120516  032777  010000  062122          BIT     #BIT12,@CSR2            ;NO BLOCK MODE SLAVE?
20171  120524  001401                          BEQ     2$                      ;IF NOT, BRANCH
20172  120526  104123                          ERROR   +123                    ;NO BLOCK MODE SLAVE
20173  120530  012701  000004          2$:     MOV     #4,R1                   ;COUNTER FOR 4 LOCATIONS
20174  120534  012702  002710                  MOV     #TEMP,R2                ;START WITH TEMP
20175  120540  042737  001000  177520  3$:     BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
20176  120546  005712                          TST     (R2)                    ;ACCESS A LOCATION
20177  120550  013737  177752  110504          MOV     HITMIS,RECDAT           ;STORE REGISTER
20178  120556  052737  001000  177520          BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
20179  120564  032737  000004  110504          BIT     #BIT02,RECDAT           ;HIT?
20180  120572  001401                          BEQ     4$                      ;IF NOT, BRANCH
20181  120574  104121                          ERROR   +121                    ;DMA DOES NOT INVALIDATE
20182  120576  022722  012525          4$:     CMP     #12525,(R2)+            ;DATO OK?
20183  120602  001401                          BEQ     5$                      ;IF SO,BRANCH
20184  120604  104123                          ERROR   +123                    ;IN DMA
20185  120606  062702  000002          5$:     ADD     #2,R2                   ;DO IN 2 WORDS
20186  120612  077126                          SOB     R1,3$                   ;DO ALL 16
```

```
                                        ;
                                        ; DO 4K OF DATO AND CHECK FOR MISS
                                        ;
120614  004737  132156                          JSR     PC,INITMM                       ;SET UP MMU
120620  012737  002000  172354                  MOV     #2000,KIPAR6                    ;START AT 32K
120626  042737  100000  172314                  BIC     #BIT15,KIPDR6                   ;NO BYPASS
120634  052737  000001  177572                  BIS     #BIT00,MMR0                     ;ENABLE MMU
120642  012737  121002  000004                  MOV     #DATI,@#4                       ;IF 32K NXW
120650  012702  140000                          MOV     #140000,R2                      ;START WITH 32K
120654  005712                                  TST     (R2)                            ;EXIT
120656  012701  010000                          MOV     #10000,R1                       ;COUNTER FOR 4K
120662  005022                          6$:     CLR     (R2)+                           ;CLEAR ALL 16
120664  077102                                  SOB     R1,6$                           ;DO ALL 16
120666  005200                                  INC     R0                              ;FLAG BLOCK MODE
120670  012777  000000  061752                  MOV     #0,@BA                          ;LOAD DMA ADDRESS
120676  012777  170000  061746                  MOV     #-10000,@WC                     ;DO 4K
120704  012777  003701  061732                  MOV     #3701,@CSR1                     ;16 WORDS FROM 32K
120712  004737  132752                          JSR     PC,DMATRN                       ;DO DATBO
120716  012701  004000                  7$:     MOV     #4000,R1                        ;COUNTER FOR 4K LOCATIONS
120722  012702  140000                          MOV     #140000,R2                      ;START WITH 0
120726  042737  001000  177520          8$:     BIC     #1000,BCSR                      ;DISABLE HALT ON BREAK
120734  005712                                  TST     (R2)                            ;ACCESS A LOCATION
120736  013737  177752  110504                  MOV     HITMIS,RECDAT                   ;STORE REGISTER
120744  052737  001000  177520                  BIS     #1000,BCSR                      ;ENABLE HALT ON BREAK
120752  032737  000004  110504                  BIT     #BIT02,RECDAT                   ;HIT?
120760  001401                                  BEQ     9$                              ;IF NOT, BRANCH
120762  104121                                  ERROR   +121                            ;DMA DOES NOT INVALIDATE
120764  022722  012525                  9$:     CMP     #12525,(R2)+                    ;DATO OK?
120770  001401                                  BEQ     10$                             ;IF SO,BRANCH
120772  104123                                  ERROR   +123                            ;IN DMA
120774  062702  000002                  10$:    ADD     #2,R2                           ;DO IN 2 WORDS
121000  077126                                  SOB     R1,8$                           ;DO ALL OF THEM
                                        ;
                                        ; CHECK DATI
                                        ;
121002  005037  177572                  DATI:   CLR     MMR0                            ;DISABLE MMU
121006  012706  001100                          MOV     #1100,SP                        ;RESTORE STACK
121012  012737  133124  000004                  MOV     #TOUT,@#4                       ;AND TIMEOUT
121020  012737  052525  002710                  MOV     #52525,TEMP                     ;LOAD MEMORY
121026  005000                                  CLR     R0                              ;JUST 1 WORD
121030  004737  133034                          JSR     PC,DMARD                        ;DO DATI
121034  022777  052525  061612                  CMP     #52525,@DATA                    ;DATI OK?
121042  001401                                  BEQ     11$                             ;IF YES, BRANCH
121044  104123                                  ERROR   +123                            ;DATI
                                        ;
                                        ; DO DATBI
                                        ;
121046  012701  000010                  11$:    MOV     #10,R1                          ;COUNTER FOR 16 LOCATIONS
121052  012702  002710                          MOV     #TEMP,R2                        ;START WITH TEMP
121056  012703  002710                          MOV     #TEMP,R3                        ;START WITH TEMP
121062  010322                          12$:    MOV     R3,(R2)+                        ;PUT ADDRESSES
121064  005723                                  TST     (R3)+                           ;INCREMENT ADDRESS
121066  077103                                  SOB     R1,12$                          ;DO ALL 16
121070  005200                                  INC     R0                              ;FLAG BLOCK MODE
121072  004737  133034                          JSR     PC,DMARD                        ;DO DATBI
```

```
20243  121076  032777  010000  061542          BIT     #BIT12,@CSR2            ;NO BLOCK MODE SLAVE?
20244  121104  001401                          BEQ     13$                     ;IF NO, BRANCH
20245  121106  104123                          ERROR   +123
20246  121110  022777  002726  061536  13$:    CMP     #TEMP+16,@DATA          ;DATI OK?
20247  121116  001401                          BEQ     14$                     ;IF SO, BRANCH
20248  121120  104123                          ERROR   +123                    ;IN DATIB
20249  121122  042737  001000  177520  14$:    BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
20250  121130  005737  002710                  TST     TEMP                    ;ACCESS
20251  121134  013737  177752  110504          MOV     HITMIS,RECDAT           ;STORE REGISTER
20252  121142  052737  001000  177520          BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
20253  121150  032737  000004  110504          BIT     #BIT02,RECDAT           ;HIT?
20254  121156  001001                          BNE     15$                     ;IF YES, BRANCH
20255  121160  104123                          ERROR   +123
20256  121162                          15$:
20257
```

```
20258                                   .SBTTL  TEST  - DIFFERENT LEVELS OF INTERRUPTS
20259                                   ;DIFFERENT LEVELS OF INTERRUPTS
20260                                   ;THIS TEST WILL PROGRAM Q22 BUS EXERCISER TO INTERRUPT AT DIFFERENT
20261                                   ;LEVELS.  ARBITRATION BETWEEN DIFFERENT LEVELS OF INTERRUPTS AND
20262                                   ;PIRQ'S WILL BE TESTED.
20263                                   ;
20264                                   ;CHECK DIFFERENT LEVELS OF INTERRUPTS.
20265                                   ;ROUTINE TEST
20266                                   ;.      SET VECTOR TO INTERRUPT_DMA
20267                                   ;.      FOR INTERRUPTS FROM 4 TO 7 DO
20268                                   ;.      .       ENABLE INTERRUPTS
20269                                   ;.      .       SET PRIORITY=INTERUPT
20270                                   ;.      .       IF INTERRUPT_FLAG SET THEN
20271                                   ;.      .       .       ERROR
20272                                   ;.      .       ENDIF
20273                                   ;.      .       ENABLE INTERRUPTS
20274                                   ;.      .       SET PRIORITY=INTERRUPT-1
20275                                   ;.      .       IF INTERRUPT_FLAG NOTSET THEN
20276                                   ;.      .       .       ERROR
20277                                   ;.      .       ENDIF
20278                                   ;.      .       LET INTERRUPT_DMA=0
20279                                   ;.      ENDDO
20280                                   ;ENDROUTINE
20281                                   ;
20282                                   ;ROUTINE INTERUPT_DMA
20283                                   ;.      LET INTERRUPT_FLAG=1
20284                                   ;RETURN
20285                                   ;ENDROUTINE
20286
20287                                   ;;*****************************************************************
20288  121162  000004                  TST61:  SCOPE
20289  121164  032737  000200  000052          BIT     #BIT07,@#52                     ;UFD MODE?
20290  121172  001122                          BNE     TST62           ;;IF SO, EXIT TEST
20291  121174  005737  002644                  TST     CSR1                            ;AT LEAST ONE Q22BE FOUND?
20292  121200  001517                          BEQ     TST62           ;;IF NOT, EXIT TEST
20293                                   ;
20294                                   ; SETUP INITIAL PRIORITY TO 7
20295                                   ;
20296  121202  013703  002644                  MOV     CSR1,R3                         ;DO FOR FIRST FOUND Q22BE
20297  121206  012777  121302  061442          MOV     #5$,@VQBE1                      ;POINT INTERRUPT VECTOR TO PROGRAM
20298  121214  012777  000340  061436          MOV     #340,@VQPR1                     ;AT PRIORITY 7
20299  121222  012700  002752                  MOV     #Q22EN,R0                       ;START WITH 7 FOR INTERRUPTS
20300  121226  012701  000340                  MOV     #340,R1                         ;LOW BOUNDARY FOR NO INTERRUPTS
20301  121232  000402                          BR      2$                              ;TRY TO DO IT FOR FIRST
20302  121234  162701  000040          1$:     SUB     #40,R1                          ;LOWER LOW BOUNDARY
20303                                   ;
20304                                   ; CHECK THAT INTERRUPTS DON'T HAPPEN AT PRIORITY HIGHER THAN BR
20305                                   ;
20306  121240  012737  000340  001160  2$:     MOV     #340,$TMP0                      ;TOP PRIORITY FOR NO INTERRUPTS
20307  121246  000403                          BR      4$                              ;DO FIRST ONE
20308  121250  162737  000040  001160  3$:     SUB     #40,$TMP0                       ;DO AT NEXT LEVEL
20309  121256  106437  001160          4$:     MTPS    $TMP0                           ;SET PRIORITY NOT TO INTERRUPT
20310  121262  004737  132732                  JSR     PC,Q22INT                       ;ENABLE INTERRUPTS
20311  121266  012077  061354                  MOV     (R0)+,@CSR2                     ;CLEAR GO BIT
20312  121272  000240                          NOP
20313  121274  000240                          NOP
```

```
20314  121276  000240                          NOP
20315  121300  000403                          BR      6$                      ;IF NO INTERUPT, BRANCH
20316  121302  104126                  5$:     ERROR   +126                    ;INTERRUPTS HAPPEN
20317  121304  005726                          TST     (SP)+                   ;RESTORE STACK
20318  121306  005726                          TST     (SP)+
20319  121310  020137  001160          6$:     CMP     R1,$TMP0                ;LAST ONE?
20320  121314  001355                          BNE     3$                      ;IF NOT BRANCH
20321  121316  022710  000002                  CMP     #2,(R0)                 ;AT BR4?
20322  121322  001344                          BNE     1$                      ;IF NOT LAST ONE, BRANCH
20323                                  ;
20324                                  ; INTERRUPT AT ALL LEVELS
20325                                  ;
20326  121324  012777  121420  061324  INQ22:  MOV     #5$,@VQBE1              ;POINT INTERRUPT VECTOR TO PROGRAM
20327  121332  012777  000340  061320          MOV     #340,@VQPR1             ;AT PRIORITY 7
20328  121340  012700  002752                  MOV     #Q22EN,R0               ;START WITH 7 FOR INTERRUPTS
20329  121344  012701  000300                  MOV     #300,R1                 ;TOP BOUNDARY FOR INTERRUPTS
20330  121350  000402                          BR      2$                      ;TRY TO DO IT FOR FIRST
20331  121352  162701  000040          1$:     SUB     #40,R1                  ;LOWER TOP BOUNDARY
20332                                  ;
20333                                  ; CHECK THAT INTERRUPTS HAPPEN AT PRIORITY LOWER THAN BR
20334                                  ;
20335  121356  010137  001160          2$:     MOV     R1,$TMP0                ;PRIORITY FOR INTERRUPTS
20336  121362  000403                          BR      4$                      ;DO FIRST ONE
20337  121364  162737  000040  001160  3$:     SUB     #40,$TMP0               ;DO AT NEXT LEVEL
20338  121372  106437  001160          4$:     MTPS    $TMP0                   ;SET PRIORITY NOT TO INTERRUPT
20339  121376  004737  132732                  JSR     PC,Q22INT               ;ENABLE INTERRUPTS
20340  121402  011077  061240                  MOV     (R0),@CSR2              ;CLEAR GO BIT
20341  121406  000240                          NOP                             ;WAIT A WHILE
20342  121410  000240                          NOP
20343  121412  000240                          NOP
20344  121414  104126                          ERROR   +126                    ;INTERRUPTS DON'T HAPPEN
20345  121416  000402                          BR      6$                      ;DON'T RESTORE STACK
20346  121420  005726                  5$:     TST     (SP)+                   ;RESTORE STACK
20347  121422  005726                          TST     (SP)+
20348  121424  005737  001160          6$:     TST     $TMP0                   ;LAST ONE 0?
20349  121430  001355                          BNE     3$                      ;IF NOT BRANCH
20350  121432  022720  000002                  CMP     #2,(R0)+                ;AT BR4?
20351  121436  001345                          BNE     1$                      ;IF NOT LAST ONE, BRANCH
20352
20353
```

```
                                        .SBTTL  TEST  - ARBITRATION BETWEEN PIRQ'S AND INTERRUPTS
                                        ;CHECK PRIORITY ORDER BETWEEN PIRQ'S AND INTERRUPTS.
                                        ;ROUTINE TEST
                                        ;.      IF UFD THEN
                                        ;.      .       EXIT TEST
                                        ;.      ENDIF
                                        ;.      DO FOR I FROM #6 DOWN TO #3
                                        ;.      .       SET PRIORITY TO I
                                        ;.      .       ENABLE INTERRUPT(I+1) AND PIRQ(I+1)
                                        ;.      .       IF INTERRUPT(I+1) WAS BEFORE PIRQ(I+1) THEN
                                        ;.      .       .       ERROR
                                        ;.      .       ENDIF
                                        ;.      ENDDO
                                        ;ENDROUTINE

                                        ;;**************************************************************
000004                                  TST62:  SCOPE
032737  000200  000052                          BIT     #BIT07,@#52                     ;UFD MODE?
001065                                          BNE     TST63           ;;IF SO, EXIT TEST
005737  002644                                  TST     CSR1                            ;AT LEAST ONE Q22BE FOUND?
001462                                          BEQ     TST63           ;;IF NOT, EXIT TEST
012777  121566  061170                          MOV     #3$,@VQBE1                      ;SETUP Q22BE VECTOR
012777  000340  061164                          MOV     #340,@VQPR1                     ;AT PRIORITY 7
012737  121576  000240                          MOV     #4$,PIRQVEC                     ;SETUP PIRQ VECTOR
012737  000340  000242                          MOV     #340,PIRQVEC+2                  ;AT PRIORITY 7
012700  002752                                  MOV     #Q22EN,R0                       ;POINT THRU PRIORITIES FOR Q22BE
012704  121614                                  MOV     #PIRQT,R4                       ;POINTER THRU PIRQ'S
013703  002644                                  MOV     CSR1,R3                         ;DO FOR FIRST Q22BE
012702  000300                                  MOV     #300,R2                         ;START WITH CPU PRIORITY AT 7
000402                                          BR      2$                              ;DO FIRST ONE
162702  000040                  1$:     SUB     #40,R2                          ;LOWER CPU PRIORITY
106427  000340                  2$:     MTPS    #340                            ;RAISE PRIORITY TO 7
012437  177772                          MOV     (R4)+,PIRQ                      ;SET PRIORITY FOR PIRQ'S
004737  132732                          JSR     PC,Q22INT                       ;INITIALISE Q22BE TO INTERRUPT
012077  061070                          MOV     (R0)+,@CSR2                     ;SET DONE BIT
106402                                  MTPS    R2                              ;LOWER PRIORITY
000240                                  NOP
000240                                  NOP
000240                                  NOP
104124                  3$:     ERROR   .124                                    ;PIRQ'S DON'T TAKE OVER BIRQ'S
005726                          TST     (SP)+                                   ;CLEAN UP STACK
005726                          TST     (SP)+
000402                                  BR      5$                              ;BRANCH AROUND PIRQ INTERRUPT
005726                  4$:     TST     (SP)+                                   ;CLEAN UP STACK
005726                          TST     (SP)+
022702  000140          5$:     CMP     #140,R2                                 ;PRIORITY 3 LAST ONE?
001351                                  BNE     1$                              ;IF NOT BRANCH
005037  177772                          CLR     PIRQ                            ;CLEAR ANY REQUESTS

100000  040000  020000  PIRQT: .WORD   100000,40000,20000,10000                 ;PIRQ'S<7-4>
010000
```

```
20406                                   .SBTTL  TEST    - POWER DOWN TEST
20407                                   ;USING Q22BE THIS TEST WILL CHECK THAT ON POWER DOWN CONDITION IF
20408                                   ;POWER UP CODE 00 IS SELECTED THE CPU TRAPS THRU 24
20409                                   ;ROUTINE TEST
20410                                   ;.      IF UFD OR POWER UP CODE 00 NOT SELECTED THEN
20411                                   ;.          .       EXIT TEST
20412                                   ;.      ENDIF
20413                                   ;.      SET 24 TO POINT TO TEST AREA
20414                                   ;.      LET CSR2<5> = #1 TO NEGATE BPOK
20415                                   ;.      IF NO TRAP TO 24 THEN
20416                                   ;.          .       ERROR IN POWER DOWN CYCLE
20417                                   ;.      ENDIF
20418                                   ;.      IF TRAP TO 24 THEN
20419                                   ;.          .       LET CSR2<5> = #0
20420                                   ;.      ENDIF
20421                                   ;ENDROUTINE
20422
20423                                   ;;*****************************************************************
20424  121624  000004                   TST63:  SCOPE
20425  121626  032737  000200  000052           BIT     #BIT07,@#52                     ;UFD MODE?
20426  121634  001033                   BNE     TST64           ;;EXIT TEST IN UFD  MODE
20427  121636  005737  002644                   TST     CSR1                            ;AT LEAST ONE Q22BE FOUND?
20428  121642  001430                   BEQ     TST64           ;;IF NOT, EXIT TEST
20429  121644  013737  000024  001160           MOV     PWRVEC,$TMP0                    ;SAVE POWER UP VECTOR
20430  121652  012737  121706  000024           MOV     #1$,PWRVEC                      ;POINT NEW TO PROGRAM
20431  121660  012737  000340  000026           MOV     #340,PWRVEC+2                   ;AT PRIORITY 7
20432  121666  012777  000040  060752           MOV     #BIT05,@CSR2                    ;DO POWER DOWN
20433  121674  000240                           NOP
20434  121676  005077  060744                   CLR     @CSR2                           ;CLEAR POWER DOWN BIT
20435  121702  104125                           ERROR   .125                            ;NO POWER DOWN TRAP
20436  121704  000404                           BR      2$                              ;SKIP RESTORING STACK
20437  121706  005077  060734           1$:     CLR     @CSR2                           ;CLEAR POWER DOWN BIT
20438  121712  005726                           TST     (SP)+                           ;RESTORE STACK POINTER
20439  121714  005726                           TST     (SP)+
20440  121716  013737  001160  000024   2$:     MOV     $TMP0,PWRVEC                    ;RESTORE POWER VECTOR
20441
20442
20443
```

```
                                .SBTTL TEST    - ARBITRATION BETWEEN DIFFERENT LEVELS OF INTERRUPTS
                                ;IF TWO Q22BE ARE AVAILABLE, THIS TEST WILL CHECK THAT HIGHER LEVEL
                                ;INTERRUPT REQUESTS TAKE PRIORITY OVER LOWER LEVEL ONES
                                ;ROUTINE TEST
                                ;.      IF UFD OR 2ND Q22BE NOT AVAILABLE THEN
                                ;.      .       EXIT TEST
                                ;.      ENDIF
                                ;.      DO FOR PRIORITY_LEVELS FROM 4 TO 6
                                ;.      .       SET UP 1ST Q22BE TO INTERRUPT AT PRIORITY_LEVEL
                                ;.      .       SET UP 2ND Q22BE TO INTERRUPT AT PRIORITY_LEVEL+1
                                ;.      .       SET UP CPU PRIORITY TO PRIORITY_LEVEL-1
                                ;.      .       IF INTERRUPTS FORM 1ST Q22BE HAPPENED BEFORE 1ST THEN
                                ;.      .       .       ERROR
                                ;.      .       ENDIF
                                ;.      ENDDO
                                ;ENDROUTINE

