11/45/55

DIAGNOSTIC TEST 2

MD-11-DEFPB-A

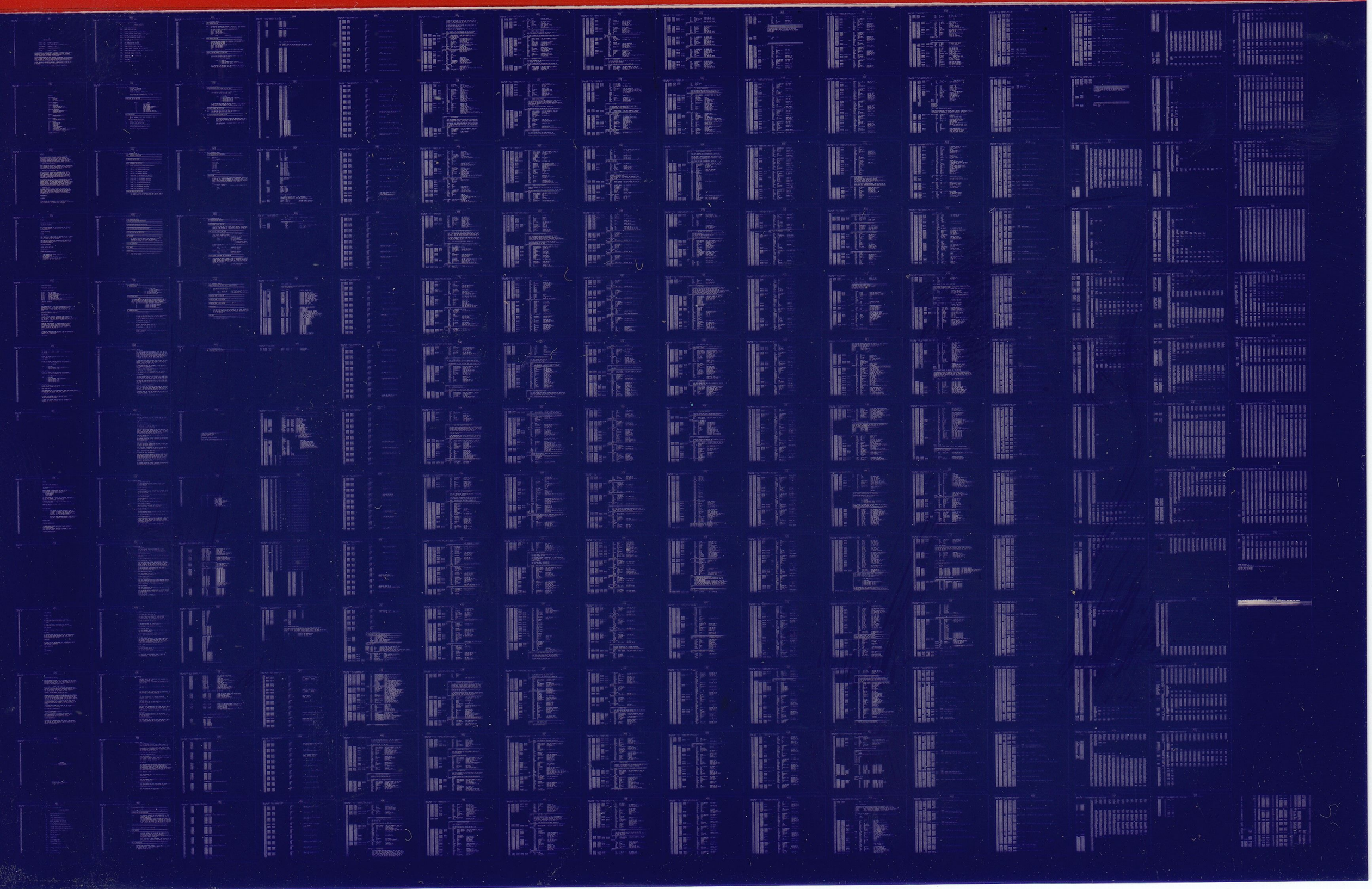
EP-DEFPB-A-DL-A

FICHE 1 OF 1

NOV 1976

digital

MADE IN USA



```
                                .REM    %
```

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DEFPB-A-D

PRODUCT NAME: PDP11-45/55/70 FP11C DIAGNOSTIC PART 2

DATE CREATED: FEBRUARY 21,1976

MAINTAINER: DIAGNOSTIC ENGINEERING

AUTHOR: DONALD W. MONROE

CONTENTS

1. ABSTRACT

DEFPA/B ARE PROGRAMS DESIGNED TO DETECT AND REPORT LOGIC FAULTS IN THE FP11-C FLOATING POINT PROCESSOR (FPP). THEY CONSIST OF 237(8) INDIVIDUAL TESTS DESIGNED AND SEQUENCED TO DETECT AND ATTEMPT TO IDENTIFY LOGIC FAULTS AT A MINIMUM HARDWARE/SOFTWARE LEVEL. THESE TESTS ARE PARTITIONED INTO TWO STAND-ALONE PROGRAMS AS DESCRIBED BELOW.

A. BASIC INSTRUCTION TESTS

DEFPA CONSISTS OF A LOGICALLY SEQUENCED SET OF INSTRUCTION TESTS DESIGNED TO VERIFY THE INTEGRITY OF THOSE INSTRUCTIONS, LOGIC OPERATIONS, AND DATA PATHS USED BY THE MORE COMPLEX INSTRUCTIONS SUCH AS MULTIPLY AND DIVIDE.

B. MULTIPLY-DIVIDE-AND ROM TESTS

DEFPB CONSISTS OF A LOGICALLY SEQUENCED SET OF TESTS FOR THE MULTIPLY, MODULO, DIVIDE INSTRUCTIONS, AND THE MEMORY MANAGEMENT ROMS FOLLOWED BY A TEST OF ALL THE LOCATIONS OF THE A-BRANCH, NO-MEM BRANCH, AND ADX ROMS THAT HAVE NOT PREVIOUSLY BEEN TESTED. THIS LAST TEST IS ALSO USED TO VERIFY THE "DISABLE INTERRUPT" BIT IN THE CONTROL STORE OF THE FPP.

UPON DETECTION OF A LOGIC FAULT AN "ERROR SERVICE" ROUTINE IS CALLED THAT REPORTS IT AS HARD COPY ON THE CONSOLE TERMINAL DEVICE. THE ERROR SERVICE ROUTINE ALSO FACILITATES OPERATOR CONTROL OF THE PROGRAM SEQUENCE VIA CONSOLE SWITCH REGISTER OPTIONS. AFTER REPORTING THE ERROR, THE PROGRAM CONTINUES ON ITS NORMAL SEQUENCE UNLESS MODIFIED BY THE OPERATOR ACTIVATING A SWITCH, SUCH AS "HALT ON ERROR".

C. IMPORTANT NOTE

THE ERROR REPORTS IN THESE PROGRAMS ARE BASED UPON THE KNOWLEDGE THAT ALL PREVIOUS TESTS ARE FAULTLESS AND THAT THERE IS ONLY ONE SINGLE POINT FAILURE IN THE PROCESSOR. THIS MEANS THAT IF EITHER PROGRAM, OR THE PROGRAMS THEMSELVES, ARE NOT RUN IN SEQUENCE, THE ERROR MESSAGE MAY NOT BE VALID.

2. REQUIREMENTS

2.1 EQUIPMENT

PDP 11/45,55, OR 70 PROCESSOR WITH OPERATORS CONSOLE, LA30 OR EQUIVALENT TERMINAL, A KW11-L OR KW11-P OR EQUIVALENT CLOCK, AND AN FP11-C FLOATING POINT PROCESSOR.

2.2 STORAGE

BOTH DEFPA AND DEFPB REQUIRE 16K OF MEMORY TO LOAD AND RUN.

2.3 PRELIMINARY PROGRAMS

BOTH PROGRAMS REQUIRE THE CPU CLUSTER AND 16K OF MEMORY TO BE WORKING PROPERLY. DEFPB REQUIRES THAT DEFPA HAS RUN SUCCESSFULLY.

3. LOADING PROCEDURE

3.1 METHOD

FOR A PDP 11/45 OR 55 SYSTEM THE PROGRAMS WILL BE ON PAPER TAPE. DEFPA WILL CONSIST OF TWO TAPES. BOTH THESE TAPES MUST BE LOADED TO RUN THE PROGRAM.

FOR A PDP 11/70 SYSTEM THE PROGRAMS WILL BE SUPPLIED ON THE DIAGNOSTIC MEDIA. REFER TO THE XXDP OPERATING MANUAL FOR FURTHER INFORMATION.

4. STARTING PROCEDURE

4.1 CONTROL SWITCH SETTINGS

SEE SECTION 5.1.

4.2 PROGRAM AND OPERATOR ACTION

1. LOAD PROGRAM INTO MEMORY (SEE SECTION 3)
2. LOAD ADDRESS 200.
3. SET SWITCHES (SEE SECTION 5.1)
4. PRESS START
5. THE PROGRAM(S) WILL LOOP AND AN END OF PASS MESSAGE WILL BE TYPED EVERY PASS.

5. OPERATING PROCEDURE

5.1 OPPERATIONAL SWITCH SETTINGS

THE SWITCH SETTINGS ARE:

SW<15>=1 ... HALT ON ERROR
SW<14>=1 ... LOOP ON CURRENT TEST
SW<13>=1 ... INHIBIT ERROR TYPE OUTS
SW<12>=1 ... SKIP MEMORY MANAGEMENT TESTS (DEFPB ONLY)
SW<11>=1 ... INHIBIT ITERATIONS
SW<10>=1 ... RING BELL ON ERROR
SW<9>=1 ... LOOP ON ERROR (ERROR COUNT IN DISPLAY REGISTER)
SW<8>=1 ... LOOP ON TEST IN SW'S<7:0>
SW<8>=0 ... LOAD MICRO-BREAK REGISTER WITH SW'S<7:0>

5.2 SUBROUTINE ABSTRACTS

5.2.1 SCOPE

THIS SUBROUTINE CALL (VIA AN IOT INSTRUCTION) IS PLACED BETWEEN EACH TEST. IT RECORDS THE STARTING ADDRESS OF EACH TEST IN LOCATION "$LPADR" AND "$LPERR" AS IT IS BEING ENTERED. IT ALSO CONTROLS TEST ITTERATION, AND LOOPING.

5.2.2 ERROR

THIS SUBROUTINE CALL (VIA A EMT INSTRUCTION) IS USED TO REPORT ALL ERRORS. (REFER TO 6)

5.2.3 TRAP CATCHER

A ".+2" - "HALT" SEQUENCE IS REPEATED FROM LOCATION 0 TO LOCATION 776 TO CATCH ANY UNEXPECTED TRAPS (EXCEPT 4, 24 114, AND 244). THUS, ANY UNEXPECTED TRAPS WILL HALT AT THE DEVICE TRAP VECTOR +2. (TRAP VECTOR+4 IN ADDRESS DISPLAY LIGHTS)

5.2.4 SPURIOUS TRAP CATCHER

THERE ARE THREE ROUTINES TO HANDLE UNEXPECTED TRAPS TO LOCATIONS 4 (CPU ERROR), 114 (PARITY ERROR), AND 244 (FPP INTERRUPT). THE ROUTINES PRINT A SHORT MESSAGE FOLLOWED BY THE PROGRAM COUNTER (PC) OF THE ERROR CALL (ERRPC), THE PC AT THE TIME OF THE TRAP (PCOFTRP), AND THE APPROPRIATE ERROR REGISTER.

5.2.5 TRAP

A NUMBER OF SUBROUTINES ARE CALLED BY THE TRAP INSTRUCTION. FOLLOWING ARE THE CALLS USED AND THE LABEL OF THE STARTING ADDRESS OF THE SUBROUTINES.

5.2.5.1 TYPE ($TYPE)

ROUTINE TO TYPE AN ASCII STRING ON THE TTY.

THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER
A LINE FEED.

5.2.5.2 TYPOC ($TYPOC)

ROUTINE TO CONVERT A 16-BIT BINARY NUMBER TO A 6-DIGIT
OCTAL NUMBER AND TYPE IT.

5.2.5.3 FL20 ($FL20)

ROUTINE TO CONVERT A 32-BIT FLOATING POINT NUMBER TO
A 13-DIGIT OCTAL ASCIZ NUMBER IN THE FOLLOWING FORMAT:

W XXX YYY ZZZZZZ

WHERE W = SIGN BIT
X = 8-BIT EXPONENT (RIGHT JUSTIFIED)
Y = FRACTION BITS <57:51> (RIGHT JUSTIFIED)
Z = FRACTION BITS <50:35>

5.2.5.4 FLD20 ($FLD20)

ROUTINE TO CONVERT A 64-BIT FLOATING POINT NUMBER TO
A 26-DIGIT OCTAL ASCIZ NUMBER IN THE FOLLOWING FORMAT:

U VVV WWW XXXXXX YYYYYY ZZZZZZ

WHERE U = SIGN BIT
W = 8-BIT EXPONENT (RIGHT JUSTIFIED)
W = FRACTION BITS <57:51> (RIGHT JUSTIFIED)
X = FRACTION BITS <50:35>
Y = FRACTION BITS <34:19>
Z = FRACTION BITS <18:03>

5.2.5.5 TYPDS ($TYPDS)

ROUTINE TO CONVERT A 16-BIT BINARY NUMBER TO A 5-DIGIT
SIGNED DECIMAL NUMBER AND TYPE IT.

5.2.6 POWER DOWN AND UP

THIS SUBROUTINE CALL (VIA A POWER DOWN) SAVES THE GENERAL PURPOSE
REGISTERS AND PSW UPON A POWER DOWN. WHEN POWER IS RESTORED A MESSAGE
IS TYPED AND THE CURRENT TEST WILL RESTART.

5.2.7 CLOCK HANDLER

IN DEFPA THIS ROUTINE SETS UP THE LINE CLOCK TO INTERRUPT
THE CENTRAL PROCESSOR EXACTLT ONE CLOCK PERIOD
AFTER THE ROUTINE IS EXITED.

IN DEFPB THIS ROUTINE HANDLES THE CLOCK INTERRUPTS AND
CONTROLS THE NUMBER OF SECONDS THAT THE ROM TEST IS
LOOPED ON.

5.3 OPERATOR ACTION

NEITHER PROGRAM REQUIRES OPERATOR INTERVENTION.

6. ERRORS

6.1 ERROR HALTS AND DESCRIPTION

THE FOLLOWING DESCRIPTION PERTAINS TO ACT11 AND STAND-ALONE OPERATION.

WHEN AN ERROR IS ENCOUNTERED THE CALL TO THE ERROR ROUTINE IS MADE AND AN ERROR MESSAGE PERTAINING TO THE ERROR WILL BE TYPED. EACH ERROR TYPE OUT WILL CONTAIN THE FOLLOWING:

1. AN ERROR MESSAGE
2. A DATA HEADER
3. A DATA STRING

THE DATA STRING WILL CONTAIN, AT A MINIMUM, THE ERROR PC AND THE TEST NUMBER. IN SOME CASES THE EXPECTED AND ACTUAL VALUES OF A REGISTER ARE ALSO INCLUDED.

FLOATING POINT DATA IS TYPED IN THE FORMAT SHOWN IN 5.2.5.3 OR 5.2.5.4.

REFER TO THE LISTING UNDER $ERRTB FOR THE TYPES OF ERRORS THAT CAN OCCUR.

6.2 ERROR RECOVERY

SW<15:9>=0 - MOST ERRORS WILL CAUSE EXECUTION TO GO TO THE START OF THE NEXT TEST AFTER THE MESSAGE IS TYPED. A FEW TESTS ARE DIVIDED INTO SECTIONS. IN THESE TESTS AN ERROR WILL CAUSE EXECUTION TO GO TO THE NEXT SECTION.

SW<15>=1 - AFTER THE ERROR MESSAGE HAS BEEN TYPED THE PROGRAM WILL HALT. PRESSING THE CONSOLE CONTINUE WILL CAUSE THE PROGRAM TO CONTINUE AS IF SW<15>=0.

7. RESTRICTIONS

7.1 STARTING RESTRICTIONS

THE USER SHOULD ENSURE THAT EITHER A LINE CLOCK OR A PROGRAMMABLE CLOCK IS INSTALLED TO OBTAIN MAXIMUM TEST EFFECTIVENESS.
IF A LINE CLOCK IS NOT INSTALLED A MESSAGE IS TYPED (ON FIRST PASS) AND THE PROGRAM CONTINUES WITH THE NEXT TEST.

7.2 OPERATING RESTRICTIONS

```
370
371
```

NONE

8. MISCELLANEOUS

8.1 EXECUTION TIME

A. DEFPA

THE FIRST PASS TAKES APPROXIMATELY 5 SECONDS.
ALL SUBSEQUENT PASSES TAKE APPROXIMATELY 2 MINUTES.

B. DEFPB

THE FIRST PASS TAKES APPROXIMATELY 30 SECONDS.
ALL SUBSEQUENT PASSES TAKE APPROXIMATELY 90 SECONDS.

8.2 STACK POINTER

THE STACK IS INITIALLY SET TO 1100.

8.3 PASS COUNT

THE PROGRAM MAKES ONE PASS FOR EACH END OF PASS MESSAGE.
THE END OF PASS MESSAGE DESCRIBES THE TOTAL NUMBER OF
PASSES COMPLETED AND THE TOTAL NUMBER OF ERRORS SINCE
THE LAST END OF PASS MESSAGE.

8.4 ITERATIONS

.

THE FIRST PASS OF THE PROGRAMS WILL AUTOMATICALLY INHIBIT
ITERATIONS. ALL SUBSEQUENT PASSES WILL PERFORM FULL,
(2000 DECIMAL) ITERATIONS.

8.5 SPECIAL REGISTERS

NONE

8.6 T-BIT TRAPPING

NONE

8.7 OSCILLOSCOPE SYNC POINTS

SW<8>=0 CAUSES SWITCHES <7:0> TO BE LOADED INTO THE MICRO-BREAK REGISTER IN THE FPP. THIS LOAD OCCURS IN THE SCOPE AND ERROR ROUTINES. WHEN THE FPP GOES THROUGH THE ROM STATE SELECTED BY SWITCHES <7:0>, A PULSE IS GENERATED AND IS AVAILABLE ON THE BACK PLANE ON PIN:

D05B2 (DB2 ON SLOT 5)

SINCE CERTAIN TESTS USE THE MICRO-TRAP FEATURE FOR FAULT ISOLATION, THE MICRO-MATCH SYNC ROM STATE CANNOT BE SELECTED FROM THE SWITCHES. THERE WILL BE A DEFAULT ROM ADDRESS SYNC PULSE DEFINED IN THE HEADER FOR THOSE TESTS.

8.8 A-BRANCH, NO-MEM BRANCH, AND ADX ROM TESTS

THE LAST 3 TESTS OF DEFPB TEST THOSE CELLS OF THE A-BRANCH, NO-MEM BRANCH, AND ADX ROMS THAT WERE NOT PREVIOUSLY TESTED. THESE TESTS ARE ALSO USED TO VERIFY THE OPERATION OF THE "DSI" BIT IN THE FPP CONTROL STORE. THIS IS DONE BY TURNING THE CLOCK ON (TO CAUSE INTERRUPTS) AND LOOPING ON THESE TESTS.

THE PROGRAM DEFAULTS TO A 30 SECOND LOOP ON THESE TESTS. IF THIS TIME NEEDS TO BE CHANGED, LOCATION "TIMES" IN THE COMMON TAGS AREA CAN BE CHANGED TO THE DESIRED NUMBER OF SECONDS.

WHILE THESE TESTS ARE BEING LOOPED ON, THE SECOND COUNT IS DISPLAYED IN THE HIGH BYTE OF THE DATA LIGHTS.

8.9 ACT11 AND APT11 COMPATABILITY

LOCATION 46 CONTAINS THE ADDRESS OF "$ENDAD" FOR THE ACT11 SEQUENCE TABLE.

LOCATION 44 CONTAINS THE ADDRESS OF "$APTHDR" FOR APT11 COMPATABILITY. THE ENVIRONMENT TABLE, MAIL BOX AND COMMUNICATIONS ROUTINE ARE ALSO INCLUDED.

9. PROGRAM DESCRIPTION

FOLLOWING IS A DESCRIPTION OF EACH TEST OF THE PROGRAM. THE TITLE OF THE TEST INDICATES THE FUNCTION BEING TESTED AND THE NARRATIVE DESCRIBES THE SPECIFIC FAULTS THAT THE PROGRAM IS LOOKING FOR.

DOCUMENT
****
PDP11-45/55/70 FP11-C DIAGNOSTIC PART 2
****

PDP11-45/55/70 FP11-C DIAGNOSTIC PART 2

TABLE OF CONTENTS
****

PDP11-45/55/70 FP11-C DIAGNOSTIC PART 2

TABLE OF CONTENTS
****

```
                        PDP11-45/55/70 FP11-C DIAGNOSTIC PART 2
                    21          COPYRIGHT (C) FEBRUARY 21,1976
                                DIGITAL EQUIPMENT CORP.
                                MAYNARD, MASS. 01754

                                PROGRAM BY DONALD W. MONROE

                                THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                                PACKAGE (MAINDEC-11-DZQAC-B2),NOV 21, 1975.


                    ;*******************************************************************
                46  ;OPERATIONAL SWITCH SETTINGS
                    ;*******************************************************************

                47
                                SWITCH          USE
                                ------          ---------------------
                                  15            HALT ON ERROR
                                  14            LOOP ON TEST
                                  13            INHIBIT ERROR TYPEOUTS
                                  12            INHIBIT MEMORY MANAGEMENT TEST
                                  11            INHIBIT ITERATIONS
                                  10            BELL ON ERROR
                                   9            LOOP ON ERROR
                                   8            LOOP ON TEST IN SWR<7:0>
                                  8=0           LOAD MICRO BREAK WITH SWR<7:0>

                    ;*******************************************************************
                60  ;BASIC DEFINITIONS
                    ;*******************************************************************

                62      INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***

                77      MISCELLANEOUS DEFINITIONS

                83      GENERAL PURPOSE REGISTER DEFINITIONS

                104     PRIORITY LEVEL DEFINITIONS

                114     "SWITCH REGISTER" SWITCH DEFINITIONS

                142     DATA BIT DEFINITIONS (BIT00 TO BIT15)

                170     BASIC "CPU" TRAP VECTOR ADDRESSES
```

```
                                PDP11-45/55/70 FP11-C DIAGNOSTIC PART 2
                                ;*****************************************************************
                          187   ;CACHE   REGISTER DEFINITIONS
                                ;*****************************************************************
                                
                                ;*****************************************************************
                          198   ;CPU REGISTER DEFINITIONS
                                ;*****************************************************************
                                
                                ;*****************************************************************
                          212   ;MEMORY MANAGEMENT DEFINITIONS
                                ;*****************************************************************
                                
                          215   ;    MEMORY MANAGEMENT STATUS REGISTER ADDRESSES
                                
                          226   ;    USER "I" PAGE DESCRIPTOR REGISTERS
                                
                          237   ;    USER "D" PAGE DESCRIPTOR REGISTORS
                                
                          248   ;    USER "I" PAGE ADDRESS REGISTERS
                                
                          259   ;    USER "D" PAGE ADDRESS REGISTERS
                                
                          270   ;    SUPERVISOR "I" PAGE DESCRIPTOR REGISTERS
                                
                          281   ;    SUPERVISOR "D" PAGE DESCRIPTOR REGISTERS
                                
                          292   ;    SUPERVISOR "I" PAGE ADDRESS REGISTERS
                                
                          303   ;    SUPERVISOR "D" PAGE ADDRESS REGISTERS
                                
                          314   ;    KERNEL "I" PAGE DESCRIPTOR REGISTERS
                                
                          325   ;    KERNEL "D" PAGE DESCRIPTOR REGISTERS
                                
                          336   ;    KERNEL "I" PAGE ADDRESS REGISTERS
                                
                          347   ;    KERNEL "D" PAGE ADDRESS REGISTERS
                                
                                ;*****************************************************************
                          361   ;UNIBUS MAP REGISTER DEFINITIONS
                                ;*****************************************************************
                                
                          364   ;    THE LOWER 16 BITS OF THE MAP REGISTERS ARE LABELED 'MAPLXX'
                                ;    THE UPPER 6 BITS OF THE MAP REGISTERS ARE LABELED 'MAPHXX'
```

```
                        PDP11-45/55/70 FP11-C DIAGNOSTIC PART 2
                        ;*******************************************************************
                448     ;FLOATING POINT REGISTER DEFINITIONS
                        ;*******************************************************************

                        ;*******************************************************************
                462     ;FLOATING POINT STATUS BIT DEFINITIONS
                        ;*******************************************************************

                        ;*******************************************************************
                478     ;FLOATING POINT EXCEPTION CODE DEFINITIONS
                        ;*******************************************************************

                        ;*******************************************************************
                487     ;FLOATING POINT VECTOR DEFINITION
                        ;*******************************************************************

                        ;*******************************************************************
                496     ;TRAP CATCHER
                        ;*******************************************************************

                499     ;       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

                        ;*******************************************************************
                506     ;STARTING ADDRESS(ES)
                        ;*******************************************************************

                        ;*******************************************************************
                509     ;ACT11 HOOKS
                        ;*******************************************************************

                        ;*******************************************************************
                520     ;COMMON TAGS
                        ;*******************************************************************

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

```
                                PDP11-45/55/70 FP11-C DIAGNOSTIC PART 2
                                ;*******************************************************************
                        581     APT MAILBOX-ETABLE
                                ;*******************************************************************

                                600                                     BITS 15-11=CPU TYPE
                                                                             11/04=01,11/05=02,11/20=03,11/40
                                                                             11/70=06,PDQ=07,Q=10
                                                                        BIT 10=REAL TIME CLOCK
                                                                        BIT  9=FLOATING POINT PROCESSOR
                                                                        BIT  8=MEMORY MANAGEMENT


                                ;*******************************************************************
                        764     ERROR POINTER TABLE
                                ;*******************************************************************

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

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

                                ;*******************************************************************
                        1477    APT PARAMETER BLOCK
                                ;*******************************************************************

                        1563    TEST 1  MULF*MO*(DST=0)*-(SRC=0)

                        1565            THE A OR NO-MEM ROMS COULD FAIL.  THE ONLY OTHER POSSIBL
                                        FAILURES WOULD BE ROM STATE 350 OR THE 4F1 BRANCH.

                                        IF THE 4F1 BRANCH FAILS THE ACD+1 WILL ALSO BE CLEARED.

                                        FPU ROM FLOW - 30, 61, 350, 351

                        1602    TEST 2  MULF*-MO*-(DST=0)*(SRC=0)

                                        THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS THE A
                                        BRANCH OR THE ADX ROMS OR STATE 344.

                                        FPU ROM FLOW - 11, 130, 61, 344, 351

                        1635    TEST 3  MULF*-MO*IMM*(DST=0)*(SRC=0)

                                        THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS THE ADX
                                        ROM OR STATE 354.

                                        FPU ROM FLOW - 11, 131, 61, 354, 351
```

```
                                PDP11-45/55/70 FP11-C DIAGNOSTIC PART 2
                        1668    TEST 4  MULF*SWR*ADD

                                THIS TEST ENSURES THAT ROM STATES 61, 340, 233, AND 112
                                SETUP THE SC AND SHIFT CONTROL PROPERLY, THAT FRHK MUL S
                                IS NOT STUCK LOW, THAT FRHL MPY/DIV ADD(1) L IS NOT STUC
                                HIGH, AND THAT FRMH FORCE MPY/DIV ADD CAUSES THE FALU TO
                                AN ADD.  THIS IS DONE BY FIRST SETTING A MICRO-TRAP ON
                                STATE 117 TO TEST MUL SWR AND THEN A TRAP IS SET ON 112
                                TO ENSURE THE OTHER FUNCTIONS ARE WORKING PROPERLY.


                        1677
                                IF FRME SHFC 0(0)H AND 1(0)H ARE TIED TOGETHER, THE SHIF
                                CONTROL WILL BE LOADED WITH THE VALUE OF THE NORM POS
                                ENCODER.  THIS WILL CAUSE A LEFT SHIFT OF 7 IN THE AR
                                AND A RIGHT SHIFT OF 8 IN THE QR.

                                IF FXPP SC00 DOES NOT GET TO FRMA AS A HIGH OR DOES NOT
                                GET TO RAD00 AS A HIGH EXECUTION WILL GO TO STATE 112
                                INSTEAD OF 113 IN SECTION 2.

                                IF FRHL MPY SIGN EXTEND DOES NOT GO LOW, THE AR WILL END
                                UP WITH 0.01 IN IT INSTEAD OF 0.10.

                                NOTE: SECTION 1 OF THIS TEST DOES NOT ALLOW SWITCHES <7:
                                TO BE LOADED INTO THE MICROBREAK FOR SYNC PURPOSES.

                        1768    TEST 5  MULF*SWR*SUB

                                THIS TEST ENSURES THAT FRHL MPY/DIV ADD (1) L GOES HIGH
                                WHEN FRHK STRING IS HIGH.  THIS SIGNAL CAUSES FRMH FORCE
                                MPY/DIV SUB TO GO LOW FORCING THE FALU TO DO A SUBTRACT.

                                SECTION 2 OF THIS TEST MICRO-TRAPS ON STATE 117 TO CHECK
                                THAT FRHK MPY SIGN EXTEND GOES HIGH.  IF THIS TRAP DOES
                                NOT OCCUR THEN FRHK E108 PIN 1 & 13 ARE NOT GOING HIGH I
                                STATE 112.

                        1861    TEST 6  LOGIC TEST OF FRHK MUL SWR*FD(0)

                                THIS TEST CHECKS THE LOGIC THAT CAUSES FRHK MUL SWR TO G
                                HIGH.  THE PARTICULAR GATES BEING TESTED ARE E107, E108,
                                AND E115.  THEY ARE TESTED BY SETTING A MICRO-TRAP ON TH
                                -SWR ROM STATE THAT WOULD BE ENTERED IF THE LOGIC FAILS.

                                IN CERTAIN CASES THE TRAP MAY OCCUR AFTER THE FIRST TIME
                                THRU THE 2F3 BRANCH.  IN THESE CASES THE QR WILL BE CHEC
                                TO ENSURE THE CORRECT DATA.

                                NOTE:   SYNC POINT NOT SELECTABLE. DEFAULT=116
```

```
                                PDP11-45/55/70 FP11-C DIAGNOSTIC PART 2
1970    TEST 7  MULF (.5*1)

                        THIS TEST MULTIPLIES 1 BY 0.5 TO ENSURE THAT ALL THE
                        ROM SIGNALS IN STATES 116, 110, AND 72 FUNCTION PROPERLY

                        FPU ROM FLOW - 11, 131, 61, 340, 233, 112, 3(116), 110,

1999    TEST 10 MULF (.111...1*1)

                        THIS TEST CHECKS ROM STATE 111.

                        FPU ROM FLOW - 11, 130, 61, 340, 233, 112, 113, 3(117),

2027    TEST 11 HIGH ORDER MUL SHIFT ENCODER ROM

                        THIS TEST CHECKS EVERY ADDRESS ON THE HIGH ORDER MUL SHI
                        ENCODER.  THIS IS DONE IN TWO SECTIONS.  THE FIRST SECTI
                        RUNS A COUNT PATTERN THROUGH QR BITS <41:35>.  THE SECON
                        SECTION STARTS EACH MULTIPLY WITH QR BITS <41:35> ALL ON
                        AND RUNS A COUNT PATTERN THRU QR BITS <48:42>.

                        SINCE THE MULCIPLICAND IS "1.0" THE RESULT SHOULD BE IDE
                        TO THE MULTIPLIER.

                        IF AN ERROR OCCURS THE TYPEOUT INDICATES THE ROM ADDRESS
                        AND THE EXPECTED OUTPUT OF THE ROM FOR EACH SHIFT OF THA
                        PARTICULAR MULTIPLY.

2084    TEST 12 LOGIC TESTS OF FRHK MUL SWR*FD(1)

                        THIS TEST CHECKS THE A-BRANCH AND ADX ROMS FOR MULD AND
                        THE GATES THAT DRIVE FRHK MUL SWR WITH FD(1).  IN PARTIC
                        FRLH E1, FRHK E107, AND E108.

                        THIS LOGIC TEST IS PERFORMED BY SETTING A MICRO-TRAP OFF
                        THE 2F3 BRANCH TO ENSURE THAT FRHK MUL SWR GOES HIGH.

                        NOTE:   SYNC POINT NOT SELECTABLE. DEFAULT=116

2246    TEST 13 LOW ORDER MUL SHIFT ENCODER ROM

                        THIS TEST CHECKS EVERY ADDRESS ON THE LOW ORDER MUL SHIF
                        ENCODER ROM.  THIS IS DONE IN TWO SECTIONS, THE FIRST
                        SECTION RUNS A COUNT PATTERN THRU QR BITS <9:3>.  THE
                        SECOND SECTION STARTS EACH MULTIPLY WITH QR BITS<9:3>
                        ALL ONE'S AND RUNS A COUNT PATTERN THRU QR BITS <16:10>.

                        SINCE THE MULTIPLICAND IS "1.0" THE RESULT SHOULD BE
                        IDENTICAL TO THE MULTIPLIER.

                        IF AN ERROR OCCURS THE TYPEOUT INDICATES THE ROM ADDRESS
                        AND THE EXPECTED OUTPUT OF THE ROM FOR EACH SHIFT OF THE
                        LOWER 24 BITS OF THAT PARTICULAR MULTIPLY.
```

```
                                PDP11-45/55/70 FP11-C DIAGNOSTIC PART 2
                        2304    TEST 14 MODF*MO*(SRC=0)*(DST=0)

                                THIS TEST ENSURES THAT FXPC FIRB08(1) GETS TO FRMA
                                AS A HIGH AND THAT THE A BRANCH AND NO-MEM ROMS
                                FUNCTION PROPERLY.

                                FPU ROM FLOW-30,61,354,355

                        2327    TEST 15 MODF*-MO*(INT=0)

                                THIS TEST ENSURES THAT AN INTEGER RESULT OF ZERO CAUSES
                                ZERO TO BE STORED IN THE DST ACC+1 AND THE FRACTION IN
                                THE DST ACCUMULATOR.

                                FPU ROM FLOW-MULTIPLY,76,266,327,305,325,NORMALIZE

                        2368    TEST 16 MODF*(INT=2**25)*-BOU

                                THIS TEST GENERATES A NUMBER WHO'S FRACTIONAL
                                PART IS ZERO.

                                FPU ROM FLOW-MULTIPLY,76,266,327,307,7,341

                        2418    TEST 17 MODF*-(INT=0)*-(FRAC=0)

                                THIS TEST CHECKS THE FLOWS THAT GET EXECUTED WHEN THE
                                RESULT CONTAINS AN INTEGER AND A FRACTION.

                                FPU ROM FLOW-MULTIPLY,76,266,327,307,5,2(332),336,170,10

                        2448    TEST 20 MODF*BOU*-(FV*FIV)

                                THIS TEST ENSURES THAT AN INTEGER OVERFLOW ON MODF CAUSE
                                THE INTEGER TO BE CLEARED.

                                FPU ROM FLOW-MULTIPLY,76,266,327,307,7,341,102,242,107

                        2477    TEST 21 MODF*-(INT=0)*(FRAC=0)

                                THIS TEST ENSURES THAT ROM STATE 105 DETECTS A ZERO FRAC
                                A MICRO TRAP IS SET ON STATE 343 (NORMALIZE) IN CASE
                                THIS FAILS.  IF THIS FAILS, THE MICRO-BREAK REGISTER CAN
                                LOADED FROM THE SWITCHES SINCE THE FPU WILL HANG IF IT G
                                THE NORMALIZE FLOWS.

                                NOTE:   SYNC POINT NOT SELECTABLE. DEFAULT=343

                                FPU ROM FLOW-MULTIPLY,76,266,327,307,5,332,336,170,105,1
```

```
                                PDP11-45/55/70 FP11-C DIAGNOSTIC PART 2
                                2511    TEST 22 MODD*(INT=0)

                                                THIS TEST ENSURES STATE 365 STORES ZERO IN THE
                                                DST ACC+1 AND THAT STATE 325 ROUNDS THE FRACTION.

                                                FPU ROM FLOW-MULTIPLY,76,266,326,365,325,NORMALIZE

                                2555    TEST 23 MODD*-(INT=0)*-(FRAC=0)

                                                THIS TEST ENSURES THAT ROM STATES 326 AND 327 ARE WORKIN
                                                PROPERLY

                                                FPU ROM FLOW-MULTIPLY,76,266,326,367,5,332,336,170,105,1

                                2593    TEST 24 FALU LOGICAL "AND" TEST

                                                THIS TEST CHECKS THE LOGICAL AND FUNCTION OF THE FALU.
                                                THIS IS DONE BY FLOATING A ONE THRU AR<57:3>, THUS
                                                MAKING THE RESULT OF THE MULTIPLY HAVE AN INTEGER OF 1.0
                                                (201*0.1) AND THE FRACTION WILL BE .5+PATTERN (200*0.1+P
                                                WHERE PATTERN IS THE VALUE OF AR <56:3> FOR THE PARTICUL

                                                THE SECOND SECTION ALSO FLOATS A ONE THRU AR<57:3>
                                                BUT, IN THIS TEST THE INTEGER WILL BE 1.0+COUNT (270*0.1
                                                AND THE FRACTION WILL BE ZERO.

                                2697    TEST 25 DIVF*-MD*(DST=0)*-(SRC=0)

                                                THIS TEST ENSURES THAT STATE 352 STORES ZERO IN THE RESU
                                                THE A-BRANCH AND ADX ROMS ARE OK.

                                                FPU ROM FLOW-11,130,73,352

                                2733    TEST 26 INITIAL TESTS OF DIVF

                                                THIS TEST ENSURE THAT SPECIFIC SIGNALS REQUIRED FOR THE
                                                FUNCTIONING PROPERLY. EACH SECTION DESCRIBES THE SIGNAL(
                                                TESTED. A MICRO-TRAP IS SET ON STATE 14 TO TEST THESE SI

                                                NOTE:   SYNC POINT NOT SELECTABLE. DEFAULT=14

                                2864    TEST 27 NORM NEG ENCODER

                                                THIS TEST RUNS A COUNT PATTERN ACROSS THE INPUTS TO THE
                                                NORM NEG ENCODED ON FRHK.

                                                THIS IS DONE BY USING A NUMERATOR OF 0.1 AND DENOMINATOR
                                                THAT COUNTS BETWEEN 0.10000001 AND 0.11111111.  THE EXEC
                                                IS INTERRUPTED AT STATE 14 AND THE QR CHECKED FOR THE CO
                                                AMOUNT OF SHIFTS

                                                THE PATTERNS FROM 1.00000000 TO 1.11111111 ARE NOT CHECK
                                                THEY ARE IMPOSSIBLE TO GENERATE.
```

```
                                PDP11-45/55/70 FP11-C DIAGNOSTIC PART 2
                                                NOTE:   SYNC POINT NOT SELECTABLE. DEFAULT=14

                        2957    TEST 30 Q REG NORM SHIFT COUNT LOOK UP

                                THIS TEST FIRST CHECKS ROM STATE 4 AND ADDRESS 4 OF THE
                                NORM SHIFT COUNT LOOK-UP ROM. IF FRHK DIV DONE
                                IS STUCK HIGH THE FPP WILL HANG IN ROM STATE 14.

                                A COUNT PATTERN IS THEN RUN THRU BITS <57:50> OF THE QR
                                CHECK ALL LOCATIONS IN THE ROM.

                        2965
                                IF AN ERROR OCCURS, THE TYPEOUT INDICATES THE ROM ADDRES
                                AND ROMOUT DATA THAT WAS BEING TESTED.

                                FPU ROM FLOW-11,130,73,342,X(14),4,NORMALIZE

                        3017    TEST 31 DIVD INITIAL TESTS

                                THIS TEST ENSURES THAT FRLH DLE PREC QT HI B(1) AND LO B
                                ARE FUNCTIONING PROPERLY.

                        3021    THIS IS DONE BY TRAPPING ON STATE 14 AND LOOKING AT THE

                                NOTE:   SYNC POINT NOT SELECTABLE. DEFAULT=14

                        3089    TEST 32 FRLJ QSI -1,-2, & -3

                                THIS TEST CHECKS THE QSI -1,-2, & -3 MUX ON FRLH WITH
                                THE DIVIDE FUNCTION. THERE ARE FOUR CONDITIONS THAT
                                ARE CHECKED: 1) LEFT SHIFT 7*FALU59; 2) LEFT SHIFT 7*-FA
                                3) LEFT SHIFT 3*FALU59;AND 4) LEFT SHIFT 3*-FALU59.
                                CONDITION 1 WAS TESTED IN THE PREVIOUS TEST.

                                NOTE:   SYNC POINT NOT SELECTABLE. DEFAULT=14

                        3177    TEST 33 DIVD*-MO*IMM

                                THIS TEST DIVIDES 127(10) BY 255(10) TO ENSURE THAT STAT
                                246 IS FUNCTIONING PROPERLY.