                                ;;*************************************************************
121724  000004                  TST64:  SCOPE
121726  032737  000200  000052          BIT     #BIT07,@#52                     ;UFD MODE?
121734  001064                          BNE     TST65           ;;IF SO, EXIT TEST
121736  005737  002664                  TST     CSR12                           ;SECOND Q22BE AVAILABLE?
121742  001002                          BNE     1$                              ;IF YES, GO DO TEST
121744  000137  133514                  JMP     $EOP                            ;OTHERWISE, GOTO EOP
                                ;
                                ; INITIALISE VECTORS FOR Q22BE'S
                                ;
121750  012777  122064  060700  1$:     MOV     #4$,@VQBE1                      ;VECTOR FOR 1ST ONE
121756  012777  000340  060674          MOV     #340,@VQPR1                     ;AT PRIORITY 7
121764  012777  122074  060704          MOV     #5$,@VQBE2                      ;VECTOR FOR 2ND ONE
121772  012777  000340  060700          MOV     #340,@VQPR2                     ;AT PRIORITY 7
122000  012700  002754                  MOV     #Q22EN+2,R0                     ;POINTER FOR Q22BE BR'S
122004  012702  000240                  MOV     #240,R2                         ;CPU AT 5
122010  000402                          BR      3$                              ;START DOING ARBITRATION
                                ;
                                ; DO FOR CPU PRIORITIES 5-3
                                ;
122012  162702  000040          2$:     SUB     #40,R2                          ;LOWER CPU PRIORITY
122016  106427  000340          3$:     MTPS    #340                            ;CPU AT 7
122022  013703  002664                  MOV     CSR12,R3                        ;Q22BE 2 AT HIGHER BR
122026  004737  132732                  JSR     PC,Q22INT                       ;INITIALISE TO INTERRUPTS
122032  011077  060610                  MOV     (R0),@CSR2                      ;START 1ST ONE
122036  013703  002644                  MOV     CSR1,R3                         ;Q22BE 1 AT LOWER BR
122042  005740                          TST     -(R0)                           ;HIGHER PRIORITY
122044  004737  132732                  JSR     PC,Q22INT                       ;INITIALISE
122050  012077  060612                  MOV     (R0)+,@CSR22                    ;START 2ND ONE
122054  106402                          MTPS    R2                              ;LOWER CPU PRIORITY
122056  000240                          NOP                                     ;WAIT A WHILE
122060  000240                          NOP
122062  000240                          NOP
122064  104126                  4$:     ERROR   #126                            ;INTERRUPTS IN WRONG ORDER
122066  005726                          TST     (SP)+                           ;RESTORE STACK
122070  005726                          TST     (SP)+
122072  000402                          BR      6$
122074  005726                  5$:     TST     (SP)+                           ;RESTORE STACK
122076  005726                          TST     (SP)+
```

```
 20500  122100  022702  000140          6$:     CMP     #140,R2                 ;PRIORITY 3 LAST?
 20501  122104  001342                          BNE     2$                      ;IF NOT, BRANCH TO CONTINUE
 20502
```

```
                                        .SBTTL  TEST    - PMG COUNTER
                                        ;USING 2 Q22BE THIS TEST WILL VALIDATE THAT PMG COUNTER IS REALLY
                                        ;CAPABLE OF GRANTING CPU BUS MASTERSHIP WHEN DMA REQUESTS ARE STILL
                                        ;PENDING.
                                        ;ROUTINE TEST
                                        ;.      IF UFD OR NO 2ND Q22BE THEN
                                        ;.      .       EXIT TEST
                                        ;.      ENDIF
                                        ;.      SET PMG COUNT TO SOME VALUE
                                        ;.      PROGRAM BOTH Q22BE TO INTERRUPT AT THE SAME LEVEL
                                        ;.      DETERMINE WHICH HAS HIGHER PRIORITY ON THE BUS
                                        ;.      PROGRAM Q22BE WITH HIGHER PRIORITY TO DO HOG MODE
                                        ;.      PROGRAM 2ND ONE TO DO A CYCLE
                                        ;.      CACHE INSTRUCTION THAT WILL STOP 2ND DMA
                                        ;.      INITIATE BOTH DMA CYCLES
                                        ;.      STOP 2ND DMA (RESET IS "STOLEN" CPU BUS CYCLE)
                                        ;.      IF 2ND DMA HAPPENED THEN
                                        ;.      .       ERROR
                                        ;.      ENDIF
                                        ;.      CLEAR PMG COUNT
                                        ;.      PROGRAM Q22BE WITH HIGHER PRIORITY TO DO HOG MODE
                                        ;.      PROGRAM 2ND ONE TO DO A CYCLE
                                        ;.      CACHE INSTRUCTION THAT WILL STOP 2ND DMA
                                        ;.      INITIATE BOTH DMA CYCLES
                                        ;.      STOP 2ND DMA (CLEARING OF CSR2<0> IS "STOLEN" CPU BUS CYCLE)
                                        ;.      IF 2ND DMA DIDN'T HAPPENED THEN
                                        ;.      .       ERROR
                                        ;.      ENDIF
                                        ;ENDROUTINE


                                        ;;*****************************************************************
122106  000004                          TST65:  SCOPE
122110  032737  000200  000052                  BIT     #BIT07,@#52                     ;UFD MODE?
122116  001402                                  BEQ     100$                            ;IF NOT, CONTINUE
122120  000137  133514                          JMP     $EOP                            ;EXIT
122124  005737  002664                  100$:   TST     CSR12                           ;SECOND Q22BE AVAILABLE?
122130  001002                                  BNE     110$                            ;IF YES, GO DO TEST
122132  000137  133514                          JMP     $EOP                            ;OTHERWISE, GOTO EOP
                                        ;
                                        ; DETERMINE WHICH Q22BE IS CLOSER TO CPU BY DOING DATO
                                        ; THE CSR1 OF THE ONE WITH HIGHER PRIORITY IS IN R4, THE SECOND IN R2
                                        ;
122136  005077  060504                  110$:   CLR     @CSR2                           ;CLEAR JUST IN CASE
122142  005077  060520                          CLR     @CSR22
122146  012777  002710  060474                  MOV     #TEMP,@BA                       ;ADDRESS TO BE USED
122154  012777  002710  060506                  MOV     #TEMP,@BA2                      ;FOR BOTH OF THEM
122162  012777  177777  060462                  MOV     #177777,@WC                     ;DO FOR 1 WORD
122170  012777  177777  060474                  MOV     #177777,@WC2                    ;IN BOTH Q22BW'S
122176  012777  012525  060450                  MOV     #12525,@DATA                    ;DATA FOR 1ST
122204  012777  052525  060462                  MOV     #52525,@DATA2                   ;DATA FOR 2ND
122212  012777  001601  060424                  MOV     #1601,@CSR1                     ;1 DATO FOR 1ST
122220  012777  001601  060436                  MOV     #1601,@CSR12                    ;AND 2ND
122226  012777  000001  060426                  MOV     #1,@SIMGOA                      ;BOTH GO
122234  105777  060404                  1$:     TSTB    @CSR1                           ;FIRST DONE?
122240  100375                                  BPL     1$                              ;IF NOT, WAIT
122242  105777  060416                  2$:     TSTB    @CSR12                          ;SECOND DONE
```

```
20559  122246  100375                          BPL     2$                      ;WAIT FOR 2ND
20560  122250  022737  052525  002710          CMP     #52525,TEMP             ;SECOND FINISHED LAST?
20561  122256  001405                          BEQ     3$                      ;IF SO, BRANCH
20562  122260  013704  002664                  MOV     CSR12,R4                ;SECOND ONE AT HIGHER LEVEL
20563  122264  013702  002644                  MOV     CSR1,R2                 ;FIRST AT LOWER
20564  122270  000404                          BR      4$                      ;DO PMG PART
20565  122272  013704  002644          3$:     MOV     CSR1,R4                 ;FIRST ONE AT HIGHER LEVEL
20566  122276  013702  002664                  MOV     CSR12,R2                ;SECOND AT LOWER
20567                                          ;
20568                                          ; INITIATE DMA CYCLES TO WORK WITH PMG COUNTER
20569                                          ;
20570  122302  013737  177520  002710  4$:     MOV     BCSR,SAVBR              ;STORE BCSR REGISTER
20571  122310  012700  000001                  MOV     #BIT00,R0               ;FIRST VALUE FOR PMG 0.4MSEC
20572  122314  050037  177520          5$:     BIS     R0,BCSR                 ;CHANGE PMG CONUTER
20573  122320  005737  122422                  TST     6$                      ;CACHE RESET INTRUCTION
20574  122324  005737  122424                  TST     6$+2                    ;AND THE FOLLOWING FEW
20575  122330  005737  122426                  TST     6$+4                    ;
20576  122334  005737  122430                  TST     6$+6                    ;
20577  122340  012714  001407                  MOV     #1407,(R4)              ;DATI IN HOG MODE FOR HIGHER ONE
20578  122344  005064  000002                  CLR     2(R4)                   ;CLEAR CSR2
20579  122350  012764  002710  000004          MOV     #TEMP,4(R4)             ;ADDRESS TO START DATI
20580  122356  012764  000000  000006          MOV     #0,6(R4)                ;DO 128 DATI'S IN HOG MODE
20581  122364  012712  001407                  MOV     #1407,(R2)              ;DATO FOR LOWER LEVEL ONE
20582  122370  005062  000002                  CLR     2(R2)                   ;CLEAR JUST IN CASE
20583  122374  012762  002710  000004          MOV     #TEMP,4(R2)             ;USE THE SAME ADDRESS
20584  122402  012762  177777  000006          MOV     #177777,6(R2)           ;DO JUST ONE DATO
20585  122410  005062  000010                  CLR     10(R2)                  ;CLEAR DATA OF 2ND Q22BE
20586  122414  012762  000001  000016          MOV     #1,16(R2)               ;BOTH GO
20587  122422  000005                  6$:     RESET                           ;STOP Q22BE AT R4
20588  122424  023762  002710  000010          CMP     TEMP,10(R2)             ;SECOND DMA DONE?
20589  122432  001001                          BNE     7$                      ;IF SET, BRANCH
20590  122434  104127                          ERROR   +127                    ;NO CYCLE STEALING
20591  122436  062700  000003          7$:     ADD     #3,R0                   ;DO FOR 1,3,7 IN PMG
20592  122442  040037  177520                  BIC     R0,BCSR                 ;CLEAR PREVOIUS BITS IN BCSR
20593  122446  022700  000007                  CMP     #7,R0                   ;LAST ONE?
20594  122452  002320                          BGE     5$                      ;IF NOT, BRANCH
20595                                          ;
20596                                          ; TRY WITHOUT PMG COUNTER OPERATING
20597                                          ;
20598  122454  042737  000007  177520          BIC     #7,BCSR                 ;TURN OF PMG COUNTER
20599  122462  005737  122564                  TST     8$                      ;CACHE RESET FETCH
20600  122466  005737  122566                  TST     8$+2                    ;AND THE FOLLOWING FEW
20601  122472  005737  122570                  TST     8$+4                    ;
20602  122476  005737  122572                  TST     8$+6                    ;
20603  122502  012714  001407                  MOV     #1407,(R4)              ;DATI IN HOG MODE FOR HIGHER ONE
20604  122506  005064  000002                  CLR     2(R4)                   ;CLEAR CSR2
20605  122512  012764  002710  000004          MOV     #TEMP,4(R4)             ;ADDRESS TO START DATI
20606  122520  012764  000000  000006          MOV     #0,6(R4)                ;DO 128 DATI'S IN HOG MODE
20607  122526  012712  001407                  MOV     #1407,(R2)              ;DATO FOR LOWER LEVEL ONE
20608  122532  005062  000002                  CLR     2(R2)                   ;CLEAR CRS2 OF 2ND
20609  122536  012762  002710  000004          MOV     #TEMP,4(R2)             ;USE THE SAME ADDRESS
20610  122544  012762  177777  000006          MOV     #177777,6(R2)           ;DO JUST ONE DATO
20611  122552  005062  000010                  CLR     10(R2)                  ;CLEAR DATA OF 2ND Q22BE
20612  122556  012777  000001  060076          MOV     #1,@SIMGOA              ;BOTH GO
20613  122564  000005                  8$:     RESET                           ;IF NOT WORKING, STOPS 2ND
20614  122566  023762  002710  000010          CMP     TEMP,10(R2)             ;2ND DMA HAPPENED?
```

```
122574  001401                          BEQ     9$                      ;IF YES, BRANCH
122576  104127                          ERROR   +127                    ;IN PMG COUNTER
122600  013737  002710  177520  9$:     MOV     SAVBR,BCSR              ;RESTORE BCSR


                                ;;*****************************************************************
122606  000004                  TST66:  SCOPE

122610  000137  133514                  JMP     $EOP
```

```
20624                                   .SBTTL  GLOBAL ERROR MESSAGES
20625  122614  040502  044523  020103  EM1:    .ASCIZ  /BASIC INSTRUCTION SET ERROR/
20626  122622  047111  052123  052522
20627  122630  052103  047511  020116
20628  122636  042523  020124  051105
20629  122644  047522  000122
20630  122650  046515  020125  051105  EM2:    .ASCIZ  /MMU ERROR/
20631  122656  047522  000122
20632  122662  050106  020120  051105  EM3:    .ASCIZ  /FPP ERROR/
20633  122670  047522  000122
20634  122674  051105  047522  020122  EM4:    .ASCIZ  /ERROR IN READ-WRITE BITS OF CCR/
20635  122702  047111  051040  040505
20636  122710  026504  051127  052111
20637  122716  020105  044502  051524
20638  122724  047440  020106  041503
20639  122732  000122
20640  122734  047506  041522  020105  EM5:    .ASCIZ  /FORCE MISS WRITES TO CACHE/
20641  122742  044515  051523  053440
20642  122750  044522  042524  020123
20643  122756  047524  041440  041501
20644  122764  042510     000
20645  122767     106  051117  042503  EM6:    .ASCIZ  /FORCE MISS WRITE INVALIDATES CACHE/
20646  122774  046440  051511  020123
20647  123002  051127  052111  020105
20648  123010  047111  040526  044514
20649  123016  040504  042524  020123
20650  123024  040503  044103  000105
20651  123032  047125  054105  042520  EM7:    .ASCIZ  /UNEXPECTED PARITY INTERRUPT/
20652  123040  052103  042105  050040
20653  123046  051101  052111  020131
20654  123054  047111  042524  051122
20655  123062  050125  000124
20656  123066  040524  020107  040520  EM10:   .ASCIZ  /TAG PARITY ERROR/
20657  123074  044522  054524  042440
20658  123102  051122  051117     000
20659  123107     104  052101  020101  EM11:   .ASCIZ  /DATA PARITY ERROR/
20660  123114  040520  044522  054524
20661  123122  042440  051122  051117
20662  123130     000
20663  123131     114  053517  041040  EM12:   .ASCIZ  /LOW BYTE PARITY ERROR/
20664  123136  052131  020105  040520
20665  123144  044522  054524  042440
20666  123152  051122  051117     000
20667  123157     110  043511  020110  EM13:   .ASCIZ  /HIGH BYTE PARITY ERROR/
20668  123164  054502  042524  050040
20669  123172  051101  052111  020131
20670  123200  051105  047522  000122
20671  123206  051105  047522  020122  EM14:   .ASCIZ  /ERROR IN DATA PATH/
20672  123214  047111  042040  052101
20673  123222  020101  040520  044124
20674  123230     000
20675  123231     106  051117  042503  EM15:   .ASCIZ  /FORCE MISS READS FROM CACHE/
20676  123236  046440  051511  020123
20677  123244  042522  042101  020123
20678  123252  051106  046517  041440
20679  123260  041501  042510     000
```

```
123265     106  051117  042503  EM16:   .ASCIZ  /FORCE MISS READS FROM CACHE AND MISS/
123272  046440  051511  020123
123300  042522  042101  020123
123306  051106  046517  041440
123314  041501  042510  040440
123322  042116  046440  051511
123330  000123
123332  051105  047522  020122  EM17:   .ASCIZ  \ERROR IN RECORDING HITS IN HIT/MISS\
123340  047111  051040  041505
123346  051117  044504  043516
123354  044040  052111  020123
123362  047111  044040  052111
123370  046457  051511  000123
123376  051127  052111  020105  EM20:   .ASCIZ  /WRITE BYTE ALLOCATES CACHE/
123404  054502  042524  040440
123412  046114  041517  052101
123420  051505  041440  041501
123426  042510     000
123431     127  044522  042524  EM21:   .ASCIZ  /WRITE BYTE HIT DOES NOT RECORD HIT/
123436  041040  052131  020105
123444  044510  020124  047504
123452  051505  047040  052117
123460  051040  041505  051117
123466  020104  044510  000124
123474  054502  042524  020123  EM22:   .ASCIZ  /BYTES REVERSED ON WRITE CYCLES/
123502  042522  042526  051522
123510  042105  047440  020116
123516  051127  052111  020105
123524  054503  046103  051505
123532     000
123533     103  047117  044504  EM23:   .ASCIZ  /CONDITIONAL BYPASS DOESN'T INVALIDATE CACHE/
123540  044524  047117  046101
123546  041040  050131  051501
123554  020123  047504  051505
123562  023516  020124  047111
123570  040526  044514  040504
123576  042524  041440  041501
123604  042510     000
123607     110  052111  020123  EM24:   .ASCIZ  /HITS RECORDED AFTER FLUSHING CACHE/
123614  042522  047503  042122
123622  042105  040440  052106
123630  051105  043040  052514
123636  044123  047111  020107
123644  040503  044103  000105
123652  054502  040520  051523  EM25:   .ASCIZ  /BYPASS DOESN'T INVALIDATE CACHE/
123660  042040  042517  047123
123666  052047  044440  053116
123674  046101  042111  052101
123702  020105  040503  044103
123710  000105
123712  051515  051105  042040  EM26:   .ASCIZ  /MSER DOES NOT CLEAR ON WRITE REFERENCE/
123720  042517  020123  047516
123726  020124  046103  040505
123734  020122  047117  053440
123742  044522  042524  051040
123750  043105  051105  047105
```

```
20736  123756  042503     000
20737  123761     120  051101  052111  EM27:   .ASCIZ  /PARITY ERROR DON'T CAUSE A MISS/
20738  123766  020131  051105  047522
20739  123774  020122  047504  023516
20740  124002  020124  040503  051525
20741  124010  020105  020101  044515
20742  124016  051523     000
20743  124021     120  051101  052111  EM30:   .ASCIZ  /PARITY ERROR DON'T SET MSER WITH CCR<7>=0/
20744  124026  020131  051105  047522
20745  124034  020122  047504  023516
20746  124042  020124  042523  020124
20747  124050  051515  051105  053440
20748  124056  052111  020110  041503
20749  124064  036122  037067  030075
20750  124072     000
20751  124073     120  051101  052111  EM31:   .ASCIZ  /PARITY ERROR IGNORED/
20752  124100  020131  051105  047522
20753  124106  020122  043511  047516
20754  124114  042522  000104
20755  124120  040520  044522  054524  EM32:   .ASCIZ  /PARITY ERROR IGNORED ON LOW BYTE/
20756  124126  042440  051122  051117
20757  124134  044440  047107  051117
20758  124142  042105  047440  020116
20759  124150  047514  020127  054502
20760  124156  042524     000
20761  124161     120  051101  052111  EM33:   .ASCIZ  /PARITY ERROR IGNORED ON HIGH BYTE/
20762  124166  020131  051105  047522
20763  124174  020122  043511  047516
20764  124202  042522  020104  047117
20765  124210  044040  043511  020110
20766  124216  054502  042524     000
20767  124223     120  051101  052111  EM34:   .ASCIZ  /PARITY ABORT LOGIC DOESN'T WORK/
20768  124230  020131  041101  051117
20769  124236  020124  047514  044507
20770  124244  020103  047504  051505
20771  124252  023516  020124  047527
20772  124260  045522     000
20773  124263     115  042523  020122  EM35:   .ASCIZ  /MSER NOT SET PROPERLY/
20774  124270  047516  020124  042523
20775  124276  020124  051120  050117
20776  124304  051105  054514     000
20777  124311     120  051101  052111  EM36:   .ASCIZ  /PARITY INTERRUPT LOGIC DOESN'T WORK/
20778  124316  020131  047111  042524
20779  124324  051122  050125  020124
20780  124332  047514  044507  020103
20781  124340  047504  051505  023516
20782  124346  020124  047527  045522
20783  124354     000
20784  124355     116  046530  040440  EM37:   .ASCIZ  /NXM AND PARITY ABORT DIN'T HAPPEN/
20785  124362  042116  050040  051101
20786  124370  052111  020131  041101
20787  124376  051117  020124  044504
20788  124404  023516  020124  040510
20789  124412  050120  047105     000
20790  124417     120  051101  052111  EM40:   .ASCIZ  /PARITY ABORT NOT BLOCKED BY NXM TRAP/
20791  124424  020131  041101  051117
```

```
20792  124432  020124  047516  020124
20793  124440  046102  041517  042513
20794  124446  020104  054502  047040
20795  124454  046530  052040  040522
20796  124462  000120
20797  124464  052515  052114  026511  EM41:   .ASCIZ  /MULTI-PROCESSOR HOOK INSTRUCTION DOESN'T CAUSE MISS/
20798  124472  051120  041517  051505
20799  124500  047523  020122  047510
20800  124506  045517  044440  051516
20801  124514  051124  041525  044524
20802  124522  047117  042040  042517
20803  124530  047123  052047  041440
20804  124536  052501  042523  046440
20805  124544  051511  000123
20806  124550  051105  047522  020122  EM42:   .ASCIZ  /ERROR IN PARITY LOGIC/
20807  124556  047111  050040  051101
20808  124564  052111  020131  047514
20809  124572  044507  000103
20810  124576  051105  047522  020122  EM43:   .ASCIZ  /ERROR IN CACHE DATA RAMS/
20811  124604  047111  041440  041501
20812  124612  042510  042040  052101
20813  124620  020101  040522  051515
20814  124626     000
20815  124627     105  051122  051117  EM44:   .ASCIZ  /ERROR IN NXM IN STANDALONE MODE/
20816  124634  044440  020116  054116
20817  124642  020115  047111  051440
20818  124650  040524  042116  046101
20819  124656  047117  020105  047515
20820  124664  042504     000
20821  124667     105  051122  051117  EM45:   .ASCIZ  \ERROR IN RECORDING HITS THROUGH HIT/MISS REGISTER\
20822  124674  044440  020116  042522
20823  124702  047503  042122  047111
20824  124710  020107  044510  051524
20825  124716  052040  051110  052517
20826  124724  044107  044040  052111
20827  124732  046457  051511  020123
20828  124740  042522  044507  052123
20829  124746  051105     000
20830  124751     110  052111  051040  EM46:   .ASCIZ  /HIT RECORDED FOR A LOCATION THAT SHOULD NOT BE IN CACHE/
20831  124756  041505  051117  042504
20832  124764  020104  047506  020122
20833  124772  020101  047514  040503
20834  125000  044524  047117  052040
20835  125006  040510  020124  044123
20836  125014  052517  042114  047040
20837  125022  052117  041040  020105
20838  125030  047111  041440  041501
20839  125036  042510     000
20840  125041     115  051511  020123  EM47:   .ASCIZ  /MISS RECORDED FOR A LOCATION THAT SHOULD BE IN CACHE/
20841  125046  042522  047503  042122
20842  125054  042105  043040  051117
20843  125062  040440  046040  041517
20844  125070  052101  047511  020116
20845  125076  044124  052101  051440
20846  125104  047510  046125  020104
20847  125112  042502  044440  020116
```

```
20848  125120  040503  044103  000105
20849  125126  051105  047522  020122  EM50:   .ASCIZ  /ERROR IN TAG STORE/
20850  125134  047111  052040  043501
20851  125142  051440  047524  042522
20852  125150     000
20853  125151     105  051122  051117  EM51:   .ASCIZ  /ERROR PCR READ-WRITE BITS/
20854  125156  050040  051103  051040
20855  125164  040505  026504  051127
20856  125172  052111  020105  044502
20857  125200  051524     000
20858  125203     105  051122  051117  EM52:   .ASCIZ  /ERROR IN BCSR READ-WRITE BITS/
20859  125210  044440  020116  041502
20860  125216  051123  051040  040505
20861  125224  026504  051127  052111
20862  125232  020105  044502  051524
20863  125240     000
20864  125241     122  051505  052105  EM53:   .ASCIZ  /RESET DOESN'T CLEAR BCSR<4>/
20865  125246  042040  042517  047123
20866  125254  052047  041440  042514
20867  125262  051101  041040  051503
20868  125270  036122  037064     000
20869  125275     103  042510  045503  EM54:   .ASCIZ  /CHECKSUM ERROR IN 16-BIT ROM /
20870  125302  052523  020115  051105
20871  125310  047522  020122  047111
20872  125316  030440  026466  044502
20873  125324  020124  047522  020115
20874  125332     000
20875  125333     103  042510  045503  EM55:   .ASCIZ  /CHECKSUM ERROR IN 8-BIT ROM/
20876  125340  052523  020115  051105
20877  125346  047522  020122  047111
20878  125354  034040  041055  052111
20879  125362  051040  046517     000
20880  125367     124  046511  047505  EM56:   .ASCIZ  /TIMEOUT READING LKS/
20881  125374  052125  051040  040505
20882  125402  044504  043516  046040
20883  125410  051513     000
20884  125413     114  051513  030074  EM57:   .ASCIZ  /LKS<07> DOES NOT BECOME 1/
20885  125420  037067  042040  042517
20886  125426  020123  047516  020124
20887  125434  042502  047503  042515
20888  125442  030440     000
20889  125445     127  044522  042524  EM60:   .ASCIZ  /WRITE REFERENCE DOESN'T CLEAR LKS<07>/
20890  125452  051040  043105  051105
20891  125460  047105  042503  042040
20892  125466  042517  047123  052047
20893  125474  041440  042514  051101
20894  125502  046040  051513  030074
20895  125510  037067     000
20896  125513     111  046114  043505  EM61:   .ASCIZ  /ILLEGAL LKS INTERRUPTS/
20897  125520  046101  046040  051513
20898  125526  044440  052116  051105
20899  125534  052522  052120  000123
20900  125542  051120  041517  051505  EM62:   .ASCIZ  /PROCESSOR INTERRUPTS DON'T CLEAR LKS<07>/
20901  125550  047523  020122  047111
20902  125556  042524  051122  050125
20903  125564  051524  042040  047117
```

```
20904  125572  052047  041440  042514
20905  125600  051101  046040  051513
20906  125606  030074  037067     000
20907  125613     114  051513  051040  EM63:   .ASCIZ  /LKS READY DOESN'T GO LOW/
20908  125620  040505  054504  042040
20909  125626  042517  047123  052047
20910  125634  043440  020117  047514
20911  125642  000127
20912  125644  051127  047117  020107  EM64:   .ASCIZ  /WRONG NUMBER OF LKS INTERRUPTS/
20913  125652  052516  041115  051105
20914  125660  047440  020106  045514
20915  125666  020123  047111  042524
20916  125674  051122  050125  051524
20917  125702     000
20918  125703     114  051513  044440  EM65:   .ASCIZ  /LKS INTERRUPTS HAPPEN AT WRONG PRIORITY/
20919  125710  052116  051105  052522
20920  125716  052120  020123  040510
20921  125724  050120  047105  040440
20922  125732  020124  051127  047117
20923  125740  020107  051120  047511
20924  125746  044522  054524     000
20925  125753     102  051503  036122  EM66:   .ASCIZ  /BCSR<12> DOES NOT DISABLES LKS/
20926  125760  031061  020076  047504
20927  125766  051505  047040  052117
20928  125774  042040  051511  041101
20929  126002  042514  020123  045514
20930  126010  000123
20931  126012  041502  051123  030474  EM67:   .ASCIZ  /BCSR<13> DOESN'T SET LKS<06>/
20932  126020  037063  042040  042517
20933  126026  047123  052047  051440
20934  126034  052105  046040  051513
20935  126042  030074  037066     000
20936  126047     122  051505  052105  EM70:   .ASCIZ  /RESET DOESN'T SET LKS<7>/
20937  126054  042040  042517  047123
20938  126062  052047  051440  052105
20939  126070  046040  051513  033474
20940  126076  000076
20941  126100  042522  042523  020124  EM71:   .ASCIZ  /RESET DOESN'T CLEAR LKS<06>/
20942  126106  047504  051505  023516
20943  126114  020124  046103  040505
20944  126122  020122  045514  036123
20945  126130  033060  000076
20946  126134  044524  042515  052517  EM72:   .ASCIZ  /TIMEOUT READING SLU REGISTERS/
20947  126142  020124  042522  042101
20948  126150  047111  020107  046123
20949  126156  020125  042522  044507
20950  126164  052123  051105  000123
20951  126172  051105  047522  020122  EM73:   .ASCIZ  /ERROR IN XMIT READY/
20952  126200  047111  054040  044515
20953  126206  020124  042522  042101
20954  126214  000131
20955  126216  041522  051123  033474  EM74:   .ASCIZ  /RCSR<7> DOESN'T BECOME 1/
20956  126224  020076  047504  051505
20957  126232  023516  020124  042502
20958  126240  047503  042515  030440
20959  126246     000
```

```
20960  126247     127  047522  043516  EM75:   .ASCIZ  /WRONG CHARACTER RECEIVED/
20961  126254  041440  040510  040522
20962  126262  052103  051105  051040
20963  126270  041505  044505  042526
20964  126276  000104
20965  126300  041522  051123  030074  EM76:   .ASCIZ  /RCSR<07> NOT CLEARED AFTER READING RBUF/
20966  126306  037067  047040  052117
20967  126314  041440  042514  051101
20968  126322  042105  040440  052106
20969  126330  051105  051040  040505
20970  126336  044504  043516  051040
20971  126344  052502  000106
20972  126350  041530  051123  030074  EM77:   .ASCIZ  /XCSR<07> NOT SET ON RESET/
20973  126356  037067  047040  052117
20974  126364  051440  052105  047440
20975  126372  020116  042522  042523
20976  126400  000124
20977  126402  041522  051123  030074  EM100:  .ASCIZ  /RCSR<07> NOT CLEARED ON RESET/
20978  126410  037067  047040  052117
20979  126416  041440  042514  051101
20980  126424  042105  047440  020116
20981  126432  042522  042523  000124
20982  126440  046123  020125  047111  EM101:  .ASCIZ  /SLU INTERRUPTS HAPPEN AT 4/
20983  126446  042524  051122  050125
20984  126454  051524  044040  050101
20985  126462  042520  020116  052101
20986  126470  032040     000
20987  126473     122  051505  052105  EM102:  .ASCIZ  /RESET DOES NOT CLEAR PROPER BITS IN SLU REGISTERS/
20988  126500  042040  042517  020123
20989  126506  047516  020124  046103
20990  126514  040505  020122  051120
20991  126522  050117  051105  041040
20992  126530  052111  020123  047111
20993  126536  051440  052514  051040
20994  126544  043505  051511  042524
20995  126552  051522     000
20996  126555     124  040522  051516  EM103:  .ASCIZ  /TRANSMIT INTERRUPT DOES NOT CLEAR XCSR<07>
20997  126562  044515  020124  047111
20998  126570  042524  051122  050125
20999  126576  020124  047504  051505
21000  126604  047040  052117  041440
21001  126612  042514  051101  054040
21002  126620  051503  036122  033460
21003  126626  000076
21004  126630  042522  042503  053111  EM104:  .ASCIZ  /RECEIVE INTERRUPTS DON'T CLEAR RCSR<07>/
21005  126636  020105  047111  042524
21006  126644  051122  050125  051524
21007  126652  042040  047117  052047
21008  126660  041440  042514  051101
21009  126666  051040  051503  036122
21010  126674  033460  000076
21011  126700  051102  040505  020113  EM105:  .ASCIZ  /BREAK CONDITION DOES NOT SET RBUF PROPERLY/
21012  126706  047503  042116  052111
21013  126714  047511  020116  047504
21014  126722  051505  047040  052117
21015  126730  051440  052105  051040
```

```
21016  126736  052502  020106  051120
21017  126744  050117  051105  054514
21018  126752     000
21019  126753     122  052502  020106  EM106:  .ASCIZ  /RBUF (15-11) WASN'T CLEARED ON NEXT CHARACTER/
21020  126760  030474  026465  030461
21021  126766  020076  040527  047123
21022  126774  052047  041440  042514
21023  127002  051101  042105  047440
21024  127010  020116  042516  052130
21025  127016  041440  040510  040522
21026  127024  052103  051105     000
21027  127031     123  052514  044440  EM107:  .ASCIZ  /SLU INTERRUPTS DON'T HAPPNEN/
21028  127036  052116  051105  052522
21029  127044  052120  020123  047504
21030  127052  023516  020124  040510
21031  127060  050120  042516  000116
21032  127066  051105  047522  020122  EM110:  .ASCIZ  /ERROR IN WRITING TO SLU REGISTERS/
21033  127074  047111  053440  044522
21034  127102  044524  043516  052040
21035  127110  020117  046123  020125
21036  127116  042522  044507  052123
21037  127124  051105  000123
21038  127130  044506  051522  020124  EM111:  .ASCIZ  /FIRST CHARACTER WAS NOT OVERRUN BY THE SECOND/
21039  127136  044103  051101  041501
21040  127144  042524  020122  040527
21041  127152  020123  047516  020124
21042  127160  053117  051105  052522
21043  127166  020116  054502  052040
21044  127174  042510  051440  041505
21045  127202  047117  000104
21046  127206  053117  051105  052522  EM112:  .ASCIZ  /OVERRUN CONDITION DOES NOT SET PROPER BITS IN RBUF/
21047  127214  020116  047503  042116
21048  127222  052111  047511  020116
21049  127230  047504  051505  047040
21050  127236  052117  051440  052105
21051  127244  050040  047522  042520
21052  127252  020122  044502  051524
21053  127260  044440  020116  041122
21054  127266  043125     000
21055  127271     117  042526  051122  EM113:  .ASCIZ  /OVERRUN BITS WERE NOT CLEARED ON THE NEXT CHARACTER
21056  127276  047125  041040  052111
21057  127304  020123  042527  042522
21058  127312  047040  052117  041440
21059  127320  042514  051101  042105
21060  127326  047440  020116  044124
21061  127334  020105  042516  052130
21062  127342  041440  040510  040522
21063  127350  052103  051105     000
21064  127355     105  051122  051117  EM114:  .ASCIZ  \ERROR ON XCSR<2>\
21065  127362  047440  020116  041530
21066  127370  051123  031074  000076
21067  127376  051105  047522  020122  EM115:  .ASCIZ  /ERROR IN TAG STORE FROM STANDALONE MODE/
21068  127404  047111  052040  043501
21069  127412  051440  047524  042522
21070  127420  043040  047522  020115
21071  127426  052123  047101  040504
```

```
21072  127434  047514  042516  046440
21073  127442  042117  000105
21074  127446  046504  020101  040524  EM116:  .ASCIZ  /DMA TAG PARITY DOES NOT GENERATE PROPER RESPONSE/
21075  127454  020107  040520  044522
21076  127462  054524  042040  042517
21077  127470  020123  047516  020124
21078  127476  042507  042516  040522
21079  127504  042524  050040  047522
21080  127512  042520  020122  042522
21081  127520  050123  047117  042523
21082  127526     000
21083  127527     115  042523  036122  EM117:  .ASCIZ  /MSER<13>NOT SET IN STANDALONE MODE/
21084  127534  031461  047076  052117
21085  127542  051440  052105  044440
21086  127550  020116  052123  047101
21087  127556  040504  047514  042516
21088  127564  046440  042117  000105
21089  127572  046504  020101  051127  EM120:  .ASCIZ  /DMA WRITE HITS DON'T INVALIDATE CACHE/
21090  127600  052111  020105  044510
21091  127606  051524  042040  047117
21092  127614  052047  044440  053116
21093  127622  046101  042111  052101
21094  127630  020105  040503  044103
21095  127636  000105
21096  127640  047111  041040  047514  EM121:  .ASCIZ  /IN BLOCK MODE ON WRITE DMA HITS NOT EVERYTHING IS INVALIDATED/
21097  127646  045503  046440  042117
21098  127654  020105  047117  053440
21099  127662  044522  042524  042040
21100  127670  040515  044040  052111
21101  127676  020123  047516  020124
21102  127704  053105  051105  052131
21103  127712  044510  043516  044440
21104  127720  020123  047111  040526
21105  127726  044514  040504  042524
21106  127734  000104
21107  127736  042522  042101  042040  EM122:  .ASCIZ  /READ DMA HIT IS WRONG/
21108  127744  040515  044040  052111
21109  127752  044440  020123  051127
21110  127760  047117  000107
21111  127764  051105  047522  020122  EM123:  .ASCIZ  /ERROR IN Q22BE DMA CYCLES/
21112  127772  047111  050440  031062
21113  130000  042502  042040  040515
21114  130006  041440  041531  042514
21115  130014  000123
21116  130016  044520  050522  044440  EM124:  .ASCIZ  /PIRQ INTERRUPTS DON'T TAKE PRIORITY OVER Q BUS INTERRUPTS/
21117  130024  052116  051105  052522
21118  130032  052117  020123  047504
21119  130040  023516  020124  040524
21120  130046  042513  050040  044522
21121  130054  051117  052111  020131
21122  130062  053117  051105  050440
21123  130070  041040  051525  044440
21124  130076  052116  051105  052522
21125  130104  052120  000123
21126  130110  047516  050040  053517  EM125:  .ASCIZ  /NO POWER DOWN TRAP TO 24 OCCUR/
21127  130116  051105  042040  053517
```

```
21128  130124  020116  051124  050101
21129  130132  052040  020117  032062
21130  130140  047440  041503  051125
21131  130146     000
21132  130147     105  051122  051117  EM126:  .ASCIZ  /ERROR DOING Q22BE INTERRUPTS/
21133  130154  042040  044517  043516
21134  130162  050440  031062  042502
21135  130170  044440  052116  051105
21136  130176  052522  052120  000123
21137  130204  051105  047522  020122  EM127:  .ASCIZ  /ERROR IN OPERATION OF PMG COUNTER/
21138  130212  047111  047440  042520
21139  130220  040522  044524  047117
21140  130226  047440  020106  046520
21141  130234  020107  047503  047125
21142  130242  042524  000122
21143  130246  047125  054105  042520  EM130:  .ASCIZ  /UNEXPECTED TRAP TO 4/
21144  130254  052103  042105  052040
21145  130262  040522  020120  047524
21146  130270  032040     000
21147  130273     105  051122  051117  EM131:  .ASCIZ  /ERROR WRITING TO LKS<6>/
21148  130300  053440  044522  044524
21149  130306  043516  052040  020117
21150  130314  045514  036123  037066
21151  130322     000
21152  130323     105  051122  051117  EM132:  .ASCIZ  /ERROR IN MAITENANCE REGISTER/
21153  130330  044440  020116  040515
21154  130336  052111  047105  047101
21155  130344  042503  051040  043505
21156  130352  051511  042524  000122
21157
21158  130360  052040  051505  004524  DH1:    .ASCII  / TEST  ERROR/<15><12>
21159  130366  051105  047522  006522
21160  130374     012
21161  130375     040  021440  020011          .ASCIZ  /  #     PC/
21162  130402  041520     000
21163  130405     040  042524  052123  DH4:    .ASCII  / TEST  ERROR   EXPCTED RECEIVED/<15><12>
21164  130412  042411  051122  051117
21165  130420  042411  050130  052103
21166  130426  042105  051011  041505
21167  130434  044505  042526  006504
21168  130442     012
21169  130443     040  021440  020011          .ASCIZ  /  #     PC      DATA     DATA/
21170  130450  041520  020011  040504
21171  130456  040524  020040  020040
21172  130464  042040  052101  000101
21173  130472  052040  051505  004524  DH5:    .ASCII  / TEST  ERROR   HITMIS  DATA IN DATA IN/<15><12>
21174  130500  051105  047522  004522
21175  130506  044510  046524  051511
21176  130514  042011  052101  020101
21177  130522  047111  042011  052101
21178  130530  020101  047111  005015
21179  130536  020040  004443  050040          .ASCIZ  /  #     PC      REG.    CACHE   MEMORY/
21180  130544  004503  051040  043505
21181  130552  004456  040503  044103
21182  130560  004505  042515  047515
21183  130566  054522     000
```

```
21184  130571    040  042524  052123  DH7:    .ASCII  / TEST  ERROR   ADDRESS MSER/<15><12>
21185  130576  042411  051122  051117
21186  130604  040411  042104  042522
21187  130612  051523  046411  042523
21188  130620  006522     012
21189  130623    040  021440  020011          .ASCIZ  /  #     PC      ACCESSED/
21190  130630  041520  040411  041503
21191  130636  051505  042523  000104
21192  130644  052040  051505  004524  DH24:   .ASCII  / TEST  ERROR   NUMBER/<15><12>
21193  130652  051105  047522  004522
21194  130660  052516  041115  051105
21195  130666  005015
21196  130670  020040  004443  050040          .ASCIZ  /  #     PC      OF HITS/
21197  130676  004503  043117  044040
21198  130704  052111  000123
21199  130710  051105  047522  004522  DH27:   .ASCII  \ERROR  ERROR   MSER    HIT/MISS\<15><12>
21200  130716  051105  047522  004522
21201  130724  051515  051105  044011
21202  130732  052111  046457  051511
21203  130740  006523     012
21204  130743    040  021440  020011          .ASCIZ  /  #     PC/
21205  130750  041520     000
21206  130753    040  042524  052123  DH41:   .ASCII  / TEST  ERROR   INSTRUCTION/<15><12>
21207  130760  042411  051122  051117
21208  130766  044411  051516  051124
21209  130774  041525  044524  047117
21210  131002  005015
21211  131004  020040  004443  050040          .ASCIZ  /  #     PC       OPCODE/
21212  131012  004503  047440  041520
21213  131020  042117  000105
21214  131024  052040  051505  004524  DH43:   .ASCII  / TEST  ERROR   EXPECTD RECEIVD CACHE/<15><12>
21215  131032  051105  047522  004522
21216  131040  054105  042520  052103
21217  131046  004504  042522  042503
21218  131054  053111  004504  040503
21219  131062  044103  006505     012
21220  131067    040  021440  020011          .ASCIZ  /  #     PC      DATA    DATA    LOCATION/
21221  131074  041520  020011  040504
21222  131102  040524  020011  040504
21223  131110  040524  046011  041517
21224  131116  052101  047511  000116
21225  131124  052040  051505  004524  DH47:   .ASCII  / TEST  ERROR   ADDRESS ADDRESS/<15><12>
21226  131132  051105  047522  004522
21227  131140  042101  051104  051505
21228  131146  004523  042101  051104
21229  131154  051505  006523     012
21230  131161    040  021440  020011          .ASCIZ  /  #     PC      <21-16>  <15-0>/
21231  131166  041520  036011  030462
21232  131174  030455  037066  020011
21233  131202  030474  026465  037060
21234  131210     000
21235  131211    040  042524  052123  DH65:   .ASCII  / TEST  ERROR   PRIORITY/<15><12>
21236  131216  042411  051122  051117
21237  131224  050011  044522  051117
21238  131232  052111  006531     012
21239  131237    040  021440  020011          .ASCIZ  /  #     PC      LEVEL/
```