                                FPU ROM FLOW-20,141,73,342,14,4,246,NORMALIZE

                        3211    TEST 34 STST

                                THIS TEST USES THE MICRO-BREAK TRAP TO CHECK THE
                                STST INSTRUCTION.  EACH SECTION OF THIS TEST CHECKS
                                A DIFFERENT MODE OF THE INSTRUCTION.
```

PDP11-45/55/70 FP11-C DIAGNOSTIC PART 2

3270 TEST 35 FIUV

THIS TEST CHECKS THE 9 CONDITIONS THAT CAN CAUSE AN
INTERRUPT ON A MINUS ZERO
IF SECTION ONE FAILS TO INTERRUPT THE PROBLEM IS MOST
LIKELY ON FRMB. ALL OTHER FAILURES SHOULD BE DUE TO THE
CONTROL STORE.

3435 TEST 36 FIU

THIS TEST CHECKS THE 4 PLACES WHERE AN UNDERFLOW TRAP
CAN OCCUR.

3557 TEST 37 FIV

THIS TEST CHECKS THE 5 POSSIBLE OVERFLOW CONDITIONS. IT
ALSO INCLUDES 3 ADDITIONAL CHECKS OF THE STCF TO ENSURE
CONTROL STORE IS FUNCTIONING PROPERLY.

3677 TEST 40 FIC

THIS TEST ENSURES THAT THE INTEGER CONVERSION
INTERRUPT OCCURS FROM ALL 3 POSSIBLE ROM STATES.

3731 TEST 41 DIVIDE BY ZERO

THIS TEST CHECKS THE TWO ROM STATES THAT CAN BE USED WHE
AN ATTEMPT TO DIVIDE BY ZERO IS MADE.

3775 TEST 42 ILLEGAL OP-CODES

THIS TEST FIRST ENSURES THAT THE ILLEGAL OP-CODE TRAP FU
PROPERLY. IT THEN CHECKS ALL THE POSSIBLE ADDRESSES OF
NO-MEM ROM THAT FORCE AN ILLEGAL OP CODE TRAP.

LAST, IT CHECKS THE FID BIT TO ENSURE IT FUNCTIONS PROPE

3834 TEST 43 NO-MEM ROM

THIS TEST COMPLETES THE TEST OF THE NO-MEM ROM. IT
CONSISTS OF TESTING THOSE OP-CODES THAT CAN USE R6 OR R7
AS LEGAL REGISTERS.

3923 TEST 44 FP PRIORITY ARBITRATION

THIS TEST CHECKS THE FLOATING POINT TRAP PRIORITY ARBITR
LOGIC ON TMCA FOR PIR7 AND STACK LIMIT YELLOW. THE
MEMORY MANAGEMENT ARBITRATION IS PERFORMED LATER.

```
                                PDP11-45/55/70 FP11-C DIAGNOSTIC PART 2
                        3976    TEST 45 MEMORY MANAGEMENT FUNCTIONS

                                THIS TEST ENSURES THAT THE SIGNAL SAPK I SPACEA (0) IS
                                ENABLED DURING THE TWO FLOATING POINT "BUST" CYCLES.

                                THIS IS DONE BY MAKING THE D SPACE PAR, THAT WOULD BE
                                REFERENCED BY THE FP INSTRUCTION, NON-RESIDENT.  SINCE
                                THE INSTRUCTION IS IMMEDIATE MODE, I SPACE SHOULD BE FOR
                                AND THE TRAP SHOULD NOT OCCUR.

                                NOTE: THIS TEST CAN BE DISABLED BY SWITCH 12.

                        4047    TEST 46 FP AND MGMT TRAP ARBITRATION

                                THIS TEST ENSURES THAT TMCA HONOR SEGT CAUSES TMCA HONOR
                                FP TRAP TO GO HIGH.
                                THIS IS DONE BY SETTING THE FP TRAP WITH A MINUS ZERO TR
                                THEN CAUSING A MEMORY MANAGEMENT TRAP.

                                NOTE: SWITCH 12 SKIPS THIS TEST

                        4092    TEST 47 FPA AND FEA

                                THIS TEST EXECUTES TWO PIECES OF CODE IN DIFFERENT PARTS
                                THE PROGRAM TO TEST THE FPA AND FEA REGISTERS.  THIS TES
                                SETS UP THE DATA AND FP VECTOR AND THEN JUMPS TO
                                THE APPROPRIATE LOCATIONS.  THESE TWO PARTICULAR LOCATIO
                                CODE TO CAUSE A 25252 (125252 IF MORE THAN 16K OF MEMORY
                                AND 52524 PATTERN TO PLACED IN THE FPA AND THE INSTRUCTI
                                WILL CAUSE THIS PATTERN TO BE PUT IN THE FEA.

                        4139    TEST 50 FXPB IMMEDIATE (MODE 1 & 4)

                        4141            THIS TEST ENSURES THAT FXPB IMMEDIATE GOES HIGH FOR
                                        ADDRESS MODE 1 AND 4.

                        4178    TEST 51 FMM *USER MODE

                                THIS TEST ENSURES THAT FRLN FMM(1) DOES NOT GO HIGH IN
                                SUPERVISOR OR USER MODE.

                        4197    TEST 52 ACCUMULATOR ADDRESSING

                                THIS TEST CHECKS THE LOGIC THAT DRIVES THE ACCUMULATOR A
                                LINES ON FXPN THAT HAVE NOT ALREADY BEEN TESTED.

                        4224    TEST 53 ACCUMULATOR VOLATILITY

                                THIS TEST EXECUTES THE WALKING ONE'S AND ZERO'S ALGORITH
                                ON THE FLOATING POINT ACCUMULATORS.
```

```
                                PDP11-45/55/70 FP11-C DIAGNOSTIC PART 2
                                ;*********************************************************************
                          4343
                                ;*********************************************************************
                                ;*********************************************************************
                          4345  A-BRANCH AND ADX ROM EXERCISER
                                ;*********************************************************************

                          4346                THE REMAINING SUBTESTS IN THIS PROGRAM TEST ALL THE
                                              LOCATIONS IN THE A-BRANCH AND ADX ROMS THAT HAVE NOT
                                              ALREADY BEEN TESTED.
                                              THE ENTIRE SEQUENCE IS FIRST EXECUTED ONCE. THEN, THE
                                              LINE (OR PROGRAMMABLE) CLOCK IS TURNED ON AND THIS SEQUE
                                              IS LOOPED ON FOR APPROXIMATELY 30 SECONDS. THE PURPOSE
                                              OF THIS IS TRY AND ENSURE THAT THE DSI BIT IN THE CONTRO
                                              STORE IS FUNCTIONAL IN ALL THE APPROPRIATE ROM STATES.

                                ;*********************************************************************
                          4355
                                ;*********************************************************************

                          4358      TEST 54 A-BRANCH*ADX ROM EXERCISER

                                ;*********************************************************************
                          5459  CLOCK HANDLER
                                ;*********************************************************************

                          5460                THIS CODE CONTROLS THE CLOCK AND THE LOOPING ON THE
                                              PREVIOUS TESTS WHILE THE CLOCK IS RUNNING.
                                              A TICK COUNT IS KEPT IN THE LOCATION CALLED "TICKS".
                                              THE LOCATION CALLED "TIME" CONTROLS THE NUMBER OF SECOND
                                              BEFORE END OF PASS IS REPORTED.

                                              WHILE THIS TEST IS BEING PERFORMED, THE HIGH BYTE OF THE
                                              DATA LIGHTS WILL INCREMENT APPROXIMATELY ONCE PER SECOND

                                ;*********************************************************************
                          5543  END OF PASS ROUTINE
                                ;*********************************************************************

                          5546      INCREMENT THE PASS NUMBER ($PASS)
                                    TYPE "END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT Y
                                    WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS
                                    IF THERES A MONITOR GO TO IT
                                    IF THERE ISN'T JUMP TO LOOP
```

```
                                PDP11-45/55/70 FP11-C DIAGNOSTIC PART 2
                                ;*********************************************************************
                        5592    ;SCOPE HANDLER ROUTINE
                                ;*********************************************************************
                                ;
                        5595    ;THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
                                ;AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7
                                ;
                        5597    ;AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
                                ;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                ;SW14=1  LOOP ON TEST
                                ;SW11=1  INHIBIT ITERATIONS
                                ;SW09=1  LOOP ON ERROR
                                ;SW08=1  LOOP ON TEST IN SWR<7:0>
                                ;CALL
                                ;       SCOPE           ;;SCOPE=IOT
                                ;
                                ;*********************************************************************
                        5665    ;ERROR HANDLER ROUTINE
                                ;*********************************************************************
                                ;
                        5668    ;THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
                                ;SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
                                ;AND GO TO $ERRTYP ON ERROR
                                ;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                ;SW15=1  HALT ON ERROR
                                ;SW13=1  INHIBIT ERROR TYPEOUTS
                                ;SW10=1  BELL ON ERROR
                                ;SW09=1  LOOP ON ERROR
                                ;CALL
                                ;       ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
                                ;
                                ;*********************************************************************
                        5727    ;CONVERT FLOATING BINARY TO OCTAL ASCIZ
                                ;*********************************************************************
                                ;
                        5728
                                ;       THIS ROUTINE CONVERTS A 32 BIT FLOATING NUMBER TO AN OCTAL
                                ;       ASCIZ STRING IN THE FOLLOWING FORMAT:
                                ;
                                ;               W  XXX  YYY ZZZZZZ
                                ;
                                ;       WHERE   W = SIGN BIT
                                ;               X = 8-BIT EXPONENT (RIGHT JUSTIFIED)
                                ;               Y = FRACTION BITS <57:51> (RIGHT JUSTIFIED)
                                ;               Z = FRACTION BITS <50:35>
                                ;
                                ;IT IS ENTERED BY A TRAP CALL WITH THE ADDRESS OF THE FLOATING
                                ;NUMBER IN THE WORD FOLLOWING THE CALL.
                                ;IT RETURNS WITH THE ADDRESS OF THE ASCIZ STRING ON THE STACK.
```

```
                                PDP11-45/55/70 FP11-C DIAGNOSTIC PART 2
                                ;*********************************************************************
                        5799    ;CONVERT FLOATING DOUBLE BINARY TO OCTAL ASCIZ
                                ;*********************************************************************

                                5800
                                ;       THIS ROUTINE CONVERTS A 64 BIT FLOATING NUMBER TO AN OCTAL
                                ;       ASCIZ STRING IN THE FOLLOWING FORMAT:
                                ;
                                ;               U  VVV  WWW XXXXXX YYYYYY ZZZZZZ
                                ;
                                ;       WHERE   U = SIGN BIT
                                ;               V = 8-BIT EXPONENT (RIGHT JUSTIFIED)
                                ;               W = FRACTION BITS <57:51> (RIGHT JUSTIFIED)
                                ;               X = FRACTION BITS <50:35>
                                ;               Y = FRACTION BITS <34:19>
                                ;               Z = FRACTION BITS <18:03>
                                ;
                                ;       IT IS ENTERED BY A TRAP CALL WITH THE ADDRESS OF THE FLOATING
                                ;       NUMBER IN THE WORD FOLLOWING THE CALL.
                                ;       IT RETURNS WITH THE ADDRESS OF THE ASCIZ STRING ON THE STACK.
                                ;
                                ;*********************************************************************
                        5851    ;ROUTINE TO START THE LINE CLOCK
                                ;*********************************************************************

                                5852    ;       THIS ROUTINE SETS UP THE LINE CLOCK TO GUARANTEE THE
                                ;       MAXIMUM DELAY BEFORE IT WILL INTERRUPT.

                                ;*********************************************************************
                        5881    ;MUL SHIFT ENCODER ROM CONVERT ROUTINE
                                ;*********************************************************************

                                5882
                                ;               THIS ROUTINE TAKES THE CONTENTS OF $REG0 AND $REG1 AS A
                                ;               MULTIPLIER AND GENERATES THE ROM ADDRESSES AND OUTPUT
                                ;               DATA OF THE MUL SHIFT ENCODER ROM FOR THE ENTIRE MULTI-
                                ;               PLICATION.

                                5930            ;       THIS ROUTINE DOES A MAINTENANCE TRAP TO LOAD THE FEC
                                ;               REGISTER WITH 2.
```

```
                                PDP11-45/55/70 FP11-C DIAGNOSTIC PART 2
                                ;*******************************************************************
                        5944    ;SAVE AND RESTORE R0-R5 ROUTINES
                                ;*******************************************************************
                                ;
                        5947    ;SAVE R0-R5
                                ;CALL:
                                ;       SAVREG
                                ;UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
                                ;
                                ;TOP---(+16)
                                ; +2---(+18)
                                ; +4---R5
                                ; +6---R4
                                ; +8---R3
                                ;+10---R2
                                ;+12---R1
                                ;+14---R0
                                ;
                        5974    ;RESTORE R0-R5
                                ;CALL:
                                ;       RESREG
                                ;
                                ;*******************************************************************
                        5990    ;TYPE ROUTINE
                                ;*******************************************************************
                                ;
                        5993    ;ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 B
                                ;THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE
                                ;NOTE1:  $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CH
                                ;NOTE2:  $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED
                                ;NOTE3:  $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
                                ;
                                ;CALL:
                                ;1) USING A TRAP INSTRUCTION
                                ;       TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN
                                ;OR
                                ;       TYPE
                                ;       MESADR
```

```
                        PDP11-45/55/70 FP11-C DIAGNOSTIC PART 2
                6070    ;*********************************************************************
                        ;APT COMMUNICATIONS ROUTINE
                        ;*********************************************************************

                        ;*********************************************************************
                6129    ;ERROR MESSAGE TYPEOUT ROUTINE
                        ;*********************************************************************

                6131    ;THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE
                        ;ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE"
                        ;AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.

                        ;*********************************************************************
                6274    ;BINARY TO OCTAL (ASCII) AND TYPE
                        ;*********************************************************************

                6277    ;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIG
                        ;OCTAL (ASCII) NUMBER AND TYPE IT.
                        ;$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS
                        ;CALL:
                        ;       MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                        ;       TYPOS                   ;;CALL FOR TYPEOUT
                        ;       .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS
                        ;       .BYTE   M               ;;M=1 OR 0
                        ;                               ;;1=TYPE LEADING ZEROS
                        ;                               ;;0=SUPPRESS LEADING ZER

                        ;$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE
                        ;$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

                        ;*********************************************************************
6352                    ;CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
                        ;*********************************************************************

                6355    ;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIG
                        ;SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER
                        ;NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE T
                        ;BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS
                        ;REPLACED WITH SPACES.
                        ;CALL:
                        ;       MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE S
                        ;       TYPDS                   ;;GO TO THE ROUTINE
```

```
                                PDP11-45/55/70 FP11-C DIAGNOSTIC PART 2
                                ;*******************************************************************
                        6420    ;DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
                                ;*******************************************************************
                                ;
                        6423    ;THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
                                ;UNSIGNED OCTAL ASCIZ NUMBER.
                                ;CALL
                                ;       MOV     #PNTR,-(SP)     ;;POINTER TO LOW WORD OF BINARY
                                ;       JSR     PC,@#$DB2O      ;;CALL THE ROUTINE
                                ;       RETURN                  ;;THE ADDRESS OF THE FIRST ASCIZ
                                ;
                                ;*******************************************************************
                        6460    ;UNEXPECTED TRAP TO 4 ROUTINE
                                ;*******************************************************************
                                ;
                                ;*******************************************************************
                        6474    ;UNEXPECTED TRAP TO 114 ROUTINE
                                ;*******************************************************************
                                ;
                                ;*******************************************************************
                        6489    ;UNEXPECTED TRAP TO 244 ROUTINE
                                ;*******************************************************************
                                ;
                                ;*******************************************************************
                        6497    ;TRAP DECODER
                                ;*******************************************************************
                                ;
                        6500    ;THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTIO
                                ;AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDR
                                ;OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
                                ;GO TO THAT ROUTINE.
                                ;
                                ;*******************************************************************
                        6514    ;TRAP TABLE
                                ;*******************************************************************
                                ;
                        6516    ;THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLE
                                ;BY THE "TRAP" INSTRUCTION.
```

```
                                PDP11-45/55/70 FP11-C DIAGNOSTIC PART 2
                                ;*****************************************************************
                        6532    POWER DOWN AND UP RO.TI
                                ;*****************************************************************
```

```
1436                                    %
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456                                    .TITLE  PDP11-45/55/70 FP11-C DIAGNOSTIC PART 2
1457                                    ;*COPYRIGHT (C) FEBRUARY 21,1976
1458                                    ;*DIGITAL EQUIPMENT CORP.
1459                                    ;*MAYNARD, MASS. 01754
1460                                    ;*
1461                                    ;*PROGRAM BY DONALD W. MONROE
1462                                    ;*
1463                                    ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
1464                                    ;*PACKAGE (MAINDEC-11-DZQAC-B2),NOV 21, 1975.
1465                                    ;*
```

```
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
```

```
.SBTTL  OPERATIONAL SWITCH SETTINGS
;*
;*      SWITCH                  USE
;*      ------          --------------------
;*        15            HALT ON ERROR
;*        14            LOOP ON TEST
;*        13            INHIBIT ERROR TYPEOUTS
;*        12            INHIBIT MEMORY MANAGEMENT TEST
;*        11            INHIBIT ITERATIONS
;*        10            BELL ON ERROR
;*         9            LOOP ON ERROR
;*         8            LOOP ON TEST IN SWR<7:0>
;*        8=0           LOAD MICRO BREAK WITH SWR<7:0>
```

```
                                .SBTTL  BASIC DEFINITIONS

                                ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100                          STACK=  1100            ;;FIRST ADDRESS OF THE STACK
001100                          KERSTK= STACK           ;;KERNEL STACK
000700                          SUPSTK= STACK-200       ;;SUPERVISOR STACK
000600                          USESTK= STACK-300       ;;USER STACK
                                .EQUIV  EMT,ERROR       ;;BASIC DEFINITION OF ERROR CALL
                                .EQUIV  IOT,SCOPE       ;;BASIC DEFINITION OF SCOPE CALL
177776                          PS=     177776          ;;PROCESSOR STATUS WORD
                                .EQUIV  PS,PSW
177774                          STKLMT= 177774          ;;STACK LIMIT REGISTER
177772                          PIRQ=   177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
177570                          DSWR=   177570          ;;HARDWARE SWITCH REGISTER
177570                          DDISP=  177570          ;;HARDWARE DISPLAY REGISTER
177546                          LKS=    177546          ;;LINE CLOCK (KW11-L) STATUS REGISTER

                                ;*MISCELLANEOUS DEFINITIONS
000011                          HT=     11              ;;CODE FOR HORIZONTAL TAB
000012                          LF=     12              ;;CODE LINE FEED
000015                          CR=     15              ;;CODE CARRIAGE RETURN
000200                          CRLF=   200             ;;CODE FOR CARRIAGE RETURN-LINE FEED

                                ;*GENERAL PURPOSE REGISTER DEFINITIONS
000000                          R0=     %0              ;;GENERAL REGISTER
000001                          R1=     %1              ;;GENERAL REGISTER
000002                          R2=     %2              ;;GENERAL REGISTER
000003                          R3=     %3              ;;GENERAL REGISTER
000004                          R4=     %4              ;;GENERAL REGISTER
000005                          R5=     %5              ;;GENERAL REGISTER
000006                          R6=     %6              ;;GENERAL REGISTER
000007                          R7=     %7              ;;GENERAL REGISTER
                                .EQUIV  R0,R10          ;;GENERAL REGISTER
                                .EQUIV  R1,R11          ;;GENERAL REGISTER
                                .EQUIV  R2,R12          ;;GENERAL REGISTER
                                .EQUIV  R3,R13          ;;GENERAL REGISTER
                                .EQUIV  R4,R14          ;;GENERAL REGISTER
                                .EQUIV  R5,R15          ;;GENERAL REGISTER
                                .EQUIV  R6,SP           ;;STACK POINTER
                                .EQUIV  SP,KSP          ;;KERNEL STACK POINTER
                                .EQUIV  SP,SSP          ;;SUPERVISOR STACK POINTER
                                .EQUIV  SP,USP          ;;USER STACK POINTER
                                .EQUIV  R7,PC           ;;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
                                        .EQUIV  SW09,SW9
                                        .EQUIV  SW08,SW8
                                        .EQUIV  SW07,SW7
                                        .EQUIV  SW06,SW6
                                        .EQUIV  SW05,SW5
                                        .EQUIV  SW04,SW4
                                        .EQUIV  SW03,SW3
                                        .EQUIV  SW02,SW2
                                        .EQUIV  SW01,SW1
                                        .EQUIV  SW00,SW0

                                        ;*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
                                        .EQUIV  BIT09,BIT9
                                        .EQUIV  BIT08,BIT8
                                        .EQUIV  BIT07,BIT7
                                        .EQUIV  BIT06,BIT6
                                        .EQUIV  BIT05,BIT5
                                        .EQUIV  BIT04,BIT4
                                        .EQUIV  BIT03,BIT3
                                        .EQUIV  BIT02,BIT2
                                        .EQUIV  BIT01,BIT1
                                        .EQUIV  BIT00,BIT0

                                        ;*BASIC "CPU" TRAP VECTOR ADDRESSES
```

```
                        ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
                        RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
                        TBITVEC=14              ;;"T" BIT
                        TRTVEC= 14              ;;TRACE TRAP
                        BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
                        IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
                        PWRVEC= 24              ;;POWER FAIL
                        EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
                        TRAPVEC=34              ;;"TRAP" TRAP
                        TKVEC=  60              ;;TTY KEYBOARD VECTOR
                        TPVEC=  64              ;;TTY PRINTER VECTOR
                        LKVEC=  100             ;;LINE CLOCK (KW11-L) VECTER
                        CACHVEC=114             ;;CACHE ERROR INTERRUPT VECTOR
                        PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
                        MMVEC=  250             ;;MEMORY MANAGEMENT VECTOR

                        .SBTTL  CACHE   REGISTER DEFINITIONS

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

                        .SBTTL  CPU REGISTER DEFINITIONS

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



                        .SBTTL  MEMORY MANAGEMENT DEFINITIONS

                        ;*MEMORY MANAGEMENT STATUS REGISTER ADDRESSES

 177572                 MMR0=   177572
 177574                 MMR1=   177574
 177576                 MMR2=   177576
 172516                 MMR3=   172516
                        .EQUIV  MMR0,SR0
                        .EQUIV  MMR1,SR1
                        .EQUIV  MMR2,SR2
                        .EQUIV  MMR3,SR3

                        ;*USER "I" PAGE DESCRIPTOR REGISTERS
```

```
                                ;*USER "D" PAGE DESCRIPTOR REGISTORS

                                ;*USER "I" PAGE ADDRESS REGISTERS

                                ;*USER "D" PAGE ADDRESS REGISTERS

                                ;*SUPERVISOR "I" PAGE DESCRIPTOR REGISTERS

                                ;*SUPERVISOR "D" PAGE DESCRIPTOR REGISTERS
```

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

                                        ;*SUPERVISOR "I" PAGE ADDRESS REGISTERS

                        172240          SIPAR0= 172240
                        172242          SIPAR1= 172242
                        172244          SIPAR2= 172244
                        172246          SIPAR3= 172246
                        172250          SIPAR4= 172250
                        172252          SIPAR5= 172252
                        172254          SIPAR6= 172254
                        172256          SIPAR7= 172256

                                        ;*SUPERVISOR "D" PAGE ADDRESS REGISTERS

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

                                        ;*KERNEL "I" PAGE DESCRIPTOR REGISTERS

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

                                        ;*KERNEL "D" PAGE DESCRIPTOR REGISTERS

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

                                        ;*KERNEL "I" PAGE ADDRESS REGISTERS

                        172340          KIPAR0= 172340
```

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

                                        ;*KERNEL "D" PAGE ADDRESS REGISTERS

                        172360          KDPAR0= 172360
                        172362          KDPAR1= 172362
                        172364          KDPAR2= 172364
                        172366          KDPAR3= 172366
                        172370          KDPAR4= 172370
                        172372          KDPAR5= 172372
                        172374          KDPAR6= 172374
                        172376          KDPAR7= 172376



                                        .SBTTL  UNIBUS MAP REGISTER DEFINITIONS

                                        ;*THE LOWER 16 BITS OF THE MAP REGISTERS ARE LABELED 'MAPLXX'
                                        ;*THE UPPER 6 BITS OF THE MAP REGISTERS ARE LABELED 'MAPHXX'


                    170200              MAPL00 = 170200
                    170202              MAPH00 = 170202
                    170204              MAPL01 = 170204
                    170206              MAPH01 = 170206
                    170210              MAPL02 = 170210
                    170212              MAPH02 = 170212
                    170214              MAPL03 = 170214
                    170216              MAPH03 = 170216
                    170220              MAPL04 = 170220
                    170222              MAPH04 = 170222
                    170224              MAPL05 = 170224
                    170226              MAPH05 = 170226
                    170230              MAPL06 = 170230
                    170232              MAPH06 = 170232
                    170234              MAPL07 = 170234
                    170236              MAPH07 = 170236
                    170240              MAPL10 = 170240
                    170242              MAPH10 = 170242
                    170244              MAPL11 = 170244
                    170246              MAPH11 = 170246
                    170250              MAPL12 = 170250
                    170252              MAPH12 = 170252
                    170254              MAPL13 = 170254
                    170256              MAPH13 = 170256
                    170260              MAPL14 = 170260
                    170262              MAPH14 = 170262
                    170264              MAPL15 = 170264
```

```
                        MAPH15 = 170266
                        MAPL16 = 170270
                        MAPH16 = 170272
                        MAPL17 = 170274
                        MAPH17 = 170276
                        MAPL20 = 170300
                        MAPH20 = 170302
                        MAPL21 = 170304
                        MAPH21 = 170306
                        MAPL22 = 170310
                        MAPH22 = 170312
                        MAPL23 = 170314
                        MAPH23 = 170316
                        MAPL24 = 170320
                        MAPH24 = 170320
                        MAPL25 = 170324
                        MAPH25 = 170326
                        MAPL26 = 170330
                        MAPH26 = 170332
                        MAPL27 = 170334
                        MAPH27 = 170336
                        MAPL30 = 170340
                        MAPH30 = 170342
                        MAPL31 = 170344
                        MAPH31 = 170346
                        MAPL32 = 170350
                        MAPH32 = 170352
                        MAPL33 = 170354
                        MAPH33 = 170356
                        MAPL34 = 170360
                        MAPH34 = 170362
                        MAPL35 = 170364
                        MAPH35 = 170366
                        MAPL36 = 170370
                        MAPH36 = 170372
                        MAPL37 = 170374
                        MAPH37 = 170376
                        .EQUIV  MAPL00,MAPL0
                        .EQUIV  MAPH00,MAPH0
                        .EQUIV  MAPL01,MAPL1
                        .EQUIV  MAPH01,MAPH1
                        .EQUIV  MAPL02,MAPL2
                        .EQUIV  MAPH02,MAPH2
                        .EQUIV  MAPL03,MAPL3
                        .EQUIV  MAPH03,MAPH3
                        .EQUIV  MAPL04,MAPL4
                        .EQUIV  MAPH04,MAPH4
                        .EQUIV  MAPL05,MAPL5
                        .EQUIV  MAPH05,MAPH5
                        .EQUIV  MAPL06,MAPL6
                        .EQUIV  MAPH06,MAPH6
                        .EQUIV  MAPL07,MAPL7
                        .EQUIV  MAPH07,MAPH7
                        .SBTTL  FLOATING POINT REGISTER DEFINITIONS
000000                  AC0     =%0
000001                  AC1     =%1
```

```
                                ;AC2     =%2
                                ;AC3     =%3
                                ;AC4     =%4
                                ;AC5     =%5
                                ;AC6     =%6
                                ;AC7     =%7
                                ;MSN     =170004
                                ;COUNT   =$REG0
                                ;STEP    =$REG1
                                ;OFFSET  =$REG2

                                .SBTTL  FLOATING POINT STATUS BIT DEFINITIONS
                                .EQUIV  BIT0,FC
                                .EQUIV  BIT1,FV
                                .EQUIV  BIT2,FZ
                                .EQUIV  BIT3,FN
                                .EQUIV  BIT4,FMM
                                .EQUIV  BIT5,FT
                                .EQUIV  BIT6,FL
                                .EQUIV  BIT7,FD
                                .EQUIV  BIT8,FIC
                                .EQUIV  BIT9,FIV
                                .EQUIV  BIT10,FIU
                                .EQUIV  BIT11,FIUV
                                .EQUIV  BIT14,FID
                                .EQUIV  BIT15,FER

                                .SBTTL  FLOATING POINT EXCEPTION CODE DEFINITIONS
                                .EQUIV  BIT1,FOCE
                                .EQUIV  BIT2,FDZ
                                FICE=6
                                FO=10
                                FU=12
                                FUV=14
                                MT=16

                                .SBTTL  FLOATING POINT VECTOR DEFINITION
                                FPPVEC=244
                                LKVEC   =100
                                LKSTAT  =177546
                                PCLKST  =172540
                                COSETB  =172542
                                COUNTER =172544
                                PKVEC   =104

                                .SBTTL  TRAP CATCHER

                                        .=0
                                ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
                                ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
                                ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
                                        .=174
000174  000000                  DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
000176  000000                  SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER

                                .SBTTL  STARTING ADDRESS(ES)
```

```
                                                JMP     @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM

                                        .SBTTL  ACT11 HOOKS

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

```
                                .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  000000                          .WORD   0
001136  177570                  SWR:    .WORD   DSWR    ;;ADDRESS OF SWITCH REGISTER
001140  177570                  DISPLAY: .WORD  DDISP   ;;ADDRESS OF DISPLAY REGISTER
001142  177560                  $TKS:   177560          ;;TTY KBD STATUS
001144  177562                  $TKB:   177562          ;;TTY KBD BUFFER
001146  177564                  $TPS:   177564          ;;TTY PRINTER STATUS REG. ADDRESS
001150  177566                  $TPB:   177566          ;;TTY PRINTER BUFFER REG. ADDRESS
001152  000                     $NULL:  .BYTE   0       ;;CONTAINS NULL CHARACTER FOR FILLS
001153  002                     $FILLS: .BYTE   2       ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001154  012                     $FILLC: .BYTE   12      ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001155  000                     $TPFLG: .BYTE   0       ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
001156  000000                  $REGAD: .WORD   0       ;;CONTAINS THE ADDRESS FROM
                                                        ;;WHICH  ($REG0) WAS OBTAINED
001160  000000                  $REG0:  .WORD   0       ;;CONTAINS (($REGAD)+0)
001162  000000                  $REG1:  .WORD   0       ;;CONTAINS (($REGAD)+2)
001164  000000                  $REG2:  .WORD   0       ;;CONTAINS (($REGAD)+4)
001166  000000                  $REG3:  .WORD   0       ;;CONTAINS (($REGAD)+6)
001170  000000                  $REG4:  .WORD   0       ;;CONTAINS (($REGAD)+10)
001172  000000                  $REG5:  .WORD   0       ;;CONTAINS (($REGAD)+12)
001174  000000                  $REG6:  .WORD   0       ;;CONTAINS (($REGAD)+14)
001176  000000                  $REG7:  .WORD   0       ;;CONTAINS (($REGAD)+16)
001200  000000                  $TMP0:  .WORD   0       ;;USER DEFINED
001202  000000                  $TMP1:  .WORD   0       ;;USER DEFINED
001204  000000                  $TMP2:  .WORD   0       ;;USER DEFINED
001206  000000                  $TMP3:  .WORD   0       ;;USER DEFINED
001210  000000                  $TMP4:  .WORD   0       ;;USER DEFINED
001212  000000                  $TMP5:  .WORD   0       ;;USER DEFINED
001214  000000                  $TMP6:  .WORD   0       ;;USER DEFINED
001216  000000                  $TMP7:  .WORD   0       ;;USER DEFINED
001220  000000                  $TIMES: 0               ;;MAX. NUMBER OF ITERATIONS
001222  000000                  $ESCAPE:0               ;;ESCAPE ON ERROR ADDRESS
```

```
 2011  001224  177607  000377          $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
 2012  001230     077                  $QUES:  .ASCII  /?/             ;;QUESTION MARK
 2013  001231     015                  $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
 2014  001232  000012                  $LF:    .ASCIZ  <12>            ;;LINE FEED
```

```
                                        ;;*****************************************************************
                                        .SBTTL  APT MAILBOX-ETABLE
                                        ;;*****************************************************************
                                        .EVEN
001234                                  $MAIL:                          ;;APT MAILBOX
001234  000000                          $MSGTY: .WORD   AMSGTY          ;;MESSAGE TYPE CODE
001236  000000                          $FATAL: .WORD   AFATAL          ;;FATAL ERROR NUMBER
001240  000000                          $TESTN: .WORD   ATESTN          ;;TEST NUMBER
001242  000000                          $PASS:  .WORD   APASS           ;;PASS COUNT
001244  000000                          $DEVCT: .WORD   ADEVCT          ;;DEVICE COUNT
001246  000000                          $UNIT:  .WORD   AUNIT           ;;I/O UNIT NUMBER
001250  000000                          $MSGAD: .WORD   AMSGAD          ;;MESSAGE ADDRESS
001252  000000                          $MSGLG: .WORD   AMSGLG          ;;MESSAGE LENGTH
001254                                  $ETABLE:                        ;;APT ENVIRONMENT TABLE
001254  000                             $ENV:   .BYTE   AENV            ;;ENVIRONMENT BYTE
001255  000                             $ENVM:  .BYTE   AENVM           ;;ENVIRONMENT MODE BITS
001256  000000                          $SWREG: .WORD   ASWREG          ;;APT SWITCH REGISTER
001260  000000                          $USWR:  .WORD   AUSWR           ;;USER SWITCHES
001262  000000                          $CPUOP: .WORD   ACPUOP          ;;CPU TYPE,OPTIONS
                                        ;*                              BITS 15-11=CPU TYPE
                                        ;*                                      11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
                                        ;*                                      11/70=06,PDQ=07,Q=10
                                        ;*                              BIT 10=REAL TIME CLOCK
                                        ;*                              BIT  9=FLOATING POINT PROCESSOR
                                        ;*                              BIT  8=MEMORY MANAGEMENT
001264                                  $ETEND:
                                        .MEXIT
001264  000000                          $ERPSW: .WORD                   ;ERROR PSW
001266  000000                          $STSTNM:.WORD                   ;TEST NUMBER FOR TYPEOUT
001270  000000                          $FEC:   .WORD                   ;FLOATING EXCEPTION CODE
001272  000000                          $FEA:   .WORD                   ;FLOATING EXCEPTION ADDRESS
001274  000                             $FT:    .BYTE                   ;TRUNCATE/ROUND MODE
001275  000                             $FL:    .BYTE                   ;INTEGER/INTEGER LONG MODE
001276  000000                          $FD:    .WORD                   ;FLOATING/FLOATING DOUBLE MODE
001300  000004                          $FLT:   .BLKW   4               ;BUFFER FOR FLOATING CONVERSIONS
001310  000042                          $FLBUFF:.BLKB   42              ;BUFFER FOR ASCII FLOATING NUMBERS
001352  000005                          $BUFF:  .BLKW   5               ;BUFFER
001364  000000                          $FPS:   .WORD
001366  000000                          TICKS:  .WORD
001370  000030                          TIMES:  .WORD   30
001372  000000                          $CPUERR:.WORD
001374  052532                          ADRPTR: .WORD   ADRTBL
001376  054532                          OUTPTR: .WORD   OUTTBL
001400    367     366     377           MULTBL: .BYTE   367,366,377,365,367,376,377,364
001403    365     367     376
001406    377     364
001410    367     366     377                   .BYTE   367,366,377,375,367,376,377,363
001413    375     367     376
001416    377     363
001420    367     366     377                   .BYTE   367,366,377,365,367,376,377,374
001423    365     367     376
001426    377     374
001430    367     366     377                   .BYTE   367,366,377,375,367,376,377,362
001433    375     367     376
```

```
2071  001436  377  362
2072  001440  367  366  377          .BYTE   367,366,377,365,367,376,377,364
2073  001443  365  367  376
2074  001446  377  364
2075  001450  367  366  377          .BYTE   367,366,377,375,367,376,377,373
2076  001453  375  367  376
2077  001456  377  373
2078  001460  367  366  377          .BYTE   367,366,377,365,367,376,377,374
2079  001463  365  367  376
2080  001466  377  374
2081  001470  367  366  377          .BYTE   367,366,377,375,367,376,377,372
2082  001473  375  367  376
2083  001476  377  372
2084  001500  367  366  377  100$:   .BYTE   367,366,377,365,367,376,377,364
2085  001503  365  367  376
2086  001506  377  364
2087  001510  367  366  377          .BYTE   367,366,377,375,367,376,377,363
2088  001513  375  367  376
2089  001516  377  363
2090  001520  367  366  377          .BYTE   367,366,377,365,367,376,377,374
2091  001523  365  367  376
2092  001526  377  374
2093  001530  367  366  377          .BYTE   367,366,377,375,367,376,377,372
2094  001533  375  367  376
2095  001536  377  372
2096  001540  367  366  377          .BYTE   367,366,377,365,367,376,377,364
2097  001543  365  367  376
2098  001546  377  364
2099  001550  367  366  377          .BYTE   367,366,377,375,367,376,377,373
2100  001553  375  367  376
2101  001556  377  373
2102  001560  367  366  377          .BYTE   367,366,377,365,367,376,377,374
2103  001563  365  367  376
2104  001566  377  374
2105  001570  367  366  377          .BYTE   367,366,377,375,367,376,377,372
2106  001573  375  367  376
2107  001576  377  372
2108  001600  362  367  366  $200:   .BYTE   362,367,366,377,365,367,376,377
2109  001603  377  365  367
2110  001606  376  377
2111  001610  364  367  366          .BYTE   364,367,366,377,375,367,376,377
2112  001613  377  375  367
2113  001616  376  377
2114  001620  363  367  366          .BYTE   363,367,366,377,365,367,376,377
2115  001623  377  365  367
2116  001626  376  377
2117  001630  374  367  366          .BYTE   374,367,366,377,375,367,376,377
2118  001633  377  375  367
2119  001636  376  377
2120  001640  362  367  366          .BYTE   362,367,366,377,365,367,376,377
2121  001643  377  365  367
2122  001646  376  377
2123  001650  364  367  366          .BYTE   364,367,366,377,375,367,376,377
2124  001653  377  375  367
2125  001656  376  377
2126  001660  373  367  366          .BYTE   373,367,366,377,365,367,376,377
```

```
2127  001663  377  365  367
2128  001666  376  377
2129  001670  374  367  366          .BYTE   374,367,366,377,375,367,376,377
2130  001673  377  375  367
2131  001676  376  377
2132  001700  362  367  366  $300:   .BYTE   362,367,366,377,365,367,376,377
2133  001703  377  365  367
2134  001706  376  377
2135  001710  364  367  366          .BYTE   364,367,366,377,375,367,376,377
2136  001713  377  375  367
2137  001716  376  377
2138  001720  363  367  366          .BYTE   363,367,366,377,365,367,376,377
2139  001723  377  365  367
2140  001726  376  377
2141  001730  374  367  366          .BYTE   374,367,366,377,375,367,376,377
2142  001733  377  375  367
2143  001736  376  377
2144  001740  372  367  366          .BYTE   372,367,366,377,365,367,376,377
2145  001743  377  365  367
2146  001746  376  377
2147  001750  364  367  366          .BYTE   364,367,366,377,375,367,376,377
2148  001753  377  375  367
2149  001756  376  377
2150  001760  373  367  366          .BYTE   373,367,366,377,365,367,376,377
2151  001763  377  365  367
2152  001766  376  377
2153  001770  374  367  366          .BYTE   374,367,366,377,375,367,376,377
2154  001773  377  375  367
2155  001776  376  377
2156  002000  171006       ILLOP:    .WORD   171006          ;22
2157  002002  171406                 .WORD   171406          ;23
2158  002004  172006                 .WORD   172006          ;24
2159  002006  172406                 .WORD   172406          ;25
2160  002010  173006                 .WORD   173006          ;26
2161  002012  173406                 .WORD   173406          ;27
2162  002014  174006                 .WORD   174006          ;30
2163  002016  174406                 .WORD   174406          ;31
2164  002020  176006                 .WORD   176006          ;34
2165  002022  177406                 .WORD   177406          ;37
2166  002024  170006                 .WORD   170006          ;46
2167  002026  170016                 .WORD   170016          ;56
2168  002030  170017                 .WORD   170017          ;57
2169  002032  170406                 .WORD   170406          ;64
2170  002034  170506                 .WORD   170506          ;65
2171  002036  170606                 .WORD   170606          ;66
2172  002040  170706                 .WORD   170706          ;67
2173  002042  170010                 .WORD   170010          ;150
2174  002044  170013                 .WORD   170013          ;153
2175  002046  170014                 .WORD   170014          ;154
2176  002050  170015                 .WORD   170015          ;155
2177  002052  170026                 .WORD   170026          ;246
2178  002054  170027                 .WORD   170027          ;247
2179  002056  170036                 .WORD   170036          ;256
2180  002060  170037                 .WORD   170037          ;257
2181  002062  170040                 .WORD   170040          ;340
2182  002064  170041                 .WORD   170041          ;341
```

```
002066  170042                          .WORD   170042          ;342
002070  170043                          .WORD   170043          ;343
002072  170044                          .WORD   170044          ;344
002074  170045                          .WORD   170045          ;345
002076  170030                          .WORD   170030          ;350
002100  170031                          .WORD   170031          ;351
002102  170032                          .WORD   170032          ;352
002104  170033                          .WORD   170033          ;353
002106  170034                          .WORD   170034          ;354
002110  170035                          .WORD   170035          ;355
002112                          ILLEND:
002112  000024                  WALKBUF:.BLKW   24
002162  000000                  $ACO:   .WORD
002164  000000                          .WORD
002166  000000                          .WORD
002170  000000                          .WORD

                                .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

002172                          $ERRTB:
```

```
                                        ;ITEM 1
002172  037633                          EM1             ;DST CHANGED ON MULF*DST=0
002174  037665                          DH1             ;ERRPC      DATA        TEST NO
                                                        ;      EXPECT      ACTUAL
002176  037752                          DT1             ;$ERRPC,$TMP0 , $TMP2,$$STSNM
002200  037764                          DF1             ;0,1,1,0

                                        ;ITEM 2
002202  037771                          EM2             ;4F1 BRANCH FAILED
002204  037665                          DH1
002206  037752                          DT1
002210  037764                          DF1

                                        ;ITEM 3
002212  040104                          EM3             ;CONDITION CODES BAD
002214  040130                          DH3             ;ERRPC      FPS         TEST NO
                                                        ;     EXPECT      ACTUAL
002216  040200                          DT3             ;$ERRPC,$TMP0,$TMP1,$$TSTNM
002220  000000                          0

                                        ;ITEM 4
002222  040212                          EM4             ;ADX ROM FAILED
002224  040231                          DH4             ;ERRPC   TEST NO
002226  040250                          DT4             ;$ERRPC,$$TSTNM
002230  000000                          0

                                        ;ITEM 5
002232  040256                          EM5             ;FRHK MUL SWR STUCK LOW
002234  040231                          DH4
002236  040250                          DT4
002240  000000                          0

                                        ;ITEM 6
002242  040305                          EM6             ;FXPP SC00 NOT GETTING TO FRMA AS A HIGH
002244  040231                          DH4
002246  040250                          DT4
002250  000000                          0

                                        ;ITEM 7
002252  040355                          EM7             ;FRME SHFC 0(0)H NOT GOING H WHEN 1(0) IS LOW
002254  037665                          DH1
002256  040436                          DT7             ;$ERRPC,$TMP0,$TMP4,$$TSTNM
002260  037764                          DF1

                                        ;ITEM 10
002262  040450                          EM10            ;QR DATA BAD, CHECK MUL SHIFT ENCODER
                                                        ;ROMOUT IN STATE 233
002264  037665                          DH1
002266  040622                          DT10            ;$ERRPC,$TMP0,$TMP4,$$TSTNM
002270  037764                          DF1

                                        ;ITEM 11
002272  040634                          EM11            ;FRHL MYP SIGN EXTEND NOT GOING LOW
002274  037665                          DH1
002276  040436                          DT7
```

```
2273  002300  037764                    DF1
2274
2275                                    ;ITEM 12
2276  002302  040677                    EM12            ;FRHL MPY/DIV ADD(1)L NOT GOING LOW, OR
2277  002304  037665                    DH1             ;NOT CAUSING FRMH FORCE MPY/DIV ADD
2278  002306  040436                    DT7             ;TO FORCE THE ADD
2279  002310  037764                    DF1
2280
2281                                    ;ITEM 13
2282  002312  041031                    EM13            ;AR DATA BAD
2283  002314  037665                    DH1
2284  002316  041046                    DT13            ;$ERRPC,$TMP2,$TMP6,$$TSTNM
2285  002320  037764                    DF1
2286
2287                                    ;ITEM 14
2288  002322  041060                    EM14            ;MUL SHF ENCODER ROMOUT BAD
2289  002324  041115                    DH14            ;ERRPC   ROMOUT  TEST NO
2290  002326  041142                    DT14            ;$ERRPC,$REG5,$$TSTNM
2291  002330  000000                    0
2292
2293                                    ;ITEM 15
2294  002332  041152                    EM15            ;FRHK MPY/DIV ADD(1)L NOT GOING HIGH
2295  002334  037665                    DH1
2296  002336  041046                    DT13
2297  002340  037764                    DF1
2298
2299                                    ;ITEM 16
2300  002342  041266                    EM16            ;FRHK MUL SWR NOT GOING LOW
2301  002344  040231                    DH4
2302  002346  040250                    DT4
2303  002350  000000                    0
2304
2305                                    ;ITEM 17
2306  002352  041407                    EM17            ;FRHK MPY SIGN EXTEND NOT GOING HIGH
2307  002354  037665                    DH1
2308  002356  041046                    DT13
2309  002360  037764                    DF1
2310
2311                                    ;ITEM 20
2312  002362  041453                    EM20            ;FRHK E115(4) NOT GOING LOW
2313  002364  040231                    DH4
2314  002366  040250                    DT4
2315  002370  000000                    0
2316
2317                                    ;ITEM 21
2318  002372  041506                    EM21            ;FRHK E115(4) NOT GOING HIGH
2319  002374  040231                    DH4
2320  002376  040250                    DT4
2321  002400  000000                    0
2322
2323                                    ;ITEM 22
2324  002402  041542                    EM22            ;FRHK E107(12) NOT GOING LOW
2325  002404  040231                    DH4
2326  002406  040250                    DT4
2327  002410  000000                    0
2328
```

```
                                        ;ITEM 23
002412  041576                          EM23            ;FRHK MUL SWR DID NOT ENABLE
002414  037665                          DH1             ;FRMH FORCE A
002416  041046                          DT13
002420  037764                          DF1

                                        ;ITEM 24
002422  041647                          EM24            ;RESULT WRONG
002424  037665                          DH1
002426  037752                          DT1
002430  037764                          DF1

                                        ;ITEM 25
002432  041664                          EM25            ;FRHK + FRLH MUL SHIFT ENCODER
002434  040231                          DH4             ;ROM FAILED
002436  040250                          DT4
002440  000000                          0

                                        ;ITEM 26
002442  042001                          EM26            ;FRHK MUL SWR NOT GOING HIGH
002444  042067                          DH26            ;                       DATA
                                                        ;       EXPECT                          ACTUAL
002446  042134                          DT26            ;$TMP0,$TMP4
002450  042142                          DF26            ;2,2

                                        ;ITEM 27
002452  042144                          EM27            ;FRLH E1(11) NOT GOING LOW
002454  042067                          DH26
002456  042134                          DT26
002460  042142                          DF26

                                        ;ITEM 30
002462  042216                          EM30            ;FRHK E107 NOT GOING LOW
002464  042067                          DH26
002466  042134                          DT26
002470  042142                          DF26

                                        ;ITEM 31
002472  042274                          EM31            ;FRLH E1(11) NOT GOING HIGH WITH
002474  042067                          DH26            ;BOTH INPUTS ON A HIGH
002476  042134                          DT26
002500  042142                          DF26

                                        ;ITEM 32
002502  042363                          EM32            ;FRHK E107(6) NOT GOING LOW
002504  042067                          DH26            ;WITH H,L,H INPUT
002506  042134                          DT26
002510  042142                          DF26

                                        ;ITEM 33
002512  042437                          EM33            ;FRHK E107(12) NOT GOING LOW
002514  042067                          DH26            ;WITH H,H,L INPUT.
002516  042134                          DT26
002520  042142                          DF26

                                        ;ITEM 34
```

```
2385  002522  042514                EM34                    ;FXPC FIR808 NOT GETTING TO FRMA AS A HIGH
2386  002524  037665                DH1
2387  002526  037752                DT1
2388  002530  037764                DF1
2389
2390                                ;ITEM 35
2391  002532  042627                EM35                    ;STATE 305 DID NOT CLEAR DST+1
2392  002534  037665                DH1
2393  002536  037752                DT1
2394  002540  037764                DF1
2395
2396                                ;ITEM 36
2397  002542  042665                EM36                    ;FRACTION IS WRONG ON MODF*INT=0
2398  002544  037665                DH1
2399  002546  037752                DT1
2400  002550  037764                DF1
2401
2402                                ;ITEM 37
2403  002552  042725                EM37                    ;STATE 266 DID NOT SET BN
2404  002554  037665                DH1
2405  002556  037752                DT1
2406  002560  037764                DF1
2407
2408                                ;ITEM 40
2409  002562  042756                EM40                    ;FRAC ACC DID NOT CLEAR IN STATE 341
2410  002564  037665                DH1
2411  002566  037752                DT1
2412  002570  037764                DF1
2413
2414                                ;ITEM 41
2415  002572  043026                EM41                    ;INTEGER PORTION WRONG IN STATE  7
2416  002574  037665                DH1
2417  002576  037752                DT1
2418  002600  037764                DF1
2419
2420                                ;ITEM 42
2421  002602  043067                EM42                    ;FRACTION WRONG ON MODF
2422  002604  037665                DH1
2423  002606  037752                DT1
2424  002610  037764                DF1
2425
2426                                ;ITEM 43
2427  002612  043115                EM43                    ;INTEGER WRONG ON MODF
2428  002614  037665                DH1
2429  002616  037752                DT1
2430  002620  037764                DF1
2431
2432                                ;ITEM 44
2433  002622  043142                EM44                    ;INTEGER DID NOT CLEAR ON OVERFLOW
2434  002624  037665                DH1
2435  002626  037752                DT1
2436  002630  037764                DF1
2437
2438                                ;ITEM 45
2439  002632  043204                EM45                    ;STATE 105 NOT DETECTING ZERO FRACTION
2440  002634  040231                DH4
```

```
002636  040250                  DT4
002640  000000                  0
                                ;ITEM 46
002642  043252                  EM46            ;LOGICAL "AND" FAILED ON FRACTION
002644  042067                  DH26
002646  042134                  DT26
002650  042142                  DF26
                                ;ITEM 47
002652  043313                  EM47            ;LOGICAL "AND" FAILED ON INTEGER
002654  042067                  DH26
002656  043354                  DT47            ;$REG4,$TMP4
002660  042142                  DF26
                                ;ITEM 50
002662  043362                  EM50            ;DIVF*DST=0 DID NOT CLR DST
002664  037665                  DH1
002666  037752                  DT1
002670  037764                  DF1
                                ;ITEM 51
002672  043415                  EM51            ;FRHC FALU59 NOT GETTING TO FRLF AS A HIGH
002674  037665                  DH1
002676  037752                  DT1
002700  037764                  DF1
                                ;ITEM 52
002702  043467                  EM52            ;FRHK DIV DONE STUCK LOW OR FRHC FALU59
002704  037665                  DH1             ;NOT GETTING TO FRHL AS A HIGH
002706  040436                  DT7
002710  037764                  DF1
                                ;ITEM 53
002712  043574                  EM53            ;QR DATA BAD
002714  037665                  DH1
002716  037752                  DT1
002720  037764                  DF1
                                ;ITEM 54
002722  043610                  EM54            ;FRLH HI QUOT GEN BIT DID NOT GO LOW
002724  037665                  DH1
002726  040436                  DT7
002730  037764                  DF1
                                ;ITEM 55
002732  043654                  EM55            ;FRHC FALU59 NOT GETTING TO FRLF AS A LOW
002734  037665                  DH1
002736  040436                  DT7
002740  037764                  DF1
                                ;ITEM 56
002742  043725                  EM56            ;FRHC FALU59 NOT GETTING TO FRHL DIV
002744  037665                  DH1             ;NORM MUX AS A HIGH
002746  040436                  DT7
002750  037764                  DF1
```

```
2497
2498                                            ;ITEM 57
2499  002752  043574                           EM53
2500  002754  037665                           DH1
2501  002756  040436                           DT7
2502  002760  037764                           DF1
2503
2504                                            ;ITEM 60
2505  002762  044014                           EM60                ;FRHL MPY/DIV ADD(1)L DID NOT GO LOW
2506  002764  037665                           DH1                 ;WITH FALU59L ON A LOW
2507  002766  041046                           DT13
2508  002770  037764                           DF1
2509
2510                                            ;ITEM 61
2511  002772  000000                           0                   ;DELETED
2512  002774  037665                           DH1
2513  002776  041046                           DT13
2514  003000  037764                           DF1
2515
2516                                            ;ITEM 62
2517  003002  044107                           EM62                ;FRLF HI QUOT GEN BIT NOT GOING HIGH
2518  003004  037665                           DH1
2519  003006  040436                           DT7
2520  003010  037764                           DF1
2521
2522                                            ;ITEM 63
2523  003012  044153                           EM63                ;FRHL MPY/DIV ADD(1)L NOT GOING HIGH
2524  003014  037665                           DH1
2525  003016  041046                           DT13
2526  003020  037764                           DF1
2527
2528                                            ;ITEM 64
2529  003022  044247                           EM64                ;FRHK NORM NEG ENCODER FAILED
2530  003024  044304                           DH64
2531  003026  044364                           DT64
2532  003030  044374                           DF64
2533
2534                                            ;ITEM 65
2535  003032  044400                           EM65                ;RESULT WRONG ON DIV
2536  003034  037665                           DH1
2537  003036  037752                           DT1
2538  003040  037764                           DF1
2539
2540                                            ;ITEM 66
2541  003042  044424                           EM66                ;NORM SHIFT COUNT ROM FAILED
2542  003044  044474                           DH66                ;DATA NORM ROM
2543                                                                ;EXPECT ACTUAL ADDRESS ROMOUT
2544  003046  044574                           DT66                ;$TMP0,$TMP2,$REG5,$REG6
2545  003050  044374                           DF64
2546
2547                                            ;ITEM 67
2548  003052  044606                           EM67                ;FRLF DLE PREC QT LO NOT GOING LOW
2549  003054  042067                           DH26
2550  003056  042134                           DT26
2551  003060  042142                           DF26
2552
```

```
2553                                            ;ITEM 70
2554  003062  044651                           EM70                    ;FRLH DLE PREC QT HI NOT GOING HIGH
2555  003064  042067                           DH26
2556  003066  042134                           DT26
2557  003070  042142                           DF26
2558
2559                                            ;ITEM 71
2560  003072  044721                           EM71                    ;FRLF E21(12) NOT GOING HIGH
2561  003074  042067                           DH26
2562  003076  042134                           DT26
2563  003100  042142                           DF26
2564
2565                                            ;ITEM 72
2566  003102  044755                           EM72                    ;FRLH DLE PREC QT HI NOT GOING LOW
2567  003104  042067                           DH26
2568  003106  042134                           DT26
2569  003110  042142                           DF26
2570
2571                                            ;ITEM 73
2572  003112  045017                           EM73                    ;FRLH DLE PREC QT LO NOT GOING HI
2573  003114  042067                           DH26
2574  003116  042134                           DT26
2575  003120  042142                           DF26
2576
2577                                            ;ITEM 74
2578  003122  043574                           EM53
2579  003124  042067                           DH26
2580  003126  042134                           DT26
2581  003130  042142                           DF26
2582
2583                                            ;ITEM 75
2584  003132  045061                           EM75                    ;FRLH QSI-1 DID NOT GO LOW WHEN
2585  003134  042067                           DH26                    ;SELECTING C1 INPUT
2586  003136  042134                           DT26
2587  003140  042142                           DF26
2588
2589                                            ;ITEM 76
2590  003142  045143                           EM76                    ;FRLJ QSI-1 DID NOT GO LOW WHEN
2591  003144  042067                           DH26                    ;SELECTING A1 INPUT
2592  003146  042134                           DT26
2593  003150  042142                           DF26
2594
2595                                            ;ITEM 77
2596  003152  045225                           EM77                    ;FRLH QSI-1 DID NOT GO HIGH WHEN
2597  003154  042067                           DH26                    ;SELECTING A1 INPUT
2598  003156  042134                           DT26
2599  003160  042142                           DF26
2600
2601                                            ;ITEM 100
2602  003162  045310                           EM100                   ;STATE 246 FAILED
2603  003164  042067                           DH26
2604  003166  042134                           DT26
2605  003170  042142                           DF26
2606
2607                                            ;ITEM 101
2608  003172  045335                           EM101                   ;FEC WRONG ON STST
```

```
2609  003174  045401                  DH101
2610  003176  045450                  DT101
2611  003200  000000                  0
2612
2613                                  ;ITEM 102
2614  003202  045357                  EM102                   ;FEA WRONG ON STST
2615  003204  045462                  DH102
2616  003206  045532                  DT102
2617  003210  000000                  0
2618
2619                                  ;ITEM 103
2620  003212  045544                  EM103                   ;FMO TRAP FAILED-CHECK FRMB FMO
2621  003214  040231                  DH4
2622  003216  040250                  DT4
2623  003220  000000                  0
2624
2625                                  ;ITEM 104
2626  003222  045603                  EM104                   ;FEC WRONG ON FMO TRAP
2627  003224  045401                  DH101
2628  003226  045450                  DT101
2629  003230  000000                  0
2630
2631                                  ;ITEM 105
2632  003232  045653                  EM105                   ;FEA WRONG ON FMO TRAP
2633  003234  045462                  DH102
2634  003236  045532                  DT102
2635  003240  000000                  0
2636
2637                                  ;ITEM 106
2638  003242  045702                  EM106                   ;FMO FAILED-CHECK APPROPRIATE ROM STATE
2639  003244  040231                  DH4
2640  003246  040250                  DT4
2641  003250  000000                  0
2642
2643                                  ;ITEM 107
2644  003252  045751                  EM107                   ;UNDERFLOW DID NOT TRAP-CHECK
2645  003254  040231                  DH4                     ;FRMA FIU(0)
2646  003256  040250                  DT4
2647  003260  000000                  0
2648
2649                                  ;ITEM 110
2650  003262  046022                  EM110                   ;FEC WRONG ON UNDERFLOW
2651  003264  045401                  DH101
2652  003266  045450                  DT101
2653  003270  000000                  0
2654
2655                                  ;ITEM 111
2656  003272  046075                  EM111                   ;FEA WRONG ON UNDERFLOW
2657  003274  045462                  DH102
2658  003276  045532                  DT102
2659  003300  000000                  0
2660
2661                                  ;ITEM 112
2662  003302  046124                  EM112                   ;EXPONENT WRONG ON UNDERFLOW
2663  003304  037665                  DH1
2664  003306  037752                  DT1
```

```
2665  003310  037764                          DF1
2666
2667                                          ;ITEM 113
2668  003312  046160                          EM113                   ;OVERFLOW DID NOT TRAP-CHECK
2669  003314  040231                          DH4                     ;FRLN FVINT GETTING TO FRMA
2670  003316  040250                          DT4
2671  003320  000000                          0
2672
2673                                          ;ITEM 114
2674  003322  046247                          EM114                   ;EXPONENT WRONG ON OVERFLOW
2675  003324  037665                          DH1
2676  003326  037752                          DT1
2677  003330  037764                          DF1
2678
2679                                          ;ITEM 115
2680  003332  046302                          EM115                   ;FEC WRONG ON OVERFLOW-CHECK
2681  003334  045401                          DH101                   ;STATE 171
2682  003336  045450                          DT101
2683  003340  000000                          0
2684
2685                                          ;ITEM 116
2686  003342  046350                          EM116                   ;FEA WRONG ON OVERFLOW
2687  003344  045462                          DH102
2688  003346  045532                          DT102
2689  003350  000000                          0
2690
2691                                          ;ITEM 117
2692  003352  046376                          EM117                   ;FEC WRONG ON OVERFLOW-CHECK
2693  003354  045401                          DH101                   ;STATE 105
2694  003356  045450                          DT101
2695  003360  000000                          0
2696
2697                                          ;ITEM 120
2698  003362  046444                          EM120                   ;CONVERSION ERROR DID NOT TRAP
2699  003364  040231                          DH4                     ;CHECK FRLN FCINT GETTING TO FRMA
2700  003366  040250                          DT4
2701  003370  000000                          0
2702
2703                                          ;ITEM 121
2704  003372  046543                          EM121                   ;FEC WRONG ON CONVERSION ERROR
2705  003374  045401                          DH101
2706  003376  045450                          DT101
2707  003400  000000                          0
2708
2709                                          ;ITEM 122
2710  003402  046601                          EM122                   ;FEA WRONG ON CONVERSION ERROR
2711  003404  045462                          DH102
2712  003406  045532                          DT102
2713  003410  000000                          0
2714
2715                                          ;ITEM 123
2716  003412  046637                          EM123                   ;ILLEGAL OP CODE DID NOT TRAP
2717  003414  046715                          DH123                   ;ERRPC OPCODE TEST NO
2718  003416  046742                          DT123                   ;$ERRPC, $TMPO, $$TSTNM
2719  003420  000000                          0
2720
```

```
2721                                            ;ITEM 124
2722  003422  046752                            EM124                   ;FEC WRONG ON ILLEGAL OP CODE
2723  003424  045401                            DH101
2724  003426  045450                            DT101
2725  003430  000000                            0
2726
2727                                            ;ITEM 125
2728  003432  047007                            EM125                   ;FEA WRONG ON ILLEGAL OP CODE
2729  003434  045462                            DH102
2730  003436  045532                            DT102
2731  003440  000000                            0
2732
2733                                            ;ITEM 126
2734  003442  047044                            EM126                   ;LEGAL OP CODE FAILED
2735  003444  047104                            DH126
2736  003446  040200                            DT3
2737  003450  000000                            0
2738
2739                                            ;ITEM 127
2740  003452  047147                            EM127                   ;LEGAL OP-CODE TRAPPED
2741  003454  047224                            DH127
2742  003456  046742                            DT123
2743  003460  000000                            0
2744
2745                                            ;ITEM 130
2746  003462  047251                            EM130                   ;STATE 32 DID NOT LOAD FEC
2747  003464  045401                            DH101
2748  003466  045450                            DT101
2749  003470  000000                            0
2750
2751                                            ;ITEM 131
2752  003472  047307                            EM131                   ;TMCA HONOR PIRT DID NOT GO HIGH
2753  003474  040231                            DH4                     ;WITH FP TRAP PRESENT
2754  003476  040250                            DT4
2755  003500  000000                            0
2756
2757                                            ;ITEM 132
2758  003502  047374                            EM132                   ;TMCA HONOR FPTRAP DID NOT GO
2759  003504  040231                            DH4                     ;HIGH WITH SL YELLOW PRESENT
2760  003506  040250                            DT4
2761  003510  000000                            0
2762
2763                                            ;ITEM 133
2764  003512  047466                            EM133                   ;SAPK I SPACEA (0) NOT GOING LOW OR
2765  003514  040231                            DH4                     ;M8138-YA NOT INSTALLED
2766  003516  040250                            DT4
2767  003520  000000                            0
2768
2769                                            ;ITEM 134
2770  003522  047570                            EM134                   ;FEC WRONG IN STATE 356
2771  003524  045401                            DH101
2772  003526  045450                            DT101
2773  003530  000000                            0
2774
2775                                            ;ITEM 135
2776  003532  047635                            EM135                   ;FEA WRONG ON DIV BY ZERO
```

```
2777  003534  045462                  DH102
2778  003536  045532                  DT102
2779  003540  000000                  0
2780
2781                                  ;ITEM 136
2782  003542  047666                  EM136                   ;FEC WRONG IN STATE 346
2783  003544  045401                  DH101
2784  003546  045450                  DT101
2785  003550  000000                  0
2786
2787                                  ;ITEM 137
2788  003552  047733                  EM137                   ;TMCA HONOR SEGT NOT DISABLING
2789  003554  040231                  DH4                     ;FP TRAP
2790  003556  040250                  DT4
2791  003560  000000                  0
2792
2793                                  ;ITEM 140
2794  003562  050001                  EM140                   ;NEGF AND FMO FAILED
2795  003564  050025                  DH140                   ;ERRPC DATA TEST NO
2796  003566  050076                  DT140                   ;$ERRPC, $TMPO, $REGO, $$TSTNM
2797  003570  000000                  0
2798
2799                                  ;ITEM 141
2800  003572  050110                  EM141                   ;FRMB IMM * REG WT NOT GOING LOW
2801  003574  040231                  DH4
2802  003576  040250                  DT4
2803  003600  000000                  0
2804
2805                                  ;ITEM 142
2806  003602  050154                  EM142                   ;THE SP IS WRONG
2807  003604  050174                  DH142
2808  003606  040200                  DT3
2809  003610  000000                  0
2810
2811                                  ;ITEM 143
2812  003612  050243                  EM143                   ;YELLOW ZONE AND FP TRAP DID NOT OCCUR
2813  003614  040231                  DH4                     ;IN CORRECT SEQUENCE
2814  003616  040250                  DT4
2815  003620  000000                  0
2816
2817                                  ;ITEM 144
2818  003622  050335                  EM144                   ;MGMT AND FP TRAP DID NOT OCCUR
2819  003624  040231                  DH4                     ;IN CORRECT SEQUENCE.
2820  003626  040250                  DT4
2821  003630  000000                  0
2822
2823                                  ;ITEM 145
2824  003632  050376                  EM145                   ;BIT STUCK IN FPA OR FEA
2825  003634  045462                  DH102
2826  003636  045532                  DT102
2827  003640  000000                  0
2828
2829                                  ;ITEM 146
2830  003642  050426                  EM146                   ;MODE 1 DID NOT CAUSE FXPB IMMEDITATE
2831  003644  037665                  DH1                     ;TO GO TRUE
2832  003646  037752                  DT1
```

```
2833  003650  037764                          DF1
2834
2835                                          ;ITEM 147
2836  003652  050505                          EM147                   ;MODE 4 DID NOT CAUSE FXPB
2837  003654  037665                          DH1                     ;IMMEDIATE TO GO TRUE
2838  003656  037752                          DT1
2839  003660  037764                          DF1
2840
2841                                          ;ITEM 150
2842  003662  050564                          EM150                   ;PDRD PS14 NOT DISABLING FXOPN
2843  003664  040130                          DH3                     ;FMM(1) H
2844  003666  040200                          DT3
2845  003670  000000                          0
2846
2847                                          ;ITEM 151
2848  003672  050631                          EM151                   ;FXPN ACS0 DID NOT GO HIGH
2849  003674  047104                          DH126
2850  003676  040200                          DT3
2851  003700  000000                          0
2852
2853                                          ;ITEM 152
2854  003702  050714                          EM152                   ;FXPN ACS1 DID NOT GO HIGH
2855  003704  047104                          DH126
2856  003706  040200                          DT3
2857  003710  000000                          0
2858
2859                                          ;ITEM 153
2860  003712  050777                          EM153                   ;SCRATCH PAD CHIP FAILED
2861  003714  051027                          DH153
2862  003716  051104                          DT153
2863  003720  042142                          DF26
2864
2865                                          ;ITEM 154
2866  003722  051114                          EM154           ;UNEXPECTED TRAP TO 4
2867  003724  051141                          DH154           ;ERRPC  PCOFTRP ERRREG  TEST NO
2868  003726  051176                          DT154           ;$ERRPC,$TMP0,$TMP1,$$TSTNM
2869  003730  000000                          0
2870
2871                                          ;ITEM 155
2872  003732  051114                          EM154           ;UNEXPECTED TRAP TO 4 (11/45)
2873  003734  051210                          DH155
2874  003736  051236                          DT155
2875  003740  000000                          0
2876
2877                                          ;ITEM 156
2878  003742  051246                          EM156           ;UNEXPECTED TRAP TO 114
2879  003744  051275                          DH156           ;ERRPC  PCOFTRP  ERRREG  LOADRS  HIADRS  TEST NO
2880  003746  051350                          DT156           ;$ERRPC,$TMP0,$TMP1,$TMP2,$TMP3,$$TSTNM
2881  003750  000000                          0
2882
2883                                          ;ITEM 157
2884  003752  051246                          EM156           ;UNEXPECTED TRAP TO 114 (11/45)
2885  003754  051210                          DH155
2886  003756  051236                          DT155
2887  003760  000000                          0
2888
```

```
                                        ;ITEM 160
003762  051366                          EM160           ;UNEXPECTED TRAP TO 244
003764  051415                          DH160           ;ERRPC  PCOFTRP   FEC     FEA   TEST NO
003766  051460                          DT160           ;$ERRPC,$TMP0,$TMP1,$TMP2,$$TSTNM
003770  000000                          0

                                        ;ITEM 161
003772  041647                          EM24
003774  051474                          DH161
003776  051554                          DT161
004000  000000                          0

                                        ;ITEM 162
004002  041647                          EM24
004004  051572                          DH162
004006  040200                          DT3
004010  000000                          0

                                        ;ITEM 163
004012  040212                          EM4
004014  051210                          DH155
004016  051236                          DT155
004020  000000                          0

                                .SBTTL  APT PARAMETER BLOCK

                                ;;**********************************************************************
                                ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                ;;**********************************************************************
        004022                          .$X=.   ;;SAVE CURRENT LOCATION
        000024                          .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
000024  000200                          200     ;;FOR APT START UP
        000044                          .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
000044  004022                          $APTHDR ;;POINT TO APT HEADER BLOCK
        004022                          .=.$X   ;;RESET LOCATION COUNTER
                                ;;**********************************************************************
                                ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                                ;INTERFACE SPEC.

004022                          $APTHD:
004022  000000                  $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
004024  001234                  $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
004026  000035                  $TSTM:  .WORD   35      ;;RUN TIM OF LONGEST TEST
004030  000036                  $PASTM: .WORD   36      ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
004032  000000                  $UNITM: .WORD           ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