```
21240  131244  041520  046011  053105
21241  131252  046105  000
21242  131255  040     042524  052123  DH72:   .ASCII  / TEST  ERROR   ADDRESS/<15><12>
21243  131262  042411  051122  051117
21244  131270  040411  042104  042522
21245  131276  051523  005015
21246  131302  020040  004443  050040          .ASCIZ  /  #     PC     FAILED/
21247  131310  004503  040506  046111
21248  131316  042105  000
21249  131321  040     042524  052123  DH105:  .ASCII  / TEST  ERROR   RBUF/<15><12>
21250  131326  042411  051122  051117
21251  131334  051011  052502  006506
21252  131342  012
21253  131343  040     021440  020011          .ASCIZ  /  #     PC/
21254  131350  041520  000
21255  131353  040     042524  052123  DH115:  .ASCII  / TEST  ERROR   MSER    ADDRESS/<15><12>
21256  131360  042411  051122  051117
21257  131366  046411  042523  004522
21258  131374  042101  051104  051505
21259  131402  006523  012
21260  131405  040     021440  020011          .ASCIZ  /  #     PC              ACCESSED/
21261  131412  041520  004440  040411
21262  131420  041503  051505  042523
21263  131426  000104
21264                                          .EVEN
21265  131430  001162  001116  000000  DT1:    .WORD   $TMP1,$ERRPC,0
21266  131436  001162  001116  000001  DT4:    .WORD   $TMP1,$ERRPC,R1,CCR,0
21267  131444  177746  000000
21268  131450  001162  001116  000002  DT5:    .WORD   $TMP1,$ERRPC,R2,R1,$GDDAT,0
21269  131456  000001  001124  000000
21270  131464  001162  001116  001122  DT7:    .WORD   $TMP1,$ERRPC,$BDADR,MSER,0
21271  131472  177744  000000
21272  131476  001162  001116  001124  DT14:   .WORD   $TMP1,$ERRPC,$GDDAT,TSTLOC,0
21273  131504  003122  000000
21274  131510  001162  001116  001124  DT17:   .WORD   $TMP1,$ERRPC,$GDDAT,RECDAT,0
21275  131516  110504  000000
21276  131522  001162  001116  000003  DT24:   .WORD   $TMP1,$ERRPC,R3,0
21277  131530  000000
21278  131532  001162  001116  177744  DT27:   .WORD   $TMP1,$ERRPC,MSER,R3,0
21279  131540  000003  000000
21280  131544  001162  001116  001124  DT35:   .WORD   $TMP1,$ERRPC,$GDDAT,MSER,0
21281  131552  177744  000000
21282  131556  001162  001116  001126  DT41:   .WORD   $TMP1,$ERRPC,$BDDAT,0
21283  131564  000000
21284  131566  001162  001116  000001  DT43:   .WORD   $TMP1,$ERRPC,R1,RECDAT,$BDADR,0
21285  131574  110504  001162  000000
21286  131602  001162  001116  172354  DT47:   .WORD   $TMP1,$ERRPC,KIPAR6,$BDADR,0
21287  131610  001122  000000
21288  131614  001162  001116  000001  DT50:   .WORD   $TMP1,$ERRPC,R1,$BDADR,0
21289  131622  001122  000000
21290  131626  001162  001116  001124  DT51:   .WORD   $TMP1,$ERRPC,$GDDAT,PCR,0
21291  131634  177522  000000
21292  131640  001162  001116  001124  DT52:   .WORD   $TMP1,$ERRPC,$GDDAT,BCSR,0
21293  131646  177520  000000
21294  131652  001162  001116  001124  DT64:   .WORD   $TMP1,$ERRPC,$GDDAT,LKSFL,0
21295  131660  002702  000000
```

```
21296  131664  001162  001116  001124  DT65:   .WORD   $TMP1,$ERRPC,$GDDAT,0
21297  131672  000000
21298  131674  001162  001116  001124  DT75:   .WORD   $TMP1,$ERRPC,$GDDAT,$BDDAT,0
21299  131702  001126  000000
21300  131706  001162  001116  177562  DT105:  .WORD   $TMP1,$ERRPC,RBUF,0
21301  131714  000000
21302  131716  001162  001116  001126  DT115:  .WORD   $TMP1,$ERRPC,$BDDAT,KIPAR6,$BDADR,0
21303  131724  172354  001122  000000
21304  131732  001162  001122  000000  DT130:  .WORD   $TMP1,$BDADR,0
```

```
                                        .SBTTL  MODIFIED ERROR MESSAGE TYPEOUT ROUTINE
                                        ;;*****************************************************************
                                        ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
                                        ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
                                        ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
                                        ;*
                                        ;*THE ONLY DIFFERENCE BETWEEN THIS ROUTINE AND THE ORIGINAL "$ERRTYP" FROM
                                        ;*SYSMAC IS THAT YOU CAN PASS INFORMATION IN GENERAL PURPOSE REGISTERS TO THIS
                                        ;*ROUTINE. THE GENERAL PURPOSE REGISTERS USED ARE TO BE SPECIFIED IN DT*
                                        ;*FORMAT. R0 SHOULD NOT BE USED.

131740                                  ERTYPE:
131740  005037  001162                          CLR     $TMP1           ;;JUST CLEAR IT
131744  113737  001102  001162                  MOVB    $TSTNM,$TMP1    ;;STORE TEST NUMBER
131752  104401  001175                          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
131756  010046                                  MOV     R0,-(SP)        ;;SAVE R0
131760  005000                                  CLR     R0              ;;PICKUP THE ITEM INDEX
131762  153700  001114                          BISB    @#$ITEMB,R0
131766  001004                                  BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
                                                                        ;;TYPE THE PC OF THE ERROR
131770  013746  001116                          MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
                                                                        ;;ERROR ADDRESS
131774  104402                                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
131776  000426                                  BR      6$              ;;GET OUT
132000  005300                          1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
132002  006300                                  ASL     R0              ;;      WORK FOR THE ERROR TABLE
132004  006300                                  ASL     R0
132006  006300                                  ASL     R0
132010  062700  001324                          ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
132014  012037  132024                          MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
132020  001404                                  BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
132022  104401                                  TYPE
132024  000000                          2$:     .WORD   0               ;;ERROR MESSAGE POINTER GOES HERE
132026  104401  001175                          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
132032  012037  132042                  3$:     MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
132036  001404                                  BEQ     5$              ;;SKIP TYPEOUT IF 0
132040  104401                                  TYPE                    ;;TYPE THE "DATA HEADER"
132042  000000                          4$:     .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
132044  104401  001175                          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
132050  011000                          5$:     MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
132052  001004                                  BNE     7$              ;;GO TYPE THE DATA
132054  012600                          6$:     MOV     (SP)+,R0        ;;RESTORE R0
132056  104401  001175                          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
132062  000207                                  RTS     PC              ;;RETURN
132064                                  7$:
132064  021027  000005                          CMP     (R0),#5         ;;GENERAL PURPOSE REGISTER?
132070  101021                                  BHI     9$              ;;IF NOT, GO TYPE DATA
132072  042737  000700  132126                  BIC     #BIT8!BIT7!BIT6,8$      ;;CLEAR BITS FOR SOURCE REGISTER
132100  011037  001160                          MOV     (R0),$TMP0      ;;SAVE (R0)
132104  000337  001160                          SWAB    $TMP0           ;;GET REGISTER NUMBER TO HIGH BYTE
132110  006237  001160                          ASR     $TMP0           ;;GET REGISTER NUMBER TO BITS 8-6
132114  006237  001160                          ASR     $TMP0
132120  053737  001160  132126                  BIS     $TMP0,8$        ;;SET BITS IN MOV INSTRUCTION
                                                                        ;;ACCORDING TO REGISTER NUMBER
132126  010046                          8$:     MOV     R0,-(SP)        ;;MOVE CONTEXT OF REGISTER TO STACK
132130  005720                                  TST     (R0)+           ;;ADVANCE POINTER
```

```
132132  000401                          BR      10$             ;;GO TYPE
132134  013046                  9$:     MOV     @(R0)+,-(SP)    ;;IF NOT GPR, SAVE @(R0)+ FOR TYPEOUT
132136  104402                  10$:    TYPOC                   ;;GO TYPE--OCTAL ASCI (ALL DIGITS)
132140  005710                          TST     (R0)            ;;IS THERE ANOTHER NUMBER?
132142  001744                          BEQ     6$              ;;BR IF NO
132144  104401  132152                  TYPE    ,11$            ;;TYPE TWO(2) SPACES
132150  000745                          BR      7$
132152  020040     000          11$:    .ASCIZ  /  /            ;;TWO(2) SPACES
        132156                          .EVEN

                                .SBTTL  GLOBAL SUBROUTINES SECTION

                                ;++
                                ; THE GLOBAL SUBROUTINES SECTION CONTAINS THE SUBROUTINES
                                ; THAT ARE USED IN MORE THAN ONE TEST.
                                ;--

                                ;++
                                ; FUNCTIONAL DESCRIPTION:
                                ;    SUBROUTINE TO INITIALIZE ALL THE MMU REGISTERS


                                ; INPUTS: NONE

                                ; OUTPUTS: NONE

                                ; SUBORDINATE ROUTINES USED: LOAD PARS
                                ;                            LOAD PDRS

                                ; FUNCTIONAL SIDE EFFECTS: NONE

                                ; CALLING SEQUENCE:     JSR     PC,INITMM

132156  012701  172240          INITMM: MOV     #172240,R1              ;BASE ADDRESS OF SIPARS
132162  004737  132320                  JSR     PC,     LDPARS
132166  012701  172260                  MOV     #172260,R1              ;BASE ADDRESS OF SDPARS
132172  004737  132320                  JSR     PC,     LDPARS
132176  012701  172340                  MOV     #172340,R1              ;BASE ADDRESS OF KIPARS
132202  004737  132320                  JSR     PC,     LDPARS
132206  012701  172360                  MOV     #172360,R1              ;BASE ADDRESS OF KDPARS
132212  004737  132320                  JSR     PC,     LDPARS
132216  012701  177640                  MOV     #177640,R1              ;BASE ADDRESS OF UIPARS
132222  004737  132320                  JSR     PC,     LDPARS
132226  012701  177660                  MOV     #177660,R1              ;BASE ADDRESS OF UDPARS
132232  004737  132320                  JSR     PC,     LDPARS
132236  012701  177600                  MOV     #177600,R1              ;BASE ADDRESS OF UIPDRS
132242  004737  132350                  JSR     PC,     LDPDRS
132246  012701  177620                  MOV     #177620,R1              ;BASE ADDRESS OF UDPDRS
132252  004737  132350                  JSR     PC,     LDPDRS
132256  012701  172300                  MOV     #172300,R1              ;BASE ADDRESS OF KIPDRS
132262  004737  132350                  JSR     PC,     LDPDRS
132266  012701  172320                  MOV     #172320,R1              ;BASE ADDRESS OF KDPDRS
132272  004737  132350                  JSR     PC,     LDPDRS
132276  012701  172200                  MOV     #172200,R1              ;BASE ADDRESS OF SIPDRS
132302  004737  132350                  JSR     PC,     LDPDRS
132306  012701  172220                  MOV     #172220,R1              ;BASE ADDRESS OF SDPDRS
```

```
 21417  132312  004737  132350                  JSR     PC,     LDPDRS
 21418  132316  000207                          RTS     PC              ;RETURN
```

```
                                        ;++
                                        ; FUNCTIONAL DESCRIPTION:
                                        ;    SUBROUTINE TO INITIALIZE ALL THE MMU PAGE ADDRESS REGISTERS (PARS).
                                        ;       THIS ROUTINE WILL INITIALIZE 8 PARS STARTING AT A BASE ADDRESS
                                        ;       SUPPLIED BY THE CALLING ROUTINE. PARS 0-5 WILL BE MAPPED FROM
                                        ;       ADDRESS 0 TO ADDRESS 137777 (0-24K). PAR 6 WILL BE MAPPED FROM
                                        ;       ADDRESS 200000 TO 217777 AND PAR 7 WILL BE MAPPED TO THE I/O
                                        ;       PAGE.

                                        ; INPUTS:
                                        ;       R1 CONTAINS THE BASE ADDRESS OF THE NEXT 8 PARS TO BE INITIALIZED

                                        ; OUTPUTS: NONE

                                        ; SUBORDINATE ROUTINES USED: NONE

                                        ; FUNCTIONAL SIDE EFFECTS: NONE

                                        ; CALLING SEQUENCE:     JSR     PC,LDPARS

132320  012702  000006                  LDPARS: MOV     #6,     R2              ;LET LOOP COUNTER COUNT FIRST 6 PARS
132324  005003                                  CLR     R3                      ;INITIALIZE INDEX VALUE
132326  010321                          1$:     MOV     R3,     (R1)+           ;LOAD PARS
132330  062703  000200                          ADD     #200,   R3              ;INDEX IN 4K INCREMENTS
132334  077204                                  SOB     R2,     1$              ;LOAD FIRST SIX PARS
132336  012721  002000                          MOV     #2000,  (R1)+           ;LET PAR6 MAP TO 200000
132342  012711  177600                          MOV     #177600,(R1)            ;LET PAR7 MAP TO I/O PAGE
132346  000207                                  RTS     PC                      ;RETURN
```

```
21447                                   ;++
21448                                   ; FUNCTIONAL DESCRIPTION:
21449                                   ;    SUBROUTINE TO INITIALIZE ALL THE MMU PAGE DECRIPTOR REGISTERS (PDRS).
21450                                   ;       THIS ROUTINE WILL INITIALIZE 8 PDRS STARTING AT A BASE ADDRESS
21451                                   ;       SUPPLIED BY THE CALLING ROUTINE. PDRS 0-5 WILL BE INITIALIZED TO
21452                                   ;       4K READ/WRITE BYPASS AND PDRS 6 AND 7 WILL BE INITIALIZED TO
21453                                   ;       4K READ/WRITE NO BYPASS.
21454                                   ;       NOTE:   THERE IS NO NEED TO BYPASS ON I/O PAGE REFERENCES BECAUSE
21455                                   ;               THE CACHE DOES NOT ALLOCATE ANY OF THESE REFERENCES.
21456
21457                                   ; INPUTS:
21458                                   ;       R1 CONTAINS THE BASE ADDRESS OF THE NEXT 8 PDRS TO BE INITIALIZED
21459
21460                                   ; OUTPUTS: NONE
21461
21462                                   ; SUBORDINATE ROUTINES USED: NONE
21463
21464                                   ; FUNCTIONAL SIDE EFFECTS: NONE
21465
21466                                   ; CALLING SEQUENCE:     JSR     PC,LDPARS
21467
21468  132350  012702  000006          LDPDRS: MOV     #6,     R2              ;LET LOOP COUNTER COUNT FIRST 6 PARS
21469  132354  012721  177406          1$:     MOV     #177406,(R1)+           ;LOAD PDRS WITH 4K READ/WRITE BYPASS
21470  132360  077203                          SOB     R2,     1$              ;LOAD FIRST SIX PDRS
21471  132362  012721  077406                  MOV     #77406, (R1)+           ;LET PAR6 BE 4K READ/WRITE NO BYPASS
21472  132366  012711  077406                  MOV     #77406, (R1)            ;LET PAR7 BE 4K READ/WRITE NO BYPASS ALSO
21473  132372  000207                          RTS     PC                      ;RETURN
```

```
                                        ;**
                                        ; FUNCTIONAL DESCRIPTION:
                                        ;    SUBROUTINE TO HANDLE PARITY ERROR ABORTS FROM THE RAM STORE RAM TESTS.

                                        ; INPUTS:
                                        ;       MEMORY SYSTEM ERROR REGISTER CONTAINS BITS INDICATING FAILURE

                                        ; OUTPUTS: NONE

                                        ; SUBORDINATE ROUTINES USED: NONE

                                        ; FUNCTIONAL SIDE EFFECTS: NONE

                                        ; CALLING SEQUENCE: CALLED BY PARITY ABORT
                                        ;       MOV     @#114,  SLOC00  ;SAVE CONTENTS OF PARITY ABORT VECTOR
                                        ;       MOV     #DSPAR, @#114   ;LET VECTOR POINT TO PARITY ABORT ROUTINE
                                        ;
                                        ;       (CACHE PARITY ERROR OCCURS)

132374  011637  001122                  RAMPAR: MOV     (SP),$BDADR             ;STOR ADDESS TRAPPED
132400  032737  000100  177744                  BIT     #BIT06, MSER            ;IF LOW BYTE PARITY ERROR
132406  001401                                  BEQ     1$                      ;THEN
132410  104007                                  ERROR   +7                      ;ERROR
132412  032737  000200  177744          1$:     BIT     #BIT07, MSER            ;IF HIGH BYTE PARITY ERROR
132420  001401                                  BEQ     2$                      ;THEN
132422  104007                                  ERROR   +7                      ;ERROR
132424  032737  000040  177744          2$:     BIT     #BIT05, MSER            ;IF TAG PARITY ERROR
132432  001401                                  BEQ     3$                      ;THEN
132434  104007                                  ERROR   +7                      ;ERROR
132436  005037  177744                  3$:     CLR     MSER                    ;INITIALIZE MSER AFTER ERROR
132442  000002                                  RTI                             ;RETURN
132444  005237  002702                  LKSINT: INC     LKSFL                   ;INCREMENT FLAG
132450  000002                                  RTI

                                        .SBTTL Q22BE SIZE ROUTINE
                                        ;THIS ROUTINE WILL AUTOSIZE FOR UP TO TWO Q22 BUS EXERCISERS.  IF NONE
                                        ;FOUND LOCATIONS CSR1 AND CSR12 WILL BE LEFT ZEROES.  THIS ROUTINE WILL
                                        ;ONLY RUN IN NOT UFD MODE.

132452  032737  001000  177750          Q22SIZ: BIT     #BIT09,MAIREG                   ;UNIBUS SYSTEM?
132460  001401                                  BEQ     1$                              ;IF NOT, ADVANCE TO ROUTINE
132462  000207                                  RTS     PC                              ;OTHERWISE, RETURN
                                        ;
                                        ; PREPARE TO DO SIZING
                                        ;
132464  013701  000004                  1$:     MOV     ERRVEC,R1                       ;STORE TIMEOUT VECTOR
132470  012737  132644  000004                  MOV     #7$,ERRVEC                      ;POINT NEW TO PROGRAM
132476  012737  000340  000006                  MOV     #340,ERRVEC+2                   ;AT PRIORITY 7
132504  005037  001160                          CLR     $TMP0                           ;CLEAR Q22BE COUNTER
132510  012702  170000                          MOV     #170000,R2                      ;FIRST POSSIBLE ADDRESS
132514  012703  000510                          MOV     #510,R3                         ;VECTOR FOR IT
132520  000404                                  BR      3$                              ;TRY THOSE VALUES
                                        ;
                                        ; NOW DO ACTUAL SIZING
                                        ;
132522  062702  000020                  2$:     ADD     #20,R2                          ;GET CSR FOR NEXT Q22BE
```

```
21550  132526  062703  000004            ADD     #4,R3                   ;GET VECTOR FOR NEXT ONE
21551  132532  005712                3$: TST     (R2)                    ;TRY TO ACCESS CSR
21552                                    ;
21553                                    ; IF NO TIMEOUT, STORE EXISTING ADDRESSES TO REGISTERS
21554                                    ;
21555  132534  005737  001160            TST     $TMP0                   ;FIRST Q22BE FOUND?
21556  132540  001010                    BNE     4$                      ;IF SECOND, BRANCH
21557  132542  012705  002644            MOV     #CSR1,R5                ;START WITH CSR1 FOR 1ST
21558  132546  010237  002662            MOV     R2,SIMGOA               ;SIMULTANEOUS GO
21559  132552  062737  000016  002662    ADD     #16,SIMGOA              ;ADDRESS
21560  132560  000402                    BR      5$                      ;BRANCH TO INITIALISE
21561  132562  012705  002664        4$: MOV     #CSR12,R5               ;START WITH CSR12 FOR 2ND
21562  132566  012704  000004        5$: MOV     #4,R4                   ;INITIALISE 5 REGISTERS
21563  132572  010215                    MOV     R2,(R5)                 ;INITIALISE CSR1
21564  132574  011565  000002        6$: MOV     (R5),2(R5)              ;STORE TO NEXT ONE
21565  132600  005725                    TST     (R5)+                   ;GET NEXT ADDRESS
21566  132602  062715  000002            ADD     #2,(R5)                 ;GET ADDRESS, POINT NEXT
21567  132606  077406                    SOB     R4,6$                   ;DO FOR NEXT 4 REGISTERS
21568  132610  010365  000002            MOV     R3,2(R5)                ;STORE INTERRUPT VECTOR
21569  132614  010365  000004            MOV     R3,4(R5)                ;AND PRIORITY
21570  132620  062765  000002  000004    ADD     #2,4(R5)
21571  132626  005237  001160            INC     $TMP0                   ;COUNT Q22BE'S
21572  132632  022737  000002  001160    CMP     #2,$TMP0                ;TWO FOUND?
21573  132640  001406                    BEQ     9$                      ;IF SO, STOP SIZING
21574  132642  000402                    BR      8$                      ;OTHERWISE, CONTINUE SIZING
21575                                    ;
21576                                    ; ON TIMEOUT TRY TO LOOK AT NEXT ADDRESS RANGE
21577                                    ;
21578  132644  005726                7$: TST     (SP)+                   ;RESTORE STACK FROM
21579  132646  005726                    TST     (SP)+                   ;TIMEOUT
21580  132650  022702  170160        8$: CMP     #170160,R2              ;AT THE LAST POSSIBLE?
21581  132654  001322                    BNE     2$                      ;IF NOT, BRANCH
21582  132656  005737  002644        9$: TST     CSR1                    ;1 FOUND?
21583  132662  001402                    BEQ     10$                     ;IF NONE, BRANCH
21584  132664  104401  132676            TYPE    ,ONOQ22                 ;TYPE  FOUND
21585  132670  010137  000004       10$: MOV     R1,ERRVEC               ;RESTORE TIMEOUT VECTOR
21586  132674  000207                    RTS     PC                      ;RETURN
21587
21588  132676  006412  031121  041062 ONOQ22: .ASCIZ  <12><15>/Q22BE USED DURING TESTING/
21589  132704  020105  051525  042105
21590  132712  042040  051125  047111
21591  132720  020107  042524  052123
21592  132726  047111  000107
21593                                .EVEN
21594                                .SBTTL Q22BE INTERRUPT INITIALISE ROUTINE
21595                                ;THIS ROUTINE WILL INITIALISE Q22BE TO INTERRUPT AT A PRIORITY AT (R0)+
21596                                ;AT THE STARTING ADDRESS IN R3. THE TEST HAVE TO SET ACTUAL DONE BIT
21597                                ;BY CLEARING GO.
21598
21599  132732  005013            Q22INT: CLR     (R3)                    ;CLEAR TRANSFER TYPE IN CSR1
21600  132734  052710  000001            BIS     #BIT00,(R0)             ;ZERO DONE
21601  132740  011063  000002            MOV     (R0),2(R3)              ;SET PRIORITY IN CSR2
21602  132744  042710  000001            BIC     #BIT00,(R0)             ;PREPARE TO SET DONE
21603  132750  000207                    RTS     PC
21604
21605                                .SBTTL DMATRN DATO CYCLE THRU Q22BE
```

```
21586                                   ;THIS ROUTINE PERFORMS DATO FROM A LOCATION TEMP THRU THE FIRST
21587                                   ;FOUND Q22BE STARTING AT LOCATION @CSR1. R0 HAS 0 IF ONLY 1 TRANSFER IS
21588                                   ;TO BE PERFORMED. OTHERWISE 16 BLOCK MODE TRANSFERS ARE TO BE PERFORMED.
21589                                   ;IN THE LATTER CASE ADDRESS AND WORD COUNT HAS TO BE LOADED BEFORE.
21590
21591  132752  012777  012525  047674  DMATRN: MOV     #12525,@DATA            ;DATA USED
21592  132760  005700                          TST     R0                      ;DO 1 WORD?
21593  132762  001404                          BEQ     1$                      ;IF YES, BRANCH
21594  132764  012777  001001  047654          MOV     #BIT09!BIT00,@CSR2      ;BLOCK MODE, GO
21595  132772  000414                          BR      2$                      ;BRANCH TO DO IT
21596  132774  012777  001601  047642  1$:     MOV     #1601,@CSR1             ;RESET LATENCY COUNT,DATO
21597  133002  012777  002710  047640          MOV     #TEMP,@BA               ;LOAD DMA ADDRESS
21598  133010  012777  177777  047634          MOV     #177777,@WC             ;DO 1 WORD
21599  133016  012777  000001  047622          MOV     #BIT00,@CSR2            ;DO IT
21600  133024  105777  047616          2$:     TSTB    @CSR2                   ;DMA DONE?
21601  133030  100375                          BPL     2$                      ;WAIT TILL DONE
21602  133032  000207                  3$:     RTS     PC                      ;RETURN FROM SUBROUTINE
21603
21604                                   .SBTTL DMARD DATI THRU Q22BE
21605                                   ;THIS ROUTINE PERFORMS DATI CYCLE THRU Q22BE IN EITHER BLOCK MODE OR A SINGLE
21606                                   ;TRANSFER MODE. MEMORY LOCATION USED IS TEMP. R0 IS ZERO FOR SINGLE TRANSFER
21607
21608  133034  012777  002710  047606  DMARD:  MOV     #TEMP,@BA               ;LOAD DMA ADDRESS
21609  133042  005700                          TST     R0                      ;DO 1 WORD?
21610  133044  001412                          BEQ     1$                      ;IF YES, BRANCH
21611  133046  012777  001507  047570          MOV     #1507,@CSR1             ;16 DATIB
21612  133054  012777  177770  047570          MOV     #177770,@WC             ;DO 8 WORD
21613  133062  012777  001001  047556          MOV     #BIT09!BIT00,@CSR2      ;BLOCK MODE GO
21614  133070  000411                          BR      2$                      ;GO CHECK
21615  133072  012777  001407  047544  1$:     MOV     #1407,@CSR1             ;RESET LATENCY COUNT,DATI
21616                                                                          ;LOAD NEW DATA TO DATA R.
21617  133100  012777  177777  047544          MOV     #177777,@WC             ;DO 1 WORD
21618  133106  012777  000001  047532          MOV     #BIT00,@CSR2            ;DO IT
21619  133114  105777  047526          2$:     TSTB    @CSR2                   ;DMA DONE?
21620  133120  100375                          BPL     2$                      ;WAIT TILL DONE
21621  133122  000207                  3$:     RTS     PC                      ;RETURN FROM SUBROUTINE
21622
21623
21624
21625  133124  011637  001122          TOUT:   MOV     (SP),$BDADR             ;STORE TRAPPED PC
21626  133130  104130                          ERROR   .130                    ;UNEXPECTED TRAP
21627  133132  000002                          RTI
21628
21629                                   ;
21630                                   ;MMU GLOBAL SUBROUTINES
21631                                   ;
21632                                   ;
21633                                   ;ROUTINE TO INITIALIZE MEMORY MANAGEMENT
21634                                   ;
21635  133134  010046                  MMU:    MOV     R0,-(SP)                ;SAVE CONTENTS OF REGISTERS
21636  133136  010146                          MOV     R1,-(SP)                ;
21637  133140  010246                          MOV     R2,-(SP)                ;
21638  133142  012700  177600                  MOV     #177600,R0              ;
21639  133146  004737  133234                  JSR     PC,PDR                  ;INIT I AND D USER PDR'S
21640  133152  004737  133256                  JSR     PC,PAR                  ;INIT I USER PAR'S
21641  133156  004737  133256                  JSR     PC,PAR                  ;INIT D USER PAR'S
```

```
21642  133162  012700  172200                  MOV     #172200,R0
21643  133166  004737  133234                  JSR     PC,PDR          ;INIT I AND D SUP PDR'S
21644  133172  004737  133256                  JSR     PC,PAR          ;INIT I SUP PAR'S
21645  133176  004737  133256                  JSR     PC,PAR          ;INIT D SUP PAR'S
21646  133202  004737  133234                  JSR     PC,PDR          ;INIT I AND D KER PDR'S
21647  133206  004737  133256                  JSR     PC,PAR          ;INIT I KER PAR'S
21648  133212  004737  133256                  JSR     PC,PAR          ;INIT D KER PAR'S
21649  133216  012737  000027  172516          MOV     #27,@#172516    ;INIT MMR3
21650  133224  012602                          MOV     (SP)+,R2        ;RESTORE REGISTERS
21651  133226  012601                          MOV     (SP)+,R1        ;
21652  133230  012600                          MOV     (SP)+,R0        ;
21653  133232  000207                          RTS     PC              ;RETURN
21654                                  ;
21655                                  ;ROUTINE TO INITIALIZE PDR'S
21656                                  ;
21657  133234  005002                  PDR:    CLR     R2              ;INIT CNTR
21658  133236  012720  077406          PDR1:   MOV     #77406,(R0)+    ;INIT PDR
21659  133242  062702  000001                  ADD     #1,R2           ;INCREMENT CNTR
21660  133246  022702  000020                  CMP     #16.,R2         ;ARE WE DONE?
21661  133252  001371                          BNE     PDR1            ;BRANCH IF NOT
21662  133254  000207                          RTS     PC              ;RETURN
21663                                  ;
21664                                  ;ROUTINE TO INITIALIZE PAR'S
21665                                  ;
21666  133256  005001                  PAR:    CLR     R1              ;SETUP TO INIT PAR
21667  133260  010120                  PAR1:   MOV     R1,(R0)+        ;INIT PAR
21668  133262  062701  000200                  ADD     #200,R1         ;GET READY FOR NEXT PAR
21669  133266  022701  001600                  CMP     #1600,R1        ;REACHED A PAR7
21670  133272  001372                          BNE     PAR1            ;BRANCH IF NOT
21671  133274  012720  177600                  MOV     #177600,(R0)+   ;INIT PAR7
21672  133300  000207                          RTS     PC              ;RETURN
21673                                  ;
21674                                  ;TIME OUT ROUTINE
21675                                  ;
21676  133302  005205                  ADDTRP: INC     R5              ;INCREMENT TIME OUT FLAG
21677  133304  000002                          RTI                     ;RETURN
21678                                  ;
21679                                  ;MMU TRAP ROUTINE
21680                                  ;
21681  133306  023727  002776  000001  MMUTRP: CMP     FLAG,#1         ;ARE WE EXPECTING AN ABORT
21682  133314  001401                          BEQ     1$              ;YES GO ON
21683  133316  104002                          ERROR   +2              ;NO GO TO ERROR
21684  133320  010046                  1$:     MOV     R0,-(SP)        ;SAVE CONTENTS OF REG 0
21685  133322  013700  177776                  MOV     @#177776,R0     ;SAVE A COPY OF PSW
21686  133326  072027  177764                  ASH     #-14,R0         ;LOOK AT BITS<15:14>
21687  133332  020027  000002                  CMP     R0,#2           ;WAS PS<15:14>=10
21688  133336  001001                          BNE     OK              ;NO GO ON
21689  133340  000411                          BR      NOTOK           ;YES CHANGE BITS TO 00
21690  133342  013700  177776          OK:     MOV     @#177776,R0     ;SAVE A COPY OF PSW
21691  133346  072027  000002                  ASH     #2,R0           ;LOOK AT BITS<13:12>
21692  133352  072027  177764                  ASH     #-14,R0         ;
21693  133356  020027  000002                  CMP     R0,#2           ;WAS PS<13:12>=10
21694  133362  001002                          BNE     OK1             ;NO GO ON
21695  133364  005066  000004          NOTOK:  CLR     4(SP)           ;CLEAR ILLEGAL MODE FFROM OLD PSW
21696  133370  013737  177572  003004  OK1:    MOV     @#177572,SAVMR0 ;SAVE A COPY OF MMR0
21697  133376  013737  177574  003006          MOV     @#177574,SAVMR1 ;SAVE A COPY OF MMR1
```

```
 21698  133404  013737  177576  003010          MOV     @#177576,SAVMR2     ;SAVE A COPY OF MMR2
 21699  133412  005037  177572                  CLR     @#177572            ;CLEAR ABORT BITS AND TURN MMU OFF
 21700  133416  005037  002776                  CLR     FLAG                ;CLEAR MMU ABORT FLAG
 21701  133422  012600                          MOV     (SP)+,R0            ;RESTORE ORIGINAL CONTENTS OF REG 0
 21702  133424  000002                          RTI                         ;RETURN
```

```
21703
21704  133426  012600                  ;FPP COMMON SUBROUTINES
21705  133430  012605                  WLDTRP: MOV     (SP)+,R0        ;SAVE PC
21706  133432  104003                          MOV     (SP)+,R5        ;SAVE STATUS AND RESTORE STACK
21707  133434  000110                          ERROR   .3
21708                                          JMP     (R0)            ;GO BACK INLINE
21709                                  ;
21710                                  ;
21711  133436  000000                  TRPFLG: .WORD   0
21712  133440  000207                  ERRFP:  RTS     R7
21713  133442  000207                  ERR:    RTS     R7
21714                                  ;
21715                                  ;
21716                                  ;
21717                                  ;
21718                                  ;
21719                                  ;SUBROUTINE DATA VERFICATION -
21720                                  ;
21721                                  ; CALLED BY     JSR     R7,DATVER
21722                                  ;
21723                                  ;INPUT:         (R4)=EXPECTED DATA
21724                                  ;               (R1)=RECEIVED DATA
21725                                  ;
21726                                  ;THIS ROUTINE VERIFIES THAT THE 4 CONSECTIVE WORDS STARTING WITH (R4) ARE
21727                                  ;EQUAL TO THE FOUR WORDS ADDRESSED BY (R1). THE CONTENTS OF R4, AND R1 ARE NOT
21728                                  ;DISTURBED.
21729                                  ;LOCATION "COUNT" , IF NOT EQUAL TO 0 SIGNIFIES DATA ERROR
21730                                  ;IF THE STATUS IS FLOATING MODE, THE LAST TWO BYTES OF RECEIEVED
21731                                  ;ARE SIMPLY CHECKED FOR ZEROS
21732                                  ;
21733
21734  133444  010446                  DATVFR: MOV     R4,-(SP)        ;SAVE R4
21735  133446  010146                          MOV     R1,-(SP)                ;SAVE R1
21736  133450  012737  000003  003060          MOV     #3,COUNT        ;SET UP ITERATION COUNT
21737  133456  000137  133474                  JMP     DAT1            ;
21738                                  ;
21739  133462  010446                  DATVER: MOV     R4,-(SP)        ;SAVE R4
21740  133464  010146                          MOV     R1,-(SP)        ;SAVE R1
21741  133466  012737  000005  003060          MOV     #5,COUNT        ;SET UP ITERATION COUNT
21742  133474  005337  003060          DAT1:   DEC     COUNT
21743  133500  001402                          BEQ     2$              ;BRANCH IF DONE
21744  133502  022421                          CMP     (R4)+,(R1)+             ;
21745  133504  001773                          BEQ     DAT1
21746  133506  012601                  2$:     MOV     (SP)+,R1        ;RESTORE R1
21747  133510  012604                          MOV     (SP)+,R4        ;RESTORE R4
21748  133512  000207                          RTS     R7              ;GO BACK TO CALLING ROUTINE
21749                                                                  ;IF DATA ERROR, COUNT NE 0
```

```
21750
21751
21752                                   .SBTTL  END OF PASS ROUTINE
21753
21754                                   ;;**********************************************************************
21755                                   ;*INCREMENT THE PASS NUMBER ($PASS)
21756                                   ;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
21757                                   ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
21758                                   ;*IF THERES A MONITOR GO TO IT
21759                                   ;*IF THERE ISN'T JUMP TO LOOP
21760
21761  133514                           $EOP:
21762  133514  032737  000100  000052          BIT     #BIT06,@#52
21763  133522  001026                        BNE     $GET42
21764  133524  005037  001102                CLR     $TSTNM          ;;ZERO THE TEST NUMBER
21765  133530  005037  001164                CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
21766  133534  005237  001206                INC     $PASS           ;;INCREMENT THE PASS NUMBER
21767  133540  042737  100000  001206        BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
21768  133546  005327                        DEC     (PC)+           ;;LOOP?
21769  133550  000001                 $EOPCT: .WORD   1
21770  133552  003022                        BGT     $DOAGN          ;;YES
21771  133554  012737                        MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
21772  133556  000001                 $ENDCT: .WORD   1
21773  133560  133550                        $EOPCT
21774  133562  104401  133627                TYPE    ,$ENDMG         ;;TYPE "END PASS #"
21775  133566  013746  001206                MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
21776  133572  104405                        TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
21777  133574  104401  133624                TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
21778  133600  013700  000042         $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
21779  133604  001405                        BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
21780  133606  000005                        RESET                   ;;CLEAR THE WORLD
21781  133610  004710                 $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
21782  133612  000240                        NOP                     ;;SAVE ROOM
21783  133614  000240                        NOP                     ;;FOR
21784  133616  000240                        NOP                     ;;ACT11
21785  133620                          $DOAGN:
21786  133620  000137                        JMP     @(PC)+          ;;RETURN
21787  133622  004562                 $RTNAD: .WORD   LOOP
21788  133624     377       377       000   $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
21789  133627     015  042412  042116  $ENDMG: .ASCIZ  <15><12>/END PASS #/
21790  133634  050040  051501  020123
21791  133642  000043
21792
21793                                   .SBTTL  SCOPE HANDLER ROUTINE
21794
21795                                   ;;**********************************************************************
21796                                   ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
21797                                   ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
21798                                   ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
21799                                   ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
21800                                   ;*SW14=1         LOOP ON TEST
21801                                   ;*SW11=1         INHIBIT ITERATIONS
21802                                   ;*SW09=1         LOOP ON ERROR
21803                                   ;*SW08=1         LOOP ON TEST IN SWR<5:0>
21804                                   ;*CALL
21805                                   ;*      SCOPE           ;;SCOPE=IOT
```

```
21806  133644                          $SCOPE:
21807  133644  104407                          CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
21808  133646  032777  040000  045264  1$:     BIT     @#BIT14,@SWR    ;;LOOP ON PRESENT TEST?
21809  133654  001117                          BNE     $OVER           ;;YES IF SW14=1
21810                                  ;@@@@@START OF CODE FOR THE XOR TESTER@@@@@
21811  133656  000416                  $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
21812                                                                  ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
21813  133660  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
21814  133664  012737  133704  000004          MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
21815  133672  005737  177060                  TST     @#177060        ;;TIME OUT ON XOR?
21816  133676  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
21817  133702  000466                          BR      $SVLAD          ;;GO TO THE NEXT TEST
21818  133704  022626                  5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
21819  133706  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
21820  133712  000426                          BR      7$              ;;LOOP ON THE PRESENT TEST
21821  133714                          6$:;@@@@@END OF CODE FOR THE XOR TESTER@@@@@
21822  133714  032777  000400  045216          BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
21823  133722  001407                          BEQ     2$              ;;BR IF NO
21824  133724  017746  045210                  MOV     @SWR,-(SP)      ;;SET DESIRED TEST NUM. FROM SWR
21825  133730  042716  000300                  BIC     #$SWRMK,(SP)    ;;STRIP AWAY UNDESIRED BITS
21826  133734  122637  001102                  CMPB    (SP)+,$TSTNM    ;;ON THE RIGHT TEST?
21827  133740  001465                          BEQ     $OVER           ;;BR IF YES
21828  133742  105737  001103          2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
21829  133746  001421                          BEQ     3$              ;;BR IF NO
21830  133750  123737  001115  001103          CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
21831  133756  101015                          BHI     3$              ;;BR IF NO
21832  133760  032777  001000  045152          BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
21833  133766  001404                          BEQ     4$              ;;BR IF NO
21834  133770  013737  001110  001106  7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
21835  133776  000446                          BR      $OVER
21836  134000  105037  001103          4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
21837  134004  005037  001164                  CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
21838  134010  000415                          BR      1$              ;;ESCAPE TO THE NEXT TEST
21839  134012  032777  004000  045120  3$:     BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
21840  134020  001011                          BNE     1$              ;;BR IF YES
21841  134022  005737  001206                  TST     $PASS           ;;IF FIRST PASS OF PROGRAM
21842  134026  001406                          BEQ     1$              ;;      INHIBIT ITERATIONS
21843  134030  005237  001104                  INC     $ICNT           ;;INCREMENT ITERATION COUNT
21844  134034  023737  001164  001104          CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
21845  134042  002024                          BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
21846  134044  012737  000001  001104  1$:     MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
21847  134052  013737  134130  001164          MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
21848  134060  105237  001102          $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
21849  134064  113737  001102  001204          MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
21850  134072  011637  001106                  MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
21851  134076  011637  001110                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
21852  134102  005037  001166                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
21853  134106  112737  000001  001115          MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
21854  134114  013777  001102  045020  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
21855  134122  013716  001106                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
21856  134126  000002                          RTI                     ;;FIXES PS
21857  134130  000001                  $MXCNT: 1                       ;;MAX. NUMBER OF ITERATIONS
21858                                  .SBTTL  ERROR HANDLER ROUTINE
21859
21860                                  ;;**************************************************************************
21861                                  ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
```