004034  000014                          .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
004036                          START:
                                ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
004036  012706  001100                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
004042  005026                  1$:     CLR     (R6)+           ;;CLEAR MEMORY LOCATION
004044  022706  001126                  CMP     #$BDDAT,R6      ;;DONE?
004050  001374                          BNE     1$-6            ;;LOOP BACK IF NO
004052  012706  001100                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                ;;INITIALIZE A FEW VECTORS
004056  012737  033122  000020          MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
004064  012737  000340  000022          MOV     #340,@#IOTVEC+2 ;;LEVEL 7
```

```
 2945  004072  012737  033442  000030          MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
 2946  004100  012737  000340  000032          MOV     #340,@#EMTVEC+2 ;;LEVEL 7
 2947  004106  012737  037320  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
 2948  004114  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
 2949  004122  012737  037364  000024          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
 2950  004130  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
 2951  004136  013737  032746  032740          MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
 2952  004144  005037  001220                  CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
 2953  004150  005037  001222                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
 2954  004154  112737  000001  001115          MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
 2955  004162  012737  004162  001106          MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
 2956  004170  012737  004170  001110          MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
 2957                                  ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
 2958                                  ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
 2959  004176  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
 2960  004202  012737  004240  000004          MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
 2961  004210  012737  177570  001136          MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
 2962  004216  012737  177570  001140          MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
 2963  004224  022777  177777  174704          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
 2964  004232  001013                          BNE     65$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
 2965                                                                  ;;AND  THE HARDWARE SWR IS NOT = -1
 2966  004234  005737  000001                  TST     @#1             ;;FORCE A TRAP THROUGH ERRVEC
 2967  004240  012737  000176  001136  64$:     MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
 2968  004246  012737  000174  001140          MOV     #DISPREG,DISPLAY ;;POINT TO SOFTWARE DISPLAY REG
 2969  004254  012716  004262                  MOV     #65$,(SP)       ;;REPLACE OLD PC WITH NEW
 2970  004260  000006                          RTT                     ;;RESTORE PC AND PSW
 2971  004262  012637  000004          65$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
 2972  004266  005037  001242                  CLR     $PASS           ;;CLEAR PASS COUNT
 2973  004272  132737  000200  001255          BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
 2974  004300  001403                          BEQ     3$              ;;YES,USE NON-APT SWITCH
 2975  004302  012737  001256  001136          MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
 2976  004310                          3$:
 2977                                  ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
 2978  004310  005227  177777                  INC     #-1             ;;FIRST TIME?
 2979  004314  001050                          BNE     66$             ;;BRANCH IF NO
 2980  004316  022737  033102  000042          CMP     #$ENDAD,@#42    ;;ACT-11?
 2981  004324  001444                          BEQ     66$             ;;BRANCH IF YES
 2982  004326  104400  004334                  TYPE    ,67$            ;;TYPE ASCIZ STRING
 2983  004332  000441                          BR      66$             ;;GET OVER THE ASCIZ
 2984                                  ;;67$:  .ASCIZ  <CRLF>"MAINDEC-11-DEFPB-A...PDP11-45/55/70...FP11-C DIAGNOSTIC PART 2"<C
 2985  004436                          66$:
 2986  004436  012737  004456  000004          MOV     #LOOP,ERRVEC    ;SETUP ERRVEC FOR TIMEOUT
 2987  004444  005737  177766                  TST     CPUERR          ;11/45 OR 11/70
 2988  004450  012737  177766  001372          MOV     #CPUERR,$CPUERR ;DO THIS IF 11/70
 2989  004456  012737  037276  000244  LOOP:   MOV     #FPSPUR,@#FPPVEC;SET VECTOR TO SPURIOUS HANDLER
 2990  004464  012737  000340  000246          MOV     #PR7,@#FPPVEC+2 ;SET PRIORITY 7 IN VECTOR PSW
 2991  004472  012737  037156  000004          MOV     #CPSPUR,@#ERRVEC;SET TRAP TO 4 VECTOR
 2992  004500  012737  000340  000006          MOV     #PR7,@#ERRVEC+2 ;SET TRAP TO 4 PSW
 2993  004506  012737  037222  000114          MOV     #CACHSPU,@#CACHVEC;SET PARITY ERROR VECTOR
 2994  004514  012737  000340  000116          MOV     #PR7,@#CACHVEC+2
 2995
 2996
 2997
 2998                                  ;;*****************************************************************
 2999                                  ;*TEST 1         MULF*MO*(DST=0)*-(SRC=0)
 3000                                  ;*
```

```
                                        ;*      THE A OR NO-MEM ROMS COULD FAIL.  THE ONLY OTHER POSSIBLE
                                        ;*      FAILURES WOULD BE ROM STATE 350 OR THE 4F1 BRANCH.
                                        ;*
                                        ;*      IF THE 4F1 BRANCH FAILS THE ACD+1 WILL ALSO BE CLEARED.
                                        ;*
                                        ;*      FPU ROM FLOW - 30, 61, 350, 351
                                        ;;*****************************************************************
004522  000004                    TST1:   SCOPE
004524  012737  000001  001240            MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
004532  012737  004652  001222            MOV     #TST2,$ESCAPE   ;;ESCAPE TO TEST 2 ON ERROR
004540  170400                            CLRF    AC0             ;CLEAR THE DST
004542  172527  040000                    LDF     #^O040000,AC1   ;ENSURE DST+1 NON ZERO
004546  172601                            LDF     AC1,AC2         ;LOAD THE SRC NON-ZERO
004550  170127  000013                    LDFPS   #13             ;LOAD COMPLIMENT CC'S
004554  171002                    1$:     MULF    AC2,AC0         ;EXECUTE INSTRUCTION UNDER TEST
004556  170237  001202                    STFPS   $TMP1           ;GET CC'S BACK
004562  170500                            TSTF    AC0             ;DST REMAIN CLEAR?
004564  170000                            CFCC
004566  001405                            BEQ     2$              ;BRANCH IF YES
004570  174037  001204                    STF     AC0,$TMP2       ;SAVE RECEIVED DATA
004574  170437  001200                    CLRF    $TMP0           ;SAVE EXPECTED DATA
004600  104001                            ERROR   1               ;DATA BAD
004602  173527  000000            2$:     CMPF    #0,AC1          ;ACD+1 CLEAR?
004606  170000                            CFCC
004610  001010                            BNE     3$              ;BRANCH IF NO
004612  174137  001204                    STF     AC1,$TMP2       ;GET RECEIVED DATA
004616  012737  040000  001200            MOV     #40000,$TMP0    ;SAVE EXPECTED
004624  005037  001202                    CLR     $TMP1           ;VALUE
004630  104002                            ERROR   2               ;4F1 BRANCH FAILED
                                    ;DATA OK - CHECK CC'S
004632  022737  000004  001202    3$:     CMP     #FZ,$TMP1       ;CC'S OK?
004640  001404                            BEQ     TST2            ;;BRANCH IF YES
004642  012737  000004  001200            MOV     #FZ,$TMP0       ;SAVE EXPECTED VALUE
004650  104003                            ERROR   3


                                        ;;*****************************************************************
                                        ;*TEST 2        MULF*-MO*-(DST=0)*(SRC=0)
                                        ;*
                                        ;*      THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS THE A
                                        ;*      BRANCH OR THE ADX ROMS OR STATE 344.
                                        ;*
                                        ;*      FPU ROM FLOW - 11, 130, 61, 344, 351
                                        ;;*****************************************************************
004652  000004                    TST2:   SCOPE
004654  012737  000002  001240            MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
004662  012737  004770  001222            MOV     #TST3,$ESCAPE   ;;ESCAPE TO TEST 3 ON ERROR
004670  172427  040000                    LDF     #^O040000,AC0   ;ENSURE DST NON-ZERO
004674  170437  001210                    CLRF    $TMP4           ;ENSURE SRC=0
004700  012700  001210                    MOV     #$TMP4,R0       ;PUT ADDRESS OF SRC IN R0
004704  170127  000013                    LDFPS   #13             ;LOAD COMPLIMENT CC'S
004710  171020                    1$:     MULF    (R0)+,AC0       ;EXECUTE INSTRUCTION UNDER TEST
004712  170237  001202                    STFPS   $TMP1           ;GET CC'S BACK
004716  022700  001214                    CMP     #$TMP6,R0       ;ADX ROM OK?
004722  001401                            BEQ     2$
004724  104004                            ERROR   4               ;ADX ROM FAILED
```

```
3057  004726  173427  000000          2$:     CMPF    #0,AC0          ;DID THE DST CLEAR?
3058  004732  170000                          CFCC
3059  004734  001405                          BEQ     3$              ;BRANCH IF YES
3060  004736  170437  001200                  CLRF    $TMP0           ;SAVE EXPECTED VALUE
3061  004742  174037  001204                  STF     AC0,$TMP2       ;SAVE RECEIVED VALUE
3062  004746  104001                          ERROR   1               ;DST DID NOT CLEAR
3063                                  ;DATA OK - CHECK CC'S
3064  004750  022737  000004  001202  3$:     CMP     #FZ,$TMP1       ;CC'S OK?
3065  004756  001404                          BEQ     TST3            ;;BRANCH IF YES
3066  004760  012737  000004  001200          MOV     #FZ,$TMP0       ;SAVE EXPECTED VALUE
3067  004766  104003                          ERROR   3               ;CC'S BAD
2068
3069
3070                                  ;;*****************************************************************
3071                                  ;*TEST 3        MULF*-MO*IMM*(DST=0)*(SRC=0)
3072                                  ;*
3073                                  ;*      THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS THE ADX
3074                                  ;*      ROM OR STATE 354.
3075                                  ;*
3076                                  ;*      FPU ROM FLOW - 11, 131, 61, 354, 351
3077                                  ;*****************************************************************
3078  004770  000004                  TST3:   SCOPE
3079  004772  012737  000003  001240          MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
3080  005000  012737  005076  001222          MOV     #TST4,$ESCAPE   ;;ESCAPE TO TEST 4 ON ERROR
3081  005006  170400                          CLRF    AC0             ;CLEAR THE DST
3082  005010  170127  000013                  LDFPS   #13             ;LOAD COMPLIMENT CC'S
3083  005014  171027                  1$:     MULF    (PC)+,AC0       ;EXECUTE INSTRUCTION UNDER TEST
3084  005016  000000                          .WORD   0               ;WILL HALT HERE IF ADX ROM FAILS
3085  005020  000403                          BR      2$
3086  005022  104004                          ERROR   4               ;ADX ROM FAILED
3087  005024  104004                          ERROR   4               ;       *
3088  005026  104004                          ERROR   4               ;       *
3089  005030  170237  001202          2$:     STFPS   $TMP1           ;GET CC'S BACK
3090  005034  173427  000000                  CMPF    #0,AC0          ;DST REMAIN ZERO?
3091  005040  170000                          CFCC
3092  005042  001405                          BEQ     3$
3093  005044  174037  001204                  STF     AC0,$TMP2       ;GET RESULT
3094  005050  170437  001200                  CLRF    $TMP0           ;SAVE EXPECTED ANSWER
3095  005054  104001                          ERROR   1               ;DST DID NOT REMAIN CLEAR
3096                                  ;DATA OK - CHECK THE CC'S
3097  005056  022737  000004  001202  3$:     CMP     #FZ,$TMP1       ;CC'S OK?
3098  005064  001404                          BEQ     TST4            ;;BRANCH IF YES
3099  005066  012737  000004  001200          MOV     #FZ,$TMP0       ;SAVE EXPECTED RESULT
3100  005074  104003                          ERROR   3               ;CC'S FAILED
3101
3102
3103                                  ;;*****************************************************************
3104                                  ;*TEST 4        MULF*SWR*ADD
3105                                  ;*
3106                                  ;*      THIS TEST ENSURES THAT ROM STATES 61, 340, 233, AND 112
3107                                  ;*      SETUP THE SC AND SHIFT CONTROL PROPERLY, THAT FRHK MUL SWR
3108                                  ;*      IS NOT STUCK LOW, THAT FRHL MPY/DIV ADD(1) L IS NOT STUCK
3109                                  ;*      HIGH, AND THAT FRMH FORCE MPY/DIV ADD CAUSES THE FALU TO DO
3110                                  ;*      AN ADD.  THIS IS DONE BY FIRST SETTING A MICRO-TRAP ON
3111                                  ;*      STATE 117 TO TEST MUL SWR AND THEN A TRAP IS SET ON 112
3112                                  ;*      TO ENSURE THE OTHER FUNCTIONS ARE WORKING PROPERLY.
```

```
                                        ;*
                                        ;*      IF FRME SHFC 0(0)H AND 1(0)H ARE TIED TOGETHER, THE SHIFT
                                        ;*      CONTROL WILL BE LOADED WITH THE VALUE OF THE NORM POS
                                        ;*      ENCODER.  THIS WILL CAUSE A LEFT SHIFT OF 7 IN THE AR
                                        ;*      AND A RIGHT SHIFT OF 8 IN THE QR.
                                        ;*
                                        ;*      IF FXPP SC00 DOES NOT GET TO FRMA AS A HIGH OR DOES NOT
                                        ;*      GET TO RAD00 AS A HIGH EXECUTION WILL GO TO STATE 112
                                        ;*      INSTEAD OF 113 IN SECTION 2.
                                        ;*
                                        ;*      IF FRHL MPY SIGN EXTEND DOES NOT GO LOW, THE AR WILL END
                                        ;*      UP WITH 0.01 IN IT INSTEAD OF 0.10.
                                        ;*
                                        ;*      NOTE: SECTION 1 OF THIS TEST DOES NOT ALLOW SWITCHES <7:0>
                                        ;*      TO BE LOADED INTO THE MICROBREAK FOR SYNC PURPOSES.
                                        ;**************************************************************
005076  000004                          TST4:   SCOPE
005100  012737  000004  001240                  MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
005106  012737  005452  001222                  MOV     #TST5,$ESCAPE   ;;ESCAPE TO TEST 5 ON ERROR
                                        ;SECTION 1 - CHECK FRHK MUL SWR
005114  012737  040007  001160                  MOV     #40007,$REG0    ;PUT MULTIPLIER IN MEMORY
005122  012737  000044  001162                  MOV     #44,$REG1       ;TO LOAD AS THE SRC.
005130  012703  000117                          MOV     #117,R3         ;LOAD MICRO-BREAK TO CATCH
005134  170003                                  LDUB                    ;MUL SWR STUCK LOW
005136  012737  005162  000244                  MOV     #2$,FPPVEC      ;SET FP VECTOR
005144  172427  040000                          LDF     #^O040000,AC0   ;PUT MULTIPLICAND IN AC0
005150  170127  000020                          LDFPS   #FMM
005154  171037  001160          1$:             MULF    $REG0,AC0       ;EXECUTE INSTRUCTION UNDER TEST
005160  000402                                  BR      3$              ;NO TRAP - MUL SWR OK.
                                        ;TRAP OCCURRED - FRHK MUL SWR NOT GOING HIGH
005162  022626                          2$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
005164  104005                                  ERROR   5               ;FRHK MUL SWR STUCK LOW
                                        ;;**************************************************************
                                        ;SECTION 2 - CHECK THE AR & QR AFTER THE FIRST SHIFT.
005166  032777  001000  173742          3$:     BIT     #SW9,@SWR       ;LOOP ON ERROR?
005174  001011                                  BNE     4$              ;BRANCH IF YES
005176  012703  000112                  5$:     MOV     #112,R3
005202  170003                                  LDUB
005204  170127  000020                          LDFPS   #FMM
005210  012737  005270  000244                  MOV     #7$,FPPVEC
005216  000405                                  BR      6$
005220  105737  001103                  4$:     TSTB    $ERFLG          ;ANY ERRORS?
005224  001764                                  BEQ     5$              ;BRANCH IF NO
005226  170127  000000                          LDFPS   #0              ;ENSURE FMM CLEAR
005232  012737  040000  001160          6$:     MOV     #40000,$REG0    ;RESTORE MULTIPLIER
005240  012737  005246  001110                  MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
005246  172427  040000                          LDF     #^O040000,AC0   ;LOAD THE MULTIPLICAND
005252  171037  001160                          MULF    $REG0,AC0       ;EXECUTE THE MULTIPLY
                                        ;NO TRAP OCCURRED
005256  170200                                  STFPS   R0
005260  032700  000020                          BIT     #FMM,R0         ;MAINTENANCE MODE?
005264  001762                                  BEQ     6$              ;BRANCH IF NO
005266  104006                                  ERROR   6               ;FXPP SC00 NOT GETTING TO FRMA AS A HIGH
                                        ;TRAP OCCURRED - CHECK QR AND AR DATA
005270  022626                          7$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
005272  170007                                  STQ0                    ;GET
```

```
3169  005274  174037  001210                    STF     AC0,$TMP4       ;QR DATA
3170  005300  042737  100000  001210            BIC     #BIT15,$TMP4    ;GET RID OF SIGN
3171  005306  012704  177776                    MOV     #-2,R4          ;PUT THE SHIFT COUNT IN R4
3172  005312  170004                            MSN                     ;SHIFT AR TO GET 59 & 58
3173  005314  170005                            STAD                    ;AGET
3174  005316  174037  001214                    STF     AC0,$TMP6       ;AR DATA
3175  005322  042737  100000  001214            BIC     #BIT15,$TMP6    ;GET RID OF SIGN
3176  005330  012737  000020  001200            MOV     #20,$TMP0       ;SAVE EXPECTED
3177  005336  005037  001202                    CLR     $TMP1           ;QR DATA
3178  005342  012737  000040  001204            MOV     #40,$TMP2       ;SAVE EXPECTED
3179  005350  005037  001206                    CLR     $TMP3           ;AR DATA
3180  005354  022737  000020  001210            CMP     #20,$TMP4       ;QR QUAD 3 OK?
3181  005362  001004                            BNE     8$              ;BRANCH IF NO
3182  005364  022737  000004  001212            CMP     #4,$TMP5        ;QR QUAD 2 OK?
3183  005372  001406                            BEQ     9$              ;BRANCH IF YES
3184  005374  022737  100000  001212  8$:       CMP     #BIT15,$TMP5    ;DID QR SHIFT RIGHT 8?
3185  005402  001001                            BNE     10$             ;BRANCH IF NO
3186  005404  104007                            ERROR   7               ;FRME SHFC 0(0) NOT GOING HIGH
3187  005406  104010                  10$:      ERROR   10              ;QR DATA BAD
3188                                            ;QR OK - CHECK AR
3189  005410  022737  000040  001214  9$:       CMP     #40,$TMP6       ;AR QUAD 3 OK?
3190  005416  001003                            BNE     11$             ;BRANCH IF NO
3191  005420  005737  001216                    TST     $TMP7           ;AR QUAD 2 OK?
3192  005424  001412                            BEQ     TST5            ;;BRANCH IF YES
3193  005426  022737  000020  001214  11$:      CMP     #20,$TMP6       ;DID MPY SIGN EXTEND FAIL?
3194  005434  001001                            BNE     12$             ;BRANCH IF NO
3195  005436  104011                            ERROR   11              ;FRHL MPY SIGN EXTEND NOT GOING LOW
3196  005440  005737  001214          12$:      TST     $TMP6           ;DID ADD OCCUR AT ALL?
3197  005444  001001                            BNE     13$             ;BRANCH IF YES
3198  005446  104012                            ERROR   12              ;FRHL MPY/DIV ADD(1) L NOT CAUSING
3199                                            ;FRMH FORCE MPY/DIV ADD TO FORCE THE ADD
3200  005450  104013                  13$:      ERROR   13              ;AR DATA BAD
3201
3202
3203                                    ;;**********************************************************************
3204                                    ;*TEST 5         MULF*SWR*SUB
3205                                    ;*
3206                                    ;*      THIS TEST ENSURES THAT FRHL MPY/DIV ADD (1) L GOES HIGH
3207                                    ;*      WHEN FRHK STRING IS HIGH.  THIS SIGNAL CAUSES FRMH FORCE
3208                                    ;*      MPY/DIV SUB TO GO LOW FORCING THE FALU TO DO A SUBTRACT.
3209                                    ;*
3210                                    ;*      SECTION 2 OF THIS TEST MICRO-TRAPS ON STATE 117 TO CHECK
3211                                    ;*      THAT FRHK MPY SIGN EXTEND GOES HIGH.  IF THIS TRAP DOES
3212                                    ;*      NOT OCCUR THEN FRHK E108 PIN 1 & 13 ARE NOT GOING HIGH IN
3213                                    ;*      STATE 112.
3214                                    ;**********************************************************************
3215  005452  000004                  TST5:     SCOPE
3216  005454  012737  000005  001240            MOV     #STN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
3217  005462  012737  006060  001222            MOV     #TST6,$ESCAPE   ;;ESCAPE TO TEST 6 ON ERROR
3218  005470  012737  000015  001172            MOV     #15,$REGS
3219  005476  032777  001000  173432            BIT     #SW9,@SWR       ;LOOP ON ERROR?
3220  005504  001011                            BNE     1$              ;BRANCH IF YES
3221  005506  012737  005606  000244  4$:       MOV     #2$,FPPVEC
3222  005514  012703  000112                    MOV     #112,R3
3223  005520  170003                            LDUB
3224  005522  170127  000020                    LDFPS   #FMM
```

```
3225  005526  000405                          BR      3$
3226  005530  105737  001103          1$:     TSTB    $ERFLG          ;ANY ERRORS
3227  005534  001764                          BEQ     4$              ;BRANCH IF NO
3228  005536  170127  000000                  LDFPS   #0              ;ENSURE FMM BIT OFF
3229  005542  012737  040177  001160  3$:     MOV     #40177,$REG0    ;PUT MULTIPLIER IN
3230  005550  012737  177774  001162          MOV     #177774,$REG1   ;MEMORY
3231  005556  012737  005564  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
3232  005564  172427  040000                  LDF     #^D040000,AC0   ;LOAD THE MULTIPLICAND
3233  005570  171037  001160                  MULF    $REG0,AC0       ;EXECUTE THE INSTRUCTION UNDER TEST
3234                          ;TRAP DID NOT OCCUR
3235  005574  170200                          STFPS   R0
3236  005576  032700  000020                  BIT     #FMM,R0
3237  005602  001757                          BEQ     3$
3238  005604  104014                          ERROR   14              ;MUL SHIFT ENCODER ROMOUT BAD
3239  005606  022626          2$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
3240  005610  012704  177776                  MOV     #-2,R4          ;PUT THE SHIFT COUNT IN R4
3241  005614  170004                          MSN                     ;SHIFT AR TO GET 59 & 58
3242  005616  170005                          STA0
3243  005620  174037  001214                  STF     AC0,$TMP6       ;GET AR
3244  005624  042737  100000  001214          BIC     #BIT15,$TMP6    ;GET RID OF SIGN
3245  005632  012737  000140  001204          MOV     #140,$TMP2      ;SAVE EXPECTED
3246  005640  005037  001206                  CLR     $TMP3           ;VALUE OF AR
3247  005644  022737  000140  001214          CMP     #140,$TMP6      ;DID SUBTRACTION WORK OK?
3248  005652  001406                          BEQ     5$              ;BRANCH IF YES
3249  005654  022737  000040  001214          CMP     #40,$TMP6       ;DID ADD OCCUR?
3250  005662  001001                          BNE     6$              ;BRANCH IF NO
3251  005664  104015                          ERROR   15              ;FRHK MPY/DIV ADD (1) L NOT GOING HIGH
3252                                                                  ;OR FRMH FORCE MPY/DIV SUB NOT GOING LOW
3253  005666  104013          6$:     ERROR   13              ;AR DATA BAD
3254                  ;;*****************************************************************
3255                  ;SECTION 2 - CHECK MPY SIGN EXTEND
3256  005670  032777  001000  173240  5$:     BIT     #SW9,@SWR       ;LOOP ON ERROR?
3257  005676  001011                          BNE     7$              ;BRANCH IF YES
3258  005700  012737  005764  000244  10$:    MOV     #8$,FPPVEC
3259  005706  012703  000113                  MOV     #113,R3
3260  005712  170003                          LDUB
3261  005714  170127  000020                  LDFPS   #FMM
3262  005720  000405                          BR      9$
3263  005722  105737  001103          7$:     TSTB    $ERFLG          ;ANY ERRORS YET?
3264  005726  001764                          BEQ     10$             ;BRANCH IF NO
3265  005730  170127  000000                  LDFPS   #0              ;ENSURE FMM CLEAR
3266  005734                          9$:
3267  005734  012737  005742  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
3268  005742  172427  040000                  LDF     #^D040000,AC0   ;LOAD MULTIPLICAND
3269  005746  171037  001160                  MULF    $REG0,AC0       ;EXECUTE INSTRUCTION UNDER TEST
3270                          ;TRAP DID NOT OCCUR
3271  005752  170200                          STFPS   R0
3272  005754  032700  000020                  BIT     #FMM,R0         ;FMM ON?
3273  005760  001765                          BEQ     9$              ;BRANCH IF NO
3274  005762  104016                          ERROR   16              ;FRHK MUL SWR NOT GOING LOW
3275                                                                  ;OR MUL SHF ENCODER ROM NOT OUTPUTTING
3276                                                                  ;12 IN STATE 112.
3277                          ;TRAP OCCURRED - SEE IF SIGNEXTEND WORKED
3278  005764  022626          8$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
3279  005766  012704  177776                  MOV     #-2,R4          ;PUT THE SHIFT COUNT IN R4
3280  005772  170004                          MSN                     ;SHIFT AR TO GET BITS 59 & 58
```

```
3281  005774  170005                          STAO
3282  005776  174037  001214                  STF     AC0,$TMP6       ;GET AR DATA
3283  006002  042737  100000  001214          BIC     #BIT15,$TMP6    ;GET RID OF SIGN
3284  006010  012737  000177  001204          MOV     #177,$TMP2      ;SAVE EXPECTED
3285  006016  012737  100000  001206          MOV     #BIT15,$TMP3    ;VALUE OF THE AR
3286  006024  022737  000177  001214          CMP     #177,$TMP6      ;AR QUAD 3 OK?
3287  006032  001004                          BNE     12$             ;BRANCH IF NO
3288  006034  022737  100000  001216          CMP     #BIT15,$TMP7    ;AR QUAD 2 OK?
3289  006042  001406                          BEQ     TST6            ;;BRANCH IF YES
3290  006044  022737  000001  001214  12$:    CMP     #1,$TMP6        ;DID SIGN EXTEND FAIL?
3291  006052  001001                          BNE     11$             ;BRANCH IF NO
3292  006054  104017                          ERROR   17              ;FRHK MPY SIGN EXTEND NOT GOING HIGH
3293  006056  104013                  11$:    ERROR   13              ;AR DATA BAD
3294
3295                                  ;;**********************************************************************
3296                                  ;*TEST 6         LOGIC TEST OF FRHK MUL SWR*FD(0)
3297                                  ;*
3298                                  ;*      THIS TEST CHECKS THE LOGIC THAT CAUSES FRHK MUL SWR TO GO
3299                                  ;*      HIGH.  THE PARTICULAR GATES BEING TESTED ARE E107, E108,
3300                                  ;*      AND E115.  THEY ARE TESTED BY SETTING A MICRO-TRAP ON THE
3301                                  ;*      -SWR ROM STATE THAT WOULD BE ENTERED IF THE LOGIC FAILS.
3302                                  ;*
3303                                  ;*      IN CERTAIN CASES THE TRAP MAY OCCUR AFTER THE FIRST TIME
3304                                  ;*      THRU THE 2F3 BRANCH.  IN THESE CASES THE QR WILL BE CHECKED
3305                                  ;*      TO ENSURE THE CORRECT DATA.
3306                                  ;*
3307                                  ;*      NOTE:   SYNC POINT NOT SELECTABLE. DEFAULT=116
3308                                  ;*
3309                                  ;**********************************************************************
3310
3311  006060  000004                  TST6:   SCOPE
3312  006062  012737  000006  001240          MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
3313                                  ;SECTION - 1
3314                                  ;       E108(1,13(LH AND E115(5,6,4)HLL
3315  006070  012737  006144  000244          MOV     #1$,FPPVEC
3316  006076  012737  040000  001160          MOV     #40000,$REG0    ;PUT MULTIPLIER IN
3317  006104  012737  000040  001162          MOV     #BIT5,$REG1     ;MEMORY (ROM ADDR=240)
3318  006112  012737  006120  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
3319  006120  012703  000116                  MOV     #116,R3
3320  006124  170003                          LDUB
3321  006126  170127  000020                  LDFPS   #FMM
3322  006132  172427  040000                  LDF     #^O040000,AC0   ;LOAD THE MULTIPLICAND
3323  006136  171037  001160                  MULF    $REG0,AC0       ;EXECUTE INSTRUCTION
3324  006142  170000                          CFCC                    ;WAIT FOR TRAP
3325  006144  022626                  1$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
3326  006146  170007                          STQ0
3327  006150  174037  001210                  STF     AC0,$TMP4       ;GET QR TO FIND OUT WHEN TRAP OCCURRED
3328  006154  042737  100000  001210          BIC     #BIT15,$TMP4    ;GET RID OF SIGN
3329  006162  022737  004000  001212          CMP     #BIT11,$TMP5    ;FIRST TIME THRU THE BRANCH?
3330  006170  001001                          BNE     2$              ;BRANCH IF NO
3331  006172  104020                          ERROR   20              ;FRHK E115(4) NOT GOING LOW
3332                                  ;;**********************************************************************
3333                                  ;SECTION - 2
3334                                  ;       E108(1,13)HH AND E115(5,6,4)LLH
3335  006174  005037  001162          2$:     CLR     $REG1           ;INITIALIZE MULTIPLIER
3336  006200  012737  006240  000244          MOV     #3$,FPPVEC
```

```
3337  006206  012737  006214  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
3338  006214  012703  000116                  MOV     #116,R3
3339  006220  170003                          LDUB
3340  006222  170127  000020                  LDFPS   #FMM
3341  006226  172427  040000                  LDF     #^O040000,AC0   ;LOAD THE MULTIPLICAND
3342  006232  171037  001160                  MULF    $REG0,AC0       ;EXECUTE THE INSTRUCTION
3343  006236  170000                          CFCC                    ;WAIT FOR TRAP
3344  006240  022626                  3$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
3345  006242  170007                          STQD
3346  006244  174037  001210                  STF     AC0,$TMP4       ;GET QR DATA
3347  006250  042737  100000  001210          BIC     #BIT15,$TMP4
3348  006256  022737  004000  001212          CMP     #BIT11,$TMP5    ;TRAP OCCUR AT CORRECT TIME?
3349  006264  001401                          BEQ     8$              ;BRANCH IF YES
3350  006266  104021                          ERROR   21              ;FRHK E115(4) NOT GOING HIGH
3351                                  ;CHECK THE AR TO ENSURE THAT MUL SWR DISABLED THE ADD
3352  006270                          8$:
3353  006270  012704  177776                  MOV     #-2,R4          ;PUT THE SHIFT COUNT IN R4
3354  006274  170004                          MSN                     ;SHIFT THE AR TO GET 59 & 58
3355  006276  170005                          STAD
3356  006300  174037  001214                  STF     AC0,$TMP6       ;GET AR DATA
3357  006304  042737  100000  001214          BIC     #BIT15,$TMP6
3358  006312  005737  001214                  TST     $TMP6           ;QUAD 3 OK?
3359  006316  001003                          BNE     9$              ;BRANCH IF NO
3360  006320  005737  001216                  TST     $TMP7           ;QUAD 2 OK?
3361  006324  001403                          BEQ     4$              ;BRANCH IF YES
3362  006326  170437  001204          9$:     CLRF    $TMP2           ;SAVE EXPECTED VALUE
3363  006332  104023                          ERROR   23              ;FRHK MUL SWR DID NOT DISABLE
3364                                                                  ;FRMH MPY/DIV ADD (OR SUB)
3365                                  ;;*****************************************************************
3366                                  ;SECTION - 3
3367                                  ;       E107(13,1,2)HLH AND E108(13,1)LH
3368  006334  012737  000002  001162  4$:     MOV     #BIT1,$REG1     ;PUT MULTIPLIER IN MEMORY
3369  006342  012737  006406  000244          MOV     #5$,FPPVEC
3370  006350  012737  006356  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
3371  006356  012703  000116                  MOV     #116,R3
3372  006362  170003                          LDUB
3373  006364  170127  000020                  LDFPS   #FMM
3374  006370  172427  040000                  LDF     #^O040000,AC0   ;LOAD THE MULTIPLICAND
3375  006374  171037  001160                  MULF    $REG0,AC0       ;EXECUTE TEST
3376  006400  170000                          CFCC                    ;WAIT FOR TRAP
3377  006402  104022                          ERROR   22              ;FRHK E107(12) NOT GOING LOW
3378  006404  000411                          BR      6$
3379  006406  022626                  5$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
3380  006410  170007                          STQD
3381  006412  174037  001214                  STF     AC0,$TMP6       ;GET QR DATA
3382  006416  022737  001000  001216          CMP     #BIT9,$TMP7     ;DID TRAP OCCUR AT CORRECT TIME?
3383  006424  001401                          BEQ     6$              ;BRANCH IF YES
3384  006426  104022                          ERROR   22              ;FRHK E107(12) NOT GOING LOW
3385                                  ;;*****************************************************************
3386                                  ;SECTION - 4
3387                                  ;       E107(13,1,2)LHH AND E108(13,1)LH
3388  006430  012737  000060  001162  6$:     MOV     #60,$REG1       ;PUT MULTIPLIER IN MEMORY
3389  006436  012737  006476  000244          MOV     #7$,FPPVEC
3390  006444  012737  006452  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
3391  006452  012703  000117                  MOV     #117,R3
3392  006456  170003                          LDUB
```

```
3393  006460  170127  000020                  LDFPS   #FMM
3394  006464  172427  040000                  LDF     #↑040000,AC0    ;LOAD THE MULTIPLICAND
3395  006470  171037  001160                  MULF    $REG0,AC0       ;EXECUTE THE INSTRUCTION
3396  006474  170000                          CFCC                    ;WAIT FOR THE TRAP
3397  006476  022626                  7$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
3398  006500  170007                          STQD
3399  006502  174037  001214                  STF     AC0,$TMP6       ;GET THE QR DATA
3400  006506  022737  000020  001216          CMP     #20,$TMP7       ;TRAP OCCUR AT THE CORRECT TIME?
3401  006514  001401                          BEQ     TST7            ;;BRANCH IF YES
3402  006516  104022                          ERROR   22              ;FRHK E107(12) NOT GOING LOW
3403
3404
3405                                  ;;*****************************************************************
3406                                  ;*TEST 7        MULF (.5*1)
3407                                  ;*
3408                                  ;*      THIS TEST MULTIPLIES 1 BY 0.5 TO ENSURE THAT ALL THE
3409                                  ;*      ROM SIGNALS IN STATES 116, 110, AND 72 FUNCTION PROPERLY.
3410                                  ;*
3411                                  ;*      FPU ROM FLOW - 11, 131, 61, 340, 233, 112, 3(116), 110, 72, NORMALIZE
3412                                  ;*****************************************************************
3413  006520  000004                  TST7:   SCOPE
3414  006522  012737  000007  001240          MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
3415  006530  012737  006622  001222          MOV     #TST10,$ESCAPE  ;;ESCAPE TO TEST 10 ON ERROR
3416  006536  012737  040000  001200          MOV     #40000,$TMP0    ;SAVE EXPECTED
3417  006544  005037  001202                  CLR     $TMP1           ;RESULT
3418  006550  172427  040200                  LDF     #↑040200,AC0    ;LOAD THE MULTIPLICAND
3419  006554  170127  000017                  LDFPS   #17             ;LOAD COMPLIMENT CC'S
3420  006560  171037  001200          1$:     MULF    $TMP0,AC0       ;EXECUTE THE INSTRUCTION UNDER TEST
3421  006564  170200                          STFPS   R0              ;GET CC'S
3422  006566  174037  001204                  STF     AC0,$TMP2       ;GET RESULT BACK
3423  006572  173437  001200                  CMPF    $TMP0,AC0       ;RESULT OK?
3424  006576  170000                          CFCC
3425  006600  001401                          BEQ     2$              ;BRANCH IF YES
3426  006602  104024                          ERROR   24              ;RESULT WRONG
3427  006604  005700                  2$:     TST     R0              ;CC'S OK?
3428  006606  001405                          BEQ     TST10           ;;BRANCH IF YES
3429  006610  005037  001200                  CLR     $TMP0           ;SAVE EXPECTED VALUE
3430  006614  010037  001202                  MOV     R0,$TMP1        ;SAVE RECEIVED VALUE
3431  006620  104003                          ERROR   3               ;CC'S BAD
3432
3433
3434                                  ;;*****************************************************************
3435                                  ;*TEST 10       MULF (.111...1*1)
3436                                  ;*
3437                                  ;*      THIS TEST CHECKS ROM STATE 111.
3438                                  ;*
3439                                  ;*      FPU ROM FLOW - 11, 130, 61, 340, 233, 112, 113, 3(117), 111, 72, NORMALIZE
3440                                  ;*****************************************************************
3441  006622  000004                  TST10:  SCOPE
3442  006624  012737  000010  001240          MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
3443  006632  012737  006734  001222          MOV     #TST11,$ESCAPE  ;;ESCAPE TO TEST 11 ON ERROR
3444  006640  012737  040177  001200          MOV     #40177,$TMP0    ;PUT MULTIPLIER
3445  006646  012737  177777  001202          MOV     #-1,$TMP1       ;IN MEMORY
3446  006654  172427  140200                  LDF     #↑0140200,AC0   ;LOAD THE MULTIPLICAND
3447  006660  170127  000007                  LDFPS   #7              ;LOAD COMPLIMENT CC'S
3448  006664  171037  001200          1$:     MULF    $TMP0,AC0       ;EXECUTE INSTRUCTION UNDER TEST
```

```
3449  006670  170200                          STFPS   R0              ;GET CC'S BACK
3450  006672  052737  100000  001200          BIS     #BIT15,$TMP0
3451  006700  173437  001200                  CMPF    $TMP0,AC0       ;RESULT OK?
3452  006704  170000                          CFCC
3453  006706  001401                          BEQ     2$              ;BRANCH IF YES
3454  006710  104024                          ERROR   24              ;RESULT BAD
3455  006712  022700  000010          2$:     CMP     #FN,R0          ;CC'S OK?
3456  006716  001406                          BEQ     TST11           ;;BRANCH IF YES
3457  006720  012737  000010  001200          MOV     #FN,$TMP0       ;SAVE EXPECTED VALUE
3458  006726  010037  001202                  MOV     R0,$TMP1        ;SAVE RECEIVED VALUE
3459  006732  104003                          ERROR   3               ;CC'S BAD
3460
3461
3462                                  ;;*****************************************************************
3463                                  ;*TEST 11       HIGH ORDER MUL SHIFT ENCODER ROM
3464                                  ;*
3465                                  ;*      THIS TEST CHECKS EVERY ADDRESS ON THE HIGH ORDER MUL SHIFT
3466                                  ;*      ENCODER.  THIS IS DONE IN TWO SECTIONS.  THE FIRST SECTION
3467                                  ;*      RUNS A COUNT PATTERN THROUGH QR BITS <41:35>.  THE SECOND
3468                                  ;*      SECTION STARTS EACH MULTIPLY WITH QR BITS <41:35> ALL ONE'S
3469                                  ;*      AND RUNS A COUNT PATTERN THRU QR BITS <48:42>.
3470                                  ;*
3471                                  ;*      SINCE THE MULCIPLICAND IS "1.0" THE RESULT SHOULD BE IDENTICAL
3472                                  ;*      TO THE MULTIPLIER.
3473                                  ;*
3474                                  ;*      IF AN ERROR OCCURS THE TYPEOUT INDICATES THE ROM ADDRESS
3475                                  ;*      AND THE EXPECTED OUTPUT OF THE ROM FOR EACH SHIFT OF THAT
3476                                  ;*      PARTICULAR MULTIPLY.
3477                                  ;;*****************************************************************
3478  006734  000004                  TST11:  SCOPE
3479  006736  012737  000011  001240          MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
3480  006744  012737  007144  001222          MOV     #TST12,$ESCAPE  ;;ESCAPE TO TEST 12 ON ERROR
3481                                  ;SECTION - 1
3482  006752  105037  001276                  CLRB    $FD             ;ENSURE $FD CLEAR INCASE OF ERROR
3483  006756  012737  040000  001160          MOV     #40000,$REG0    ;INITIALIZE THE
3484  006764  005037  001162                  CLR     $REG1           ;MULTIPLIER
3485  006770  012737  006776  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
3486  006776  172427  040200          1$:     LDF     #^O040200,AC0   ;INITIALIZE THE MULTIPLICAND
3487  007002  171037  001160                  MULF    $REG0,AC0       ;EXECUTE THE MULTIPLY
3488  007006  173437  001160                  CMPF    $REG0,AC0       ;RESULT OK?
3489  007012  170000                          CFCC
3490  007014  001013                          BNE     2$              ;BRANCH IF NO
3491  007016  032777  001000  172112          BIT     #SW9,@SWR       ;LOOP ON ERROR?
3492  007024  001403                          BEQ     3$              ;BRANCH IF NO
3493  007026  105737  001103                  TSTB    $ERFLG          ;ANY ERRORS YET?
3494  007032  001361                          BNE     1$              ;BRANCH IF YES
3495  007034  105237  001162          3$:     INCB    $REG1           ;SELECT THE NEXT MULTIPLIER
3496  007040  100356                          BPL     1$              ;BRANCH IF NOT FINISHED
3497  007042  000401                          BR      4$              ;GO TO NEXT SECTION
3498                                  ;;*****************************************************************
3499  007044  104025                  2$:     ERROR   25              ;MUL SHIFT ENCODER ROM FAILED
3500                                  ;;*****************************************************************
3501                                  ;SECTION - 2
3502  007046  012737  040000  001160  4$:     MOV     #40000,$REG0    ;INITIALIZE THE
3503  007054  012737  000177  001162          MOV     #177,$REG1      ;MULTIPLIER
3504  007062  012737  007070  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
```

```
3505  007070  172427  040200          6$:     LDF     #↑040200,AC0    ;INITIALIZE THE MULTIPLICAND
3506  007074  171037  001160                  MULF    $REG0,AC0       ;EXECUTE THE MULTIPLY
3507  007100  173437  001160                  CMPF    $REG0,AC0       ;RESULT OK?
3508  007104  170000                          CFCC
3509  007106  001356                          BNE     2$              ;BRANCH IF NO
3510  007110  032777  001000  172020          BIT     #SW9,@SWR       ;LOOP ON ERROR?
3511  007116  001403                          BEQ     5$              ;BRANCH IF NO
3512  007120  105737  001103                  TSTB    $ERFLG          ;ANY ERRORS YET?
3513  007124  001361                          BNE     6$              ;BRANCH IF YES
3514  007126  062737  000200  001162  5$:     ADD     #BIT7,$REG1     ;SELECT NEXT MULTIPLIER
3515  007134  032737  040000  001162          BIT     #BIT14,$REG1    ;DONE YET?
3516  007142  001752                          BEQ     6$              ;BRANCH IF NO
3517
3518
3519                                  ;;*****************************************************************
3520                                  ;*TEST 12       LOGIC TESTS OF FRHK MUL SWR*FD(1)
3521                                  ;*
3522                                  ;*      THIS TEST CHECKS THE A-BRANCH AND ADX ROMS FOR MULD AND
3523                                  ;*      THE GATES THAT DRIVE FRHK MUL SWR WITH FD(1).  IN PARTICULAR
3524                                  ;*      FRLH E1, FRHK E107, AND E108.
3525                                  ;*
3526                                  ;*      THIS LOGIC TEST IS PERFORMED BY SETTING A MICRO-TRAP OFF
3527                                  ;*      THE 2F3 BRANCH TO ENSURE THAT FRHK MUL SWR GOES HIGH.
3528                                  ;*
3529                                  ;*      NOTE:   SYNC POINT NOT SELECTABLE. DEFAULT=116
3530                                  ;;*****************************************************************
3531  007144  000004          TST12:  SCOPE
3532  007146  012737  000012  001240          MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
3533  007154  012737  010166  001222          MOV     #TST13,$ESCAPE  ;;ESCAPE TO TEST 13 ON ERROR
3534  007162  005037  001200                  CLR     $TMP0
3535  007166  012737  004000  001202          MOV     #BIT11,$TMP1
3536  007174  005037  001204                  CLR     $TMP2
3537  007200  005037  001206                  CLR     $TMP3
3538  007204  012737  040000  001160          MOV     #40000,$REG0    ;PUT THE
3539  007212  012737  000040  001162          MOV     #BIT5,$REG1     ;MULTIPLIER IN
3540  007220  005037  001164                  CLR     $REG2           ;MEMORY
3541  007224  005037  001166                  CLR     $REG3           ;       *
3542  007230  012737  007272  000244          MOV     #2$,FPPVEC
3543  007236  012737  007244  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
3544  007244  012703  000116                  MOV     #116,R3         ;SET MICRO-TRAP
3545  007250  170003                          LDUB
3546  007252  170127  000220                  LDFPS   #FMM+FD
3547  007256  172427  040000                  LDD     #↑040000,AC0    ;LOAD THE MULTIPLICAND
3548  007262  012700  001160                  MOV     #$REG0,R0
3549  007266  171020                  1$:     MULD    (R0)+,AC0       ;EXECUTE TEST
3550  007270  170000                          CFCC                    ;WAIT FOR TRAP
3551  007272  012706  001100          2$:     MOV     #1100,SP        ;RESTORE THE SP
3552  007276  022700  001170                  CMP     #$REG4,R0       ;ADX ROM OK?
3553  007302  001401                          BEQ     100$            ;BRANCH IF YES
3554  007304  104004                          ERROR   4               ;ADX ROM FAILED
3555  007306  170007                  100$:   STQ0
3556  007310  174037  001210                  STD     AC0,$TMP4       ;GET QR
3557  007314  042737  100000  001210          BIC     #BIT15,$TMP4    ;GET RID OF SIGN