```
21862                                   ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
21863                                   ;*AND GO TO ERTYPE ON ERROR
21864                                   ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
21865                                   ;*SW15=1        HALT ON ERROR
21866                                   ;*SW13=1        INHIBIT ERROR TYPEOUTS
21867                                   ;*SW10=1        BELL ON ERROR
21868                                   ;*SW09=1        LOOP ON ERROR
21869                                   ;*CALL
21870                                   ;*      ERROR   *N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
21871
21872  134132                           $ERROR:
21873  134132  005737  004060                   TST     UQUIET          ;;TEST FOR USER-QUIET MODE
21874  134136  001403                           BEQ     9$              ;;BRANCH IF FIELD-SERVICE MODE
21875  134140  005000                           CLR     R0              ;;IN CASE R0 HAS A #3 IN IT (^C)
21876  134142  004737  134354                   JSR     PC,ABORT        ;;TEST FOR ABORT CONDITION
21877  134146                           9$:
21878  134146  104407                           CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
21879  134150  052737  001000  177520           BIS     #1000,@CSR              ;ENABLE HALT ON BREAK
21880  134156  105237  001103           7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
21881  134162  001775                           BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
21882  134164  013777  001102  044750           MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
21883  134172  032777  002000  044740           BIT     #BIT10,@SWR     ;;BELL ON ERROR?
21884  134200  001402                           BEQ     1$              ;;NO - SKIP
21885  134202  104401  001170                   TYPE    ,$BELL          ;;RING BELL
21886  134206  005237  001112           1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
21887  134212  011637  001116                   MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
21888  134216  162737  000002  001116           SUB     #2,$ERRPC
21889  134224  117737  044666  001114           MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
21890  134232  032777  020000  044700           BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
21891  134240  001004                           BNE     20$             ;;SKIP TYPEOUTS
21892  134242  004737  131740                   JSR     PC,ERTYPE       ;;GO TO USER ERROR ROUTINE
21893  134246  104401  001175                   TYPE    ,$CRLF
21894  134252                           20$:
21895  134252  122737  000001  001220           CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
21896  134260  001007                           BNE     2$              ;;NO,SKIP APT ERROR REPORT
21897  134262  113737  001114  134274           MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
21898  134270  004737  134532                   JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
21899  134274     000                   21$:    .BYTE   0
21900  134275     000                           .BYTE   0
21901  134276  000777                   22$:    BR      22$             ;;APT ERROR LOOP
21902  134300  005777  044634           2$:     TST     @SWR            ;;HALT ON ERROR
21903  134304  100002                           BPL     3$              ;;SKIP IF CONTINUE
21904  134306  000000                           HALT                    ;;HALT ON ERROR!
21905  134310  104407                           CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
21906  134312  032777  001000  044620   3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
21907  134320  001402                           BEQ     4$              ;;BR IF NO
21908  134322  013716  001110                   MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
21909  134326  005737  001166           4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
21910  134332  001402                           BEQ     5$              ;;BR IF NONE
21911  134334  013716  001166                   MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
21912  134340                           5$:
21913  134340  022737  133610  000042           CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
21914  134346  001001                           BNE     6$              ;;BRANCH IF NO
21915  134350  000000                           HALT                    ;;YES
21916  134352                           6$:
21917  134352  000002                           RTI                     ;;RETURN
```

```
21918
21919
21920
21921                                   .SBTTL  ABORT ROUTINE FOR LCP/ORION UFD MODE
21922
21923
21924  134354  005737  004056          ABORT:  TST     UFDFLG          ;TEST FOR USER FRIENDLY MODE
21925  134360  001454                          BEQ     NOABRT          ;IF NOT UFD THEN CONTINUE NORMAL OPERATION
21926  134362  020027  000032                  CMP     R0,#32          ;IS IT A ↑Z ?
21927  134366  001443                          BEQ     ABORTZ          ;JUST GO BACK TO CHAIN IF IT IS (NO ERROR)
21928  134370  020027  000003                  CMP     R0,#3           ;IS IS A ↑C ?
21929  134374  001404                          BEQ     ABORTC          ;BR TO LOAD ↑C ON XXDP+ STACK (NO ERROR)
21930  134376  005737  004060                  TST     UQUIET          ;TEST FOR USER-QUIET MODE
21931  134402  001443                          BEQ     NOABRT          ;IF FIELD-SERVICE MODE, CONTINUE NORMAL OPERATION
21932                                                                  ; BECAUSE FIELD-SERVICE MODE DOES NOT QUIT ON ERROR
21933  134404  000422                          BR      ABORTE          ;SET DRSERR THEN LEAVE
21934
21935
21936
21937  134406  013737  004052  000030  ABORTC: MOV     SAV30,30        ;RESTORE EMT LOCATION (30)
21938  134414  013737  004054  000032          MOV     SAV32,32        ;RESTORE EMT PRIORITY LOCATION (32)
21939  134422  104043                          EMT     +43             ;GET XXDP STACK LOC. INTO R0 FROM MONITOR
21940  134424  005720                  1$:     TST     (R0)+           ;FIND END OF STACK
21941  134426  001376                          BNE     1$
21942  134430  112760  000057  177777          MOVB    #'/,-1(R0)      ;LOAD SLASH OVER ZERO
21943  134436  112720  000136                  MOVB    #'↑,(R0)+       ;LOAD UPARROW
21944  134442  112720  000103                  MOVB    #'C,(R0)+       ;LOAD C
21945  134446  105010                          CLRB    (R0)            ;MAKE NEW END TO STACK
21946  134450  000412                          BR      ABORTZ          ;NOW LEAVE
21947  134452  013737  004052  000030  ABORTE: MOV     SAV30,30        ;RESTORE EMT LOCATION (30)
21948  134460  013737  004054  000032          MOV     SAV32,32        ;RESTORE EMT PRIORITY LOCATION (32)
21949  134466  104042                          EMT     +42             ;GET DCA LOCATION INTO R0 FROM MONITOR
21950  134470  012760  177777  000042          MOV     #-1,42(R0)      ;SET A -1 INTO LOCATION DRSERR IN MONITOR
21951  134476  013700  000042          ABORTZ: MOV     @#42,R0         ;AND PUT THE MONITOR RETURN ADDRESS IN R0
21952  134502  005037  000042                  CLR     @#42            ;CLEAR MONITOR RETURN FLAG
21953  134506  000137  133610                  JMP     $ENDAD          ;RETURN TO MONITOR-DO NOT PUSH STACK HERE
21954  134512  000207                  NOABRT: RTS     PC              ;IF NOTUFD RETURN TO MAINLINE
21955
21956
21957                                   .SBTTL  APT COMMUNICATIONS ROUTINE
21958
21959                                   ;;*****************************************************************
21960  134514  112737  000001  134760  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
21961  134522  112737  000001  134756  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
21962  134530  000403                          BR      $ATYC
21963  134532  112737  000001  134760  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
21964  134540                          $ATYC:
21965  134540  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
21966  134542  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
21967  134544  105737  134756                  TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
21968  134550  001450                          BEQ     5$              ;;IF NOT:  BR
21969  134552  122737  000001  001220          CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
21970  134560  001031                          BNE     3$              ;;IF NOT:  BR
21971  134562  132737  000100  001221          BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
21972  134570  001425                          BEQ     3$              ;;IF NOT:  BR
21973  134572  017600  000004                  MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
```

```
21974  134576  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
21975  134604  005737  001200          1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
21976  134610  001375                          BNE     1$              ;;IF NOT:  WAIT
21977  134612  010037  001214                  MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
21978  134616  105720                  2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
21979  134620  001376                          BNE     2$
21980  134622  163700  001214                  SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
21981  134626  006200                          ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
21982  134630  010037  001216                  MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
21983  134634  012737  000004  001200          MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
21984  134642  000413                          BR      5$
21985  134644  017637  000004  134670  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
21986  134652  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
21987  134660  013746  177776                  MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
21988  134664  004737  134762                  JSR     PC,$TYPE        ;;CALL TYPE MACRO
21989  134670  000000                  4$:     .WORD   0
21990  134672                          5$:
21991  134672  105737  134760          10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
21992  134676  001416                          BEQ     12$             ;;IF NOT:  BR
21993  134700  005737  001220                  TST     $ENV            ;;RUNNING UNDER APT?
21994  134704  001413                          BEQ     12$             ;;IF NOT:  BR
21995  134706  005737  001200          11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
21996  134712  001375                          BNE     11$             ;;IF NOT:  WAIT
21997  134714  017637  000004  001202          MOV     @4(SP),$FATAL   ;;GET ERROR #
21998  134722  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
21999  134730  005237  001200                  INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
22000  134734  105037  134760          12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
22001  134740  105037  134757                  CLRB    $LFLG           ;;CLEAR LOG FLAG
22002  134744  105037  134756                  CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
22003  134750  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
22004  134752  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
22005  134754  000207                          RTS     PC              ;;RETURN
22006  134756     000          $MFLG:  .BYTE   0               ;;MESSG. FLAG
22007  134757     000          $LFLG:  .BYTE   0               ;;LOG FLAG
22008  134760     000          $FFLG:  .BYTE   0               ;;FATAL FLAG
22009          134762                  .EVEN
22010          000200          APTSIZE=200
22011          000001          APTENV=001
22012          000100          APTSPOOL=100
22013          000040          APTCSUP=040
22014                          .SBTTL  TYPE ROUTINE
22015
22016                          ;;******************************************************************************
22017                          ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
22018                          ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
22019                          ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
22020                          ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
22021                          ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
22022                          ;*
22023                          ;*CALL:
22024                          ;*1) USING A TRAP INSTRUCTION
22025                          ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
22026                          ;*OR
22027                          ;*      TYPE
22028                          ;*      MESADR
22029                          ;*
```

```
22030
22031  134762  105737  001157          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
22032  134766  100002                  BPL     1$              ;;BR IF YES
22033  134770  000000                  HALT                    ;;HALT HERE IF NO TERMINAL
22034  134772  000430                  BR      3$              ;;LEAVE
22035  134774  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
22036  134776  017600  000002          MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
22037  135002  122737  000001  001220  CMPB    @#APTENV,$ENV   ;;RUNNING IN APT MODE
22038  135010  001011                  BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
22039  135012  132737  000100  001221  BITB    @#APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
22040  135020  001405                  BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
22041  135022  010037  135032          MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
22042  135026  004737  134522          JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
22043  135032  000000                  61$:    .WORD   0               ;;MESSAGE ADDRESS
22044  135034  132737  000040  001221  62$:    BITB    @#APTCSUP,$ENVM ;;APT CONSOLE SUPPRESSED
22045  135042  001003                  BNE     60$             ;;YES,SKIP TYPE OUT
22046  135044  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
22047  135046  001005                  BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
22048  135050  005726                  TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
22049  135052  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
22050  135054  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
22051  135060  000002                  RTI                     ;;RETURN
22052  135062  122716  000011          4$:     CMPB    @#HT,(SP)       ;;BRANCH IF <HT>
22053  135066  001430                  BEQ     8$
22054  135070  122716  000200          CMPB    @#CRLF,(SP)     ;;BRANCH IF NOT <CRLF>
22055  135074  001006                  BNE     5$
22056  135076  005726                  TST     (SP)+           ;;POP  <CR><LF> EQUIV
22057  135100  104401                  TYPE                    ;;TYPE A CR AND LF
22058  135102  001175                  $CRLF
22059  135104  105037  135312          CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
22060  135110  000755                  BR      2$              ;;GET NEXT CHARACTER
22061  135112  004737  135174          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
22062  135116  123726  001156          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
22063  135122  001350                  BNE     2$              ;;IF NO GO GET NEXT CHAR.
22064  135124  013746  001154          MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
22065                                                          ;;AND THE NULL CHAR.
22066  135130  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
22067  135134  002770                  BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
22068  135136  004737  135174          JSR     PC,$TYPEC       ;;GO TYPE A NULL
22069  135142  105337  135312          DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
22070  135146  000770                  BR      7$              ;;LOOP
22071
22072                                  ;HORIZONTAL TAB PROCESSOR
22073
22074  135150  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
22075  135154  004737  135174          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
22076  135160  132737  000007  135312  BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
22077  135166  001372                  BNE     9$              ;;TAB STOP
22078  135170  005726                  TST     (SP)+           ;;POP SPACE OFF STACK
22079  135172  000724                  BR      2$              ;;GET NEXT CHARACTER
22080  135174                          $TYPEC:
22081  135174  105777  043744          TSTB    @$TKS           ;;CHAR IN KYBD BUFFER?          ;MJD001
22082  135200  100022                  BPL     10$             ;;BR IF NOT                     ;MJD001
22083  135202  017746  043740          MOV     @$TKB,-(SP)     ;;GET CHAR                      ;MJD001
22084  135206  042716  177600          BIC     #177600,(SP)    ;;STRIP EXTRANEOUS BITS         ;MJD001
22085  135212  122716  000023          CMPB    #$XOFF,(SP)     ;;WAS CHAR XOFF                 ;MJD001
```

```
135216  001012                          BNE     102$            ;;BR IF NOT                                  ;MJD001
135220                          101$:                                                                        ;MJD001
135220  105777  043720                  TSTB    @$TKS           ;;WAIT FOR CHAR                              ;MJD001
135224  100375                          BPL     101$                                                         ;MJD001
135226  117716  043714                  MOVB    @$TKB,(SP)      ;;GET CHAR                                   ;MJD001
135232  042716  177600                  BIC     #177600,(SP)    ;;STRIP IT                                   ;MJD001
135236  122716  000021                  CMPB    #$XON,(SP)      ;;WAS IT XON?                                ;MJD001
135242  001366                          BNE     101$            ;;BR IF NOT                                  ;MJD001
135244                          102$:                                                                        ;MJD001
135244  005726                          TST     (SP)+           ;;FIX STACK                                  ;MJD001
135246                          10$:                                                                         ;MJD001
135246  105777  043676                  TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
135252  100375                          BPL     10$                                                          ;MJD001
135254  116677  000002  043670          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
135262  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
135270  001003                          BNE     1$              ;;BRANCH IF NO
135272  105037  135312                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
135276  000406                          BR      $TYPEX          ;;EXIT
135300  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
135306  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
135310  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
135312  000000                  $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
135314  000207                  $TYPEX: RTS     PC

                                .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

135316  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
135322  116637  000001  135541          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
135330  112637  135543                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
135334  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
135340  000406                          BR      $TYPON
135342  112737  000001  135541  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
135350  112737  000006  135543          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
```

```
22142  135356  112737  000005  135540  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
22143  135364  010346                  MOV     R3,-(SP)        ;;SAVE R3
22144  135366  010446                  MOV     R4,-(SP)        ;;SAVE R4
22145  135370  010546                  MOV     R5,-(SP)        ;;SAVE R5
22146  135372  113704  135543          MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
22147  135376  005404                  NEG     R4
22148  135400  062704  000006          ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
22149  135404  110437  135542          MOVB    R4,$OMODE       ;;SAVE IT FOR USE
22150  135410  113704  135541          MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
22151  135414  016605  000012          MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
22152  135420  005003                  CLR     R3              ;;CLEAR THE OUTPUT WORD
22153  135422  006105          1$:     ROL     R5              ;;ROTATE MSB INTO "C"
22154  135424  000404                  BR      3$              ;;GO DO MSB
22155  135426  006105          2$:     ROL     R5              ;;FORM THIS DIGIT
22156  135430  006105                  ROL     R5
22157  135432  006105                  ROL     R5
22158  135434  010503                  MOV     R5,R3
22159  135436  006103          3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
22160  135440  105337  135542          DECB    $OMODE          ;;TYPE THIS DIGIT?
22161  135444  100016                  BPL     7$              ;;BR IF NO
22162  135446  042703  177770          BIC     #177770,R3      ;;GET RID OF JUNK
22163  135452  001002                  BNE     4$              ;;TEST FOR 0
22164  135454  005704                  TST     R4              ;;SUPPRESS THIS 0?
22165  135456  001403                  BEQ     5$              ;;BR IF YES
22166  135460  005204          4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
22167  135462  052703  000060          BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
22168  135466  052703  000040          BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
22169  135472  110337  135536          MOVB    R3,8$           ;;SAVE FOR TYPING
22170  135476  104401  135536          TYPE    ,8$             ;;GO TYPE THIS DIGIT
22171  135502  105337  135540  7$:     DECB    $OCNT           ;;COUNT BY 1
22172  135506  003347                  BGT     2$              ;;BR IF MORE TO DO
22173  135510  002402                  BLT     6$              ;;BR IF DONE
22174  135512  005204                  INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
22175  135514  000744                  BR      2$              ;;GO DO THE LAST DIGIT
22176  135516  012605          6$:     MOV     (SP)+,R5        ;;RESTORE R5
22177  135520  012604                  MOV     (SP)+,R4        ;;RESTORE R4
22178  135522  012603                  MOV     (SP)+,R3        ;;RESTORE R3
22179  135524  016666  000002  000004  MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
22180  135532  012616                  MOV     (SP)+,(SP)
22181  135534  000002                  RTI                     ;;RETURN
22182  135536     000          8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
22183  135537     000                  .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
22184  135540     000          $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
22185  135541     000          $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
22186  135542  000000          $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
22187                          .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
22188
22189                          ;;**************************************************************
22190                          ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
22191                          ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
22192                          ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
22193                          ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
22194                          ;*REPLACED WITH SPACES.
22195                          ;*CALL:
22196                          ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
22197                          ;*      TYPDS                   ;;GO TO THE ROUTINE
```

```
22198
22199  135544                          $TYPDS:
22200  135544  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
22201  135546  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
22202  135550  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
22203  135552  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
22204  135554  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
22205  135556  012746  020200                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
22206  135562  016605  000020                  MOV     20(SP),R5       ;;GET THE INPUT NUMBER
22207  135566  100004                          BPL     1$              ;;BR IF INPUT IS POS.
22208  135570  005405                          NEG     R5              ;;MAKE THE BINARY NUMBER POS.
22209  135572  112766  000055  000001          MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
22210  135600  005000                  1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
22211  135602  012703  135760                  MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
22212  135606  112723  000040                  MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
22213  135612  005002                  2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
22214  135614  016001  135750                  MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
22215  135620  160105                  3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
22216  135622  002402                          BLT     4$              ;;BR IF DONE
22217  135624  005202                          INC     R2              ;;INCREASE THE BCD DIGIT BY 1
22218  135626  000774                          BR      3$
22219  135630  060105                  4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
22220  135632  005702                          TST     R2              ;;CHECK IF BCD DIGIT=0
22221  135634  001002                          BNE     5$              ;;FALL THROUGH IF 0
22222  135636  105716                          TSTB    (SP)            ;;STILL DOING LEADING 0'S?
22223  135640  100407                          BMI     7$              ;;BR IF YES
22224  135642  106316                  5$:     ASLB    (SP)            ;;MSD?
22225  135644  103003                          BCC     6$              ;;BR IF NO
22226  135646  116663  000001  177777          MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
22227  135654  052702  000060          6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
22228  135660  052702  000040          7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
22229  135664  110223                          MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
22230  135666  005720                          TST     (R0)+           ;;JUST INCREMENTING
22231  135670  020027  000010                  CMP     R0,#10          ;;CHECK THE TABLE INDEX
22232  135674  002746                          BLT     2$              ;;GO DO THE NEXT DIGIT
22233  135676  003002                          BGT     8$              ;;GO TO EXIT
22234  135700  010502                          MOV     R5,R2           ;;GET THE LSD
22235  135702  000764                          BR      6$              ;;GO CHANGE TO ASCII
22236  135704  105726                  8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
22237  135706  100003                          BPL     9$              ;;BR IF NO
22238  135710  116663  177777  177776          MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
22239  135716  105013                  9$:     CLRB    (R3)            ;;SET THE TERMINATOR
22240  135720  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
22241  135722  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
22242  135724  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
22243  135726  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
22244  135730  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
22245  135732  104401  135760                  TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
22246  135736  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
22247  135744  012616                          MOV     (SP)+,(SP)
22248  135746  000002                          RTI                     ;;RETURN TO USER
22249  135750  023420                  $DTBL:  10000.
22250  135752  001750                          1000.
22251  135754  000144                          100.
22252  135756  000012                          10.
22253  135760  000004                  $DBLK:  .BLKW   4
```

```
                                        .SBTTL  TTY INPUT ROUTINE
                                        ;;*****************************************************************
                                        .ENABL  LSB
                                        ;;*****************************************************************
                                        ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
                                        ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
                                        ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
                                        ;*WHEN OPERATING IN TTY FLAG MODE.
135770  022737  000176  001140  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
135776  001074                          BNE     15$             ;;BRANCH IF NO
136000  105777  043140                  TSTB    @$TKS           ;;CHAR THERE?
136004  100071                          BPL     15$             ;;IF NO, DON'T WAIT AROUND
136006  117746  043134                  MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
136012  042716  177600                  BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
136016  022726  000007                  CMP     #7,(SP)+        ;;IS IT A CONTROL G?
136022  001062                          BNE     15$             ;;NO, RETURN TO USER
136024  123727  001134  000001          CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
136032  001456                          BEQ     15$             ;;BRANCH IF YES
136034  104401  136525                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
136040  104401  136532          $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
136044  013746  000176                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
136050  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
136052  104401  136543                  TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
136056  005046                  19$:    CLR     -(SP)           ;;CLEAR COUNTER
136060  005046                          CLR     -(SP)           ;;THE NEW SWR
136062  105777  043056          7$:     TSTB    @$TKS           ;;CHAR THERE?
136066  100375                          BPL     7$              ;;IF NOT TRY AGAIN

136070  117746  043052                  MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
136074  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII


136100  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
136104  001005                          BNE     10$             ;;BRANCH IF NOT
136106  104401  136520                  TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
136112  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
136116  000757                          BR      19$             ;;LET'S TRY IT AGAIN


136120  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
136124  001022                          BNE     16$             ;;BRANCH IF NO
136126  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
136132  001403                          BEQ     11$             ;;BRANCH IF YES
136134  016677  000002  042776          MOV     2(SP),@SWR      ;;SAVE NEW SWR
136142  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
136146  104401  001175          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
136152  123727  001135  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
136160  001003                          BNE     15$             ;;BRANCH IF NOT
136162  012777  000100  042754          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
136170  000002                  15$:    RTI                     ;;RETURN
136172  004737  135174          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
136176  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
```

```
22310  136202  002420                          BLT     18$             ;;BRANCH IF YES
22311  136204  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
22312  136210  003015                          BGT     18$             ;;BRANCH IF YES
22313  136212  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
22314  136216  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
22315  136222  001403                          BEQ     17$             ;;BRANCH IF YES
22316  136224  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
22317  136226  006316                          ASL     (SP)            ;;   CHAR OVER TO MAKE
22318  136230  006316                          ASL     (SP)            ;;   ROOM FOR NEW ONE.
22319  136232  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
22320  136236  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
22321  136242  000707                          BR      7$              ;;GET THE NEXT ONE
22322  136244  104401  001174          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
22323  136250  000720                          BR      20$             ;;SIMULATE CONTROL-U
22324                                  .DSABL  LSB
22325
22326
22327                                  ;;*****************************************************************
22328                                  ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
22329                                  ;*CALL:
22330                                  ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
22331                                  ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
22332                                  ;*                              ;;WITH PARITY BIT STRIPPED OFF
22333                                  ;
22334
22335  136252  011646          $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
22336  136254  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
22337  136262  105777  042656          1$:     TSTB    @$TKS           ;;WAIT FOR
22338  136266  100375                          BPL     1$              ;;A CHARACTER
22339  136270  117766  042652  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
22340  136276  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
22341  136304  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
22342  136312  001013                          BNE     3$              ;;BRANCH IF NO
22343  136314  105777  042624          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
22344  136320  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
22345  136322  117746  042620                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
22346  136326  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
22347  136332  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
22348  136336  001366                          BNE     2$              ;;IF NOT DISCARD IT
22349  136340  000750                          BR      1$              ;;YES, RESUME
22350  136342  026627  000004  000021  3$:     CMP     4(SP),#$XON     ;;IS IT A RANDOM XON?           ;RAN001
22351  136350  001744                          BEQ     1$              ;;BRANCH IF YES                 ;RAN001
22352  136352  026627  000004  000140          CMP     4(SP),#140      ;;IS IT UPPER CASE?
22353  136360  002407                          BLT     4$              ;;BRANCH IF YES
22354  136362  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
22355  136370  003003                          BGT     4$              ;;BRANCH IF YES
22356  136372  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
22357  136400  000002                  4$:     RTI                     ;;GO BACK TO USER
22358                                  ;;*****************************************************************
22359                                  ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
22360                                  ;*CALL:
22361                                  ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
22362                                  ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
22363                                  ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
22364
22365  136402  010346          $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
```

```
136404  012703  136510          1$:     MOV     #$TTYIN,R3          ;;GET ADDRESS
136410  022703  136520          2$:     CMP     #$TTYIN+8.,R3       ;;BUFFER FULL?
136414  101405                          BLOS    4$                  ;;BR IF YES
136416  104410                          RDCHR                       ;;GO READ ONE CHARACTER FROM THE TTY
136420  112613                          MOVB    (SP)+,(R3)          ;;GET CHARACTER
136422  122713  000177          10$:    CMPB    #177,(R3)           ;;IS IT A RUBOUT
136426  001003                          BNE     3$                  ;;SKIP IF NOT
136430  104401  001174          4$:     TYPE    ,$QUES              ;;TYPE A '?'
136434  000763                          BR      1$                  ;;CLEAR THE BUFFER AND LOOP
136436  111337  136506          3$:     MOVB    (R3),9$             ;;ECHO THE CHARACTER
136442  104401  136506                  TYPE    ,9$
136446  122723  000015                  CMPB    #15,(R3)+           ;;CHECK FOR RETURN
136452  001356                          BNE     2$                  ;;LOOP IF NOT RETURN
136454  105063  177777                  CLRB    -1(R3)              ;;CLEAR RETURN (THE 15)
136460  104401  001176                  TYPE    ,$LF                ;;TYPE A LINE FEED
136464  012603                          MOV     (SP)+,R3            ;;RESTORE R3
136466  011646                          MOV     (SP),-(SP)          ;;ADJUST THE STACK AND PUT ADDRESS OF THE
136470  016666  000004  000002          MOV     4(SP),2(SP)         ;;      FIRST ASCII CHARACTER ON IT
136476  012766  136510  000004          MOV     #$TTYIN,4(SP)
136504  000002                          RTI                         ;;RETURN
136506     000                  9$:     .BYTE   0                   ;;STORAGE FOR ASCII CHAR. TO TYPE
136507     000                          .BYTE   0                   ;;TERMINATOR
136510  000010                  $TTYIN: .BLKB   8.                  ;;RESERVE 8 BYTES FOR TTY INPUT
136520  052536  005015     000  $CNTLU: .ASCIZ  /↑U/<15><12>        ;;CONTROL "U"
136525     136  006507  000012  $CNTLG: .ASCIZ  /↑G/<15><12>        ;;CONTROL "G"
136532  005015  053523  020122  $MSWR:  .ASCIZ  <15><12>/SWR = /
136540  020075     000
136543     040  047040  053505  $MNEW:  .ASCIZ  /  NEW = /
136550  036440  000040
                                .SBTTL  READ AN OCTAL NUMBER FROM THE TTY

                                ;;*****************************************************************
                                ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
                                ;*CHANGE IT TO BINARY.
                                ;*CALL:
                                ;*      RDOCT                       ;;READ AN OCTAL NUMBER
                                ;*      RETURN HERE                 ;;LOW ORDER BITS ARE ON TOP OF THE STACK
                                ;*                                  ;;HIGH ORDER BITS ARE IN $HIOCT

136554  011646                  $RDOCT: MOV     (SP),-(SP)          ;;PROVIDE SPACE FOR THE
136556  016666  000004  000002          MOV     4(SP),2(SP)         ;;INPUT NUMBER
136564  010046                          MOV     R0,-(SP)            ;;PUSH R0 ON STACK
136566  010146                          MOV     R1,-(SP)            ;;PUSH R1 ON STACK
136570  010246                          MOV     R2,-(SP)            ;;PUSH R2 ON STACK
136572  104411                  1$:     RDLIN                       ;;READ AN ASCIZ LINE
136574  012600                          MOV     (SP)+,R0            ;;GET ADDRESS OF 1ST CHARACTER
136576  005001                          CLR     R1                  ;;CLEAR DATA WORD
136600  005002                          CLR     R2
136602  112046                  2$:     MOVB    (R0)+,-(SP)         ;;PICKUP THIS CHARACTER
136604  001412                          BEQ     3$                  ;;IF ZERO GET OUT
136606  006301                          ASL     R1                  ;;*2
136610  006102                          ROL     R2
136612  006301                          ASL     R1                  ;;*4
136614  006102                          ROL     R2
136616  006301                          ASL     R1                  ;;*8
136620  006102                          ROL     R2
```

```
22422  136622  042716  177770                BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
22423  136626  062601                        ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
22424  136630  000764                        BR      2$              ;;LOOP
22425  136632  005726                3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
22426  136634  010166  000012              MOV     R1,12(SP)       ;;SAVE THE RESULT
22427  136640  010237  136654              MOV     R2,$HIOCT
22428  136644  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
22429  136646  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
22430  136650  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
22431  136652  000002                        RTI                     ;;RETURN
22432  136654  000000                $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
22433                                 .SBTTL  TRAP DECODER
22434
22435                                 ;;**************************************************************************
22436                                 ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
22437                                 ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
22438                                 ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
22439                                 ;*GO TO THAT ROUTINE.
22440
22441  136656  010046                $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
22442  136660  016600  000002                MOV     2(SP),R0        ;;GET TRAP ADDRESS
22443  136664  005740                        TST     -(R0)           ;;BACKUP BY 2
22444  136666  111000                        MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
22445  136670  006300                        ASL     R0              ;;POSITION FOR INDEXING
22446  136672  016000  136712              MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
22447  136676  000200                        RTS     R0              ;;GO TO ROUTINE
22448
22449
22450                                 ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
22451
22452  136700  011646                $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
22453  136702  016666  000004  000002        MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
22454  136710  000002                        RTI                     ;;RESTORE THE PSW
22455
22456                                 .SBTTL  TRAP TABLE
22457
22458                                 ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
22459                                 ;*BY THE "TRAP" INSTRUCTION.
22460
22461                                 ;       ROUTINE
22462                                 ;       -------
22463  136712  136700                $TRPAD: .WORD   $TRAP2
22464  136714  134762                        $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
22465  136716  135342                        $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
22466  136720  135316                        $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
22467  136722  135356                        $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
22468  136724  135544                        $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
22469
22470  136726  136040                        $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING
22471
22472  136730  135770                        $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
22473  136732  136252                        $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
22474  136734  136402                        $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
22475  136736  136554                        $RDOCT  ;;CALL=RDOCT    TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
22476                                 .SBTTL  POWER DOWN AND UP ROUTINES
22477
```

```
                                        ;;*****************************************************************
                                        ;POWER DOWN ROUTINE
136740  012737  137100  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
136746  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
136754  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
136756  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
136760  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
136762  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
136764  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
136766  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
136770  017746  042144                  MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
136774  010637  137104                  MOV     SP,$SAVR6       ;;SAVE SP
137000  012737  137012  000024          MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
137006  000000                          HALT
137010  000776                          BR      .-2             ;;HANG UP

                                        ;;*****************************************************************
                                        ;POWER UP ROUTINE
137012  012737  137100  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
137020  013706  137104                  MOV     $SAVR6,SP       ;;GET SP
137024  005037  137104                  CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
137030  005237  137104          1$:     INC     $SAVR6          ;;WAIT FOR THE INC
137034  001375                          BNE     1$              ;;OF  WORD
137036  012677  042076                  MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
137042  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
137044  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
137046  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
137050  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
137052  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
137054  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
137056  012737  136740  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
137064  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
137072  104401                          TYPE                    ;REPORT THE POWER FAILURE
137074  137106                  $PWRMG: .WORD   $POWER          ;;POWER FAIL MESSAGE POINTER
137076  000002                          RTI
137100  000000                  $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
137102  000776                          BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
137104  000000                  $SAVR6: 0                       ;;PUT THE SP HERE
137106  005015  047520  042527  $POWER: .ASCIZ  <15><12>"POWER"
137114  000122
                                        .EVEN
        000001                          .END
```

```
ABASE = 000000
ABOEXT  105774
ABORT   134354
ABORTC  134406
ABORTE  134452
ABORTI  104522
ABORTR  104556
ABORTZ  134476
ABORTO  043636
ABORT7  044024
ABROUT  105652
ACDW1 = 000000
ACDW2 = 000000
ACPUOP= 000000
ACTCHS  002704
AC0   =%000000
AC1   =%000001
AC2   =%000002
AC3   =%000003
AC4   =%000004
AC5   =%000005
AC6   =%000006
AC7   =%000007
ADDLSB  110510
ADDT    052020
ADDTRP  133302
ADDW0 = 000000
ADDW1 = 000000
ADDW10= 000000
ADDW11= 000000
ADDW12= 000000
ADDW13= 000000
ADDW14= 000000
ADDW15= 000000
ADDW2 = 000000
ADDW3 = 000000
ADDW4 = 000000
ADDW5 = 000000
ADDW6 = 000000
ADDW7 = 000000
ADDW8 = 000000
ADDW9 = 000000
ADEVCT= 000000
ADEVM = 000000
AENV  = 000000
AENVM = 000000
AFATAL= 000000
ALLCTR  003112
ALR0TS  021350
ALR1TS  021426
ALR2TS  021504
ALR3TS  021562
ALR4TS  021640
ALR5TS  021716
AMADR1= 000000
AMADR2= 000000
AMADR3= 000000
AMADR4= 000000
AMAMS1= 000000
AMAMS2= 000000
AMAMS3= 000000
AMAMS4= 000000
AMSGAD= 000000
AMSGLG= 000000
AMSGTY= 000000
AMTYP1= 000000
AMTYP2= 000000
AMTYP3= 000000
AMTYP4= 000000
APASS = 000000
APRIOR= 000000
APTCSU= 000040
APTENV= 000001
APTSIZ= 000200
APTSPO= 000100
ASWREG= 000000
ATESTN= 000000
AUNIT = 000000
AUSWR = 000000
AVECT1= 000000
AVECT2= 000000
A126    025066
BA      002650
BACDAT  102644
BA2     002670
BCR   = 177524
BCSR  = 177520
BDR   = 177524
BFA     052052
BFAC1   051640
BFAC2   051660
BFAC3   051700
BFAC4   051720
BFAC5   051742
BFAE    052112
BFB     052110
BGNTLP  110026
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BNO     043566
BN1     043756
BPTOA   030500
BPTOB   030506
BPTVEC= 000014
BTER    051202
BTEXP   003040
BTGO    051002
BTRES   003050
BTTST   051446
BTTSTE  051474
BYPARR  103134
CCR   = 177746
CCRTBL  105612
CHECK   015160
CHECK1  015236
CHECK2  015314
CHECK7  053222
CHEC10  053442
CHEC26  056224
CHEC27  056532
CHEC30  057042
CHEC32  057434
CHEC33  057710
CHEK7   053226
CHK10   053454
CHK7    053234
CH10    053446
CKSWR = 104407
CMPRTN  064232
COUNT   003060
CPEREG= 177766
CR    = 000015
CRLF  = 000200
CSR1    002644
CSR12   002664
CSR2    002646
CSR22   002666
CURADD  110524
CURDAT  110516
C121A   023102
C121B   023122
C121C   023142
DAPABO  104300
DATA    002654
DATA2   002674
DATBO   120452
DATI    121002
DATVER  133462
DATVFR  133444
DAT1    133474
DCOUNT  110500
DDISP = 177570
DH1     130360
DH105   131321
DH115   131353
DH24    130644
DH27    130710
DH4     130405
DH41    130753
DH43    131024
DH47    131124
DH5     130472
DH65    131211
DH7     130571
DH72    131255
DISPLA  001142
DISPRE  000174
DMAPAR  117670
DMARD   133034
DMATRN  132752
DPAREN  117730
DSWR  = 177570
DT1     131430
DT105   131706
DT115   131716
DT130   131732
DT14    131476
DT17    131510
DT24    131522
DT27    131532
DT35    131544
DT4     131436
DT41    131556
DT43    131566
DT47    131602
DT5     131450
DT50    131614
DT51    131626
DT52    131640
DT64    131652
DT65    131664
DT7     131464
DT75    131674
DVDSUB  066232
DVFSUB  065270
D1      052432
D2      052444
D3      052446
D4      052460
D5      052474
D6      052504
D7      052510
EMTOA   030304
EMTOB   030312
EMTSAV  003764
EMTVEC= 000030
EM1     122614
EM10    123066
EM100   126402
EM101   126440
EM102   126473
EM103   126555
EM104   126630
EM105   126700
EM106   126753
EM107   127031
EM11    123107
EM110   127066
EM111   127130
EM112   127206
EM113   127271
EM114   127355
EM115   127376
EM116   127446
EM117   127527
EM12    123131
EM120   127572
EM121   127640
EM122   127736
EM123   127764
EM124   130016
EM125   130110
EM126   130147
EM127   130204
EM13    123157
EM130   130246
EM131   130273
EM132   130323
EM14    123206
EM15    123231
EM16    123265
```

J2


```
EM17    123332          EM76    126300          FIN6    053054          KDPAR2= 172364          MAC3    051076
EM2     122650          EM77    126350          FIN7    053274          KDPAR3= 172366          MAC4    051112
EM20    123376          ENDHRT  102634          FLAG    002776          KDPAR4= 172370          MAC5    051126
EM21    123431          ENDLUP  110444          FLO     003022          KDPAR5= 172372          MADCC   015760
EM22    123474          ENDMOV  110526          FLOAT   003012          KDPAR6= 172374          MAIREG= 177750
EM23    123533          ENDTAG  111630          FMPARR  102306          KDPAR7= 172376          MALCC   016606
EM24    123607          ENDTLP  110414          FPVEC = 000244          KDPDR0= 172320          MARCC   016666
EM25    123652          ERR     133442          FRSTST  004504          KDPDR1= 172322          MASK    003120
EM26    123712          ERRFP   133440          FSTADD  110520          KDPDR2= 172324          MA11    010632
EM27    123761          ERROR = 104000          FWDSEQ  110506          KDPDR3= 172326          MA55    011536
EM3     122662          ERROUT  101700          GOODAD  002770          KDPDR4= 172330          MBB11   010730
EM30    124021          ERRVEC= 000004          GPROTS  004726          KDPDR5= 172332          MBB22   011212
EM31    124073          ERTYPE  131740          GPR1TS  005004          KDPDR6= 172334          MBCCC   015502
EM32    124120          EXBAD   110460          GPR2TS  005062          KDPDR7= 172336          MBC00   010434
EM33    124161          EXBAD2  111602          GPR3TS  005140          KIPAR0= 172340          MBC11   011010
EM34    124223          EXITST  110470          GPR4TS  005216          KIPAR1= 172342          MBC22   011300
EM35    124263          EXPBDT  105632          GPR5TS  005274          KIPAR2= 172344          MBI00   010354
EM36    124311          EXPDAT  110502          GPR6TS  005352          KIPAR3= 172346          MBPTC   030434
EM37    124355          EXPIR1  034164          G1SWR = 104406          KIPAR4= 172350          MBSCC   015554
EM4     122674          EXPTBL  102672          HITMIS= 177752          KIPAR5= 172352          MBT     032544
EM40    124417          EXPWDT  105622          HMPARR  102460          KIPAR6= 172354          MBTA    032566
EM41    124464          FINNOP  021070          HOP10   065502          KIPAR7= 172356          MBTB    032570
EM42    124550          FIN1    052174          HOP11   066444          KIPDR0= 172300          MBTCC   015432
EM43    124576          FIN10   053514          HOP12   067452          KIPDR1= 172302          MBTD    032600
EM44    124627          FIN11   053610          HOP13   070450          KIPDR2= 172304          MBTE    051476
EM45    124667          FIN116  022560          HOP14   071510          KIPDR3= 172306          MBTF    032610
EM46    124751          FIN117  022674          HOP15   073150          KIPDR4= 172310          MBTO    032622
EM47    125041          FIN120  023016          HOP16   073450          KIPDR5= 172312          MBTOA   032652
EM5     122734          FIN121  023414          HOP17   074102          KIPDR6= 172314          MBTOB   032654
EM50    125126          FIN122  023664          HOP18   076102          KIPDR7= 172316          MBTOC   032666
EM51    125151          FIN125  025026          HOP19   076504          KMCR  = 177734          MBTOD   032712
EM52    125203          FIN126  026066          HOP20   100074          LASTCH  112426          MBTOE   032714
EM53    125241          FIN127  026526          HOP21   101130          LCDSUB  076372          MBTOF   032726
EM54    125275          FIN13   053774          HOP22   101556          LCFSUB  075770          MBT1    050760
EM55    125333          FIN130  027416          HOP44   064350          LDPARS  132320          MBT2    050762
EM56    125367          FIN14   054124          HT    = 000011          LDPDRS  132350          MBT2A   050774
EM57    125413          FIN15   054312          ILAOA   030600          LEDS    116672          MBT8    051204
EM6     122767          FIN16   054456          ILBOB   030606          LF    = 000012          MBT8A   051234
EM60    125445          FIN17   054640          ILL     052674          LKS   = 177546          MBT8B   051256
EM61    125513          FIN2    052270          ILLBOA  030674          LKSFL   002702          MBT8C   051300
EM62    125542          FIN20   055012          ILLBOB  030702          LKSINT  132444          MBT8D   051322
EM63    125613          FIN21   055202          ILLOP1  052530          LOOP    004562          MBT8E   051346
EM64    125644          FIN22   055346          ILLOP2  052604          LOOPIN  003114          MBT8F   051372
EM65    125703          FIN23   055466          INITMM  132156          LOST    052160          MBT8FG  051406
EM66    125753          FIN24   055642          INQ22   121324          LOWADD  002766          MBT8I   051432
EM67    126012          FIN26   056322          INTERR  104730          LSTADD  110522          MB66    011612
EM7     123032          FIN27   056630          INTRPC  104706          LXPSUB  077654          MCB44   011456
EM70    126047          FIN30   057140          IOTOA   030206          MACCC   016034          MCLRD   074136
EM71    126100          FIN31   057260          IOTOB   030214          MACE    051142          MCLRI   074214
EM72    126134          FIN32   057532          IOTVEC= 000020          MACO    051032          MCMPD   064050
EM73    126172          FIN33   060006          IOXXX   033340          MACOA   051036          MCTSCC  015704
EM74    126216          FIN4    052610          KDPAR0= 172360          MAC1    051046          MDA0    010230
EM75    126247          FIN5    052700          KDPAR1= 172362          MAC2    051062          MDCCC   015622
```

K2