3558  007322  022737  004000  001212          CMP     #BIT11,$TMP5    ;QR OK?
3559  007330  001401                          BEQ     3$              ;BRANCH IF YES
3560  007332  104026                          ERROR   26              ;FRHK MUL SWR NOT GOING LOW
```

```
                                                                ;WITH PIN 9 AND 10 ON A HIGH.
                                        ;;**************************************************************
                                        ;SECTION - 2
                                        ;
                                                FRHK E108(9,10,13,1)H,L,L,L, AND FRLH E1(12,13)H,L
007334  012737  000040  001166  3$:     MOV     #40,$REG3       ;SET THE MULTIPLIER
007342  012737  007402  000244          MOV     #4$,FPPVEC
007350  012737  007356  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
007356  012703  000116                  MOV     #116,R3
007362  170003                          LDUB
007364  170127  000220                  LDFPS   #FD+FMM
007370  172427  040000                  LDD     #^O040000,AC0   ;LOAD THE MULTIPLICAND
007374  171037  001160                  MULD    $REG0,AC0       ;EXECUTE THE TEST
007400  170000                          CFCC                    ;WAIT FOR TRAP
007402  012706  001100          4$:     MOV     #1100,SP        ;RESTORE THE SP
007406  170007                          STQD
007410  174037  001210                  STD     AC0,$TMP4       ;GET THE QR
007414  042737  100000  001210          BIC     #BIT15,$TMP4    ;GET RID OF SIGN
007422  022737  000040  001212          CMP     #BIT5,$TMP5     ;TRAP OCCUR AT CORRECT TIME?
007430  001411                          BEQ     5$              ;BRANCH IF YES
007432  012737  000040  001202          MOV     #BIT5,$TMP1
007440  012737  000010  001204          MOV     #10,$TMP2
007446  005037  001200                  CLR     $TMP0
007452  104027                          ERROR   27              ;FRLH E1(11) DID NOT GO LOW
                                        ;;**************************************************************
                                        ;SECTION - 3
                                        ;
                                                FRHK E108(9,10,13,1)L,H,L,L
007454  012737  000060  001166  5$:     MOV     #60,$REG3       ;SET THE MULTIPLIER
007462  012737  007522  000244          MOV     #6$,FPPVEC
007470  012737  007476  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
007476  012703  000117                  MOV     #117,R3
007502  170003                          LDUB
007504  170127  000220                  LDFPS   #FD+FMM
007510  172427  040000                  LDD     #^O040000,AC0   ;LOAD THE MULTIPLICAND
007514  171037  001160                  MULD    $REG0,AC0       ;EXECUTE THE TEST
007520  170000                          CFCC                    ;WAIT FOR THE TRAP
007522  012706  001100          6$:     MOV     #1100,SP        ;RESTORE THE SP
007526  170007                          STQD
007530  174037  001210                  STD     AC0,$TMP4       ;GET THE QR
007534  042737  100000  001210          BIC     #BIT15,$TMP4    ;GET RID OF THE SIGN
007542  022737  000020  001212          CMP     #BIT4,$TMP5     ;TRAP OCCUR AT THE CORRECT TIME?
007550  001411                          BEQ     7$              ;BRANCH IF YES
007552  005037  001200                  CLR     $TMP0           ;SAVE EXPECTED
007556  012737  000020  001202          MOV     #BIT4,$TMP1     ;DATA
007564  012737  000010  001204          MOV     #10,$TMP2       ;       *
007572  104030                          ERROR   30              ;FRHK E107 NOT GOING LOW WITH
                                                                ;MUL SHF 0 ON A HIGH
                                        ;;**************************************************************
                                        ;SECTION - 4
                                        ;
                                                FRLH E1(12,13)H,H AND FRHK E107(3,4,5)H,H,H
007574  012737  000377  001166  7$:     MOV     #377,$REG3      ;LOAD THE MULTIPLIER
007602  012737  007642  000244          MOV     #8$,FPPVEC
007610  012737  007616  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
007616  012703  000117                  MOV     #117,R3
007622  170003                          LDUB
007624  170127  000220                  LDFPS   #FD+FMM
007630  172427  040000                  LDD     #^O040000,AC0   ;LOAD THE MULTIPLICAND
```

```
3617  007634  171037  001160              MULD    $REG0,AC0       ;EXECUTE THE TEST
3618  007640  170000                      CFCC                    ;WAIT FOR THE TRAP
3619  007642  012706  001100        8$:   MOV     #1100,SP        ;RESTORE THE SP
3620  007646  170007                      STQ0
3621  007650  174037  001210              STD     AC0,$TMP4       ;GET THE QR
3622  007654  042737  100000  001210      BIC     #BIT15,$TMP4    ;GET RID OF SIGN
3623  007662  022737  040000  001212      CMP     #BIT14,$TMP5    ;QR DATA OK?
3624  007670  001411                      BEQ     9$              ;BRANCH IF YES
3625  007672  012737  040000  001202      MOV     #BIT14,$TMP1    ;SAVE EXPECTED
3626  007700  005037  001200              CLR     $TMP0           ;DATA
3627  007704  012737  010000  001204      MOV     #BIT12,$TMP2    ;          *
3628  007712  104031                      ERROR   31              ;FRLH E1 NOT GOING HIGH WITH BOTH
3629                                                              ;INPUTS ON A HIGH
3630                                ;;****************************************************************
3631                                ;SECTION - 5
3632                                ;       FRHK E107(3,4,5)H,L,H
3633  007714  012737  000002  001166  9$:   MOV     #2,$REG3        ;LOAD MULTIPLIER
3634  007722  012737  007762  000244      MOV     #10$,FPPVEC
3635  007730  012737  007736  001110      MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
3636  007736  012703  000116              MOV     #116,R3
3637  007742  170003                      LDUB
3638  007744  170127  000220              LDFPS   #FD+FMM
3639  007750  172427  040000              LDD     #^O040000,AC0   ;LOAD THE MULTIPLICAND
3640  007754  171037  001160              MULD    $REG0,AC0       ;EXECUTE THE TEST
3641  007760  170000                      CFCC                    ;WAIT FOR THE TRAP
3642  007762  012706  001100        10$:  MOV     #1100,SP        ;RESTORE THE SP
3643  007766  170007                      STQ0
3644  007770  174037  001210              STD     AC0,$TMP4       ;GET THE QR
3645  007774  042737  100000  001210      BIC     #BIT15,$TMP4    ;GET RID OF SIGN
3646  010002  022737  001000  001212      CMP     #BIT9,$TMP5     ;TRAP AT CORRECT TIME?
3647  010010  001413                      BEQ     11$             ;BRANCH IF YES
3648  010012  012737  001000  001202      MOV     #BIT9,$TMP1
3649  010020  012737  000200  001204      MOV     #BIT7,$TMP2
3650  010026  005037  001206              CLR     $TMP3
3651  010032  005037  001200              CLR     $TMP0
3652  010036  104032                      ERROR   32              ;FRHK E107(6) NOT GOING LOW WITH
3653                                                              ;H,L,H INPUT
3654                                ;;****************************************************************
3655                                ;SECTION - 6
3656                                ;       FRHK E107(13,1,2)H,H,L
3657  010040  005037  001162        11$:  CLR     $REG1           ;LOAD THE
3658  010044  012737  000040  001166      MOV     #BIT5,$REG3     ;MULTIPLIER
3659  010052  012737  010112  000244      MOV     #12$,FPPVEC
3660  010060  012737  010066  001110      MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
3661  010066  012703  000116              MOV     #116,R3
3662  010072  170003                      LDUB
3663  010074  170127  000220              LDFPS   #FD+FMM
3664  010100  172427  040000              LDD     #^O040000,AC0   ;LOAD THE MULTIPLICAND
3665  010104  171037  001160              MULD    $REG0,AC0       ;EXECUTE THE TEST
3666  010110  170000                      CFCC                    ;WAIT FOR TRAP
3667  010112  012706  001100        12$:  MOV     #1100,SP        ;RESTORE THE SP
3668  010116  170007                      STQ0
3669  010120  174037  001210              STD     AC0,$TMP4       ;GET QR
3670  010124  042737  100000  001210      BIC     #BIT15,$TMP4    ;GET RID OF SIGN
3671  010132  022737  000040  001212      CMP     #BIT5,$TMP5     ;TRAP OCCUR AT CORRECT TIME?
3672  010140  001412                      BEQ     TST13           ;;BRANCH IF YES
```

```
 3673  010142  005037  001200                CLR     $TMP0
 3674  010146  012737  000040  001202        MOV     #BITS,$TMP1
 3675  010154  005037  001204                CLR     $TMP2
 3676  010160  005037  001206                CLR     $TMP3
 3677  010164  104032                        ERROR   32              ;FRHK E107(12) NOT GOING LOW
                                                                     ;WITH H,H,L INPUT.
 3678
 3679
 3680
 3681                                ;;*****************************************************************
 3682                                ;*TEST 13       LOW ORDER MUL SHIFT ENCODER ROM
 3683                                ;*
 3684                                ;*      THIS TEST CHECKS EVERY ADDRESS ON THE LOW ORDER MUL SHIFT
 3685                                ;*      ENCODER ROM.  THIS IS DONE IN TWO SECTIONS, THE FIRST
 3686                                ;*      SECTION RUNS A COUNT PATTERN THRU QR BITS <9:3>.  THE
 3687                                ;*      SECOND SECTION STARTS EACH MULTIPLY WITH QR BITS<9:3>
 3688                                ;*      ALL ONE'S AND RUNS A COUNT PATTERN THRU QR BITS <16:10>.
 3689                                ;*
 3690                                ;*      SINCE THE MULTIPLICAND IS "1.0" THE RESULT SHOULD BE
 3691                                ;*      IDENTICAL TO THE MULTIPLIER.
 3692                                ;*
 3693                                ;*      IF AN ERROR OCCURS THE TYPEOUT INDICATES THE ROM ADDRESS
 3694                                ;*      AND THE EXPECTED OUTPUT OF THE ROM FOR EACH SHIFT OF THE
 3695                                ;*      LOWER 24 BITS OF THAT PARTICULAR MULTIPLY.
 3696                                ;;*****************************************************************
 3697  010166  000004                TST13:  SCOPE
 3698  010170  012737  000013  001240        MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
 3699  010176  112737  000001  001276        MOVB    #1,$FD          ;SET SOFTWARE FD INDICATOR
 3700                                ;SECTION - 1
 3701  010204  012737  040000  001160        MOV     #40000,$REG0    ;INITIALIZE
 3702  010212  005037  001162                CLR     $REG1           ;THE MULTIPLIER
 3703  010216  005037  001164                CLR     $REG2           ;       *
 3704  010222  005037  001166                CLR     $REG3           ;       *
 3705  010226  012737  010234  001110        MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
 3706  010234  170127  000200        1$:     LDFPS   #FD
 3707  010240  172427  040200                LDD     #^O40200,AC0    ;INITIALIZE THE MULTIPLICAND
 3708  010244  171037  001160                MULD    $REG0,AC0       ;EXECUTE THE MULTIPLY
 3709  010250  173437  001160                CMPD    $REG0,AC0       ;RESULT OK?
 3710  010254  170000                        CFCC
 3711  010256  001013                        BNE     2$              ;BRANCH IF NO
 3712  010260  032777  001000  170650        BIT     #SW9,@SWR       ;LOOP ON ERROR?
 3713  010266  001403                        BEQ     3$              ;BRANCH IF NO
 3714  010270  105737  001103                TSTB    $ERFLG          ;ANY ERRORS?
 3715  010274  001357                        BNE     1$              ;BRANCH IF YES
 3716  010276  105237  001166        3$:     INCB    $REG3           ;SELECT THE NEXT MULTIPLIER
 3717  010302  100354                        BPL     1$              ;BRANCH IF NOT FINISHED
 3718  010304  000401                        BR      4$              ;GO TO SECTION 2
 3719                                ;;*****************************************************************
 3720  010306  104025                2$:     ERROR   25              ;MUL SHIFT ENCODER ROM FAILED
 3721                                ;;*****************************************************************
 3722                                ;SECTION - 2
 3723  010310  012737  000177  001166 4$:     MOV     #177,$REG3      ;INIT THE MULTIPLIER
 3724  010316  012737  010324  001110        MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
 3725  010324  170011                6$:     SETD
 3726  010326  172427  040200                LDD     #^O40200,AC0    ;LOAD THE MULTIPLICAND
 3727  010332  171037  001160                MULD    $REG0,AC0       ;EXECUTE THE MULTIPLY
 3728  010336  173437  001160                CMPD    $REG0,AC0       ;RESULT OK?
```

```
                                        CFCC
                                        BNE     2$                      ;BRANCH IF NO
                                        BIT     #SW9,@SWR               ;LOOP ON ERROR?
                                        BEQ     5$                      ;BRANCH IF NO
                                        TSTB    $ERFLG                  ;ANY ERRORS?
                                        BNE     6$                      ;BRANCH IF YES
                                5$:     ADD     #BIT7,$REG3             ;SELECT THE NEXT MULTIPLIER
                                        BIT     #BIT14,$REG3            ;DONE YET?
                                        BEQ     6$                      ;BRANCH IF NO

                        ;;*******************************************************************
                        ;*TEST 14       MODF*MO*(SRC=0)*(DST=0)
                        ;*
                        ;*      THIS TEST ENSURES THAT FXPC FIRB08(1) GETS TO FRMA
                        ;*      AS A HIGH AND THAT THE A BRANCH AND NO-MEM ROMS
                        ;*      FUNCTION PROPERLY.
                        ;*
                        ;*      FPU ROM FLOW-30,61,354,355
                        ;*******************************************************************
                        TST14:  SCOPE
                                MOV     #$TN-1,$TESTN           ;;SET TEST NUMBER IN  MAIL BOX
                                MOV     #TST15,$ESCAPE          ;;ESCAPE TO TEST 15 ON ERROR
                                CLRF    AC0                     ;ENSURE DST ZERO
                                LDF     #^O40000,AC1            ;ENSURE DST+1 NON-ZERO
                                CLRF    AC2                     ;ENSURE SRC=0
                        1$:     MODF    AC2,AC0                 ;EXECUTE INSTRUCTION UNDER TEST
                                CMPF    #0,AC1                  ;DID DST+1 GET CLEARED?
                                CFCC
                                BEQ     TST15                   ;;BRANCH IF YES
                                STF     AC1,$TMP2               ;GET DST+1
                                CLRF    $TMP0                   ;SAVE EXPECTED DATA
                                ERROR   34                      ;MODF DID NOT CLEAR DST+1

                        ;;*******************************************************************
                        ;*TEST 15       MODF*-MO*(INT=0)
                        ;*
                        ;*      THIS TEST ENSURES THAT AN INTEGER RESULT OF ZERO CAUSES
                        ;*      ZERO TO BE STORED IN THE DST ACC+1 AND THE FRACTION IN
                        ;*      THE DST ACCUMULATOR.
                        ;*
                        ;*      FPU ROM FLOW-MULTIPLY,76,266,327,305,325,NORMALIZE
                        ;*******************************************************************
                        TST15:  SCOPE
                                MOV     #$TN-1,$TESTN           ;;SET TEST NUMBER IN  MAIL BOX
                                MOV     #TST16,$ESCAPE          ;;ESCAPE TO TEST 16 ON ERROR
                                MOV     #40007,$REG0            ;PUT THE SRC
                                CLR     $REG1                   ;DATA IN MEMORY
                                MOV     #307,R3                 ;SET MICRO-BREAK TO
                                LDUB                            ;LATCH 3F2 BRANCH FAILURE
                                LDFPS   #FMM
                                MOV     #2$,FPPVEC
                                MOV     #.+6,@#$LPERR           ;SET ERROR LOOP
                                LDF     #^O140200,AC0           ;LOAD THE DST
                                LDF     $REG0,AC1               ;ENSURE ACD+1 IS NON ZERO
                                MOV     #$REG0,R0               ;PUT ADDRESS OF SRC IN R0
                        1$:     MODF    (R0)+,AC0               ;EXECUTE INSTRUCTION UNDER TEST
```

```
3785  010550  022700  001164                    CMP     #$REG2,R0       ;ADX ROM OK?
3786  010554  001401                            BEQ     3$              ;BRANCH IF YES
3787  010556  104004                            ERROR   4               ;ADX ROM FAILED
3788  010560  174137  001204            3$:     STF     AC1,$TMP2       ;GET DST ACC+1
3789  010564  005737  001204                    TST     $TMP2           ;DID IT GET CLEARED?
3790  010570  001403                            BEQ     4$              ;BRANCH IF YES
3791  010572  170437  001200                    CLRF    $TMP0           ;SAVE EXPECTED VALUE
3792  010576  104035                            ERROR   35              ;STATE 305 DID NOT CLEAR DST ACC+1
3793  010600  173427  140007            4$:     CMPF    #^O140007,AC0   ;IS FRACTION OK?
3794  010604  001412                            BEQ     TST16           ;;BRANCH IF YES
3795  010606  174037  001204                    STF     AC0,$TMP2       ;SAVE RECEIVED DATA
3796  010612  012737  014007  001200            MOV     #14007,$TMP0    ;SAVE EXPECTED
3797  010620  005037  001202                    CLR     $TMP1           ;DATA
3798  010624  104036                            ERROR   36              ;FRACTION IS WRONG
3799                                    ;ERROR - 3F2 BRANCH FAILED
3800  010626  022626                    2$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
3801  010630  104037                            ERROR   37              ;STATE 266 DID NOT CAUSE BN TO SET
3802
3803                                    ;;*****************************************************************
3804                                    ;*TEST 16       MODF*(INT=2**25)*-BOU
3805                                    ;*
3806                                    ;*      THIS TEST GENERATES A NUMBER WHO'S FRACTIONAL
3807                                    ;*      PART IS ZERO.
3808                                    ;*
3809                                    ;*      FPU ROM FLOW-MULTIPLY,76,266,327,307,7,341
3810                                    ;;*****************************************************************
3811  010632  000004                    TST16:  SCOPE
3812  010634  012737  000016  001240            MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
3813  010642  012737  011052  001222            MOV     #TST17,$ESCAPE  ;;ESCAPE TO TEST 17 ON ERROR
3814  010650  012737  046207  001160            MOV     #46207,$REG0    ;PUT THE SRC
3815  010656  005037  001162                    CLR     $REG1           ;IN MEMORY
3816  010662  012703  000005                    MOV     #5,R3
3817  010666  170003                            LDUB
3818  010670  012737  011046  000244            MOV     #2$,FPPVEC
3819  010676  012737  010704  001110            MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
3820  010704  172427  040200                    LDF     #^O040200,AC0   ;LOAD THE DST
3821  010710  032777  001000  170220            BIT     #SW9,@SWR       ;LOOP ON ERROR?
3822  010716  001406                            BEQ     3$              ;BRANCH IF NO
3823  010720  105737  001103                    TSTB    $ERFLG          ;ANY ERRORS?
3824  010724  001403                            BEQ     3$              ;BRANCH IF NO
3825  010726  170127  000013                    LDFPS   #13             ;LOAD COMPLIMENT CC'S
3826  010732  000402                            BR      4$
3827  010734  170127  000033            3$:     LDFPS   #FMM+13         ;LOAD COMPLIMENT CC'S
3828  010740  171437  001160            4$:     MODF    $REG0,AC0       ;EXECUTE INSTRUCTION UNDER TEST
3829  010744  170200                            STFPS   R0              ;GET CC'S
3830  010746  173427  000000                    CMPF    #0,AC0          ;FRACTION PART ZERO?
3831  010752  170000                            CFCC
3832  010754  001405                            BEQ     5$              ;BRANCH IF YES
3833  010756  174037  001204                    STF     AC0,$TMP2       ;GET DATA
3834  010762  170437  001200                    CLRF    $TMP0           ;SAVE EXPECTED VALUE
3835  010766  104040                            ERROR   40              ;FRACTION DID NOT CLEAR
3836  010770  173527  046207            5$:     CMPF    #^O046207,AC1   ;INTEGER PORTION CORRECT?
3837  010774  170000                            CFCC
3838  010776  001410                            BEQ     6$              ;BRANCH IF YES
3839  011000  174137  001204                    STF     AC1,$TMP2       ;GET RESULT
3840  011004  012737  046207  001200            MOV     #46207,$TMP0    ;SAVE
```

```
3841  011012  005037  001202                CLR     $TMP1           ;EXPECTED VALUE
3842  011016  104041                       ERROR   41              ;INTEGER PORTION WRONG
3843                                  ;DATA OK - CHECK CC'S
3844  011020  042700  000020          6$:   BIC     #FMM,R0         ;GET RID OF FMM BIT
3845  011024  022700  000004                CMP     #FZ,R0          ;CC'S OK?
3846  011030  001410                       BEQ     TST17           ;;BRANCH IF YES
3847  011032  010037  001202                MOV     R0,$TMP1        ;SAVE RECEIVED VALUE
3848  011036  012737  000004  001200        MOV     #FZ,$TMP0       ;SAVE EXPECTED  VALUE
3849  011044  104003                       ERROR   3               ;CC'S BAD
3850  011046  022626                 2$:   CMP     (SP)+,(SP)+     ;RESTORE THE SP
3851  011050  104000                       ERROR                   ;STATE 327 DID NOT CAUSE BN TO CLEAR
3852
3853                                  ;;*****************************************************************
3854                                  ;*TEST 17       MODF*-(INT=0)*-(FRAC=0)
3855                                  ;*
3856                                  ;*      THIS TEST CHECKS THE FLOWS THAT GET EXECUTED WHEN THE
3857                                  ;*      RESULT CONTAINS AN INTEGER AND A FRACTION.
3858                                  ;*
3859                                  ;*      FPU ROM FLOW-MULTIPLY,76,266,327,307,5,2(332),336,170,105,172,NORMALIZE
3860                                  ;;*****************************************************************
3861  011052  000004                 TST17:  SCOPE
3862  011054  012737  000017  001240        MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
3863  011062  012737  011174  001222        MOV     #TST20,$ESCAPE  ;;ESCAPE TO TEST 20 ON ERROR
3864  011070  170127  000000                LDFPS   #0
3865  011074  005037  001202                CLR     $TMP1
3866  011100  012737  042001  001160        MOV     #42001,$REG0    ;PUT SRC DATA IN MEMORY
3867  011106  012737  100000  001162        MOV     #BIT15,$REG1    ;INT=200 AND FRAC=0.1
3868  011114  172627  040200                LDF     #^O040200,AC2   ;LOAD THE DST
3869  011120  171637  001160          1$:   MODF    $REG0,AC2       ;EXECUTE INSTRUCTION UNDER TEST
3870  011124  173627  040000                CMPF    #^O040000,AC2   ;FRACTION OK?
3871  011130  170000                       CFCC
3872  011132  001406                       BEQ     2$              ;BRANCH IF YES
3873  011134  174237  001204                STF     AC2,$TMP2       ;GET RECEIVED FRACTION
3874  011140  012737  040000  001200        MOV     #40000,$TMP0    ;SAVE EXPECTED FRACTION
3875  011146  104042                       ERROR   42              ;FRACTION WRONG
3876  011150  173727  042001          2$:   CMPF    #^O042001,AC3   ;INTEGER PORTION OK?
3877  011154  170000                       CFCC
3878  011156  001406                       BEQ     TST20           ;;BRANCH IF YES
3879  011160  174337  001204                STF     AC3,$TMP2       ;GET RECEIVED INTEGER
3880  011164  012737  042001  001200        MOV     #42001,$TMP0    ;SAVE EXPECTED INTEGER
3881  011172  104043                       ERROR   43              ;INTEGER WRONG
3882
3883                                  ;;*****************************************************************
3884                                  ;*TEST 20       MODF*BOU*-(FV*FIV)
3885                                  ;*
3886                                  ;*      THIS TEST ENSURES THAT AN INTEGER OVERFLOW ON MODF CAUSES
3887                                  ;*      THE INTEGER TO BE CLEARED.
3888                                  ;*
3889                                  ;*      FPU ROM FLOW-MULTIPLY,76,266,327,307,7,341,102,242,107
3890                                  ;;*****************************************************************
3891  011174  000004                 TST20:  SCOPE
3892  011176  012737  000020  001240        MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
3893  011204  012737  011306  001222        MOV     #TST21,$ESCAPE  ;;ESCAPE TO TEST 21 ON ERROR
3894  011212  012737  011220  001110        MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
3895  011220  172527  040000                LDF     #^O040000,AC1   ;ENSURE ACC+1 NOT ALREADY ZERO
3896  011224  172427  060000                LDF     #^O060000,AC0   ;LOAD THE DST
```

```
3897  011230  170127  000011                  LDFPS   #11                     ;LOAD COMPLEMENT CC'S
3898  011234  171427  060400          1$:     MODF    #^O060400,AC0           ;EXECUTE INSTRUCTION UNDER TEST
3899  011240  170200                          STFPS   R0                      ;GET CC'S
3900  011242  173527  000000                  CMPF    #0,AC1                  ;INTEGER CLEAR?
3901  011246  170000                          CFCC
3902  011250  001405                          BEQ     2$                      ;BRANCH IF YES
3903  011252  174137  001204                  STF     AC1,$TMP2               ;SAVE RECEIVED VALUE
3904  011256  170437  001200                  CLRF    $TMP0                   ;SAVE EXPECTED VALUE
3905  011262  104044                          ERROR   44                      ;INTEGER DID NOT CLEAR ON OVERFLOW
3906  011264  022700  000006          2$:     CMP     #FZ+FV,R0               ;CC'S OK?
3907  011270  001406                          BEQ     TST21                   ;;BRANCH IF YES
3908  011272  010037  001202                  MOV     R0,$TMP1                ;SAVE RECEIVED CC'S
3909  011276  012737  000006  001200          MOV     #FZ+FV,$TMP0            ;SAVE EXPECTED VALUE
3910  011304  104003                          ERROR   3                       ;CC'S BAD
3911
3912                                  ;;**********************************************************************
3913                                  ;*TEST 21       MODF*-(INT=0)*(FRAC=0)
3914                                  ;*
3915                                  ;*      THIS TEST ENSURES THAT ROM STATE 105 DETECTS A ZERO FRACTION.
3916                                  ;*      A MICRO TRAP IS SET ON STATE 343 (NORMALIZE) IN CASE
3917                                  ;*      THIS FAILS.  IF THIS FAILS, THE MICRO-BREAK REGISTER CANNOT BE
3918                                  ;*      LOADED FROM THE SWITCHES SINCE THE FPU WILL HANG IF IT GETS INTO
3919                                  ;*      THE NORMALIZE FLOWS.
3920                                  ;*
3921                                  ;*      NOTE:   SYNC POINT NOT SELECTABLE. DEFAULT=343
3922                                  ;*
3923                                  ;*      FPU ROM FLOW-MULTIPLY,76,266,327,307,5,332,336,170,105,172,341
3924                                  ;;**********************************************************************
3925  011306  000004          TST21:  SCOPE
3926  011310  012737  000021  001240          MOV     #STN-1,$TESTN           ;;SET TEST NUMBER IN  MAIL BOX
3927  011316  012737  011422  001222          MOV     #TST22,$ESCAPE          ;;ESCAPE TO TEST 22 ON ERROR
3928  011324  012737  040200  001160          MOV     #40200,$REG0            ;PUT THE SRC
3929  011332  005037  001162                  CLR     $REG1                   ;DATA IN MEMORY
3930  011336  012737  011416  000244          MOV     #2$,FPPVEC
3931  011344  012737  011352  001110          MOV     #.+6,@#$LPERR           ;SET ERROR LOOP
3932  011352  012703  000343                  MOV     #343,R3
3933  011356  170003                          LDUB
3934  011360  170127  000020                  LDFPS   #FMM
3935  011364  172427  040200                  LDF     #^O040200,AC0           ;LOAD THE DST
3936  011370  171437  001160          1$:     MODF    $REG0,AC0               ;EXECUTE INSTRUCTION UNDER TEST
3937  011374  173427  000000                  CMPF    #0,AC0                  ;DID FRACTION CLEAR?
3938  011400  170000                          CFCC
3939  011402  001407                          BEQ     TST22                   ;;BRANCH IF YES
3940  011404  174037  001204                  STF     AC0,$TMP2
3941  011410  170437  001200                  CLRF    $TMP0
3942  011414  104042                          ERROR   42                      ;FRACTION DID NOT CLEAR
3943  011416  022626                  2$:     CMP     (SP)+,(SP)+             ;RESTORE THE SP
3944  011420  104045                          ERROR   45                      ;STATE 105 NOT DETECTING ZERO FRACTION
3945
3946                                  ;;**********************************************************************
3947                                  ;*TEST 22       MODD*(INT=0)
3948                                  ;*
3949                                  ;*      THIS TEST ENSURES STATE 365 STORES ZERO IN THE
3950                                  ;*      DST ACC+1 AND THAT STATE 325 ROUNDS THE FRACTION.
3951                                  ;*
3952                                  ;*      FPU ROM FLOW-MULTIPLY,76,266,326,365,325,NORMALIZE
```

```
3953                                    ;;**********************************************************************
3954  011422  000004                    TST22:  SCOPE
3955  011424  012737  000022  001240            MOV     #STN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
3956  011432  012737  011636  001222            MOV     #TST23,$ESCAPE  ;;ESCAPE TO TEST 23 ON ERROR
3957  011440  012737  040177  001160            MOV     #40177,$REG0    ;PUT MULTIPLICAND
3958  011446  012737  177777  001162            MOV     #-1,$REG1       ;IN MEMORY
3959  011454  012737  177777  001164            MOV     #-1,$REG2       ;    *
3960  011462  012737  177776  001166            MOV     #-2,$REG3       ;    *
3961  011470  012737  140000  001170            MOV     #140000,$REG4   ;PUT MULTIPLIER
3962  011476  005037  001172                    CLR     $REG5           ;IN MEMORY
3963  011502  005037  001174                    CLR     $REG6           ;    *
3964  011506  012737  000040  001176            MOV     #40,$REG7       ;    *
3965  011514  012737  011522  001110            MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
3966  011522  170011                            SETD
3967  011524  172437  001160                    LDD     $REG0,AC0       ;LOAD THE MULTIPLICAND
3968  011530  012700  001170                    MOV     #$REG4,R0       ;PUT ADDRESS OF MULTIPLIER IN R0
3969  011534  172527  044034                    LDD     #40000,AC1      ;ENSURE ACC+1 NON ZERO
3970  011540  171420                    1$:     MODD    (R0)+,AC0       ;EXECUTE INSTRUCTION UNDER TEST
3971  011542  022700  001200                    CMP     #$REG7+2,R0     ;ADX ROM OK?
3972  011546  001401                            BEQ     2$              ;BRANCH IF YES
3973  011550  104004                            ERROR   4               ;ADX ROM FAILED
3974  011552  012737  140000  001200  2$:      MOV     #140000,$TMP0   ;SAVE
3975  011560  005037  001202                    CLR     $TMP1           ;EXPECTED VALUE
3976  011564  005037  001204                    CLR     $TMP2           ;OF DATA
3977  011570  012737  000037  001206            MOV     #37,$TMP3       ;    *
3978  011576  173437  001200                    CMPD    $TMP0,AC0       ;FRACTION OK?
3979  011602  170000                            CFCC
3980  011604  001403                            BEQ     3$              ;BRANCH IF YES
3981  011606  174037  001210                    STD     AC0,$TMP4       ;GET DATA
3982  011612  104046                            ERROR   46              ;FRACTION WRONG ON MODD
3983  011614  173527  000000            3$:     CMPD    #0,AC1          ;INTEGER 0?
3984  011620  170000                            CFCC
3985  011622  001405                            BEQ     TST23           ;;BRANCH IF YES
3986  011624  174137  001210                    STD     AC1,$TMP4       ;SAVE RECEIVED VALUE
3987  011630  170437  001200                    CLRD    $TMP0           ;SAVE EXPECTED VALUE
3988  011634  104047                            ERROR   47              ;INTEGER DID NOT CLEAR ON MODD
3989
3990                                    ;;**********************************************************************
3991                                    ;*TEST 23       MODD*-(INT=0)*-(FRAC=0)
3992                                    ;*
3993                                    ;*      THIS TEST ENSURES THAT ROM STATES 326 AND 327 ARE WORKING
3994                                    ;*      PROPERLY
3995                                    ;*
3996                                    ;*      FPU ROM FLOW-MULTIPLY,76,266,326,367,5,332,336,170,105,172,NORM
3997                                    ;;**********************************************************************
3998  011636  000004                    TST23:  SCOPE
3999  011640  012737  000023  001240            MOV     #STN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
4000  011646  012737  012034  001222            MOV     #TST24,$ESCAPE  ;;ESCAPE TO TEST 24 ON ERROR
4001  011654  012737  142177  001160            MOV     #142177,$REG0   ;PUT
4002  011662  012737  177777  001162            MOV     #-1,$REG1       ;MULTIPLIER
4003  011670  012737  177777  001164            MOV     #-1,$REG2       ;IN
4004  011676  012737  177777  001166            MOV     #-1,$REG3       ;MEMORY
4005  011704  012737  011712  001110            MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
4006  011712  170127  000200                    LDFPS   #FD
4007  011716  172427  040200                    LDD     #^O040200,AC0   ;LOAD THE MULTIPLICAND
4008  011722  171437  001160            1$:     MODD    $REG0,AC0       ;EXECUTE INSTRUCTION UNDER TEST
```

```
4009  011726  173527  142177                          CMPD    #10142177,AC1   ;INTEGER OK?
4010  011732  170000                                  CFCC
4011  011734  001414                                  BEQ     2$              ;BRANCH IF YES
4012  011736  012737  142177  001200                  MOV     #142177,$TMP0   ;SAVE
4013  011744  005037  001202                          CLR     $TMP1           ;EXPECTED
4014  011750  005037  001204                          CLR     $TMP2           ;VALUE
4015  011754  005037  001206                          CLR     $TMP3           ;     *
4016  011760  174137  001210                          STD     AC1,$TMP4       ;GET RECEIVED VALUE
4017  011764  104047                                  ERROR   47              ;INTEGER WRONG
4018  011766  012737  140177  001200  2$:             MOV     #140177,$TMP0   ;SAVE
4019  011774  012737  177777  001202                  MOV     #-1,$TMP1       ;EXPECTED
4020  012002  012737  177777  001204                  MOV     #-1,$TMP2       ;VALUE OF
4021  012010  012737  177400  001206                  MOV     #177400,$TMP3   ;FRACTION
4022  012016  173437  001200                          CMPD    $TMP0,AC0       ;FRACTION OK?
4023  012022  170000                                  CFCC
4024  012024  001403                                  BEQ     TST24           ;;BRANCH IF YES
4025  012026  174037  001210                          STD     AC0,$TMP4       ;SAVE RECEIVED DATA
4026  012032  104046                                  ERROR   46              ;FRACTION WRONG
4027
4028                                          ;;*****************************************************************
4029                                          ;*TEST 24       FALU LOGICAL "AND" TEST
4030                                          ;*
4031                                          ;*      THIS TEST CHECKS THE LOGICAL AND FUNCTION OF THE FALU.
4032                                          ;*      THIS IS DONE BY FLOATING A ONE THRU AR<57:3>, THUS
4033                                          ;*      MAKING THE RESULT OF THE MULTIPLY HAVE AN INTEGER OF 1.0
4034                                          ;*      (201*0.1) AND THE FRACTION WILL BE .5+PATTERN (200*0.1+PATTERN)
4035                                          ;*      WHERE PATTERN IS THE VALUE OF AR <56:3> FOR THE PARTICULAR LOOP.
4036                                          ;*
4037                                          ;*      THE SECOND SECTION ALSO FLOATS A ONE THRU AR<57:3>
4038                                          ;*      BUT, IN THIS TEST THE INTEGER WILL BE 1.0+COUNT (270*0.1+COUNT)
4039                                          ;*      AND THE FRACTION WILL BE ZERO.
4040                                          ;*****************************************************************
4041  012034  000004                          TST24:  SCOPE
4042  012036  012737  000024  001240                  MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
4043  012044  012737  012512  001222                  MOV     #TST25,$ESCAPE  ;;ESCAPE TO TEST 25 ON ERROR
4044  012052  012737  040300  001160                  MOV     #40300,$REG0    ;INITIALIZE
4045  012060  005037  001162                          CLR     $REG1           ;THE
4046  012064  005037  001164                          CLR     $REG2           ;MULTIPLICAND
4047  012070  012737  000001  001166                  MOV     #1,$REG3        ;     *
4048  012076  012737  040000  001200                  MOV     #40000,$TMP0    ;INITIALIZE
4049  012104  005037  001202                          CLR     $TMP1           ;CHECK DATA
4050  012110  005037  001204                          CLR     $TMP2           ;FOR FRACTION
4051  012114  012737  000002  001206                  MOV     #2,$TMP3        ;     *
4052  012122  012737  040200  001170                  MOV     #40200,$REG4    ;INITIALIZE
4053  012130  005037  001172                          CLR     $REG5           ;CHECK DATA
4054  012134  005037  001174                          CLR     $REG6           ;FOR INTEGER
4055  012140  005037  001176                          CLR     $REG7           ;   *
4056  012144  012737  012152  001110                  MOV     #.+6,@#SLPERR   ;SET ERROR LOOP
4057  012152  170011                          1$:     SETD
4058  012154  172437  001160                          LDD     $REG0,AC0       ;LOAD THE MULTIPLICAND
4059  012160  171427  040200                          MODD    #^O40200,AC0    ;EXECUTE TEST
4060  012164  173437  001200                          CMPD    $TMP0,AC0       ;FRACTION OK?
4061  012170  170000                                  CFCC
4062  012172  001403                                  BEQ     2$              ;BRANCH IF YES
4063  012174  174037  001210                          STD     AC0,$TMP4       ;GET RECEIVED DATA
4064  012200  104046                                  ERROR   46              ;LOGICAL "AND" FAILED IN FALU ON FRACTION
```

```
4065  012202  173527  040200          2$:     CMPD    #^O040200,AC1   ;INTEGER OK?
4066  012206  170000                          CFCC
4067  012210  001403                          BEQ     3$              ;BRANCH IF YES
4068  012212  174137  001210                  STD     AC1,$TMP4       ;GET RECEIVED DATA
4069  012216  104047                          ERROR   47              ;LOGICAL "AND" FAILED IN FALU ON INTEGER
4070  012220  032777  001000  166710  3$:     BIT     #SW9,@SWR       ;LOOP ON ERROR?
4071  012226  001403                          BEQ     4$              ;BRANCH IF NO
4072  012230  105737  001103                  TSTB    $ERFLG          ;ANY ERRORS YET?
4073  012234  001346                          BNE     1$              ;BRANCH IF YES
4074  012236  006137  001166          4$:     ROL     $REG3           ;SELECT
4075  012242  006137  001164                  ROL     $REG2           ;NEXT
4076  012246  006137  001162                  ROL     $REG1           ;MULTIPLICAND
4077  012252  106137  001160                  ROLB    $REG0           ;    *
4078  012256  052737  000100  001160          BIS     #BIT6,$REG0
4079  012264  000241                          CLC
4080  012266  006137  001206                  ROL     $TMP3           ;SELECT
4081  012272  006137  001204                  ROL     $TMP2           ;NEXT
4082  012276  006137  001202                  ROL     $TMP1           ;CHECK FRACTION
4083  012302  106137  001200                  ROLB    $TMP0           ;    *
4084  012306  100321                          BPL     1$              ;BRANCH IF NOT FINISHED
4085                                  ;;*****************************************************************
4086                                  ;SECTION - 2
4087  012310  012700  001160                  MOV     #$REG0,R0
4088  012314  012720  056000                  MOV     #56000,(R0)+    ;INITIALIZE
4089  012320  005020                          CLR     (R0)+           ;MULTIPLICAND
4090  012322  005020                          CLR     (R0)+           ;    *
4091  012324  012720  000001                  MOV     #1,(R0)+        ;        *
4092  012330  012720  056000                  MOV     #56000,(R0)+    ;INITIALIZE
4093  012334  005020                          CLR     (R0)+           ;INTEGER
4094  012336  005020                          CLR     (R0)+           ;CHECK
4095  012340  012710  000001                  MOV     #1,(R0)         ;DATA
4096  012344  005037  001200                  CLR     $TMP0           ;INITIALIZE
4097  012350  005037  001202                  CLR     $TMP1           ;FRACTION
4098  012354  005037  001204                  CLR     $TMP2           ;CHECK DATA
4099  012360  005037  001206                  CLR     $TMP3           ;    *
4100  012364  005001                          CLR     R1
4101  012366  012737  012374  001110          MOV     #.+6,@#$'PERR   ;SET ERROR LOOP
4102  012374  170011                  5$:     SETD
4103  012376  172437  001160                  LDD     $REG0,AC0       ;LOAD THE MULTIPLICAND
4104  012402  171427  040200                  MODD    #^O040200,AC0   ;EXECUTE THE TEST
4105  012406  173537  001170                  CMPD    $REG4,AC1       ;INTEGER OK?
4106  012412  170000                          CFCC
4107  012414  001403                          BEQ     6$              ;BRANCH IF YES
4108  012416  174137  001210                  STD     AC1,$TMP4       ;SAVE RECEIVED VALUE
4109  012422  104047                          ERROR   47              ;LOGICAL "AND" FAILED
4110  012424  173437  001200          6$:     CMPD    $TMP0,AC0       ;FRACTION OK?
4111  012430  170000                          CFCC
4112  012432  001403                          BEQ     7$              ;BRANCH IF YES
4113  012434  174037  001210                  STD     AC0,$TMP4       ;SAVE EXPECTED VALUE
4114  012440  104046                          ERROR   46              ;LOGICAL "AND" FAILED
4115  012442  032777  001000  166466  7$:     BIT     #SW9,@SWR       ;LOOP ON ERROR?
4116  012450  001403                          BEQ     8$              ;BRANCH IF NO
4117  012452  105737  001103                  TSTB    $ERFLG          ;ANY ERRORS?
4118  012456  001346                          BNE     5$              ;BRANCH IF YES
4119  012460  012700  001200          8$:     MOV     #$REG7+2,R0
4120  012464  006140                          ROL     -(R0)           ;SELECT NEXT
```

```
                                        ROL     -(R0)           ;CHECK
                                        ROL     -(R0)           ;INTEGER
                                        XOR     R1,-(R0)        ;DECREMENT R0
                                        ROLB    (R0)            ;    *
                                        ROL     -(R0)           ;SELECT
                                        ROL     -(R0)           ;NEXT
                                        ROL     -(R0)           ;MULTIPLICAND
                                        XOR     R1,-(R0)        ;DECREMENT R0 WITHOUT AFFECTING CARRY BIT
                                        ROLB    (R0)            ;
                                        BPL     5$              ;BRANCH IF NOT FINISHED

                                ;;**************************************************************
                                ;*TEST 25        DIVF*-M0*(DST=0)*-(SRC=0)
                                ;*
                                ;*      THIS TEST ENSURES THAT STATE 352 STORES ZERO IN THE RESULT AND THAT
                                ;*      THE A-BRANCH AND ADX ROMS ARE OK.
                                ;*
                                ;*      FPU ROM FLOW-11,130,73,352
                                ;**************************************************************
000000                  TST25:  SCOPE
012512  000004                  MOV     #STN-1,STESTN   ;;SET TEST NUMBER IN  MAIL BOX
012514  012737  000025  001240  MOV     #TST26,SESCAPE  ;;ESCAPE TO TEST 26 ON ERROR
012522  012737  012646  001222
                                ;;**************************************************************
012530  012737  040000  001160  MOV     #40000,SREG0    ;PUT THE
012536  005037  001162          CLR     SREG1           ;SRC IN MEMORY
012542  012737  012550  001110  MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
012550  170400                  CLRF    AC0             ;ENSURE DST ZERO
012552  012700  001160          MOV     #SREG0,R0       ;PUT ADDRESS OF SRC IN R0
012556  170127  000013          LDFPS   #13             ;LOAD COMPLIMENT CC'S
012562  174420          1$:     DIVF    (R0)+,AC0       ;EXECUTE INSTRUCTION UNDER TEST
012564  170201                  STFPS   R1              ;GET CC'S
012566  022700  001164          CMP     #SREG2,R0       ;ADX ROM OK?
012572  001401                  BEQ     2$              ;BRANCH IF YES
012574  104004                  ERROR   4               ;ADX ROM FAILED
012576  174037  001204  2$:     STF     AC0,STMP2       ;GET RESULT
012602  005737  001204          TST     STMP2           ;QUAD 3 OK?
012606  001003                  BNE     3$              ;BRANCH IF NO
012610  005737  001206          TST     STMP3           ;QUAD 2 OK?
012614  001403                  BEQ     4$              ;BRANCH IF YES
012616  170437  001200  3$:     CLRF    STMP0           ;SAVE EXPECTED VALUE
012622  104050                  ERROR   50              ;DST DID NOT CLEAR
012624  022701  000004  4$:     CMP     #FZ,R1          ;CC'S OK?
012630  001406                  BEQ     TST26           ;;BRANCH IF YES
012632  010137  001202          MOV     R1,STMP1        ;SAVE RECEIVED DATA
012636  012737  000004  001200  MOV     #FZ,STMP0       ;SAVE EXPECTED DATA
012644  104003                  ERROR   3               ;CC'S BAD

                                ;;**************************************************************
                                ;*TEST 26        INITIAL TESTS OF DIVF
                                ;*
                                ;*      THIS TEST ENSURE THAT SPECIFIC SIGNALS REQUIRED FOR THE DIV ARE
                                ;*      FUNCTIONING PROPERLY. EACH SECTION DESCRIBES THE SIGNAL(S) BEING
                                ;*      TESTED. A MICRO-TRAP IS SET ON STATE 14 TO TEST THESE SIGNALS.
                                ;*
                                ;*      NOTE:   SYNC POINT NOT SELECTABLE. DEFAULT=14
                                ;**************************************************************
```

```
 4177  012646  000004                    TST26:  SCOPE
 4178  012650  012737  000026  001240            MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
 4179  012656  012737  013442  001222            MOV     #TST27,$ESCAPE  ;;ESCAPE TO TEST 27 ON ERROR
 4180                                    ;;**************************************************************
 4181                                    ;SECTION 1
 4182                                    ;       TEST THAT FRLH HI QUOT GEN BIT GOES HIGH & GETS INSERTED INTO QR31.
 4183                                    ;;**************************************************************
 4184  012664  012737  040000  001160            MOV     #40000,$REG0    ;INITIALIZE
 4185  012672  005037  001162                    CLR     $REG1           ;THE SRC (DENOMINATOR)
 4186  012676  012737  012736  000244            MOV     #2$,FPPVEC
 4187  012704  012737  012712  001110            MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
 4188  012712  012703  000014                    MOV     #14,R3
 4189  012716  170003                            LDUB
 4190  012720  170127  000020                    LDFPS   #FMM
 4191  012724  172427  040000                    LDF     #^040000,AC0    ;INITIALIZE THE DST (NUMERATOR)
 4192  012730  174437  001160            1$:     DIVF    $REG0,AC0       ;EXECUTE TEST INSTRUCTION
 4193  012734  170000                            CFCC                    ;WAIT FOR TRAP
 4194  012736  022626                    2$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
 4195  012740  170007                            STQ0                    ;
 4196  012742  174037  001204                    STF     AC0,$TMP2       ;GET THE QR
 4197  012746  042737  177600  001204            BIC     #177600,$TMP2
 4198  012754  005037  001200                    CLR     $TMP0           ;SAVE EXPECTED
 4199  012760  012737  000010  001202            MOV     #BIT3,$TMP1     ;VALUE OF QR
 4200  012766  022737  000010  001206            CMP     #BIT3,$TMP3     ;DID QUOT BIT GET INSERTED?
 4201  012774  001430                            BEQ     4$              ;BRANCH IF YES
 4202  012776  022737  000017  001204            CMP     #17,$TMP2       ;DID FRHC FALU59 FAIL TO GET
 4203  013004  001001                            BNE     3$              ;TO FRLF AS A HIGH?
 4204  013006  104051                            ERROR   51              ;YES
 4205  013010                            3$:
 4206  013010  012704  000004                    MOV     #+4,R4          ;PUT THE SHIFT COUNT IN R4
 4207  013014  170004                            MSN                     ;SHIFT QR TO GET BIT 31
 4208  013016  170007                            STQ0
 4209  013020  174037  001210                    STF     AC0,$TMP4       ;GET QR
 4210  013024  022737  000001  001210            CMP     #1,$TMP4        ;DID FRHL DIV SHF SELECT WRONG?
 4211  013032  001004                            BNE     5$              ;BRANCH IF NO
 4212  013034  012737  000200  001202            MOV     #BIT7,$TMP1     ;SAVE EXPECTED DATA
 4213  013042  104052                            ERROR   52              ;FRHK DIV DONE STUCK LOW OR FRHC
 4214                                                                    ;FALU59 NOT GETTING TO FRHL DIV
 4215                                                                    ;NORM MUX AS A LOW
 4216  013044  005737  001206            5$:     TST     $TMP3           ;DID FRLF HI QUOT GEN BIT FAIL TO GO HI?
 4217  013050  001001                            BNE     .+4             ;BRANCH IF NO
 4218  013052  104062                            ERROR   62              ;FRLF HI QUOT GEN BIT NOT GOING HIGH
 4219  013054  104053                            ERROR   53              ;QR DATA BAD
 4220                                    ;;**************************************************************
 4221                                    ;SECTION - 2
 4222                                    ;       TEST FRHL MPY/DIV ADD(1)L GOING LOW, FRHC FALU59 GETTING TO FRHL AS A
 4223                                    ;       HIGH AND TO FRLH AS A LOW.
 4224                                    ;;**************************************************************
 4225  013056  012737  040100  001160    4$:     MOV     #40100,$REG0    ;INITIALIZE THE DENOMINATOR
 4226  013064  012737  013124  000244            MOV     #6$,FPPVEC
 4227  013072  012737  013100  001110            MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
 4228  013100  012703  000014                    MOV     #14,R3
 4229  013104  170003                            LDUB
 4230  013106  170127  000020                    LDFPS   #FMM
 4231  013112  172427  040000                    LDF     #^040000,AC0    ;LOAD THE NUMERATOR
 4232  013116  174437  001160                    DIVF    $REG0,AC0       ;EXECUTE TEST INSTRUCTION
```

```
4233  013122  170000                          CFCC                    ;WAIT FOR TRAP
4234  013124  022626                  6$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
4235  013126  012704  177776                  MOV     #-2,R4          ;PUT THE SHIFT COUNT IN R4
4236  013132  170004                          MSN                     ;SHIFT AR TO GET 59 & 58
4237  013134  170005                          STAD                    ;GET
4238  013136  174037  001214                  STF     AC0,$TMP6       ;AR DATA
4239  013142  042737  177600  001214          BIC     #177600,$TMP6
4240  013150  012704  000007                  MOV     #7,R4           ;PUT THE SHIFT COUNT IN R4
4241  013154  170004                          MSN                     ;SHIFT QR TO GET
4242  013156  012704  000004                  MOV     #4,R4           ;PUT THE SHIFT COUNT IN R4
4243  013162  170004                          MSN                     ;BIT 26
4244  013164  170007                          STQD                    ;GET QR
4245  013166  174037  001210                  STF     AC0,$TMP4       ;DATA
4246  013172  042737  177600  001210          BIC     #177600,$TMP4
4247  013200  022737  000137  001212          CMP     #137,$TMP5      ;QR OK?
4248  013206  001424                          BEQ     7$              ;BRANCH IF YES
4249  013210  005037  001200                  CLR     $TMP0           ;SAVE EXPECTED
4250  013214  012737  000137  001202          MOV     #137,$TMP1      ;QR DATA
4251  013222  022737  000377  001212          CMP     #377,$TMP5      ;FRLH HI QUOT GEN BIT FAIL TO GO LOW?
4252  013230  001001                          BNE     8$              ;BRANCH IF NO
4253  013232  104054                          ERROR   54              ;FRLH HI QUOT GEN BIT DID NOT GO LOW
4254  013234  005737  001212          8$:     TST     $TMP5
4255  013240  001001                          BNE     9$
4256  013242  104055                          ERROR   55              ;FRHC FALU59 NOT GETTING TO FRLF AS A LOW
4257  013244  022737  000057  001212  9$:     CMP     #57,$TMP5       ;DID FRHL DIV NORM MUX SELECT WRONG?
4258  013252  001001                          BNE     10$             ;BRANCH IF NO
4259  013254  104056                          ERROR   56              ;FRHC FALU59 NOT GETTING TO FRHL
4260                                                                  ;DIV NORM MUX AS A HIGH
4261  013256  104057                  10$:    ERROR   57              ;QR DATA BAD
4262                                  ;QR OK - CHECK THE AR
4263  013260  022737  000160  001214  7$:     CMP     #160,$TMP6      ;AR DATA OK?
4264  013266  001413                          BEQ     12$             ;BRANCH IF YES
4265  013270  012737  000160  001204          MOV     #160,$TMP2      ;SAVE EXPECTED
4266  013276  005037  001206                  CLR     $TMP3           ;DATA
4267  013302  022737  000020  001214          CMP     #20,$TMP6       ;DID FRHL MPY/DIV ADD(1) FAIL TO GO LOW?
4268  013310  001001                          BNE     11$             ;BRANCH IF NO
4269  013312  104060                          ERROR   60              ;FRHL MPY/DIV ADD(1)L DID NOT GO LOW
4270  013314  104013                  11$:    ERROR   13              ;AR DATA BAD
4271                                  ;;*****************************************************************
4272                                  ;SECTION - 3
4273                                  ;       CHECK THAT FRHL MPY/DIV ADD(1)L GOES HIGH WITH FALU59L ON A HIGH
4274                                  ;;*****************************************************************
4275  013316  012737  040000  001160  12$:    MOV     #40000,$REG0    ;INIT THE DENOMINATOR
4276  013324  012737  013364  000244          MOV     #13$,FPPVEC
4277  013332  012737  013340  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
4278  013340  012703  000014                  MOV     #14,R3
4279  013344  170003                          LDUB
4280  013346  170127  000020                  LDFPS   #FMM
4281  013352  172427  040100                  LDF     #^O040100,AC0   ;LOAD THE NUMERATOR
4282  013356  174437  001160                  DIVF    $REG0,AC0       ;EXECUTE THE TEST INSTRUCTION
4283  013362  170000                          CFCC                    ;WAIT FOR TRAP
4284  013364  022626                  13$:    CMP     (SP)+,(SP)+     ;RESTORE THE SP
4285  013366  012704  177776                  MOV     #-2,R4          ;PUT THE SHIFT COUNT IN R4
4286  013372  170004                          MSN                     ;GET AR BITS 59 & 58
4287  013374  170005                          STAD
4288  013376  174037  001214                  STF     AC0,$TMP6       ;GET AR DATA
```

```
                                        B1C     #177600,$TMP6
                                        CLR     $TMP3           ;SAVE EXPECTED
                                        CLR     $TMP2           ;RESULT
                                        TST     $TMP6           ;AR DATA OK?
                                        BEQ     TST27           ;.BRANCH IF YES
                                        CMP     #100,$TMP6      ;DID SUBTRACT OCCUR INSTEAD OF ADD?
                                        BNE     14$             ;BRANCH IF NO
                                        ERROR   63              ;FRHL MPY/DIV ADD(1)L NOT GOING HIGH
                                14$:    ERROR   13              ;AR DATA BAD

                                ;;*****************************************************************
                                ;*TEST 27       NORM NEG ENCODER
                                ;*
                                ;*      THIS TEST RUNS A COUNT PATTERN ACROSS THE INPUTS TO THE
                                ;*      NORM NEG ENCODED ON FRHK.
                                ;*
                                ;*      THIS IS DONE BY USING A NUMERATOR OF 0.1 AND DENOMINATOR
                                ;*      THAT COUNTS BETWEEN 0.10000001 AND 0.11111111.  THE EXECUTION
                                ;*      IS INTERRUPTED AT STATE 14 AND THE QR CHECKED FOR THE CORRECT
                                ;*      AMOUNT OF SHIFTS
                                ;*
                                ;*      THE PATTERNS FROM 1.00000000 TO 1.11111111 ARE NOT CHECKED BECAUSE
                                ;*      THEY ARE IMPOSSIBLE TO GENERATE.
                                ;*
                                ;*      NOTE:   SYNC POINT NOT SELECTABLE. DEFAULT=14
                                ;;*****************************************************************
013442  000004                TST27:  SCOPE
013444  012737  000027  001240        MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
013452  012737  014120  001222        MOV     #TST30,$ESCAPE  ;;ESCAPE TO TEST 30 ON ERROR
013460  012737  040001  001160        MOV     #40001,$REG0    ;INITIALIZE THE
013466  005037  001162                CLR     $REG1           ;DENOMINATOR
013472  012737  000377  001172        MOV     #377,$REG5
013500  012737  037777  001202        MOV     #37777,$TMP1    ;INITIALIZE
013506  005037  001200                CLR     $TMP0           ;CHECK DATA
013512  012702  000002                MOV     #2,R2           ;SET LOOP COUNT
013516  012700  000003                MOV     #3,R0           ;INITIALIZE CHANGE CHECK DATA WORD
013522  012701  000177                MOV     #127.,R1        ;SET SOB LOOP COUNT
013526  012737  013566  000244        MOV     #1$,FPPVEC
013534  012737  013542  001110        MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
013542  012703  000014                MOV     #14,R3
013546  170003                        LDUB
013550  170127  000020                LDFPS   #FMM
013554  172427  040000        4$:     LDF     #1040000,AC0    ;LOAD THE NUMERATOR
013560  174437  001160                DIVF    $REG0,AC0       ;EXECUTE INSTRUCTION
013564  170000                        CFCC                    ;WAIT FOR TRAP
013566  022626                1$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
013570  012704  000007                MOV     #7,R4           ;PUT THE SHIFT COUNT IN R4
013574  170004                        MSN                     ;SHIFT QR TO PUT
013576  012704  000004                MOV     #4,R4           ;PUT THE SHIFT COUNT IN R4
013602  170004                        MSN                     ;BIT 26 INTO BIT 35
013604  170007                        STQD
013606  174037  001204                STF     AC0,$TMP2       ;GET QR DATA
013612  042737  177600  001204        BIC     #177600,$TMP2   ;GET RID OF EXPONENT
013620  023737  001202  001206        CMP     $TMP1,$TMP3     ;SHIFT OK?
013626  001133                        BNE     2$              ;BRANCH IF NO
013630  032777  001000  165300        BIT     #SW9,@SWR       ;LOOP ON ERROR?
```

```
4345  013636  001403                        BEQ     3$              ;BRANCH IF NO
4346  013640  105737  001103                TSTB    $ERFLG          ;ANY ERRORS?
4347  013644  001343                        BNE     4$              ;BRANCH IF YES
4348  013646  005237  001160          3$:   INC     $REG0           ;SELECT NEXT DENOMINATOR
4349  013652  022737  040023  001160        CMP     #40023,$REG0    ;TIME TO CHANGE CHECK DATA?
4350  013660  001003                        BNE     12$             ;BRANCH IF NO
4351  013662  012737  001577  001202        MOV     #1577,$TMP1     ;CHANGE CHECK DATA
4352  013670  022737  037777  001202  12$:  CMP     #37777,$TMP1
4353  013676  001003                        BNE     20$
4354  013700  042737  000200  001202        BIC     #BIT7,$TMP1
4355  013706  022737  040053  001160  20$:  CMP     #40053,$REG0    ;TIME TO CHANGE CHECK DATA?
4356  013714  001003                        BNE     8$              ;BRANCH IF NO
4357  013716  012737  000577  001202        MOV     #577,$TMP1      ;CHANGE IT
4358  013724  120037  001160          8$:   CMPB    R0,$REG0        ;
4359  013730  001045                        BNE     5$
4360  013732  022737  003577  001202        CMP     #3577,$TMP1
4361  013740  001421                        BEQ     7$
4362  013742  022737  001577  001202        CMP     #1577,$TMP1
4363  013750  001421                        BEQ     9$
4364  013752  022737  000577  001202        CMP     #577,$TMP1
4365  013760  001421                        BEQ     10$
4366  013762  052737  000200  001202        BIS     #BIT7,$TMP1
4367  013770  006237  001202                ASR     $TMP1           ;SELECT NEXT CHECK DATA
4368  013774  042737  000200  001202        BIC     #BIT7,$TMP1
4369  014002  000413                        BR      11$
4370  014004  012737  001600  001202  7$:   MOV     #1600,$TMP1
4371  014012  000407                        BR      11$
4372  014014  012737  000600  001202  9$:   MOV     #600,$TMP1
4373  014022  000403                        BR      11$
4374  014024  012737  000200  001202  10$:  MOV     #200,$TMP1
4375  014032  042700  000001          11$:  BIC     #BIT0,R0        ;SELECT NEXT
4376  014036  006300                        ASL     R0              ;CHANGE CHECK DATA
4377  014040  052700  000001                BIS     #BIT0,R0        ;VALUE
4378  014044  005337  001172          5$:   DEC     $REG5
4379  014050  005301                        DEC     R1              ;CONTINUE
4380  014052  001240                        BNE     4$
4381                                  ;       CHECK LAST PATTERN OF 1.10000000
4382  014054  012737  040177  001160  6$:   MOV     #40177,$REG0
4383  014062  012737  100000  001162        MOV     #BIT15,$REG1
4384  014070  012737  000200  001202        MOV     #200,$TMP1
4385  014076  012701  000001                MOV     #1,R1
4386  014102  012737  000200  001172        MOV     #200,$REG5
4387  014110  005302                        DEC     R2
4388  014112  001220                        BNE     4$              ;CONTINUE
4389  014114  000401                        BR      TST30           ;;GO TO NEXT TEST
4390  014116  104064                  2$:   ERROR   64              ;FRHK NORM NEG ENCODER FAILED
4391
4392                                  ;;**************************************************************
4393                                  ;;*TEST 30       Q REG NORM SHIFT COUNT LOOK UP
4394                                  ;;*
4395                                  ;;*       THIS TEST FIRST CHECKS ROM STATE 4 AND ADDRESS 4 OF THE
4396                                  ;;*       NORM SHIFT COUNT LOOK-UP ROM. IF FRHK DIV DONE
4397                                  ;;*       IS STUCK HIGH THE FPP WILL HANG IN ROM STATE 14.
4398                                  ;;*
4399                                  ;;*       A COUNT PATTERN IS THEN RUN THRU BITS <57:50> OF THE QR TO
4400                                  ;;*       CHECK ALL LOCATIONS IN THE ROM.
```

```
                                        ;*
                                        ;*      IF AN ERROR OCCURS, THE TYPEOUT INDICATES THE ROM ADDRESS
                                        ;*      AND ROMOUT DATA THAT WAS BEING TESTED.
                                        ;*
                                        ;*      FPU ROM FLOW-11,130,73,342,X(14),4,NORMALIZE
                                        ;;*****************************************************************
014120  000004                        TST30:  SCOPE
014122  012737  000030  001240                MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
014130  012737  140200  001200                MOV     #140200,$TMP0   ;INITIALIZE THE
014136  170127  000000                        LDFPS   #0
014142  005037  001202                        CLR     $TMP1           ;NUMERATOR
014146  012737  014154  001110                MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
014154  172427  040200                        LDF     #^O040200,AC0   ;INITIALIZE THE NUMERATOR
014160  174437  001200                1$:     DIVF    $TMP0,AC0       ;EXECUTE INSTRUCTION UNDER TEST
014164  173427  140200                        CMPF    #^O0140200,AC0  ;RESULT OK?
014170  170000                        CFCC
014172  001403                        BEQ     2$                      ;BRANCH IF YES
014174  174037  001204                        STF     AC0,$TMP2       ;GET RESULT
014200  104065                        ERROR   65                      ;RESULT WRONG
                                        ;;*****************************************************************
014202  012737  040004  001200        2$:     MOV     #40004,$TMP0    ;INITIALIZE THE
014210  005037  001202                        CLR     $TMP1           ;DENOMINATOR
014214  012701  000200                        MOV     #BIT7,R1        ;INITIALIZE COUNT BIT
014220  012737  000010  001174                MOV     #10,$REG6
014226  012700  000377                        MOV     #255.,R0        ;SET LOOP COUNT
014232  012737  000001  001172                MOV     #1,$REG5
014240  012737  014246  001110                MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
014246  172437  001200                3$:     LDF     $TMP0,AC0       ;LOAD THE NUMERATOR
014252  174427  040200                        DIVF    #^O040200,AC0   ;EXECUTE INSTRUCTION UNDER TEST
014256  173437  001200                        CMPF    $TMP0,AC0       ;RESULT OK?
014262  170000                        CFCC
014264  001032                        BNE     4$                      ;BRANCH IF NO
014266  032777  001000  164642                BIT     #SW9,@SWR       ;LOOP ON ERROR?
014274  001403                        BEQ     5$                      ;BRANCH IF NO
014276  105737  001103                        TSTB    $ERFLG          ;ANY ERRORS?
014302  001361                        BNE     3$                      ;BRANCH IF YES
014304  005237  001172                5$:     INC     $REG5
014310  060137  001200                        ADD     R1,$TMP0        ;SELECT NEXT NUMERATOR
014314  105737  001200                        TSTB    $TMP0           ;DID FRACTION CHANGE SIGN?
014320  100012                        BPL     6$                      ;BRANCH IF NO
014322  042737  000200  001200                BIC     #BIT7,$TMP0     ;GET RID OF OVERFLOW BIT
014330  106037  001200                        RORB    $TMP0           ;SELECT NEXT
014334  006037  001202                        ROR     $TMP1           ;FRACTION
014340  006001                        ROR     R1                      ;SELECT NEXT COUNT BIT
014342  005337  001174                        DEC     $REG6
014346  077041                        6$:     SOB     R0,3$
014350  000404                        BR      TST31                   ;;GO TO NEXT TEST
014352  174037  001204                4$:     STF     AC0,$TMP2       ;SAVE RECEIVED DATA
014356  104066                        ERROR   66                      ;NORMALIZE ROM FAILED
014360  000751                        BR      5$

                                        ;;*****************************************************************
                                        ;*TEST 31       DIVD INITIAL TESTS
                                        ;*
                                        ;*      THIS TEST ENSURES THAT FRLH DLE PREC QT HI B(1) AND LO B(1)
                                        ;*      ARE FUNCTIONING PROPERLY.
```

```
                                        ;*      THIS IS DONE BY TRAPPING ON STATE 14 AND LOOKING AT THE QR
                                        ;*
                                        ;*      NOTE:   SYNC POINT NOT SELECTABLE. DEFAULT=14
                                        ;;*****************************************************************
014262  000004                    TST31:  SCOPE
014364  012737  000031  001240            MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
014372  012737  014724  001222            MOV     #TST32,$ESCAPE  ;;ESCAPE TO TEST 32 ON ERROR
014400  012737  040000  001160            MOV     #40000,$REG0    ;INITIALIZE
014406  005037  001162                    CLR     $REG1           ;THE DENOMINATOR
014412  005037  001164                    CLR     $REG2           ;    *
014416  005037  001166                    CLR     $REG3           ;    *
014422  012737  014464  000244            MOV     #2$,FPPVEC
014430  012737  014436  001110            MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
014436  012703  000014                    MOV     #14,R3
014442  170003                            LDUB
014444  012700  001160                    MOV     #$REG0,R0       ;PUT ADDRESS OF DENOM IN R0
014450  170127  000220                    LDFPS   #FD+FMM
014454  172427  040000                    LDD     #^O40000,AC0    ;LOAD THE NUMERATOR
014460  174420                    1$:     DIVD    (R0)+,AC0       ;EXECUTE INSTRUCTION UNDER TEST
014462  170000                            CFCC                    ;WAIT FOR TRAP
014464  022626                    2$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
014466  022700  001170                    CMP     #$REG4,R0       ;ADX ROM OK?
014472  001401                            BEQ     3$              ;BRANCH IF YES
014474  104004                            ERROR   4               ;ADX ROM FAILED
014476  170007                    3$:     STQ0
014500  174037  001210                    STD     AC0,$TMP4       ;GET QR DATA
014504  042737  177600  001210            BIC     #177600,$TMP4
014512  005037  001200                    CLR     $TMP0           ;SAVE EXPECTED
014516  005037  001202                    CLR     $TMP1           ;VALUE OF DATA
014522  005037  001204                    CLR     $TMP2           ;    *
014526  012737  000010  001206            MOV     #10,$TMP3       ;    *
014534  022737  000010  001216            CMP     #10,$TMP7       ;DID HI BIT GET INSERTED?
014542  001406                            BEQ     4$              ;BRANCH IF YES
014544  022737  000017  001216            CMP     #17,$TMP7       ;DID FRLF DLE PREC QT LO GO LOW?
014552  001001                            BNE     6$              ;BRANCH IF YES
014554  104067                            ERROR   67              ;FRLF DLE PREC QT LO DID NOT GO LOW
014556  104070                    6$:     ERROR   70              ;FRLH DLE PREC QT HI B(1) NOT GOING HIGH
014560  005737  001212            4$:     TST     $TMP5           ;DID SINGLE PREC LOGIC DISABLE>
014564  001401                            BEQ     5$              ;BRANCH IF YES
014566  104071                            ERROR   71              ;FRLF E21(12) NOT GOING HIGH
                                    ;NOW CHECK FRLH DLE PREC QT LO
014570  012737  040177  001160    5$:     MOV     #40177,$REG0    ;INIT THE DENOMINATOR
014576  012737  014636  000244            MOV     #7$,FPPVEC
014604  012737  014612  001110            MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
014612  012703  000014                    MOV     #14,R3
014616  170003                            LDUB
014620  170127  000220                    LDFPS   #FD+FMM
014624  172427  040176                    LDD     #^O40176,AC0    ;LOAD THE NUMERATOR
014630  174437  001160                    DIVD    $REG0,AC0       ;EXECUTE INSTRUCTION UNDER TEST
014634  170000                            CFCC                    ;WAIT FOR TRAP
014636  022626                    7$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
014640  012704  000003                    MOV     #3,R4           ;PUT THE SHIFT COUNT IN R4
014644  170004                            MSN                     ;GET QR BITS <2:0>
014646  170007                            STQ0
014650  174037  001210                    STD     AC0,$TMP4       ;GET QR DATA
014654  042737  177600  001210            BIC     #177600,$TMP4
```

```
                                MOV     #77,$TMP3
                                CMP     #77,$TMP7       ;QR DATA OK?
                                BEQ     TST32           ;;BRANCH IF YES
                                CMP     #177,$TMP7      ;DID HI BIT FAIL TO GO LO?
                                BNE     8$              ;BRANCH IF NO
                                ERROR   72              ;FRLH DOUBLE PREC QUOT HI BIT NOT GOING L
                        8$:     TST     $TMP7           ;DID LOW BIT GO HIGH?
                                BNE     9$              ;BRANCH IF YES
                                ERROR   73              ;FRLH DLE PREC QT LO DID NOT GO HIGH
                        9$:     ERROR   74              ;QR DATA BAD

                        ;;*****************************************************************
                        ;*TEST 32       FRLJ QSI -1,-2, & -3
                        ;*
                        ;*      THIS TEST CHECKS THE QSI -1,-2, & -3 MUX ON FRLH WITH
                        ;*      THE DIVIDE FUNCTION. THERE ARE FOUR CONDITIONS THAT
                        ;*      ARE CHECKED: 1) LEFT SHIFT 7*FALU59; 2) LEFT SHIFT 7*-FALU59;
                        ;*      3) LEFT SHIFT 3*FALU59;AND 4) LEFT SHIFT 3*-FALU59.
                        ;*      CONDITION 1 WAS TESTED IN THE PREVIOUS TEST.
                        ;*
                        ;*      NOTE:   SYNC POINT NOT SELECTABLE. DEFAULT=14
                        ;*****************************************************************
                        TST32:  SCOPE
                                MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
                                MOV     #TST33,$ESCAPE  ;;ESCAPE TO TEST 33 ON ERROR
                        ;SECTION - 1
                        ;LEFT SHIFT 7*-FALU59
                                MOV     #14,R3
                                LDUB
                                MOV     #1$,FPPVEC
                                LDFPS   #FD+FMM
                                LDD     #^O040000,AC0   ;LOAD THE NUMERATOR
                                DIVD    #^O040000,AC0   ;EXECUTE INSTRUCTION UNDER TEST
                                CFCC                    ;WAIT FOR TRAP
                        1$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
                                MOV     #3,R4           ;PUT THE SHIFT COUNT IN R4
                                MSN                     ;GET QR BITS 2:0
                                STQD
                                STD     AC0,$TMP4       ;GET QR DATA
                                BIC     #177600,$TMP4   ;GET RID OF EXPONENT
                                CLR     $TMP0           ;SAVE EXPECTED
                                CLR     $TMP1           ;VALUE OF QR
                                CLR     $TMP2           ;     *
                                MOV     #100,$TMP3      ;     *
                                CMP     #100,$TMP7      ;QR DATA OK?
                                BEQ     2$              ;BRANCH IF YES
                                CMP     #104,$TMP7      ;DID QSI-1 FAIL?
                                BNE     3$              ;BRANCH IF NO
                                ERROR   75              ;FRLJ QSI-1 DID NOT GO LOW WHEN SELECTING C1
                        3$:     ERROR   74              ;QR DATA BAD
                        ;;*****************************************************************
                        ;SECTION - 2
                        ;LEFT SHIFT 3*FALU59
                        2$:     MOV     #3,$TMP3        ;SAVE EXPECTED DATA
                                MOV     #4$,FPPVEC
                                MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
```

```
4569  015110  012703  000014                        MOV     #14,R3
4570  015114  170003                                LDUB
4571  015116  170127  000220                        LDFPS   #FD+FMM
4572  015122  172427  040100                        LDD     #^O040100,AC0   ;LOAD THE NUMERATOR
4573  015126  174427  040140                        DIVD    #^O040140,AC0   ;EXECUTE THE TEST
4574  015132  170000                                CFCC                    ;WAIT FOR TRAP
4575  015134  022626                        4$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
4576  015136  012704  000003                        MOV     #3,R4           ;PUT THE SHIFT COUNT IN R4
4577  015142  170004                                MSN                     ;GET QR BITS 2:0
4578  015144  170007                                STQD
4579  015146  174037  001210                        STD     AC0,$TMP4       ;GET QR DATA
4580  015152  042737  177600  001210                BIC     #177600,$TMP4   ;GET RID OF EXPONENT
4581  015160  022737  000003  001216                CMP     #3,$TMP7        ;QR DATA OK?
4582  015166  001406                                BEQ     5$              ;BRANCH IF YES
4583  015170  022737  000007  001216                CMP     #7,$TMP7        ;DID QSI-1 FAIL?
4584  015176  001001                                BNE     6$              ;BRANCH IF NO
4585  015200  104076                                ERROR   76              ;FRLJ QSI-1 NOT GOING LOW WHEN SELECTING A1
4586  015202  104074                        6$:     ERROR   74
4587                                        ;;*****************************************************************
4588                                        ;SECTION - 3
4589                                        ;LEFT SHIFT 3*-FALU59
4590  015204  012737  000004  001206        5$:     MOV     #4,$TMP3        ;SAVE EXPECTED VALUE OF QR DATA
4591  015212  012737  015252  000244                MOV     #7$,FPPVEC
4592  015220  012737  015226  001110                MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
4593  015226  012703  000014                        MOV     #14,R3
4594  015232  170003                                LDUB
4595  015234  170127  000220                        LDFPS   #FD+FMM
4596  015240  172427  040160                        LDD     #^O040160,AC0   ;LOAD THE NUMERATOR
4597  015244  174427  040140                        DIVD    #^O040140,AC0   ;EXECUTE TEST INSTRUCTION
4598  015250  170000                                CFCC                    ;WAIT FOR TRAP
4599  015252  022626                        7$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
4600  015254  012704  000003                        MOV     #3,R4           ;PUT THE SHIFT COUNT IN R4
4601  015260  170004                                MSN                     ;GET QR BITS 2:0
4602  015262  170007                                STQD
4603  015264  174037  001210                        STD     AC0,$TMP4       ;GET QR DATA
4604  015270  042737  177600  001210                BIC     #177600,$TMP4   ;GET RID OF EXPONENT
4605  015276  022737  000004  001216                CMP     #4,$TMP7        ;QR DATA OK?
4606  015304  001405                                BEQ     TST33           ;;BRANCH IF YES
4607  015306  005737  001216                        TST     $TMP7           ;DID QSI-1 FAIL?
4608  015312  001001                                BNE     8$              ;BRANCH IF NO
4609  015314  104077                                ERROR   77              ;FRLJ QSI-1 NOT GOING HIGH WHEN SELECTING A1
4610  015316  104074                        8$:     ERROR   74              ;QR DATA BAD
4611
4612                                        ;;*****************************************************************
4613                                        ;*TEST 33       DIVD*-MD*IMM
4614                                        ;*
4615                                        ;*      THIS TEST DIVIDES 127(10) BY 255(10) TO ENSURE THAT STATE
4616                                        ;*      246 IS FUNCTIONING PROPERLY.
4617                                        ;*
4618                                        ;*      FPU ROM FLOW-20,141,73,342,14,4,246,NORMALIZE
4619                                        ;;*****************************************************************
4620  015320  000004                        TST33:  SCOPE
4621  015322  012737  000033  001240                MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
4622  015330  012737  015450  001222                MOV     #TST34,$ESCAPE  ;;ESCAPE TO TEST 34 ON ERROR
4623  015336  012737  015444  000004                MOV     #2$,ERRVEC      ;SET ERRVEC TO CATCH ADX ROM FAILURE
4624  015344  012737  037776  001200                MOV     #37776,$TMP0    ;SAVE EXPECTED
```

```
 4625  015352  012737  177376  001202          MOV     #177376,$TMP1   ;VALUE OF RESULT
 4626  015360  012737  177376  001204          MOV     #177376,$TMP2   ;    *
 4627  015366  012737  177377  001206          MOV     #177377,$TMP3   ;    *
 4628  015374  012737  015402  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
 4629  015402  170127  000200                  LDFPS   #FD
 4630  015406  172427  041776                  LDD     #^D041776,AC0   ;LOAD THE NUMERATOR
 4631  015412  174427                  1$:     DIVD    (PC)+,AC0       ;EXECUTE INSTRUCTION UNDER TEST
 4632  015414  042177                          .WORD   42177           ;WILL EXECUTE THIS IF ADX ROM FAILS
 4633  015416  000403                          .WORD   403             ;THIS WILL CAUSE THE PREVIOUS WORD
 4634                                                                  ;TO ODD ADDRESS TRAP IF IT GETS EXEC.
 4635  015420  104004                          ERROR   4               ;ADX ROM FAILED
 4636  015422  104004                          ERROR   4               ;       *
 4637  015424  104004                          ERROR   4               ;       *
 4638  015426  173437  001200                  CMPD    $TMP0,AC0       ;RESULT OK?
 4639  015432  170000                          CFCC
 4640  015434  001405                          BEQ     TST34           ;;BRANCH IF YES
 4641  015436  174037  001210                  STD     AC0,$TMP4       ;GET RESULT
 4642  015442  104100                          ERROR   100             ;ROM STATE 246 FAILED.
 4643  015444  022626                  2$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
 4644  015446  104004                          ERROR   4               ;ADX ROM FAILED
 4645
 4646                                  ;;**********************************************************************
 4647                                  ;*TEST 34       STST
 4648                                  ;*
 4649                                  ;*      THIS TEST USES THE MICRO-BREAK TRAP TO CHECK THE
 4650                                  ;*      STST INSTRUCTION.  EACH SECTION OF THIS TEST CHECKS
 4651                                  ;*      A DIFFERENT MODE OF THE INSTRUCTION.
 4652                                  ;;**********************************************************************
 4653  015450  000004                  TST34:  SCOPE
 4654  015452  012737  000034  001240          MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
 4655  015460  012737  015666  001222          MOV     #TST35,$ESCAPE  ;;ESCAPE TO TEST 35 ON ERROR
 4656                                  ;;**********************************************************************
 4657                                  ;SECTION - 1 STST*-MD-IMM
 4658                                  ;FPU ROM FLOW-24,175,212
 4659  015466  012737  015474  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
 4660  015474  004737  034716                  JSR     PC,CHGFEC
 4661  015500  012700  001204                  MOV     #$TMP2,R0       ;PUT ADDRESS OF DST IN R0
 4662  015504  170320                          STST    (R0)+           ;EXECUTE INSTRUCTION UNDER TEST
 4663  015506  022700  001210                  CMP     #$TMP4,R0       ;ADX ROM OK?
 4664  015512  001401                          BEQ     3$              ;BRANCH IF YES
 4665  015514  104004                          ERROR   4               ;ADX ROM FAILED
 4666  015516  012737  000016  001200  3$:     MOV     #MT,$TMP0       ;SAVE EXPECTED FEC
 4667  015524  012737  034742  001202          MOV     #$$FEA,$TMP1    ;SAVE EXPECTED FEA
 4668  015532  022737  000016  001204          CMP     #MT,$TMP2       ;FEC OK?
 4669  015540  001401                          BEQ     4$              ;BRANCH IF YES
 4670  015542  104101                          ERROR   101             ;FEC WRONG
 4671  015544  022737  034742  001206  4$:     CMP     #$$FEA,$TMP3    ;FEA OK?
 4672  015552  001401                          BEQ     5$              ;BRANCH IF YES
 4673  015554  104102                          ERROR   102             ;FEA WRONG.
 4674                                  ;;**********************************************************************
 4675                                  ;SECTION - 2 STST*-MD*IMM
 4676                                  ;FPU ROM FLOW-24,175
 4677  015556  012737  015630  000010  5$:     MOV     #6$,RESVEC      ;SET RESVEC TO CATCH ADX ROM FAILURE
 4678  015564  012737  015572  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
 4679  015572  004737  034716                  JSR     PC,CHGFEC
 4680  015576  005037  015604                  CLR     7$
```

```
                                        STST    (PC)+           ;EXECUTE INSTRUCTION UNDER TEST
                                7$:     .WORD   0               ;WILL STORE 16 HERE & TRY AND EXEC IF ADX ROM FAILS
                                        BR      8$
                                        ERROR   4               ;ADX ROM FAILED
                                        ERROR   4               ;       *
                                        ERROR   4               ;       *
                                8$:     CMP     #MT,7$          ;INSTRUCTION OK?
                                        BEQ     9$              ;BRANCH IF YES
                                        ERROR   101
                                6$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
                                        ERROR   4               ;ADX ROM FAILED
                                ;;**************************************************************
                                ;SECTION - 3 STST*MD
                                ;FPU ROM FLOW-30,54,126
                                9$:
                                        MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
                                        JSR     PC,CHGFEC
                                        CLR     R0              ;INIT R0
                                        STST    R0              ;EXECUTE ISTRUCTION UNDER TEST
                                        CMP     #MT,R0          ;INSTRUCTION OK?
                                        BEQ     TST35           ;;BRANCH IF YES
                                        MOV     R0,$TMP2        ;SACE RECEIVED DATA
                                        ERROR   101

                                ;;**************************************************************
                                ;*TEST 35        FIUV
                                ;*
                                ;*      THIS TEST CHECKS THE 9 CONDITIONS THAT CAN CAUSE AN
                                ;*      INTERRUPT ON A MINUS ZERO
                                ;*      IF SECTION ONE FAILS TO INTERRUPT THE PROBLEM IS MOST
                                ;*      LIKELY ON FRMB.  ALL OTHER FAILURES SHOULD BE DUE TO THE
                                ;*      CONTROL STORE.
                                ;;**************************************************************
                                TST35:  SCOPE
                                        MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
                                        MOV     #TST36,$ESCAPE  ;;ESCAPE TO TEST 36 ON ERROR
                                ;SECTION - 1
                                ;LDAC6*FD(0)
                                        MOV     #100177,$REG0   ;PUT MINUS ZERO IN MEMORY
                                        MOV     #1$,FPPVEC
                                        MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
                                        JSR     PC,CHGFEC
                                        LDF     #^O40000,AC1    ;INIT AC1
                                        LDF     AC1,AC3         ;INIT AC3
                                        STF     AC1,AC5         ;INIT AC5
                                        LDFPS   #FIUV
                                2$:     ADDF    $REG0,AC0       ;EXECUTE INSTRUCTION THAT SHOULD TRAP
                                        CFCC                    ;WAIT FOR TRAP
                                ;FAILURE-NO TRAP OCCURRED
                                        ERROR   103
                                1$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
                                        STST    $TMP2           ;GET STATUS
                                        CMP     #FUV,$TMP2      ;FEC OK?
                                        BEQ     3$              ;BRANCH IF YES
                                        MOV     #FUV,$TMP0
                                        ERROR   104
```

```
4737  016004  022737  015746  001206  3$:     CMP     #2$,$TMP3       ;FEA OK?
4738  016012  001404                          BEQ     17$             ;BRANCH IF YES
4739  016014  012737  015746  001202          MOV     #2$,$TMP1
4740  016022  104105                          ERROR   105
4741  016024  173527  *040000          17$:    CMPF    #^O040000,AC1   ;AC1 REMAIN UNCHANGED?
4742  016030  170000                          CFCC
4743  016032  001010                          BNE     20$             ;BRANCH IF NO
4744  016034  173701                          CMPF    AC1,AC3         ;AC3 REMAIN UNCHANGED?
4745  016036  170000                          CFCC
4746  016040  001004                          BNE     21$             ;BRANCH IF NO
4747  016042  173505                          CMPF    AC5,AC1         ;AC5 REMAIN UNCHANGED?
4748  016044  170000                          CFCC
4749  016046  001403                          BEQ     4$              ;BRANCH IF YES
4750  016050  104154                  22$:    ERROR   154             ;FXPN ACD1 DID NOT GO HIGH
4751  016052  104155                  21$:    ERROR   155             ;FXPN ACD2 DID NOT GO HIGH
4752  016054  104156                  20$:    ERROR   156             ;FXPN ACF1*ACF2 DID NOT GO LOW
4753                                  ;;*****************************************************************
4754                                  ;SECTION - 2
4755                                  ;LDF*-MO
4756  016056  012737  016112  000244  4$:     MOV     #5$,FPPVEC
4757  016064  012737  016072  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
4758  016072  004737  034716                  JSR     PC,CHGFEC
4759  016076  170127  004000                  LDFPS   #FIUV
4760  016102  172437  001160                  LDF     $REGO,AC0       ;EXECUTE INSTRUCTION THAT SHOULD TRAP
4761  016106  170000                          CFCC
4762                                  ;FAILURE-NO TRAP
4763  016110  104106                          ERROR   106
4764  016112  022626                  5$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
4765                                  ;;*****************************************************************
4766                                  ;SECTION - 3
4767                                  ;LDAC6*FD(1)
4768  016114  012737  016150  000244          MOV     #6$,FPPVEC
4769  016122  012737  016130  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
4770  016130  004737  034716                  JSR     PC,CHGFEC
4771  016134  170127  004200                  LDFPS   #FIUV+FD
4772  016140  172437  001160                  LDD     $REGO,AC0       ;EXECUTE INSTRUCTION THAT SHOULD TRAP
4773  016144  170000                          CFCC
4774                                  ;FAILURE-NO TRAP
4775  016146  104106                          ERROR   106
4776  016150  022626                  6$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
4777                                  ;;*****************************************************************
4778                                  ;SECTION - 4
4779                                  ;LDD*-MO
4780  016152  012737  016206  000244          MOV     #7$,FPPVEC
4781  016160  012737  016166  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
4782  016166  004737  034716                  JSR     PC,CHGFEC
4783  016172  170127  004200                  LDFPS   #FIUV+FD
4784  016176  172437  001160                  LDD     $REGO,AC0       ;EXECUTE INSTRUCTION THAT SHOULD TRAP
4785  016202  170000                          CFCC
4786                                  ;FAILURE-NO TRAP
4787  016204  104106                          ERROR   106
4788  016206  022626                  7$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
4789                                  ;;*****************************************************************
4790                                  ;SECTION - 5
4791                                  ;LDCF*-MO
4792  016210  012737  016244  000244          MOV     #8$,FPPVEC
```

```
 4793  016216  012737  016224  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
 4794  016224  004737  034716                  JSR     PC,CHGFEC
 4795  016230  170127  004000                  LDFPS   #FIUV
 4796  016234  177437  001160                  LDCDF   $REG0,AC0       ;EXECUTE INSTRUCTION THAT SHOULD TRAP
 4797  016240  170000                          CFCC
 4798                                  ;FAILURE-NO TRAP
 4799  016242  104106                          ERROR   106
 4800  016244  022626                  8$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
 4801                                  ;;*****************************************************************
 4802                                  ;SECTION - 6
 4803                                  ;TSTX*-MD
 4804  016246  012737  016302  000244          MOV     #9$,FPPVEC
 4805  016254  012737  016262  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
 4806  016262  004737  034716                  JSR     PC,CHGFEC
 4807  016266  170127  004000                  LDFPS   #FIUV
 4808  016272  170537  001160                  TSTF    $REG0           ;EXECUTE THE INSTRUCTION THAT SHOULD TRAP
 4809  016276  170000                          CFCC
 4810                                  ;FAILURE-NO TRAP
 4811  016300  104106                          ERROR   106
 4812  016302  022626                  9$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
 4813                                  ;;*****************************************************************
 4814                                  ;SECTION - 7
 4815                                  ;NEGF*-MD
 4816  016304  012737  016346  000244          MOV     #10$,FPPVEC
 4817  016312  012737  016320  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
 4818  016320  004737  034716                  JSR     PC,CHGFEC
 4819  016324  012737  100177  001160          MOV     #100177,$REG0   ;INIT $REG0 WITH MINUS ZERO
 4820  016332  170127  004000                  LDFPS   #FIUV
 4821  016336  170737  001160                  NEGF    $REG0
 4822  016342  170000                          CFCC
 4823                                  ;FAILURE-TRAP DID NOT OCCUR
 4824  016344  104106                          ERROR   106
 4825  016346  022626                  10$:    CMP     (SP)+,(SP)+
 4826  016350  005737  001160                  TST     $REG0           ;DID DST GET NEGATTED?
 4827  016354  001403                          BEQ     12$             ;BRANCH IF YES
 4828  016356  005037  001200                  CLR     $TMP0
 4829  016362  104140                          ERROR   140             ;DON'T KNOW WHAT HAPPENED
 4830                                  ;;*****************************************************************
 4831                                  ;SECTION - 8
 4832                                  ;NEGF*-MD*IMM
 4833  016364  012737  016430  000244  12$:    MOV     #11$,FPPVEC
 4834  016372  012737  016400  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
 4835  016400  004737  034716                  JSR     PC,CHGFEC
 4836  016404  012737  100177  016420          MOV     #100177,13$     ;MAKE DATA A MINUS ZERO