```
MDDSUB  072704
MDFSUB  071234
MDIVD   065502
MDIVF   064350
MDMO    010204
MDM27   010314
MDSO    010260
MEMK    117310
MEMQ    117340
MEMTO   030240
MET     032262
META    032320
METB    032330
METD    032340
METF    032350
METO    032370
METOA   032434
METOB   032444
METOC   032456
METOD   032502
METOE   032512
METOF   032524
MFA     051500
MFACU   051476
MFSRCM  060010
MIALL   033140
MIALLA  033164
MIALLB  033166
MIALLD  033176
MIALLF  033206
MIL     032740
MILA    032764
MILAO   030532
MILB    032766
MILD    032776
MILF    033006
MILLBO  030632
MILLO   030044
MILLOA  030110
MILLOB  030116
MILO    033020
MILOA   033052
MILOB   033054
MILOC   033066
MILOD   033112
MILOE   033114
MILOF   033126
MIOT    032066
MIOTA   032110
MIOTB   032112
MIOTD   032122
MIOTF   032132
MIOTO   030142
MITO    032144
MITOA   032174
MITOB   032176
MITOC   032210
MITOD   032234
MITOE   032236
MITOF   032250
MJ      012542
MJP     013374
MJP17   013404
MJP27   013426
MJP27A  013440
MJP37   013512
MJP37A  013524
MJP67   013450
MJP67A  013462
MJP67B  013476
MJP77   013500
MJP77E  013534
MJRA    014312
MJR27   014346
MJR27A  014402
MJR27B  014422
MJR37   014504
MJR37A  014540
MJR6A   014500
MJR6B   014502
MJR67   014424
MJR67A  014460
MJR77   014560
MJR77A  014614
MJSI    033220
MJSIA   033252
MJSIB   033254
MJSIC   033266
MJSID   033312
MJSIE   033314
MJSIF   033326
MJSR    013534
MJSRA   013652
MJSRB   014016
MJSRC   014164
MJSR1   013654
MJSR1A  013666
MJSR1B  013712
MJSR2   013572
MJSR2A  013604
MJSR2B  013630
MJSR3   014020
MJSR3A  014032
MJSR3B  014056
MJSR4   013734
MJSR4A  013746
MJSR4B  013772
MJSR5   014166
MJSR5A  014200
MJSR5B  014224
MJSR6   014102
MJSR6A  014114
MJSR6B  014140
MJSR7   014250
MJSR7A  014262
MJSR7E  014306
MJU1    012614
MJU1A   012626
MJU2    012556
MJU2A   012570
MJU2B   012600
MJU3    012640
MJU3A   012652
MJU3B   012662
MJU4    012734
MJU4A   012746
MJU4B   012760
MJU5    012674
MJU5A   012706
MJU5B   012720
MJU6    012774
MJU6A   013006
MJU7    013026
MJU7E   013040
MJ2     012612
MJ5     012772
MJ7     013022
MLCD    076102
MLCF    075264
MLDC    063502
MLDC2   075164
MLDDM2  060060
MLDDM3  060146
MLDDM4  060262
MLDDM5  060366
MLDDM6  060456
MLDDM7  060542
MLDM27  060630
MLDSUB  070236
MLFSUB  067240
MLS1    074250
MLS2    074342
MLS3    074434
MLS4    074544
MLS5    074640
MLS6    074750
MLS7    075050
MLXP    076504
MMARK   017372
MMODD   071510
MMODF   070450
MMRL5   012274
MMR0  = 177572
MMR1  = 177574
MMR2  = 177576
MMR3  = 172516
MMU     133134
MMULD   067452
MMULF   066444
MMUTRP  133306
MMVCC   015366
MMVEC = 000250
MM11    010522
MM22    011070
MNCCCC  016116
MNGOP   061746
MNNRM1  060722
MNNRM2  061142
MNNRM3  061274
MNNRM4  061416
MNRM    062600
MODE1   045654
MODE2   047234
MODGAR  071500
MRLB1   012022
MRLCC   016370
MRL0    011750
MRL2    012102
MRL3    012160
MRL4    012234
MRL6    012336
MRL7    012374
MRRB1   012514
MRRCC   016450
MRR0    012472
MRT     031344
MRTA    031366
MRTB    031370
MRTE    031400
MRTF    031410
MRTO    031422
MRTOA   031452
MRTOB   031454
MRTOC   031466
MRTOD   031512
MRTOE   031514
MRTOF   031526
MRTS    014620
MSB     062456
MSBCC   016234
MSBCCC  016304
MSCD    100074
MSCF    101130
MSDF    073450
MSER  = 177744
MSFD    073150
MSFDI   074102
MSOB    017316
MSPAA   007462
MSPAU   027416
MSPB    005554
MSPBB   007546
MSPC    005600
MSPD    005630
MSPEO   005666
MSPF    005730
MSPG    005772
MSPH    006030
MSPI    006074
MSPJ    006134
MSPL    006254
MSPM    006322
MSPN    006400
MSPO    006446
MSPP    006552
MSPQ    006650
MSPR    006756
MSPS    007044
MSPT    007112
MSPU    007130
MSPV    007230
MSPVO   007174
MSPX    007274
MSPY    007344
MSPZ    007412
MSP0    005532
MSTB3   007616
MSTO    030722
MSTOE   030776
MSTOEE  031004
MST4    007670
MST4B   007726
MST5    007766
MST5B   010020
MST6    010062
MST7    010130
MSW37   012436
MSXP    101272
MSXT    017120
MSXTCC  016756
MS11    010670
MS22    011142
MS33    011372
MS77    011662
MTP     031540
```

L2


```
MTPA    031742          NULL   = 000000        PR2    = 000100        SDPAR6= 172274        SWREG   000176
MTPAA   032016          NXMFIN  047456        PR3    = 000140        SDPAR7= 172276        SW0    = 000001
MTPAE   032046          NXMPAR  106022        PR4    = 000200        SDPDR0= 172220        SW00   = 000001
MTPAH   032014          NXMTRP  047064        PR5    = 000240        SDPDR1= 172222        SW01   = 000002
MTPAL   032004          ODDXX   033440        PR6    = 000300        SDPDR2= 172224        SW02   = 000004
MTPB    031572          OK      133342        PR7    = 000340        SDPDR3= 172226        SW03   = 000010
MTPF    031612          OKAY7    042676       PS     = 177776        SDPDR4= 172230        SW04   = 000020
MTPO    031624          OKAY7A   042706       PSW    = 177776        SDPDR5= 172232        SW05   = 000040
MTPOA   031654          OKA7     042664       PSWBTS  005434        SDPDR6= 172234        SW06   = 000100
MTPOB   031656          OK1      133370       PWRVEC= 000024        SDPDR7= 172236        SW07   = 000200
MTPOC   031670          OK7      042646       Q22EN   002752        SEQ     003032        SW08   = 000400
MTPOD   031714          ONOQ22   132676       Q22INT  132732        SFDSUB  073344        SW09   = 001000
MTPOE   031716          PAR      133256       Q22SIZ  132452        SIMGOA  002662        SW1    = 000002
MTPOF   031730          PARAD1   045504       RAMPAR  132374        SIPAR0= 172240        SW10   = 002000
MTPQ    031602          PARAD2   047116       RBUF   = 177562        SIPAR1= 172242        SW11   = 004000
MTPR    031570          PARVA1   045536       RCSR   = 177560        SIPAR2= 172244        SW12   = 010000
MTRPO   030336          PARVA2   047150       RDCHR  = 104410        SIPAR3= 172246        SW13   = 020000
MTRY    027560          PARVA3   047266       RDLIN  = 104411        SIPAR4= 172250        SW14   = 040000
MTRYA   027634          PAR1     133260       RDOCT  = 104412        SIPAR5= 172252        SW15   = 100000
MTRYB   027654          PCR    = 177522       RECDAT  110504        SIPAR6= 172254        SW2    = 000004
MTRYM   027706          PDR      133234       RECDST  003102        SIPAR7= 172256        SW3    = 000010
MTSO    015366          PDR1     133236       RECFEC  003062        SIPDR0= 172200        SW4    = 000020
MTT     031016          PHY1     045622       RECST   003072        SIPDR1= 172202        SW5    = 000040
MTTA    031052          PIR    = 177772       RESVEC= 000010        SIPDR2= 172204        SW6    = 000100
MTTB    031054          PIRQ   = 177772       RET1    015104        SIPDR3= 172206        SW7    = 000200
MTTD    031064          PIRQNX   034042       RET2    015172        SIPDR4= 172210        SW8    = 000400
MTTE    031074          PIRQT    121614       RET3    015250        SIPDR5= 172212        SW9    = 001000
MTTR    031200          PIRQVE= 000240        RITEDA  110512        SIPDR6= 172214        SXPSUB  101452
MTTRA   031244          PIRRTN   034342       RTSE    014672        SIPDR7= 172216        TAB1    003174
MTTRB   031246          PIRTBL   034144       RTS1    014640        SIXBIT  111774        TAB10   003324
MTTRC   031260          PIRTEX   034044       RTS6    014650        SLEND   116510        TAB11   003334
MTTRD   031316          PIRXXX   034006       RXXX    033660        SLOC00  002762        TAB11A  003344
MTTRE   031320          PIR1     034044       R6     =%000006       SLOC01  002764        TAB12   003354
MTTRF   031332          PIR2     034164       R7     =%000007       SPAU1   027434        TAB13   003364
MTTS    031106          PIR2EX   034272       SAVBR   002710        SPAU2   027456        TAB13B  003374
MTTSA   031142          PIR3     034272       SAVMR0  003004        SPAU3   027476        TAB14   003404
MTTSB   031146          PIR3EX   034366       SAVMR1  003006        SPAU4   027522        TAB15   003414
MTTSD   031156          PIR4     034366       SAVMR2  003010        SPAU5   027536        TAB16   003424
MTTSE   031166          PIR5     034444       SAVPCR  002706        SPAU6   027554        TAB17   003434
MTTSQ   031144          PIR5EX   034654       SAVPOS  003116        SPS     003034        TAB18   003444
MUVAD   062710          PIR6     034654       SAVSUP  003000        SPSJ    003036        TAB2    003204
MXDF1   061526          PIR6EX   034756       SAVUSE  003002        SR0    = 177572       TAB21   003454
MXOR    017236          PITBL1   034250       SAV30   004052        SR1    = 177574       TAB22   003464
MXRCC   017026          PITBL2   034640       SAV32   004054        SR2    = 177576       TAB23   003474
M2      004610          PI1      034314       SCDSUB  100714        SR3    = 172516       TAB24   003504
M3      004624          PI2      034326       SCOPE  = 000004        STACK  = 001100       TAB25   003514
M4      004644          PI3      034330       SDFSUB  073700        START   003764        TAB26   003524
M5      004664          PLF0     043514       SDPAR0= 172260        STBOT  = 001000       TAB27   003534
M6      004704          PLF1     043706       SDPAR1= 172262        STKLMT= 177774        TAB28   003544
NEWADD  002774          POLY   = 120001       SDPAR2= 172264        STMOVI  107662        TAB29   003554
NEWDAT  110514          PROCNT   022744       SDPAR3= 172266        STMOVT  110764        TAB29A  003564
NOABRT  134512          PR0    = 000000       SDPAR4= 172270        SUBT    051766        TAB3    003214
NOTOK   133364          PR1    = 000040       SDPAR5= 172272        SWR     001140        TAB30   003574
```

M2


```
TAB31   003604
TAB32   003614
TAB33   003624
TAB34   003634
TAB4    003224
TAB40   003644
TAB41   003654
TAB42   003664
TAB43   003674
TAB45   003704
TAB46   003714
TAB47   003724
TAB47A  003734
TAB48   003744
TAB49   003754
TAB5    003234
TAB5A   003244
TAB6    003254
TAB6A   003264
TAB7    003274
TAB8    003304
TAB9    003314
TAPAB0  104426
TA114   021006
TA116   022462
TBITVE= 000014
TB114   021016
TC114   021026
TD114   021036
TEMP    002710
TE102   017576
TE103   017620
TE104   017642
TE105   017664
TE106   017706
TE107   017730
TE110   017752
TE111   017774
TE112   020016
TE113   020240
TE113A  020460
TE114   020632
TE115   022002
TE115A  022212
TE115B  022222
TE115C  022230
TE115D  022236
TE115F  022244
TE116   022246
TE116A  022522
TE116B  022526
TE116C  022540
TE116D  022550
TE117   022560
TE117A  022672
TE120   022676
TE120A  022752
TE121   023026
TE122   023414
TE123   023666
TE124   024106
TE125   024340
TE125A  024600
TE126   025026
TE126A  025074
TE126B  025550
TE127   026066
TE127A  026266
TE130   026530
TE130A  027006
TF114   021046
TG114   021056
THRBIT  112252
TH114   021066
TIMDEL  114452
TIMEOU  052600
TIMOUT  002750
TKVEC = 000060
TMM16A  050354
TMM16B  050234
TMM16C  050264
TMM16D  050314
TMM16E  050344
TMM16F  050346
TM16A   050740
TOUT    133124
TPVEC = 000064
TRAPVE= 000034
TRPFLG  133436
TRP0A   030402
TRP0B   030410
TRTVEC= 000014
TRYMA   027746
TRYMB   030000
TRYMC   030026
TSD31   057166
TSEND   111612
TSFP1   052112
TSFP10  053274
TSFP11  053514
TSFP12  053610
TSFP13  053650
TSFP14  053774
TSFP15  054126
TSFP16  054314
TSFP17  054460
TSFP2   052176
TSFP20  054642
TSFP21  055014
TSFP22  055204
TSFP23  055350
TSFP24  055470
TSFP25  055644
TSFP26  055762
TSFP27  056324
TSFP3   052270
TSFP30  056632
TSFP31  057142
TSFP32  057262
TSFP33  057534
TSFP4   052364
TSFP5   052610
TSFP6   052700
TSFP7   053054
TSF10   053460
TSF13   053770
TSF14   054120
TSF15   054306
TSF16   054452
TSF17   054634
TSF2    052244
TSF20   055006
TSF21   055176
TSF22   055342
TSF23   055462
TSF24   055636
TSF31   057254
TSF6    053050
TSF7    053240
TSLOOP  111072
TSMA    042736
TSMB    042756
TSMC    042766
TSMMU0  014674
TSMMU1  034772
TSMMU2  035056
TSMMU3  036256
TSMMU4  036556
TSMMU5  037506
TSMMU6  037640
TSMMU7  042344
TSMMU8  042772
TSMMU9  043174
TSMM10  044074
TSMM11  044456
TSMM12  044714
TSMM13  045302
TSMM14  046350
TSMM15  047474
TSMM16  050014
TSMM6A  037746
TSMM6B  040404
TSMM6C  041144
TSMM6D  041644
TSM16A  050110
TSM16B  050136
TSM16C  050162
TSM16D  050226
TSM7    042724
TSM9    043330
TSTADD  002772
TSTLOC  003122
TSTLUP  107744
TST1    004562
TST10   103146
TST11   103372
TST12   103630
TST13   104012
TST14   104310
TST15   104436
TST16   104632
TST17   105000
TST2    101556
TST20   105172
TST21   105654
TST22   106032
TST23   106230
TST24   106550
TST25   107306
TST26   107432
TST27   110526
TST3    101704
TST30   111630
TST31   112002
TST32   112260
TST33   112564
TST34   112716
TST35   113066
TST36   113352
TST37   113520
TST4    101752
TST40   114014
TST41   114172
TST42   114462
TST43   114522
TST44   114620
TST45   114702
TST46   115116
TST47   115160
TST5    102320
TST50   115534
TST51   116002
TST52   116272
TST53   116510
TST54   117016
TST55   117372
TST56   117640
TST57   117760
TST6    102466
TST60   120142
TST61   121162
TST62   121440
TST63   121624
TST64   121724
TST65   122106
TST66   122606
TST7    102710
TS10    044374
TS10D1  053464
TS10D2  053474
TS10D4  053504
TS11    044574
TS11D1  053600
TS12    045232
TS14    046754
TS15    047640
TS16    050644
TS16A   050636
TS1822  045706
TS26D0  056242
TS26D1  056252
TS26D2  056262
TS26D3  056272
TS26D4  056302
TS26D5  056312
TS27D0  056550
TS27D1  056560
TS27D2  056570
TS27D3  056600
TS27D4  056610
TS27D5  056620
TS30D0  057060
TS30D1  057070
TS30D2  057100
TS30D3  057110
TS30D4  057120
TS30D5  057130
TS32D0  057452
TS32D1  057462
TS32D3  057502
TS32D4  057512
TS32D5  057522
TS33D0  057726
TS33D1  057736
TS33D2  057746
```

N2


```
TS33D3  057756
TS33D4  057766
TS33D5  057776
TS6DA   053030
TS6DAT  053040
TS7     042614
TS7DA1  053244
TS7DA2  053254
TS7DA4  053264
TS7FIN  042770
TS9FIN  044072
TYPDS   = 104405
TYPE    = 104401
TYPOC   = 104402
TYPON   = 104404
TYPOS   = 104403
T10FIN  044456
T11FIN  044714
T114    020764
T116    022430
T12FIN  045300
T122A   023512
T122B   023644
T123A   024050
T123B   024060
T123C   024066
T123D   024074
T123E   024076
T123F   024102
T124A   024502
T124B   024512
T124C   024520
T124D   024526
T124E   024530
T124F   024534
T13FIN  045706
T14     047000
T14FIN  047552
T15     047710
T15A    047766
T15FIN  050014
UDPAR0= 177660
UDPAR1= 177662
UDPAR2= 177664
UDPAR3= 177666
UDPAR4= 177670
UDPAR5= 177672
UDPAR6= 177674
UDPAR7= 177676
UDPDR0= 177620
UDPDR1= 177622
UDPDR2= 177624
UDPDR3= 177626
UDPDR4= 177630
UDPDR5= 177632
UDPDR6= 177634
UDPDR7= 177636
UFDFLG  004056
UFDSET= 000001
UIPAR0= 177640
UIPAR1= 177642
UIPAR2= 177644
UIPAR3= 177646
UIPAR4= 177650
UIPAR5= 177652
UIPAR6= 177654
UIPAR7= 177656
UIPDR0= 177600
UIPDR1= 177602
UIPDR2= 177604
UIPDR3= 177606
UIPDR4= 177610
UIPDR5= 177612
UIPDR6= 177614
UIPDR7= 177616
UNXPIR  034270
UQUIET  004060
VIR1    045570
VIR2    047202
VIR3    047320
VMXOR   004062
VQBE1   002656
VQBE2   002676
VQPR1   002660
VQPR2   002700
WC      002652
WC2     002672
WLDTRP  133426
XBUF    = 177566
XCBIT   016526
XCSR    = 177564
XME100  017532
XME101  017554
$APTHD  000232
$ATYC   134540
$ATY1   134514
$ATY3   134522
$ATY4   134532
$AUTOB  001134
$BASE   00'254
$BDADR  00'122
$BUDAT  00 126
$BELL   001170
$CDW1   001260
$CDW2   001262
$CHARC  135312
$CKSWR  135770
$CMTAG  001100
$CM3    = 000000
$CM4    = 000002
$CNTLG  136525
$CNTLU  136520
$CPUOP  001226
$CRLF   001175
$DBLK   135760
$DDW0   001264
$DDW1   001266
$DDW10  001310
$DDW11  001312
$DDW12  001314
$DDW13  001316
$DDW14  001320
$DDW15  001322
$DDW2   001270
$DDW3   001272
$DDW4   001274
$DDW5   001276
$DDW6   001300
$DDW7   001302
$DDW8   001304
$DDW9   001306
$DEVCT  001210
$DEVM   001256
$DOAGN  133620
$DTBL   135750
$ENDAD  133610
$ENDCT  133556
$ENDMG  133627
$ENULL  133624
$ENV    001220
$ENVM   001221
$EOP    133514
$EOPCT  133550
$ERFLG  001103
$ERMAX  001115
$ERROR  134132
$ERRPC  001116
$ERRTB  001324
$ERTTL  001112
$ESCAP  001166
$ETABL  001220
$ETEND  001324
$FATAL  001202
$FFLG   134760
$FILLC  001156
$FILLS  001155
$GDADR  001120
$GDDAT  001124
$GET42  133600
$GTSWR  136040
$HD     = 000001
$HIBTS  000232
$HIOCT  136654
$ICNT   001104
$ILLUP  137100
$INTAG  001135
$ITEMB  001114
$LF     001176
$LFLG   134757
$LPADR  001106
$LPERR  001110
$MADR1  001232
$MADR2  001236
$MADR3  001242
$MADR4  001246
$MAIL   001200
$MAMS1  001230
$MAMS2  001234
$MAMS3  001240
$MAMS4  001244
$MBADR  000234
$MFLG   134756
$MNEW   136543
$MSGAD  001214
$MSGLG  001216
$MSGTY  001200
$MSWR   136532
$MTYP1  001231
$MTYP2  001235
$MTYP3  001241
$MTYP4  001245
$MXCNT  134130
$NULL   001154
$NWTST= 000000
$OCNT   135540
$OMODE  135542
$OVER   134114
$PASS   001206
$PASTM  000240
$POWER  137106
$PWRDN  136740
$PWRMG  137074
$PWRUP  137012
$QUES   001174
$RDCHR  136252
$RDLIN  136402
$RDOCT  136554
$RDSZ   = 000010
$RTNAD  133622
$SAVR6  137104
$SCOPE  133644
$SETUP= 000137
$STUP   = 177777
$SVLAD  134060
$SVPC   = 000232
$SWR    = 167400
$SWREG  001222
$SWRMK= 000300
$TESTN  001204
$TIMES  001164
$TKB    001146
$TKS    001144
$TMP0   001160
$TMP1   001162
$TN     = 000067
$TPB    001152
$TPFLG  001157
$TPS    001150
$TRAP   136656
$TRAP2  136700
$TRP    = 000013
$TRPAD  136712
$TSTM   000236
$TSTNM  001102
$TTYIN  136510
$TYPDS  135544
$TYPE   134762
$TYPEC  135174
$TYPEX  135314
$TYPOC  135342
$TYPON  135356
$TYPOS  135316
$UNIT   001212
$UNITM  000242
$USWR   001224
$VECT1  001250
$VECT2  001252
$XOFF   = 000023
$XON    = 000021
$XTSTR  133656
$$GET4= 000000
$OFILL  135541
.       = 137116
.$X     = 000232

. ABS.  137116      000

ERRORS DETECTED:  0
```

```
COKDAB,COKDAB.PRT/SOL/NL:TOC=SYSMAC.SML,COKDAB.P11
RUN-TIME:  101 53 1 SECONDS
RUN-TIME RATIO: 214/156=1.3
CORE USED:  43K  (86 PAGES)
```