 4837  016412  170127  004000                  LDFPS   #FIUV
 4838  016416  170727                          NEGF    (PC)+           ;EXECUTE INSTRUCTION
 4839  016420  100177                  13$:    .WORD   100177
 4840  016422  170000                          CFCC
 4841                                  ;FAILURE-NO TRAP
 4842  016424  170000                          CFCC
 4843  016426  104141                          ERROR   141             ;FRMB IMMEDIATE*REG WRITE NOT GOING LOW
 4844  016430  022626                  11$:    CMP     (SP)+,(SP)+     ;RESTORE THE SP
 4845  016432  005737  016420                  TST     13$             ;DATA OK?
 4846  016436  001406                          BEQ     14$             ;BRANCH IF YES
 4847  016440  005037  001200                  CLR     $TMP0           ;SAVE EXPECTED VALUE
 4848  016444  012737  016004  001160          MOV     #3$,$REG0       ;SAVE RECEIVED VALUE
```

```
                                        ERROR   140             ;DATA BAD
                                ;;**************************************************************
                                ;SECTION - 9
                                ;ABSF*-MD*IMM
016454  012737  016516  000244  14$:    MOV     #15$,FPPVEC     ;SET FP VECTOR
016462  012737  016470  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
016470  004737  034716                  JSR     PC,CHGFEC
016474  012737  100177  016510          MOV     #100177,16$     ;INIT DATA
016502  170127  004000                  LDFPS   #FIUV
016506  170627                          ABSF    (PC)+           ;EXECUTE INSTRUCTION UNDER TEST
016510  100177                  16$:    .WORD   100177
016512  170000                          CFCC                    ;WAIT FOR TRAP
                                ;FAILURE NO TRAP
016514  104106                          ERROR   106             ;NO TRAP
016516  022626                  15$:    CMP     (SP)+,(SP)+     ;RESTORE THE SP
016520  005737  016510                  TST     16$             ;DATA GET CLEARED?
016524  001406                          BEQ     TST36           ;;BRANCH IF YES
016526  013737  016510  001160          MOV     16$,$REGO       ;SAVE RECEIVED DATA
016534  005037  001200                  CLR     $TMP0           ;SAVE EXPECTED DATA
016540  104140                          ERROR   140             ;ABSF DID NOT CLEAR DATA

                                ;;**************************************************************
                                ;*TEST 36        FIU
                                ;*
                                ;*      THIS TEST CHECKS THE 4 PLACES WHERE AN UNDERFLOW TRAP
                                ;*      CAN OCCUR.
                                ;;**************************************************************
016542  000004                  TST36:  SCOPE
016544  012737  000036  001240          MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
016552  012737  017312  001222          MOV     #TST37,$ESCAPE  ;;ESCAPE TO TEST 37 ON ERROR
                                ;SECTION - 1
                                ;LDEXP*DIMUX<15>=1*DIMUX<14:7>=0
016560  012737  016620  000244          MOV     #1$,FPPVEC
016566  012737  016574  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
016574  004737  034716                  JSR     PC,CHGFEC
016600  172427  040077                  LDF     #^O40077,AC0    ;INITIALIZE AC0
016604  170127  002017                  LDFPS   #FIU+17         ;LOAD COMPLIMENT CC'S
016610  176427  100177          2$:     LDEXP   #100177,AC0     ;EXECUTE INSTRUCTION TO CAUSE TRAP
016614  170000                          CFCC
                                ;FAILURE-NO TRAP
016616  104107                          ERROR   107             ;UNDERFLOW DID NOT TRAP
016620  022626                  1$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
016622  170200                          STFPS   R0              ;GET CC'S
016624  170337  001204                  STST    $TMP2           ;GET FEC & FEA
016630  012737  000012  001200          MOV     #FU,$TMP0       ;SAVE EXPECTED
016636  012737  016610  001202          MOV     #2$,$TMP1       ;VALUE OF FEC & FEA
016644  022737  000012  001204          CMP     #FU,$TMP2       ;FEC OK?
016652  001401                          BEQ     3$              ;BRANCH IF YES
016654  104110                          ERROR   110             ;FEC WRONG ON UNDERFLOW
016656  022737  016610  001206  3$:     CMP     #2$,$TMP3       ;FEA OK?
016664  001401                          BEQ     4$              ;BRANCH IF YES
016666  104111                          ERROR   111             ;FEA WRONG ON UNDERFLOW
016670  174037  001204          4$:     STF     AC0,$TMP2       ;GET RESULT
016674  022737  037677  001204          CMP     #37677,$TMP2    ;DID EXPONENT LOAD CORRECTLY?
016702  001406                          BEQ     5$              ;BRANCH IF YES
016704  012737  037677  001200          MOV     #37677,$TMP0    ;SAVE EXPECTED
```

```
 4905  016712  005037  001202                CLR     $TMP1           ;VALUE OF RESULT
 4906  016716  104112                        ERROR   112             ;EXPONENT WRONG
 4907  016720  042700  102000        5$:     BIC     #FER+FIU,R0
 4908  016724  005700                        TST     R0              ;CC'S OK?
 4909  016726  001405                        BEQ     6$              ;BRANCH IF YES
 4910  016730  005037  001200                CLR     $TMP0           ;SAVE EXPECTED CC'S
 4911  016734  010037  001202                MOV     R0,$TMP1        ;SAVE RECEIVED CC'S
 4912  016740  104003                        ERROR   3               ;CC'S BAD
 4913                                ;;*****************************************************************
 4914                                ;SECTION - 2
 4915                                ;LDEXP*(EXP=-200)
 4916  016742  012737  017002  000244 6$:     MOV     #7$,FPPVEC
 4917  016750  012737  016756  001110        MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
 4918  016756  004737  034716                JSR     PC,CHGFEC
 4919  016762  172427  037677                LDF     #^O37677,AC0    ;INITIALIZE AC0
 4920  016766  170127  002013                LDFPS   #FIU+13         ;LOAD COMPLIMENT CC'S
 4921  016772  176427  177600                LDEXP   #177600,AC0     ;EXECUTE INSTRUCTION TO CAUSE TRAP
 4922  016776  170000                        CFCC
 4923                                ;FAILURE-NO TRAP
 4924  017000  104107                        ERROR   107
 4925  017002  022626                7$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
 4926  017004  170200                        STFPS   R0              ;GET CC'S BACK
 4927  017006  174037  001204                STF     AC0,$TMP2       ;GET DATA
 4928  017012  022737  000077  001204        CMP     #77,$TMP2       ;EXPONENT OK?
 4929  017020  001406                        BEQ     8$              ;BRANCH IF YES
 4930  017022  012737  000077  001200        MOV     #77,$TMP0       ;SAVE EXPECTED
 4931  017030  005037  001202                CLR     $TMP1           ;VALUE FO DATA
 4932  017034  104112                        ERROR   112             ;EXPONENT WRONG
 4933  017036  042700  102000        8$:     BIC     #FER+FIU,R0
 4934  017042  022700  000004                CMP     #FZ,R0          ;CC'S OK?
 4935  017046  001406                        BEQ     9$              ;BRANCH IF YES
 4936  017050  010037  001202                MOV     R0,$TMP1        ;SAVE RECEIVED VALUE
 4937  017054  012737  000004  001200        MOV     #FZ,$TMP0       ;SAVE EXPECTED VALUE
 4938  017062  104003                        ERROR   3               ;CC'S WRONG
 4939                                ;;*****************************************************************
 4940                                ;SECTION - 3
 4941                                ;LDEXP*(EXP<-200)
 4942  017064  012737  017124  000244 9$:     MOV     #10$,FPPVEC
 4943  017072  012737  017100  001110        MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
 4944  017100  004737  034716                JSR     PC,CHGFEC
 4945  017104  172427  137677                LDF     #^O137677,AC0   ;INITIALIZE AC0
 4946  017110  170127  002007                LDFPS   #FIU+7          ;LOAD COMPLIMENT CC'S
 4947  017114  176427  177577                LDEXP   #177577,AC0     ;EXECUTE INSTRUCTION TO CAUSE TRAP
 4948  017120  170000                        CFCC
 4949                                ;FAILURE-NO TRAP
 4950  017122  104107                        ERROR   107
 4951  017124  022626                10$:    CMP     (SP)+,(SP)+     ;RESTORE THE SP
 4952  017126  170200                        STFPS   R0              ;SAVE CC'S
 4953  017130  174037  001204                STF     AC0,$TMP2       ;GET DATA
 4954  017134  022737  137677  001204        CMP     #137677,$TMP2   ;EXPONENT OK?
 4955  017142  001406                        BEQ     11$             ;BRANCH IF YES
 4956  017144  012737  137677  001200        MOV     #137677,$TMP0   ;SAVE EXPECTED
 4957  017152  005037  001202                CLR     $TMP1           ;VALUE OF RESULT
 4958  017156  104112                        ERROR   112             ;RESULT WRONG
 4959  017160  042700  102000        11$:    BIC     #FER+FIU,R0
 4960  017164  022700  000010                CMP     #FN,R0          ;CC'S OK?
```

```
 4961  017170  001406                          BEQ     12$             ;BRANCH IF YES
 4962  017172  010037  001202                  MOV     R0,$TMP1        ;SAVE RECEIVED
 4963  017176  012737  000010  001200          MOV     #FN,$TMPO       ;SAVE EXPECTED VALUE
 4964  017204  104003                          ERROR   3               ;CC'S BAD
 4965                                  ;;**************************************************************
 4966                                  ;SECTION - 4
 4967                                  ;NORMALIZATION FLOW*EXP<-129
 4968  017206  012737  017246  000244  12$:    MOV     #13$,FPPVEC
 4969  017214  012737  017222  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
 4970  017222  004737  034716                  JSR     PC,CHGFEC
 4971  017226  172427  000377                  LDF     #^O377,AC0      ;INIT THE DST
 4972  017232  170127  002000                  LDFPS   #FIU
 4973  017236  171027  000377                  MULF    #^O377,AC0      ;EXECUTE INSTRUCTION TO CAUSE UNDERFLOW
 4974  017242  170000                          CFCC
 4975                                  ;FAILURE-NO TRAP
 4976  017244  104107                          ERROR   107
 4977  017246  022626                  13$:    CMP     (SP)+,(SP)+     ;RESTORE THE SP
 4978                                  ;;**************************************************************
 4979                                  ;SECTION - 5
 4980                                  ;NORMALIZATION FLOW*EXP=-129
 4981  017250  012737  017310  000244          MOV     #14$,FPPVEC
 4982  017256  012737  017264  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
 4983  017264  004737  034716                  JSR     PC,CHGFEC
 4984  017270  172427  000400                  LDF     #^O400,AC0      ;INITIALIZE THE DST
 4985  017274  170127  002000                  LDFPS   #FIU
 4986  017300  173027  000300                  SUBF    #^O300,AC0      ;EXECUTE INSTRUCTION TO CAUSE TRAP
 4987  017304  170000                          CFCC
 4988                                  ;FAILURE-NO TRAP
 4989  017306  104107                          ERROR   107
 4990  017310  022626                  14$:    CMP     (SP)+,(SP)+     ;RESTORE THE SP
 4991
 4992                                  ;;**************************************************************
 4993                                  ;*TEST 37        FIV
 4994                                  ;*
 4995                                  ;*      THIS TEST CHECKS THE 5 POSSIBLE OVERFLOW CONDITIONS.  IT
 4996                                  ;*      ALSO INCLUDES 3 ADDITIONAL CHECKS OF THE STCF TO ENSURE THE
 4997                                  ;*      CONTROL STORE IS FUNCTIONING PROPERLY.
 4998                                  ;;**************************************************************
 4999  017312  000004                  TST37:  SCOPE
 5000  017314  012737  000037  001240          MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
 5001  017322  012737  020036  001222          MOV     #TST40,$ESCAPE  ;;ESCAPE TO TEST 40 ON ERROR
 5002                                  ;SECTION - 1
 5003                                  ;LDEXP*(EXP > 177)
 5004  017330  004737  034716                  JSR     PC,CHGFEC
 5005  017334  012737  017362  000244          MOV     #1$,FPPVEC
 5006  017342  172427  040177                  LDF     #^O40177,AC0    ;INIT AC0
 5007  017346  170127  001017                  LDFPS   #FIV+17         ;LOAD COMPLIMENT CC'S
 5008  017352  176427  000377          4$:     LDEXP   #377,AC0        ;EXECUTE INSTRUCTION TO CAUSE TRAP
 5009  017356  170000                          CFCC
 5010                                  ;FAILURE-NO TRAP
 5011  017360  104113                          ERROR   113
 5012  017362  022626                  1$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
 5013  017364  170237  001202                  STFPS   $TMP1           ;GET CC'S
 5014  017370  174037  001204                  STF     AC0,$TMP2       ;GET DATA
 5015  017374  022737  037777  001204          CMP     #37777,$TMP2    ;EXP OK?
 5016  017402  001406                          BEQ     2$              ;BRANCH IF YES
```

```
5017  017404  012737  037777  001200          MOV     #37777,$TMP0    ;SAVE EXPECTED
5018  017412  005037  001202                  CLR     $TMP1           ;VALUE OF DATA
5019  017416  104114                          ERROR   114             ;EXPONENT WRONG
5020  017420  170337  001204          2$:     STST    $TMP2           ;GET FEC & FEA
5021  017424  022737  000010  001204          CMP     #F0,$TMP2       ;FEC OK?
5022  017432  001404                          BEQ     3$              ;BRANCH IF YES
5023  017434  012737  000010  001200          MOV     #F0,$TMP0       ;SAVE EXPECTED VALUE
5024  017442  104115                          ERROR   115             ;FEC WRONG ON OVERFLOW
5025  017444  022737  017352  001206  3$:     CMP     #4$,$TMP3       ;FEA OK?
5026  017452  001404                          BEQ     5$              ;BRANCH IF YES
5027  017454  012737  017352  001202          MOV     #4$,$TMP1       ;SAVE EXPECTED VALUE
5028  017462  104116                          ERROR   116             ;FEA WRONG ON OVERFLOW
5029  017464  042737  001000  001202  5$:     BIC     #FIV,$TMP1
5030  017472  022737  100002  001202          CMP     #FER+FV,$TMP1   ;CC'S OK?
5031  017500  001404                          BEQ     6$              ;BRANCH IF YES
5032  017502  012737  100002  001200          MOV     #FER+FV,$TMP0   ;SAVE EXPECTED RESULT
5033  017510  104003                          ERROR   3               ;CC'S BAD
5034                                  ;;**********************************************************************
5035                                  ;SECTION - 2
5036                                  ;MOD*(INTG>2**127)
5037  017512  012737  017552  000244  6$:     MOV     #7$,FPPVEC
5038  017520  012737  017526  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
5039  017526  004737  034716                  JSR     PC,CHGFEC
5040  017532  172427  077777                  LDF     #^077777,AC0    ;LOAD THE DEST
5041  017536  170127  001000                  LDFPS   #FIV
5042  017542  171427  077777                  MODF    #^077777,AC0    ;EXECUTE INSTRUCTION TO CAUSE TRAP
5043  017546  170000                          CFCC
5044                                  ;FAILURE-NO TRAP
5045  017550  104113                          ERROR   113             ;NO TRAP
5046  017552  022626                  7$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
5047  017554  170337  001204                  STST    $TMP2
5048  017560  022737  000010  001204          CMP     #F0,$TMP2       ;STATE 105 WORK OK?
5049  017566  001404                          BEQ     8$              ;BRANCH IF YES
5050  017570  012737  000010  001200          MOV     #F0,$TMP0       ;SAVE EXPECTED FEC
5051  017576  104117                          ERROR   117             ;FEC WRONG IN STATE 105
5052                                  ;;**********************************************************************
5053                                  ;SECTION - 3
5054                                  ;NORMALIZATION FLOW
5055  017600  012737  017640  000244  8$:     MOV     #9$,FPPVEC
5056  017606  012737  017614  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
5057  017614  004737  034716                  JSR     PC,CHGFEC
5058  017620  172427  077600                  LDF     #^077600,AC0    ;LOAD THE DST
5059  017624  170127  001000                  LDFPS   #FIV
5060  017630  172027  077600                  ADDF    #^077600,AC0    ;EXECUTE INSTRUCTION TO CAUSE TRAP
5061  017634  170000                          CFCC
5062                                  ;FAILURE-NO TRAP
5063  017636  104113                          ERROR   113             ;NO TRAP
5064  017640  022626                  9$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
5065                                  ;;**********************************************************************
5066                                  ;SECTION - 4
5067                                  ;STCF*MO
5068  017642  012737  017724  000244          MOV     #10$,FPPVEC
5069  017650  012737  077777  001160          MOV     #77777,$REG0    ;PUT # IN MEMORY
5070  017656  012737  177777  001162          MOV     #-1,$REG1       ;TO CAUSE OVERFLOW
5071  017664  012737  177777  001164          MOV     #-1,$REG2       ;WHEN CONVERTING
5072  017672  170011                          SETD
```

```
5073  017674  172437  001160                  LDD     $REG0,AC0       ;LOAD THE SRC ACC.
5074  017700  012737  017706  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
5075  017706  004737  034716                  JSR     PC,CHGFEC
5076  017712  170127  001200                  LDFPS   #FIV+FD
5077  017716  176001                          STCDF   AC0,AC1         ;EXECUTE INSTRUCTION TO CAUSE TRAP
5078  017720  170000                          CFCC                    ;WAIT FOR TRAP
5079                                  ;FAILURE, NO TRAP
5080  017722  104113                          ERROR   113             ;NO TRAP
5081  017724  022626                  10$:    CMP     (SP)+,(SP)+     ;RESTORE THE SP
5082                                  ;;**************************************************************
5083                                  ;SECTION - 5
5084                                  ;STCF*IMMEDIATE
5085  017726  012737  017770  000244          MOV     #11$,FPPVEC
5086  017734  170011                          SETD
5087  017736  172437  001160                  LDD     $REG0,AC0       ;LOAD THE SRC
5088  017742  012737  017750  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
5089  017750  004737  034716                  JSR     PC,CHGFEC
5090  017754  170127  001200                  LDFPS   #FIV+FD
5091  017760  176027                          STCDF   AC0,(PC)+       ;EXECUTE INSTRUCTION TO CAUSE TRAP
5092  017762  000000                          .WORD
5093  017764  170000                          CFCC
5094                                  ;FAILURE-NO TRAP
5095  017766  104113                          ERROR   113             ;NO TRAP
5096  017770  022626                  11$:    CMP     (SP)+,(SP)+     ;RESTORE THE SP
5097                                  ;;**************************************************************
5098                                  ;SECTION - 6
5099                                  ;STCF*-MO
5100  017772  012737  020034  000244          MOV     #12$,FPPVEC
5101  020000  170011                          SETD
5102  020002  172437  001160                  LDD     $REG0,AC0       ;LOAD THE SRC
5103  020006  012737  020014  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
5104  020014  004737  034716                  JSR     PC,CHGFEC
5105  020020  170127  001200                  LDFPS   #FIV+FD
5106  020024  176037  001204                  STCDF   AC0,$TMP2       ;EXECUTE INSTRUCTION TO CAUSE TRAP
5107  020030  170000                          CFCC
5108                                  ;FAILURE-NO TRAP
5109  020032  104113                          ERROR   113             ;NO TRAP
5110  020034  022626                  12$:    CMP     (SP)+,(SP)+     ;RESTORE THE SP
5111
5112                                  ;;**************************************************************
5113                                  ;*TEST 40        FIC
5114                                  ;*
5115                                  ;*      THIS TEST ENSURES THAT THE INTEGER CONVERSION
5116                                  ;*      INTERRUPT OCCURS FROM ALL 3 POSSIBLE ROM STATES.
5117                                  ;;**************************************************************
5118  020036  000004                  TST40:  SCOPE
5119  020040  012737  000040  001240          MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
5120  020046  012737  020244  001222          MOV     #TST41,$ESCAPE  ;;ESCAPE TO TEST 41 ON ERROR
5121                                  ;SECTION - 1
5122                                  ;STCI*MO*IL(0)
5123  020054  004737  034716                  JSR     PC,CHGFEC
5124  020060  012737  020104  000244          MOV     #1$,FPPVEC
5125  020066  172427  060000                  LDF     #^O60000,AC0    ;INIT THE SRC
5126  020072  170127  000400                  LDFPS   #FIC
5127  020076  175400                  3$:     STCFI   AC0,R0          ;EXECUTE INSTRUCTION TO CAUSE TRAP
5128  020100  170000                          CFCC
```

```
5129                                    ;FAILURE-NO TRAP
5130  020102  104120                            ERROR   120             ;NO TRAP ON CONVERSION ERROR
5131  020104  022626                    1$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
5132  020106  170337  001204                    STST    $TMP2
5133  020112  022737  000006  001204            CMP     #FICE,$TMP2     ;FEC OK?
5134  020120  001404                            BEQ     2$              ;BRANCH IF YES
5135  020122  012737  000006  001200            MOV     #FICE,$TMP0     ;SAVE EXPECTED VALUE
5136  020130  104121                            ERROR   121             ;FEC WRONG
5137  020132  022737  020076  001206    2$:     CMP     #3$,$TMP3       ;FEA OK?
5138  020140  001404                            BEQ     4$              ;BRANCH IF YES
5139  020142  012737  020076  001202            MOV     #3$,$TMP1       ;SAVE EXPECTED RESULT
5140  020150  104122                            ERROR   122             ;FEA WRONG
5141                                    ;;**************************************************************
5142                                    ;SECTION - 2
5143                                    ;STCI*M0*IL(1)
5144  020152  012737  020204  000244    4$:     MOV     #5$,FPPVEC
5145  020160  012737  020166  001110            MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
5146  020166  004737  034716                    JSR     PC,CHGFEC
5147  020172  170127  000500                    LDFPS   #FIC+FL
5148  020176  175400                            STCFI   AC0,R0          ;EXECUTE INSTRUCTION TO CAUSE TRAP
5149  020200  170000                            CFCC
5150                                    ;FAILURE-NO TRAP
5151  020202  104120                            ERROR   120             ;NO TRAP
5152  020204  022626                    5$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
5153                                    ;;**************************************************************
5154                                    ;SECTION - 3
5155                                    ;STCFI*-M0
5156  020206  012737  020242  000244            MOV     #6$,FPPVEC
5157  020214  012737  020222  001110            MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
5158  020222  004737  034716                    JSR     PC,CHGFEC
5159  020226  170127  000400                    LDFPS   #FIC
5160  020232  175437  001200                    STCFI   AC0,$TMP0       ;EXECUTE INSTRUCTION TO CAUSE TRAP
5161  020236  170000                            CFCC
5162                                    ;FAILURE-NO TRAP
5163  020240  104120                            ERROR 120               ;NO TRAP
5164  020242  022626                    6$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
5165
5166                                    ;;**************************************************************
5167                                    ;*TEST 41        DIVIDE BY ZERO
5168                                    ;*
5169                                    ;*      THIS TEST CHECKS THE TWO ROM STATES THAT CAN BE USED WHEN
5170                                    ;*      AN ATTEMPT TO DIVIDE BY ZERO IS MADE.
5171                                    ;;**************************************************************
5172  020244  000004                    TST41:  SCOPE
5173  020246  012737  000041  001240            MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
5174  020254  012737  020464  001222            MOV     #TST42,$ESCAPE  ;;ESCAPE TO TEST 42 ON ERROR
5175  020262  012737  020314  000244            MOV     #1$,FPPVEC
5176  020270  172427  000000                    LDF     #0,AC0          ;LOAD THE NUMERATOR WITH ZERO
5177  020274  012737  020302  001110            MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
5178  020302  004737  034716                    JSR     PC,CHGFEC
5179  020306  174427  000000            3$:     DIVF    #0,AC0          ;EXECUTE INSTRUCTION UNDER TEST
5180  020312  170000                            CFCC                    ;WAIT FOR TRAP
5181  020314  022626                    1$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
5182  020316  170337  001204                    STST    $TMP2           ;GET THE FP STATUS
5183  020322  022737  000004  001204            CMP     #4,$TMP2        ;FEC OK?
5184  020330  001404                            BEQ     2$              ;BRANCH IF YES
```

```
5185  020332  012737  000004  001200          MOV     #4,$TMP0        ;SAVE EXPECTED VALUE
5186  020340  104134                          ERROR   134             ;FEC WRONG
5187  020342  022737  020306  001206  2$:     CMP     #3$,$TMP3       ;FEA OK?
5188  020350  001404                          BEQ     4$              ;BRANCH IF YES
5189  020352  012737  020306  001202          MOV     #3$,$TMP1       ;SAVE EXPECTED VALUE
5190  020360  104135                          ERROR   135             ;FEA WRONG
5191                                  ;;**********************************************************************
5192  020362  012737  020402  000244  4$:     MOV     #5$,FPPVEC
5193  020370  170127  004000                  LDFPS   #FIUV
5194  020374  172427  100000                  LDF     #↑O100000,AC0   ;CHANGE FEC
5195  020400  170000                          CFCC                    ;WAIT FOR TRAP
5196  020402  022626                  5$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
5197  020404  172427  040000                  LDF     #↑O40000,AC0    ;ENSURE AC0 NON-ZERO
5198  020410  012737  020436  000244          MOV     #6$,FPPVEC
5199  020416  012737  020424  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
5200  020424  004737  034716                  JSR     PC,CHGFEC
5201  020430  174427  000000                  DIVF    #0,AC0          ;EXECUTE INSTRUCTION UNDER TEST
5202  020434  170000                          CFCC                    ;WAIT FOR TRAP
5203  020436  022626                  6$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
5204  020440  170337  001204                  STST    $TMP2           ;GET FEC
5205  020444  022737  000004  001204          CMP     #4,$TMP2        ;FEC OK?
5206  020452  001404                          BEQ     TST42           ;;BRANCH IF YES
5207  020454  012737  000004  001200          MOV     #4,$TMP0        ;SAVE EXPECTED VALUE
5208  020462  104136                          ERROR   136             ;FEC WRONG
5209
5210                                  ;;**********************************************************************
5211                                  ;*TEST 42        ILLEGAL OP-CODES
5212                                  ;*
5213                                  ;*      THIS TEST FIRST ENSURES THAT THE ILLEGAL OP-CODE TRAP FUNCTIONS
5214                                  ;*      PROPERLY.  IT THEN CHECKS ALL THE POSSIBLE ADDRESSES OF THE
5215                                  ;*      NO-MEM ROM THAT FORCE AN ILLEGAL OP CODE TRAP.
5216                                  ;*
5217                                  ;*      LAST, IT CHECKS THE FID BIT TO ENSURE IT FUNCTIONS PROPERLY.
5218                                  ;;**********************************************************************
5219  020464  000004                  TST42:  SCOPE
5220  020466  012737  000042  001240          MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
5221  020474  012737  020736  001222          MOV     #TST43,$ESCAPE  ;;ESCAPE TO TEST 43 ON ERROR
5222  020502  004737  034716                  JSR     PC,CHGFEC
5223  020506  012737  020530  000244          MOV     #1$,FPPVEC
5224  020514  170006                  2$:     .WORD   170006          ;EXECUTE THE ILL-OP CODE
5225  020516  000240                          NOP
5226                                  ;FAILURE-NO TRAP
5227  020520  012737  170006  001200          MOV     #170006,$TMP0   ;SAVE OP CODE THAT FAILED
5228  020526  104123                          ERROR   123             ;NO-MEM ROM FAILED
5229  020530  022626                  1$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
5230  020532  170337  001204                  STST    $TMP2           ;GET STATUS
5231  020536  022737  000002  001204          CMP     #2,$TMP2        ;FEC OK?
5232  020544  001404                          BEQ     3$              ;BRANCH IF YES
5233  020546  012737  000002  001200          MOV     #2,$TMP0        ;SAVE EXPECTED VALUE
5234  020554  104124                          ERROR   124             ;FEC WRONG ON ILLEGAL OP-CODE
5235  020556  022737  020514  001206  3$:     CMP     #2$,$TMP3       ;FEA OK?
5236  020564  001404                          BEQ     4$              ;BRANCH IF YES
5237  020566  012737  020514  001202          MOV     #2$,$TMP1       ;SAVE EXPECTED FEA
5238  020574  104125                          ERROR   125             ;FEA WRONG ON ILL OP CODE
5239                                  ;;**********************************************************************
5240                                  ;CHECK THE NO-MEM ROM
```

```
5241  020576  012737  020630  000244  4$:     MOV     #5$,FPPVEC
5242  020604  012700  002000                  MOV     #ILLOP,R0       ;PUT ADDRESS OF TBL OF OP-CODES IN R0
5243  020610  012737  020616  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
5244  020616  011037  020622          8$:     MOV     (R0),6$         ;LOAD THE ILLEGAL OP CODE
5245  020622  000000                  6$:     .WORD                   ;EXECUTE THE OP-CODE
5246  020624  000240                          NOP
5247  020626  000412                          BR      7$              ;DID NOT TRAP
5248  020630  022626                  5$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
5249  020632  105737  001103                  TSTB    $ERFLG          ;ANY ERRORS?
5250  020636  001371                          BNE     6$              ;BRANCH IF YES
5251  020640  062700  000002                  ADD     #2,R0           ;SELECT NEXT OP-CODE
5252  020644  022700  002112                  CMP     #ILLEND,R0      ;DONE?
5253  020650  001362                          BNE     8$              ;BRANCH IF NO
5254  020652  000406                          BR      9$
5255  020654  012706  001100          7$:     MOV     #1100,SP
5256  020660  013737  020622  001200          MOV     6$,$TMP0        ;SAVE OP-CODE THAT FAILED
5257  020666  104123                          ERROR   123             ;NO TRAP
5258  020670  012737  020710  000244  9$:     MOV     #10$,FPPVEC
5259  020676  012737  020704  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
5260  020704  176006                          STCFD   AC0,AC6         ;EXECUTE ILLEGAL INSTRUCTION
5261  020706  170000                          CFCC                    ;WAIT FOR TRAP
5262  020710  022626                  10$:    CMP     (SP)+,(SP)+     ;RESTORE THE SP
5263  020712  170337  001204                  STST    $TMP2           ;CHECK ROM STATE 32
5264  020716  022737  000002  001204          CMP     #2,$TMP2        ;IS IT OK?
5265  020724  001404                          BEQ     TST43           ;;BRANCH IF YES
5266  020726  012737  000002  001200          MOV     #2,$TMP0
5267  020734  104130                          ERROR   130             ;STATE 32 DID NOT LOAD FEC
5268
5269                                  ;;*****************************************************************
5270                                  ;*TEST 43       NO-MEM ROM
5271                                  ;*
5272                                  ;*      THIS TEST COMPLETES THE TEST OF THE NO-MEM ROM.  IT
5273                                  ;*      CONSISTS OF TESTING THOSE OP-CODES THAT CAN USE R6 OR R7
5274                                  ;*      AS LEGAL REGISTERS.
5275                                  ;;*****************************************************************
5276  020736  000004          TST43:  SCOPE
5277  020740  012737  000043  001240          MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
5278  020746  012737  021304  001222          MOV     #TST44,$ESCAPE  ;;ESCAPE TO TEST 44 ON ERROR
5279  020754  012737  021264  000244          MOV     #1$,FPPVEC
5280                                  ;SECTION - 1
5281                                  ;LDFPS
5282  020762  012706  001100                  MOV     #STACK,SP       ;INITIALIZE THE SP
5283  020766  170106                          LDFPS   R6              ;EXECUTE INSTRUCTION UNDER TEST
5284  020770  170237  001202                  STFPS   $TMP1           ;GET DATA BACK
5285  020774  022737  001100  001202          CMP     #1100,$TMP1     ;DATA OK?
5286  021002  001404                          BEQ     2$              ;BRANCH IF YES
5287  021004  012737  001100  001200          MOV     #1100,$TMP0     ;SAVE EXPECTED VALUE
5288  021012  104126                          ERROR   126             ;LEGAL OP-CODE FAILED
5289                                  ;;*****************************************************************
5290                                  ;SECTION - 2
5291                                  ;STFPS
5292  021014                          2$:
5293  021014  012737  021022  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
5294  021022  170127  001056                  LDFPS   #1056           ;INITIALIZE FPS
5295  021026  170206                          STFPS   R6              ;EXECUTE INSTRUCTION
5296  021030  022706  001056                  CMP     #1056,R6        ;WORK OK?
```

```
5297  021034  001410                        BEQ     3$                      ;BRANCH IF YES
5298  021036  010637  001202                MOV     R6,$TMP1                ;SAVE RECEIVED VALUE
5299  021042  012737  001056  001200        MOV     #1056,$TMPO             ;SAVE EXPECTED VALUE
5300  021050  012706  001100                MOV     #STACK,SP
5301  021054  104126                        ERROR   126                     ;LEGAL OP-CODE FAILED
5302                                ;;*****************************************************************
5303                                ;SECTION - 3
5304                                ;STST
5305  021056                        3$:
5306  021056  012737  021064  001110        MOV     #.+6,@#$LPERR           ;SET ERROR LOOP
5307  021064  170306                        STST    R6                      ;EXECUTE INSTRUCTION FEC=2
5308  021066  010637  001202                MOV     R6,$TMP1                ;SAVE DATA
5309  021072  012706  001100                MOV     #STACK,R6               ;RESTORE R6
5310  021076  022737  000002  001202        CMP     #2,$TMP1                ;DATA OK?
5311  021104  001404                        BEQ     4$                      ;BRANCH IF YES
5312  021106  012737  000002  001200        MOV     #2,$TMPO                ;SAVE EXPECTED VALUE
5313  021114  104126                        ERROR   126                     ;LEGAL OP CODE FAILED
5314                                ;;*****************************************************************
5315                                ;SECTION - 4
5316                                ;STCFI
5317  021116                        4$:
5318  021116  012737  021124  001110        MOV     #.+6,@#$LPERR           ;SET ERROR LOOP
5319  021124  177027  001076                LDCIF   #1076,AC0               ;INITIALIZE AC0
5320  021130  175406                        STCFI   AC0,R6                  ;EXECUTE INSTRUCTION
5321  021132  022706  001076                CMP     #1076,SP                ;DATA OK?
5322  021136  001410                        BEQ     5$                      ;BRANCH IF YES
5323  021140  010637  001202                MOV     R6,$TMP1                ;SAVE RECEIVED DATA
5324  021144  012737  001076  001200        MOV     #1076,$TMPO             ;SAVE EXPECTED DATA
5325  021152  012706  001100                MOV     #STACK,SP               ;RESTORE SP
5326  021156  104126                        ERROR   126                     ;LEGAL OP CODE FAILED
5327                                ;;*****************************************************************
5328                                ;SECTION - 5
5329                                ;LDEXP
5330  021160                        5$:
5331  021160  012737  021166  001110        MOV     #.+6,@#$LPERR           ;SET ERROR LOOP
5332  021166  012706  177776                MOV     #177776,SP              ;PUT EXPONENT TO LOAD IN SP
5333  021172  176406                        LDEXP   R6,AC0                  ;EXECUTE INSTRUCTION
5334  021174  012706  001100                MOV     #STACK,SP               ;RESTORE THE SP
5335  021200  175037  001202                STEXP   AC0,$TMP1               ;GET EXPONENT BACK
5336  021204  022737  177776  001202        CMP     #-2,$TMP1               ;DATA OK?
5337  021212  001404                        BEQ     6$                      ;BRANCH IF YES
5338  021214  012737  177776  001200        MOV     #-2,$TMPO               ;SAVE EXPECTED VALUE
5339  021222  104126                        ERROR   126                     ;LEGAL OP-CODE FAILED
5340                                ;;*****************************************************************
5341                                ;SECTION-6
5342                                ;LDCIF
5343  021224                        6$:
5344  021224  012737  021232  001110        MOV     #.+6,@#$LPERR           ;SET ERROR LOOP
5345  021232  012706  001100                MOV     #STACK,SP               ;INITIALIZE THE SP
5346  021236  177006                        LDCIF   R6,AC0                  ;EXECUTE INSTRUCTION
5347  021240  175437  001202                STCFI   AC0,$TMP1               ;GET DATA BACK
5348  021244  022737  001100  001202        CMP     #1100,$TMP1             ;DATA OK?
5349  021252  001414                        BEQ     TST44                   ;;BRANCH IF YES
5350  021254  012737  001100  001200        MOV     #1100,$TMPO             ;SAVE EXPECTED VALUE
5351  021262  104126                        ERROR   126                     ;LEGAL OP-CODE FAILED
5352  021264  011646                1$:     MOV     (SP),-(SP)
```

```
5353  021266  162716  000002                  SUB     #2,(SP)
5354  021272  013637  001200                  MOV     @(SP)+,$TMP0
5355  021276  012706  001100                  MOV     #STACK,SP
5356  021302  104127                          ERROR   127             ;LEGAL OP CODE TRAPPED
5357
5358                                  ;;*****************************************************************
5359                                  ;*TEST 44       FP PRIORITY ARBITRATION
5360                                  ;*
5361                                  ;*      THIS TEST CHECKS THE FLOATING POINT TRAP PRIORITY ARBITRATION
5362                                  ;*      LOGIC ON TMCA FOR PIR7 AND STACK LIMIT YELLOW.  THE
5363                                  ;*      MEMORY MANAGEMENT ARBITRATION IS PERFORMED LATER.
5364                                  ;;*****************************************************************
5365  021304  000004                  TST44:  SCOPE
5366  021306  012737  000044  001240          MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
5367  021314  012737  021532  001222          MOV     #TST45,$ESCAPE  ;;ESCAPE TO TEST 45 ON ERROR
5368                                  ;SECTION - 1 --- PIR 7
5369  021322  012737  021366  000244          MOV     #2$,FPPVEC      ;SETUP
5370  021330  012737  021366  000240          MOV     #2$,PIRQVEC     ;VECTORS
5371  021336  012706  001100                  MOV     #1100,SP        ;MAKE SURE SP=1100
5372  021342  012737  021350  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
5373  021350  000237                          SPL     7               ;SET CPU AT LEVEL 7
5374  021352  012737  100000  177772          MOV     #BIT15,PIRQ     ;SET PIR 7
5375  021360  170016                          .WORD   170016          ;EXECUTE ILLEGAL OP CODE
5376  021362  000230                          SPL     0               ;LOWER PRIORITY
5377  021364  000240                          NOP                     ;WAIT FOR TRAP
5378  021366  022706  001074          2$:     CMP     #1074,SP        ;DID ONLY ONE TRAP OCCUR?
5379  021372  001405                          BEQ     1$              ;BRANCH IF YES
5380  021374  012706  001100                  MOV     #1100,SP        ;RESTORE THE SP
5381  021400  005037  177772                  CLR     PIRQ            ;ENSURE PIRQ OFF
5382  021404  104131                          ERROR   131             ;FP TRAP DID NOT DISABLE
5383                                                                  ;PIR7 ON TMCA.
5384                                  ;SECTION - 2
5385                                  ;STACK LIMIT YELLOW
5386  021406  012706  001100          1$:     MOV     #1100,SP        ;RESTORE THE SP
5387  021412  005037  177772                  CLR     PIRQ            ;ENSURE PIRQ REG CLEAR
5388  021416  012737  021452  000244          MOV     #3$,FPPVEC      ;SETUP
5389  021424  012737  021452  000004          MOV     #3$,ERRVEC      ;VECTORS
5390  021432  012737  021440  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
5391  021440  012706  000400                  MOV     #400,SP         ;SET SP TO YELLOW ZONE BOUNDRY
5392  021444  170016                          .WORD   170016          ;EXECUTE INSTR TO CAUSE FP TRAP
5393  021446  005046                          CLR     -(SP)           ;EXECUTE INSTR TO CAUSE YEL ZONE TRAP
5394  021450  000240                          NOP
5395  021452  022706  000362          3$:     CMP     #362,SP         ;DID CORRECT NO. OF TRAPS OCCUR?
5396  021456  001416                  6$:     BEQ     4$              ;BRANCH IF YES
5397  021460  022706  000372                  CMP     #372,SP         ;DID FLOATING POINT FAIL TO TRAP?
5398  021464  001003                          BNE     5$              ;BRANCH IF NO
5399  021466  012706  001100                  MOV     #STACK,SP
5400  021472  104132                          ERROR   132             ;YELLOW ZONE DID NOT DISABLE
5401                                                                  ;TMCA HONOR FP
5402  021474  010637  001202          5$:     MOV     SP,$TMP1        ;SAVE SP
5403  021500  012737  000362  001200          MOV     #362,$TMP0      ;SAVE EXPECTED VALUE
5404  021506  012706  001100                  MOV     #STACK,SP       ;RESTORE THE SP
5405  021512  104142                          ERROR   142             ;THE SP IS NOT WHAT I EXPECTED
5406  021514  012706  001100          4$:     MOV     #STACK,SP
5407  021520  022737  021456  000366          CMP     #6$,@#366       ;DID TRAPS OCCUR IN CORRECT SEQ?
5408  021526  001401                          BEQ     TST45           ;;BRANCH IF YES
```

```
 5409  021530  104143                          ERROR   143             ;TRAPS DID NOT OCCUR IN SEQUENCE
 5410
 5411                                  ;;*****************************************************************
 5412                                  ;*TEST 45       MEMORY MANAGEMENT FUNCTIONS
 5413                                  ;*
 5414                                  ;*      THIS TEST ENSURES THAT THE SIGNAL SAPK I SPACEA (0) IS
 5415                                  ;*      ENABLED DURING THE TWO FLOATING POINT "BUST" CYCLES.
 5416                                  ;*
 5417                                  ;*      THIS IS DONE BY MAKING THE D SPACE PAR, THAT WOULD BE
 5418                                  ;*      REFERENCED BY THE FP INSTRUCTION, NON-RESIDENT.  SINCE
 5419                                  ;*      THE INSTRUCTION IS IMMEDIATE MODE, I SPACE SHOULD BE FORCED
 5420                                  ;*      AND THE TRAP SHOULD NOT OCCUR.
 5421                                  ;*
 5422                                  ;*      NOTE: THIS TEST CAN BE DISABLED BY SWITCH 12.
 5423                                  ;;*****************************************************************
 5424  021532  000004                  TST45:  SCOPE
 5425  021534  012737  000045  001240          MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
 5426  021542  012737  022046  001222          MOV     #TST46,$ESCAPE  ;;ESCAPE TO TEST 46 ON ERROR
 5427  021550  012737  021564  000004          MOV     #4$,ERRVEC      ;SET ERRVEC
 5428  021556  005737  177572                  TST     MMR0            ;IS MEMORY MGMT THERE?
 5429  021562  000403                          BR      5$              ;BRANCH IF YES
 5430  021564  012706  001100          4$:     MOV     #STACK,SP
 5431  021570  000526                          BR      TST46           ;;MGMT NOT AVAILABLE
 5432  021572  032777  010000  157336  5$:     BIT     #SW12,@SWR      ;INHIBIT THIS TEST?
 5433  021600  001122                          BNE     TST46           ;;BRANCH IF YES
 5434                                  ;SETUP MEMORY MANAGEMENT
 5435  021602  012700  172340                  MOV     #KIPAR0,R0
 5436  021606  005020                          CLR     (R0)+           ;SETUP THE PAR'S (KERNEL I)
 5437  021610  012720  000200                  MOV     #200,(R0)+      ;       *
 5438  021614  012720  000400                  MOV     #400,(R0)+      ;       *
 5439  021620  012720  000600                  MOV     #600,(R0)+      ;       *
 5440  021624  012720  001000                  MOV     #1000,(R0)+     ;       *
 5441  021630  012720  001200                  MOV     #1200,(R0)+     ;       *
 5442  021634  012720  001400                  MOV     #1400,(R0)+     ;       *
 5443  021640  012720  177600                  MOV     #177600,(R0)+   ;       *
 5444  021644  005020                          CLR     (R0)+           ;SETUP KERNEL D PAR'S
 5445  021646  012720  000200                  MOV     #200,(R0)+      ;       *
 5446  021652  012720  000400                  MOV     #400,(R0)+      ;       *
 5447  021656  012720  000600                  MOV     #600,(R0)+      ;       *
 5448  021662  012720  001000                  MOV     #1000,(R0)+     ;       *
 5449  021666  012720  001200                  MOV     #1200,(R0)+     ;       *
 5450  021672  012720  001400                  MOV     #1400,(R0)+     ;       *
 5451  021676  012720  177600                  MOV     #177600,(R0)+   ;       *
 5452  021702  012700  172300                  MOV     #KIPDR0,R0
 5453  021706  012701  077406                  MOV     #77406,R1
 5454  021712  012702  000010                  MOV     #10,R2
 5455  021716  010120          1$:     MOV     R1,(R0)+        ;SETUP KERNEL PDR'S (I SPACE)
 5456  021720  077202                          SOB     R2,1$
 5457  021722  012702  000010                  MOV     #10,R2
 5458  021726  012701  077400                  MOV     #77400,R1
 5459  021732  010120          2$:     MOV     R1,(R0)+        ;SETUP KERNEL D PDR'S (NON-RESIDENT)
 5460  021734  077202                          SOB     R2,2$
 5461  021736  012737  077406  172320          MOV     #77406,KDPDR0   ;MAKE TRAP VECTORS RESIDENT
 5462  021744  012737  077406  172336          MOV     #77406,KDPDR7   ;MAKE I/O PAGE RESIDENT
 5463  021752  012737  022032  000250          MOV     #3$,MMVEC       ;SETUP MEMORY MANAGEMENT VECTOR
 5464  021760  012737  021774  001106          MOV     #6$,$LPADR
```

```
                                                MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
                                        6$:     MOV     #BIT0,MMR0      ;TURN ON MEMORY MGMT
                                                MOV     #BIT2,MMR3      ;ENABLE KERNEL D SPACE
                                                NEGF    (PC)+           ;EXECUTE TEST INSTRUCTION
                                                .WORD   40000
                                                NOP
                                                NOP
                                                CLR     MMR3            ;TURN OFF D SPACE
                                                CLR     MMR0
                                                BR      TST46           ;;GO TO NEXT TEST
                                        3$:     CLR     MMR3
                                                CLR     MMR0
                                                CMP     (SP)+,(SP)+     ;RESTORE THE SP
                                                ERROR   133             ;EITHER SAPK I SPACEA (0) NOT GOING
                                                                        ;LOW OR MB138-YA
                                                                        ;NOT INSTALLED.

                                        ;;**************************************************************
                                        ;*TEST 46       FP AND MGMT TRAP ARBITRATION
                                        ;*
                                        ;*      THIS TEST ENSURES THAT TMCA HONOR SEGT CAUSES TMCA HONOR
                                        ;*      FP TRAP TO GO HIGH.
                                        ;*      THIS IS DONE BY SETTING THE FP TRAP WITH A MINUS ZERO TRAP AND
                                        ;*      THEN CAUSING A MEMORY MANAGEMENT TRAP.
                                        ;*
                                        ;*      NOTE: SWITCH 12 SKIPS THIS TEST
                                        ;;**************************************************************
022046  000004                        TST46:  SCOPE
022050  012737  000046  001240                MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
022056  012737  022252  001222                MOV     #TST47,$ESCAPE  ;;ESCAPE TO TEST 47 ON ERROR
022064  032777  010000  157044                BIT     #SW12,@SWR      ;DISABLE THIS TEST?
022072  001067                                BNE     TST47           ;;BRANCH IF YES
022074  012737  022110  000004                MOV     #3$,ERRVEC      ;SET ERRVEC
022102  005737  177572                        TST     MMR0            ;MEMORY MANAGEMENT AVAILBLE?
022106  000402                                BR      4$              ;BRANCH IF YES
022110  022626                        3$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
022112  000457                                BR      TST47           ;;GO TO NEXT TEST
022114  012737  022206  000250        4$:     MOV     #1$,MMVEC       ;SET UP
022122  012737  022216  000244                MOV     #2$,FPPVEC      ;VECTORS
022130  012706  001100                        MOV     #STACK,SP
022134  012737  100000  077776                MOV     #100000,@#77776 ;PU DATA IN TEST ADDRESS
022142  012737  077404  172326                MOV     #77404,KDPDR3   ;MAKE PAGE 3 TRAP
022150  012737  022156  001110                MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
022156  052737  000004  172516                BIS     #BIT2,MMR3      ;ENABLE D SPACE
022164  012737  001001  177572                MOV     #1001,MMR0      ;TURN ON MEMORY MGMT
022172  170127  004000                        LDFPS   #FIUV
022176  170537  077776                        TSTF    @#77776         ;EXECUTE FP INSTRUCTION TO CAUSE
                                                                        ;MGMT & FP TRAP.
022202  000240                        NOP
022204  000240                        6$:     NOP                     ;WAIT FOR TRAP
022206  005037  177572                1$:     CLR     MMR0
022212  022626                                CMP     (SP)+,(SP)+     ;RESTORE THE SP
022214  104137                                ERROR   137             ;TMCA HONOR FP TRAP NOT GOING HIGH
022216  005037  177572                2$:     CLR     MMR0
022222  005037  172516                        CLR     MMR3
022226  022716  022206                        CMP     #1$,(SP)        ;DID MGMT TRAP OCCUR FIRST?
```

```
5521  022232  001405                        BEQ     5$              ;BRANCH IF YES
5522  022234  012706  001100                MOV     #STACK,SP       ;RESTORE THE SP
5523  022240  005037  177572                CLR     MMR0
5524  022244  104144                        ERROR   144             ;OUT OF SEQUENCE
5525  022246  012706  001100        5$:     MOV     #STACK,SP       ;RESTORE THE SP
5526
5527                                ;;**************************************************************
5528                                ;*TEST 47       FPA AND FEA
5529                                ;*
5530                                ;*      THIS TEST EXECUTES TWO PIECES OF CODE IN DIFFERENT PARTS OF
5531                                ;*      THE PROGRAM TO TEST THE FPA AND FEA REGISTERS.  THIS TEST
5532                                ;*      SETS UP THE DATA AND FP VECTOR AND THEN JUMPS TO
5533                                ;*      THE APPROPRIATE LOCATIONS.  THESE TWO PARTICULAR LOCATIONS CONTAIN
5534                                ;*      CODE TO CAUSE A 25252 (125252 IF MORE THAN 16K OF MEMORY IS AVAILABLE)
5535                                ;*      AND 52524 PATTERN TO PLACED IN THE FPA AND THE INSTRUCTION
5536                                ;*      WILL CAUSE THIS PATTERN TO BE PUT IN THE FEA.
5537                                ;;**************************************************************
5538  022252  000004                TST47:  SCOPE
5539  022254  012737  000047  001240        MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
5540  022262  012737  022504  001222        MOV     #TST50,$ESCAPE  ;;ESCAPE TO TEST 50 ON ERROR
5541  022270  012737  022406  000244        MOV     #1$,FPPVEC      ;SETUP FP VECTOR
5542  022276  012737  022356  000004        MOV     #4$,ERRVEC      ;SETUP ERROR VEC
5543  022304  005737  125252                TST     @#125252        ;16K AVAILABLE?
5544  022310  012737  125252  022404        MOV     #125252,5$      ;YES
5545  022316  012737  125252  022420        MOV     #125252,6$
5546  022324  012737  125252  022426        MOV     #125252,7$
5547  022332  012737  170016  125252        MOV     #170016,@#125252;PUT ILLEGAL INSTRUCTION AT ADDRESS
5548  022340  012737  000240  125254        MOV     #240,@#125254
5549  022346  012737  000240  125256        MOV     #240,@#125256
5550  022354  000412                        BR      8$
5551  022356  022626                4$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
5552  022360  012737  025252  022404        MOV     #25252,5$
5553  022366  012737  025252  022420        MOV     #25252,6$
5554  022374  012737  025252  022426        MOV     #25252,7$
5555  022402  000137                8$:     JMP     @(PC)+          ;GO EXECUTE ILLEGAL INSTRUCTION
5556  022404  125252                5$:     .WORD   125252          ;MIGHT GET CHANGED
5557  022406  022626                1$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
5558  022410  170337  001204                STST    $TMP2           ;GET FEA
5559  022414  023727  001206                CMP     $TMP3,(PC)+     ;FEA OK?
5560  022420  125252                6$:     .WORD   125252
5561  022422  001405                        BEQ     2$              ;BRANCH IF YES
5562  022424  012700                        MOV     (PC)+,R0        ;
5563  022426  125252                7$:     .WORD   125252
5564  022430  010037  001200                MOV     R0,$TMP0
5565  022434  104145                        ERROR   145             ;BIT STUCK IN FPA OR FEA REGISTER
5566  022436  012737  022456  000244  2$:     MOV     #3$,FPPVEC
5567  022444  012737  022452  001110        MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
5568  022452  000137  052524                JMP     @#52524         ;GO EXECUTE INSTRUCTION TO LOAD FEA
5569  022456  022626                3$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
5570  022460  170337  001204                STST    $TMP2           ;GET THE FEA
5571  022464  022737  052524  001206        CMP     #52524,$TMP3    ;FEA OK?
5572  022472  001404                        BEQ     TST50           ;;BRANCH IF YES
5573  022474  012737  052524  001200        MOV     #52524,$TMP0    ;SAVE EXPECTED VALUE
5574  022502  104145                        ERROR   145             ;BIT STUCK IN FEA
5575
5576                                ;;**************************************************************
```

```
                                        ;*TEST 50       FXPB IMMEDIATE (MODE 1 & 4)
                                        ;*
                                        ;*      THIS TEST ENSURES THAT FXPB IMMEDIATE GOES HIGH FOR
                                        ;*      ADDRESS MODE 1 AND 4.
                                        ;;*****************************************************************
022504  000004                  TST50:  SCOPE
022506  012737  000200  001220          MOV     #200,$TIMES     ;;DO 200 ITERATIONS
022514  012737  000050  001240          MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
022522  012737  022660  001222          MOV     #TST51,$ESCAPE  ;;ESCAPE TO TEST 51 ON ERROR
022530  012737  174037  001200          MOV     #174037,$TMP0   ;SAVE EXPECTED VALUE
022536  005037  001202                  CLR     $TMP1           ;OF RESULT
022542  012737  022550  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
022550  170400                          CLRF    ACO             ;INITIALIZE ACO
022552  172417                  1$:     LDF     (PC),ACO        ;EXECUTE INSTRUCTION UNDER TEST
022554  174037  001204                  STF     ACO,$TMP2       ;GET DATA BACK
022560  173437  001200                  CMPF    $TMP0,ACO       ;RESULT OK?
022564  170000                          CFCC
022566  001401                          BEQ     2$              ;BRANCH IF YES
022570  104146                          ERROR   146             ;MODE 1 DID NOT CAUSE FXPB IMMEDIATE
                                                                ;TO GO TRUE
                                        ;MODE 1 OK-NOW CHECK MODE4
022572  012737  172447  001200  2$:     MOV     #172447,$TMP0   ;SAVE EXPECTED
022600  012737  022632  000100          MOV     #3$,LKVEC
022606  005037  001202                  CLR     $TMP1           ;VALUE OF DATA
022612  012737  022620  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
022620  170400                          CLRF    ACO             ;INITIALIZE ACO
022622  000230                          SPL     0               ;ENSURE PRIORITY AT 0
022624  004737  034324                  JSR     PC,GETTIK       ;START CLOCK
022630  172447                          LDF     -(PC),ACO       ;EXECUTE INSTRUCTION UNDER TEST
022632  022626                  3$:     CMP     (SP)+,(SP)+     ;RESTORE THE SP
022634  042737  000100  177546          BIC     #BIT6,LKSTAT    ;CLEAR I.E. BIT IN CLOCK STATUS
022642  174037  001204                  STD     ACO,$TMP2       ;GET DATA BACK
022646  173437  001200                  CMPF    $TMP0,ACO       ;RESULT OK?
022652  170000                          CFCC
022654  001401                          BEQ     TST51           ;;BRANCH IF YES
022656  104147                          ERROR   147             ;MODE 4 DID NOT CAUSE FXPB
                                                                ;IMMEDIATE TO GO TRUE.

                                        ;;*****************************************************************
                                        ;*TEST 51       FMM *USER MODE
                                        ;*
                                        ;*      THIS TEST ENSURES THAT FRLN FMM(1) DOES NOT GO HIGH IN
                                        ;*      SUPERVISOR OR USER MODE.
                                        ;*****************************************************************
022660  000004                  TST51:  SCOPE
022662  012737  000051  001240          MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
022670  012737  022740  001222          MOV     #TST52,$ESCAPE  ;;ESCAPE TO TEST 52 ON ERROR
022676  012737  147757  001200          MOV     #147757,$TMP0   ;SAVE EXPECTED VALUE
022704  012737  140000  177776          MOV     #140000,PSW     ;PUT PROCESSOR IN USER MODE
022712  170127  177777                  LDFPS   #-1             ;TRY AND LOAD FMM BIT
022716  170237  001202                  STFPS   $TMP1           ;GET FPS BACK
022722  005037  177776                  CLR     PSW             ;GO BACK TO KERNEL MODE
022726  022737  147757  001202          CMP     #147757,$TMP1   ;FPS LOAD OK?
022734  001401                          BEQ     TST52           ;;BRANCH IF YES
022736  104150                          ERROR   150             ;PDRD PS14 NOT DISABLING FXPN
                                                                ;FMM(1) H
```

```
                                ;;*****************************************************************
                                ;*TEST 52        ACCUMULATOR ADDRESSING
                                ;*
                                ;*      THIS TEST CHECKS THE LOGIC THAT DRIVES THE ACCUMULATOR ADDRESS
                                ;*      LINES ON FXPN THAT HAVE NOT ALREADY BEEN TESTED.
                                ;;*****************************************************************
022740  000004                  TST52:  SCOPE
022742  012737  000052  001240          MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
022750  012737  023034  001222          MOV     #TST53,$ESCAPE  ;;ESCAPE TO TEST 53 ON ERROR
                                ;SECTION 1
                                ;FXPN ACS1=ACF0(1)H *FIRB07(1)H
                                ;FXPN ACS0=ACF0(1)H *FIRB06(1)H
022756  172727  037600                  LDF     #^O037600,AC3   ;INITIALIZE AC3
022762  172627  040000                  LDF     #^O040000,AC2   ;INITIALIZE AC2
022766  172527  040200                  LDF     #^O040200,AC1   ;INITIALIZE AC1
022772  012737  023000  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
023000  005000                  1$:     CLR     R0              ;INIT R0
023002  175300                          STEXP   AC3,R0          ;EXECUTE INSTRUCTION UNDER TEST
023004  022700  177777                  CMP     #-1,R0          ;RESULT OK?
023010  001411                          BEQ     TST53           ;;BRANCH IF YES
023012  012737  177777  001200          MOV     #-1,$TMP0       ;SAVE EXPECTED VALUE
023020  010037  001202                  MOV     R0,$TMP1        ;SAVE RECEIVED VALUE
023024  005700                          TST     R0              ;DID R0 GET EXP OF AC2?
023026  001001                          BNE     2$              ;BRANCH IF NO
023030  104151                          ERROR   151             ;FXPN ACS0 NOT GOING HIGH
023032  104152                  2$:     ERROR   152             ;FXPN ACS1 NOT GOING HIGH

                                ;;*****************************************************************
                                ;*TEST 53        ACCUMULATOR VOLATILITY
                                ;*
                                ;*      THIS TEST EXECUTES THE WALKING ONE'S AND ZERO'S ALGORITHM
                                ;*      ON THE FLOATING POINT ACCUMULATORS.
                                ;;*****************************************************************
023034  000004                  TST53:  SCOPE
023036  012737  000053  001240          MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
023044  012737  023524  001222          MOV     #TST54,$ESCAPE  ;;ESCAPE TO TEST 54 ON ERROR
                                ;INITIALIZE THE BUFFER
023052  012700  002112                  MOV     #WALKBUF,R0     ;PUT ADDRESS OF BUFFER IN R0
023056  012701  000006                  MOV     #6,R1           ;SET LOOP COUNT
023062  170011                          SETD
023064  172427  040000                  LDD     #^O040000,AC0   ;PUT INITAIL DATA IN R0
023070  174020                  12$:    STD     AC0,(R0)+       ;INITAILIZE BUFFER
023072  077102                          SOB     R1,12$          ;CONTINUE
023074  012700  002112                  MOV     #WALKBUF,R0
023100  010002                          MOV     R0,R2           ;SAVE ADDRESS OF ONE'S
023102  012720  077777                  MOV     #77777,(R0)+    ;INITIALIZE
023106  012720  177777                  MOV     #-1,(R0)+       ;THE DATA
023112  012720  177777                  MOV     #-1,(R0)+       ;FOR AC5
023116  012720  177777                  MOV     #-1,(R0)+       ;   *
023122  012737  023130  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
                                ;LOAD THE ACCUMULATORS
023130  012700  002112          5$:     MOV     #WALKBUF,R0
023134  172420                          LDD     (R0)+,AC0
023136  174005                          STD     AC0,AC5
023140  172420                          LDD     (R0)+,AC0
```

```
5689  023142  174004                          STD     AC0,AC4
5690  023144  172720                          LDD     (R0)+,AC3
5691  023146  172620                          LDD     (R0)+,AC2
5692  023150  172520                          LDD     (R0)+,AC1
5693  023152  172420                          LDD     (R0)+,AC0
5694                                ;NOW CHECK THE DATA IN THE ACCUMULATORS
5695  023154  173440                          CMPD    -(R0),AC0
5696  023156  170000                          CFCC
5697  023160  001064                          BNE     6$
5698  023162  173540                          CMPD    -(R0),AC1
5699  023164  170000                          CFCC
5700  023166  001061                          BNE     6$
5701  023170  173640                          CMPD    -(R0),AC2
5702  023172  170000                          CFCC
5703  023174  001056                          BNE     6$
5704  023176  173740                          CMPD    -(R0),AC3
5705  023200  170000                          CFCC
5706  023202  001053                          BNE     6$
5707  023204  172404                          LDD     AC4,AC0
5708  023206  173440                          CMPD    -(R0),AC0
5709  023210  170000                          CFCC
5710  023212  001047                          BNE     6$
5711  023214  172405                          LDD     AC5,AC0
5712  023216  173440                          CMPD    -(R0),AC0
5713  023220  170000                          CFCC
5714  023222  001043                          BNE     6$
5715                                ;NOW CHANGE THE DATA PATTERN
5716  023224  032777  001000  155704          BIT     #BIT9,@SWR      ;LOOP ON ERROR?
5717  023232  001403                          BEQ     1$              ;BRANCH IF NO
5718  023234  105737  001103                  TSTB    $ERFLG          ;ANY ERRORS?
5719  023240  001333                          BNE     5$              ;BRANCH IF YES
5720  023242  012700  002112          1$:     MOV     #WALKBUF,R0
5721  023246  020002                  3$:     CMP     R0,R2           ;IS THIS WORD THE TEST PATTERN?
5722  023250  001403                          BEQ     2$              ;BRANCH IF YES
5723  023252  062700  000010                  ADD     #10,R0
5724  023256  000773                          BR      3$
5725  023260  022700  002162          2$:     CMP     #$AC0,R0        ;LAST ACCUMULATOR?
5726  023264  001465                          BEQ     4$              ;BRANCH IF YES
5727  023266  172422                          LDD     (R2)+,AC0       ;SAVE THIS PATTERN
5728  023270  174012                          STD     AC0,(R2)        ;MOVE TO NEXT WORD
5729  023272  005760  000002                  TST     2(R0)           ;IS BACKGROUND ONE'S OR ZERO'S?
5730  023276  001404                          BEQ     7$              ;BRANCH IF ONE'S(PATTERN IS ZERO)
5731  023300  170410                          CLRD    (R0)            ;BACKGROUND IS ZERO
5732  023302  052710  040000                  BIS     #BIT14,(R0)     ;MAKE EXPONENT NOW ZERO
5733  023306  000710                          BR      5$              ;CONTINUE
5734  023310  012720  077777          7$:     MOV     #77777,(R0)+    ;BACKGROUND IS
5735  023314  012720  177777                  MOV     #-1,(R0)+       ;ALL ONE'S
5736  023320  012720  177777                  MOV     #-1,(R0)+       ;    *
5737  023324  012720  177777                  MOV     #-1,(R0)+       ;    *
5738  023330  000677                          BR      5$              ;CONTINUE
5739                                ;;*****************************************************************
5740                                ;THIS ROUTINE DETERMINES WHICH ACCUMULATOR FAILED
5741                                ;AND SAVES THE CONTENTS OF THAT ACCUMULATOR IN $TMP4
5742                                ;AND SAVES THE EXPECTED VALUE IN $TMP0.
5743  023332  010003                  6$:     MOV     R0,R3           ;SAVE R0
5744  023334  042737  000300  023424          BIC     #300,8$         ;INSURE SRC FIELD OF STD INSTRUCTION = AC0
```

```
5745  023342  042737  000007  023422          BIC     #7,9$           ;INSURE SRC FIELD OF LDD INSTR = ACO
5746  023350  162700  002172                  SUB     #WALKBUF+60,R0  ;CALCULATE THE
5747  023354  005400                          NEG     R0              ;ACC NUMBER THAT
5748  023356  072027  177775                  ASH     #-3,R0          ;FAILED
5749  023362  010037  001172                  MOV     R0,$REG5        ;SAVE FOR TYPEOUT
5750  023366  000337  001172                  SWAB    $REG5           ;PUT IN HIGH
5751  023372  006337  001172                  ASL     $REG5           ;BYTE OF REG5
5752  023376  022700  000003                  CMP     #3,R0           ;IS IT GREATER THAN 3?
5753  023402  002405                          BLT     10$             ;BRANCH IF YES
5754  023404  072027  000006                  ASH     #6,R0           ;SHIFT TO SRC FIELD OF STD INSTR.
5755  023410  050037  023424                  BIS     R0,8$           ;INSERT INTO SRC FIELD
5756  023414  000403                          BR      8$              ;GO SAVE IT
5757  023416  050037  023422          10$:    BIS     R0,9$           ;INSERT INTO SRC FIELD OF LDD INSTR
5758  023422  172400          9$:     LDD     AC0,AC0         ;SRC GETS CHANGED
5759  023424  174037  001210          8$:     STD     AC0,$TMP4       ;SRC GETS CHANGED
5760  023430  172440                  LDD     -(R0),AC0       ;GET EXPECTED DATA
5761  023432  174037  001200                  STD     AC0,$TMP0       ;SAVE FOR TYPEOUT
5762  023436  104153                          ERROR   153             ;ACCUMULATOR CHIP FAILED
5763                                  ;;**************************************************************
5764  023440  012700  002112          4$:     MOV     #WALKBU,R0
5765  023444  022710  077777                  CMP     #77777,(R0)     ;DONE WITH TEST?
5766  023450  001425                          BEQ     TST54           ;;BRANCH IF YES
5767                                  ;CHANGE TO A BACKGROUND OF ALL ONE'S
5768  023452  012701  000006                  MOV     #6,R1           ;SET LOOP COUNT
5769  023456  012720  077777                  MOV     #77777,(R0)+
5770  023462  012720  177777                  MOV     #-1,(R0)+
5771  023466  012720  177777                  MOV     #-1,(R0)+
5772  023472  012720  177777                  MOV     #-1,(R0)+
5773  023476  172440                          LDD     -(R0),AC0
5774  023500  174020                  11$:    STD     AC0,(R0)+       ;INITIALIZE BUFFER
5775  023502  077102                          SOB     R1,11$          ;CONTINUE
5776  023504  012700  002112                  MOV     #WALKBU,R0      ;
5777  023510  010002                          MOV     R0,R2           ;
5778  023512  170410                          CLRD    (R0)            ;PUT ZERO PATTERN IN FIRST WORD
5779  023514  052710  040000                  BIS     #BIT14,(R0)     ;INSURE EXPONENT NON-ZERO
5780  023520  000137  023130                  JMP     5$              ;GO EXECUTE TEST
5781                                  .SBTTL
5782                                  ;;**************************************************************
5783                                  .SBTTL  A-BRANCH AND ADX ROM EXERCISER
5784                                  ;*      THE REMAINING SUBTESTS IN THIS PROGRAM TEST ALL THE
5785                                  ;*      LOCATIONS IN THE A-BRANCH AND ADX ROMS THAT HAVE NOT
5786                                  ;*      ALREADY BEEN TESTED.
5787                                  ;*      THE ENTIRE SEQUENCE IS FIRST EXECUTED ONCE. THEN, THE
5788                                  ;*      LINE (OR PROGRAMMABLE) CLOCK IS TURNED ON AND THIS SEQUENCE
5789                                  ;*      IS LOOPED ON FOR APPROXIMATELY 30 SECONDS. THE PURPOSE
5790                                  ;*      OF THIS IS TRY AND ENSURE THAT THE DSI BIT IN THE CONTROL
5791                                  ;*      STORE IS FUNCTIONAL IN ALL THE APPROPRIATE ROM STATES.
5792                                  ;;**************************************************************
5793                                  .SBTTL
5794
5795                                  ;;**************************************************************
5796                                  ;*TEST 54       A-BRANCH*ADX ROM EXERCISER
5797                                  ;;**************************************************************
5798  023524  000004          TST54:  SCOPE
5799  023526  012737  000001  001220          MOV     #1,$TIMES       ;;DO 1 ITERATION
5800  023534  012737  000054  001240          MOV     #$TN-1,$TESTN   ;;SET TEST NUMBER IN  MAIL BOX
```

```
5801                                            ;FLOATING*INTEGER*MODE 0 INSTRUCTIONS
5802  023542  012737  032712  001222            MOV     #$EOP,$ESCAPE   ;;ESCAPE TO $EOP ON ERROR
5803  023550  170011                            SETD
5804  023552  170400                            CLRD    AC0
5805  023554  170401                            CLRD    AC1
5806  023556  170402                            CLRD    AC2
5807  023560  170403                            CLRD    AC3
5808  023562  170404                            CLRD    AC4
5809  023564  170405                            CLRD    AC5
5810  023566  170001                            SETF
5811  023570  005037  001366                    CLR     TICKS
5812  023574  005037  032606                    CLR     OUT
5813  023600  012737  023624  000244            MOV     #1$,FPPVEC      ;SET FPPVEC TO INSURE
5814  023606  012703  000042                    MOV     #42,R3          ;FEC CONTAINS A 16
5815  023612  170003                            LDUB
5816  023614  170127  000020                    LDFPS   #FMM
5817  023620  172500                            LDF     AC0,AC1
5818  023622  170000                            CFCC
5819  023624  012706  001100            1$:     MOV     #STACK,SP       ;RESTORE THE SP
5820  023630  012737  040200  001200  LOOP0:   MOV     #40200,$TMP0    ;INITIALIZE
5821  023636  005037  001202                    CLR     $TMP1           ;DATA IN MEMORY
5822  023642  170127  000000                    LDFPS   #0
5823  023646  172627  040000                    LDF     #^040000,AC2    ;LOAD THE SRC
5824  023652  172702                            LDF     AC2,AC3         ;LOAD THE DST
5825  023654  174702                    1$:     DIVF    AC2,AC3         ;EXECUTE TEST
5826  023656  173737  001200                    CMPF    $TMP0,AC3       ;RESULT OK?
5827  023662  170000                            CFCC
5828  023664  001403                            BEQ     100$            ;BRANCH IF YES
5829  023666  174337  001204                    STF     AC3,$TMP2       ;SAVE RECEIVED DATA
5830  023672  104024                            ERROR   24              ;RESULT WRONG
5831
5832                                            ;;*****************************************************************
5833                                            ;FLOATING*INTEGER*NOT MODE 0*NOT IMMEDIATE INSTRUCTIONS
5834  023674  012700  001200            100$:   MOV     #$TMP0,R0       ;PUT ADDRESS OF DATA IN R0
5835  023700                            LABLE1:
5836  023700  170127  000017                    LDFPS   #17             ;LOAD COMPLIMENT CC'S
5837  023704  170520                    1$:     TSTF    (R0)+           ;EXECUTE TEST
5838  023706  170237  001202                    STFPS   $TMP1           ;SAVE CC'S
5839  023712  022700  001204                    CMP     #$TMP2,R0       ;ADX ROM OK?
5840  023716  001401                            BEQ     2$              ;BRANCH IF YES
5841  023720  104004                            ERROR   4               ;ADX ROM FAILED
5842  023722  005737  001202            2$:     TST     $TMP1           ;CC'S OK?
5843  023726  001403                            BEQ     99$             ;BRANCH IF YES
5844  023730  005037  001200                    CLR     $TMP0           ;SAVE EXPECTED VALUE
5845  023734  104003                            ERROR   3               ;CC'S BAD
5846
5847                                            ;;*****************************************************************
5848                                            ;FLOATING*INTEGER*NOT MODE ZERO*IMMEDIATE
5849  023736  012737  023722  000010    99$:    MOV     #2$,RESVEC      ;SET RESVEC
5850  023744                            LABLE2:
5851  023744  170127  000017                    LDFPS   #17             ;LOAD COMPLIMENT CC'S
5852  023750  170527                    1$:     TSTF    (PC)+           ;EXECUTE TEST
5853  023752  076777                            .WORD   76777
5854  023754  000405                            BR      3$
5855  023756  104004                            ERROR   4               ;ADX ROM FAILED
5856  023760  104004                            ERROR   4               ;       *
```

```
5857  023762  104004                        ERROR   4               ;       *
5858  023764  022626                  2$:   CMP     (SP)+,(SP)+     ;RESTORE THE SP
5859  023766  104004                        ERROR   4               ;ADX ROM FAILED
5860  023770  012737  024016  000010  3$:   MOV     #4$,RESVEC      ;SET RESVEC
5861  023776  172627  040000                LDF     #^O040000,AC2   ;LOAD THE DST
5862  024002  171627                        MODF    (PC)+,AC2       ;EXECUTE THE TEST
5863  024004  076777                        .WORD   76777
5864  024006  000405                        BR      5$
5865  024010  104004                        ERROR   4               ;ADX ROM FAILED
5866  024012  104004                        ERROR   4               ;       *
5867  024014  104004                        ERROR   4               ;       *
5868  024016  022626                  4$:   CMP     (SP)+,(SP)+     ;RESTORE THE SP
5869  024020  104004                        ERROR   4               ;ADX ROM FAILED
5870  024022  012737  024054  000010  5$:   MOV     #6$,RESVEC
5871  024030  005037  024042                CLR     8$
5872  024034  172727  041600                LDF     #^O041600,AC3   ;INIT THE SRC
5873  024040  175327                        STEXP   AC3,(PC)+       ;EXECUTE THE TEST
5874  024042  000000                  8$:   .WORD   0
5875  024044  000405                        BR      7$
5876  024046  104004                        ERROR   4               ;ADX ROM FAILED
5877  024050  104004                        ERROR   4               ;       *
5878  024052  104004                        ERROR   4               ;       *
5879  024054  022626                  6$:   CMP     (SP)+,(SP)+     ;RESTORE THE SP
5880  024056  104004                        ERROR   4               ;ADX ROM FAILED
5881  024060  022737  000007  024042  7$:   CMP     #7,8$           ;DATA OK?
5882  024066  001407                        BEQ     9$              ;BRANCH IF YES
5883  024070  012737  000007  001200        MOV     #7,$TMP0
5884  024076  013737  024042  001202        MOV     8$,$TMP1
5885  024104  104162                        ERROR   162             ;DATA BAD
5886  024106  012737  024140  000010  9$:   MOV     #10$,RESVEC
5887  024114  005037  024126                CLR     11$
5888  024120  177227  000007                LDCIF   #7,AC2          ;INIT AC2
5889  024124  175627                        STCFI   AC2,(PC)+       ;EXECUTE TEST
5890  024126  000000                 11$:   .WORD   0
5891  024130  000405                        BR      12$
5892  024132  104004                        ERROR   4               ;ADX ROM FAILED
5893  024134  104004                        ERROR   4               ;       *
5894  024136  104004                        ERROR   4               ;       *
5895  024140  022626                 10$:   CMP     (SP)+,(SP)+     ;RESTORE THE SP
5896  024142  104004                        ERROR   4               ;ADX ROM FAILED
5897  024144  022737  000007  024126  12$:  CMP     #7,11$          ;RESULT OK?
5898  024152  001407                        BEQ     13$             ;BRANCH IF YES
5899  024154  012737  000007  001200        MOV     #7,$TMP0
5900  024162  013737  024126  001202        MOV     11$,$TMP1
5901  024170  104162                        ERROR   162             ;RESULT WRONG
5902  024172  012737  024216  000010  13$:  MOV     #14$,RESVEC
5903  024200  170401                        CLRF    AC1             ;INIT AC1
5904  024202  177527                        LDCDF   (PC)+,AC1       ;EXECUTE TEST
5905  024204  076777                        .WORD   76777
5906  024206  000405                        BR      15$
5907  024210  104004                        ERROR   4               ;ADX ROM FAILED
5908  024212  104004                        ERROR   4               ;       *
5909  024214  104004                        ERROR   4               ;       *
5910  024216  022626                 14$:   CMP     (SP)+,(SP)+     ;RESTORE THE SP
5911  024220  104004                        ERROR   4               ;ADX ROM FAILED
5912  024222  173527  076777         15$:   CMPF    #^O076777,AC1   ;RESULT OK?
```

```
5913  024226  170000                          CFCC
5914  024230  001406                          BEQ     98$                     ;BRANCH IF YES
5915  024232  012737  076777  001200          MOV     #76777,$TMP0
5916  024240  174137  001204                  STF     AC1,$TMP2
5917  024244  104024                          ERROR   24                      ;RESULT WRONG
5918
5919                                  ;;**************************************************************
5920                                  ;MODE 0 INSTRUCTIONS
5921  024246  012737  000140  001364 98$:    MOV     #140,$FPS               ;INIT TEST FPS
5922  024254                          LABLE3:
5923  024254  005004                  LOOP1:  CLR     R4                      ;INIT R4
5924  024256  052737  000017  001364          BIS     #17,$FPS                ;SET CC'S IN FPS
5925  024264  013737  001364  001200          MOV     $FPS,$TMP0
5926  024272  042737  000300  001200          BIC     #FL+FD,$TMP0
5927  024300  012737  024306  001110          MOV     #.+6,@#$LPERR           ;SET ERROR LOOP
5928  024306  170137  001364                  LDFPS   $FPS                    ;SET FPS
5929  024312  170001                          SETF
5930  024314  170002                          SETI                            ;EXECUTE TEST
5931  024316  170237  001202                  STFPS   $TMP1                   ;GET FPS BACK
5932  024322  023737  001200  001202          CMP     $TMP0,$TMP1             ;FPS OK?
5933  024330  001401                          BEQ     1$                      ;BRANCH IF YES
5934  024332  104003                          ERROR   3                       ;DATA WRONG
5935  024334  013700  001364          1$:     MOV     $FPS,R0
5936  024340  170100                          LDFPS   R0
5937  024342  170201                          STFPS   R1                      ;EXECUTE TEST
5938  024344  023701  001364                  CMP     $FPS,R1                 ;RESULT OK?
5939  024350  001406                          BEQ     2$                      ;BRANCH IF YES
5940  024352  010137  001202                  MOV     R1,$TMP1                ;SAVE RECEIVED DATA
5941  024356  013737  001364  001200          MOV     $FPS,$TMP0              ;SAVE EXPECTED DATA
5942  024364  104162                          ERROR   162                     ;DATA WRONG
5943  024366                          2$:
5944  024366  012737  024374  001110          MOV     #.+6,@#$LPERR           ;SET ERROR LOOP
5945  024374  005000                          CLR     R0                      ;INIT R0
5946  024376  170104                          LDFPS   R4                      ;CLEAR FPS
5947  024400  170137  001364                  LDFPS   $FPS
5948  024404  170300                          STST    R0                      ;EXECUTE TEST
5949  024406  022700  000016                  CMP     #16,R0                  ;FEC OK?
5950  024412  001406                          BEQ     3$                      ;BRANCH IF YES
5951  024414  010037  001204                  MOV     R0,$TMP2                ;SAVE RECEIVED VALUE
5952  024420  012737  000016  001200          MOV     #16,$TMP0               ;SAVE EXPECTED VALUE
5953  024426  104101                          ERROR   101                     ;FEC WRONG
5954  024430                          3$:
5955  024430  012737  024436  001110          MOV     #.+6,@#$LPERR           ;SET ERROR LOOP
5956  024436  170104                          LDFPS   R4                      ;CLEAR FPS
5957  024440  005037  001200                  CLR     $TMP0                   ;INIT $TEMP0
5958  024444  172727  040000                  LDF     #^040000,AC3            ;INIT AC3
5959  024450  170137  001364                  LDFPS   $FPS
5960  024454  170403                          CLRF    AC3                     ;EXECUTE TEST
5961  024456  170104                          LDFPS   R4                      ;CLEAR FPS
5962  024460  170503                          TSTF    AC3                     ;RESULT OK?
5963  024462  170000                          CFCC
5964  024464  001405                          BEQ     4$                      ;BRANCH IF YES
5965  024466  170437  001200                  CLRF    $TMP0                   ;SAVE EXPECTED VALUE
5966  024472  174337  001204                  STF     AC3,$TMP2               ;SAVE RECEIVED VALUE
5967  024476  104024                          ERROR   24                      ;RESULT WRONG
5968  024500                          4$:
```

```
 5969  024500  012737  024506  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
 5970  024506  172627  040000                  LDF     #^O040000,AC2   ;INIT AC2
 5971  024512  170137  001364                  LDFPS   $FPS
 5972  024516  170502                          TSTF    AC2             ;EXECUTE TEST
 5973  024520  170200                          STFPS   R0              ;GET CC'S BACK
 5974  024522  042700  177760                  BIC     #177760,R0
 5975  024526  170104                          LDFPS   R4
 5976  024530  005700                          TST     R0              ;CC'S OK?
 5977  024532  001405                          BEQ     5$              ;BRANCH IF YES
 5978  024534  005037  001200                  CLR     $TMP0           ;SAVE EXPECTED VALUE
 5979  024540  010037  001202                  MOV     R0,$TMP1        ;SAVE RECEIVED VALUE
 5980  024544  104003                          ERROR   3               ;CC'S BAD
 5981  024546                          5$:
 5982  024546  012737  024554  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
 5983  024554  172427  140000                  LDF     #^O0140000,AC0  ;INIT AC0
 5984  024560  170137  001364                  LDFPS   $FPS            ;
 5985  024564  170600                          ABSF    AC0             ;EXECUTE TEST
 5986  024566  170104                          LDFPS   R4
 5987  024570  170500                          TSTF    AC0             ;RESULT OK?
 5988  024572  170000                          CFCC
 5989  024574  100006                          BPL     6$              ;BRANCH IF YES
 5990  024576  174037  001204                  STF     AC0,$TMP2       ;SAVE RECEIVED VALUE
 5991  024602  012737  040000  001200          MOV     #40000,$TMP0
 5992  024610  104024                          ERROR   24              ;DATA WRONG
 5993  024612                          6$:
 5994  024612  012737  024620  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
 5995  024620  172527  140000                  LDF     #^O0140000,AC1  ;INIT AC1
 5996  024624  170137  001364                  LDFPS   $FPS
 5997  024630  170701                          NEGF    AC1             ;EXECUTE TEST
 5998  024632  170104                          LDFPS   R4
 5999  024634  170501                          TSTF    AC1             ;RESULT OK?
 6000  024636  170000                          CFCC
 6001  024640  100006                          BPL     7$              ;BRANCH IF YES
 6002  024642  174137  001204                  STF     AC1,$TMP2       ;SAVE RECEIVED VALUE
 6003  024646  012737  040000  001200          MOV     #40000,$TMP0
 6004  024654  104024                          ERROR   24
 6005  024656                          7$:
 6006  024656  012737  024664  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
 6007  024664  172727  140177                  LDF     #^O0140177,AC3  ;INIT AC3
 6008  024670  012737  040200  001200          MOV     #40200,$TMP0
 6009  024676  005037  001202                  CLR     $TMP1
 6010  024702  172637  001200                  LDF     $TMP0,AC2       ;INIT AC2
 6011  024706  170137  001364                  LDFPS   $FPS
 6012  024712  171203                          MULF    AC3,AC2         ;EXECUTE TEST
 6013  024714  170104                          LDFPS   R4
 6014  024716  173627  140177                  CMPF    #^O0140177,AC2  ;RESULT OK?
 6015  024722  170000                          CFCC
 6016  024724  001406                          BEQ     8$              ;BRANCH IF YES
 6017  024726  174237  001204                  STF     AC2,$TMP2       ;SAVE RESULT
 6018  024732  012737  140177  001200          MOV     #140177,$TMP0
 6019  024740  104024                          ERROR   24              ;RESULT WRONG
 6020  024742                          8$:
 6021  024742  012737  024750  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
 6022  024750  172627  142177                  LDF     #^O0142177,AC2  ;INIT AC2
 6023  024754  172537  001200                  LDF     $TMP0,AC1       ;INIT AC1
 6024  024760  170137  001364                  LDFPS   $FPS
```

```
6025  024764  171601                          MODF    AC1,AC2         ;EXECUTE TEST
6026  024766  170104                          LDFPS   R4
6027  024770  170502                          TSTF    AC2             ;FRACTION ZERO?
6028  024772  170000                          CFCC
6029  024774  001531                          BEQ     9$              ;BRANCH IF YES
6030  024776  174237  001204                  STF     AC2,$TMP2       ;SAVE RESULT
6031  025002  005037  001200                  CLR     $TMP0
6032  025006  104024                          ERROR   24              ;RESULT WRONG
6033          025010                  X14=.
6034                                  ;;**********************************************************************
6035                                  ;;**********************************************************************
6036                                  ;;**********************************************************************
6037                                  ;;**********************************************************************
6038                                  ;THIS WORD IS USED TO TEST THE FPC AND FEA REGISTERS.
6039                                  ;IT MUST BE ASSEMBLED PRIOR TO THE PREVIOUS CODE REACHING
6040                                  ;ADDRESS 25252. IF NOT, AN ERROR MESSAGE IS TYPED DURING
6041                                  ;THE ASSEMBLY.
6042
6043
6044
6045
6046          025252                  .=25252
6047  025252  170016                          .WORD   170016
6048  025254  000240                          NOP
6049  025256  000240                          NOP
6050
6051
6052
6053                                  ;;**********************************************************************
6054                                  ;;**********************************************************************
6055  025260                          9$:
6056  025260  012737  025266  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
6057  025266  170011                          SETD
6058  025270  170403                          CLRD    AC3
6059  025272  170001                          SETF
6060  025274  170401                          CLRF    AC1             ;INIT AC1
6061  025276  172727  040200                  LDF     #^O040200,AC3   ;INIT AC3
6062  025302  170137  001364                  LDFPS   $FPS
6063  025306  172503                          LDF     AC3,AC1         ;EXECUTE TEST
6064  025310  170104                          LDFPS   R4
6065  025312  173527  040200                  CMPF    #^O040200,AC1   ;RESULT OK?
6066  025316  170000                          CFCC
6067  025320  001403                          BEQ     10$             ;BRANCH IF YES
6068  025322  174137  001204                  STF     AC1,$TMP2       ;SAVE RESULT
6069  025326  104024                          ERROR   24              ;RESULT WRONG
6070  025330                          10$:
6071  025330  012737  025336  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
6072  025336  172627  040400                  LDF     #^O040400,AC2   ;INIT AC2
6073  025342  172727  040200                  LDF     #^O040200,AC3   ;INIT AC3
6074  025346  170137  001364                  LDFPS   $FPS
6075  025352  173203                          SUBF    AC3,AC2         ;EXECUTE TEST
6076  025354  170104                          LDFPS   R4
6077  025356  173627  040200                  CMPF    #^O040200,AC2   ;RESULT OK?
6078  025362  170000                          CFCC
6079  025364  001403                          BEQ     11$             ;BRANCH IF YES
6080  025366  174237  001204                  STF     AC2,$TMP2       ;SAVE RESULT
```

```
                                                ERROR   24              ;RESULT WRONG
                                        11$:
                                                MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
                                                LDF     #^O40200,AC3    ;INIT AC3
                                                LDF     AC3,AC0         ;INIT AC0
                                                BIC     #BIT2,$FPS
                                                LDFPS   $FPS
                                                CMPF    AC3,AC0         ;EXECUTE TEST
                                                CFCC
                                                BEQ     12$             ;BRANCH IF RESULT OK
                                                STFPS   R0
                                                BIC     #177760,R0
                                                MOV     R0,$TMP1
                                                MOV     #F2,$TMP0
                                                ERROR   3               ;CC'S BAD
                                        12$:
                                                MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
                                                LDFPS   R4
                                                CLRF    AC1             ;INIT AC1
                                                LDF     #^O40200,AC2    ;INIT AC2
                                                LDFPS   $FPS
                                                STF     AC2,AC1         ;EXECUTE TEST
                                                CMPF    AC2,AC1         ;RESULT OK?
                                                CFCC
                                                BEQ     13$             ;BRANCH IF YES
                                                LDFPS   R4
                                                STF     AC1,$TMP2       ;SAVE RESULT
                                                ERROR   24              ;RESULT WRONG
                                        13$:
                                                MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
                                                LDFPS   R4
                                                LDF     #^O40400,AC2    ;INIT AC2
                                                LDF     AC2,AC3         ;INITAC3
                                                LDFPS   $FPS
                                                DIVF    AC3,AC2         ;EXECUTE TEST
                                                LDFPS   R4
                                                CMPF    $TMP0,AC2       ;RESULT OK?
                                                CFCC
                                                BEQ     14$             ;BRANCH IF YES
                                                STF     AC2,$TMP2       ;SAVE RESULT
                                                ERROR   24              ;RESULT WRONG
                                        14$:
                                                MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
                                                CLR     R0              ;INIT R0
                                                LDF     #^O77777,AC3    ;INIT AC3
                                                LDFPS   $FPS
                                                STEXP   AC3,R0          ;EXECUTE TEST
                                                CMP     #177,R0         ;RESULT OK?
                                                BEQ     15$             ;BRANCH IF YES
                                                MOV     #177,$TMP0
                                                MOV     R0,$TMP1
                                                ERROR   162             ;RESULT WRONG
                                        ;;*****************************************************************
                                        15$:
                                                MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
                                                LDFPS   R4
```

```
6137  025634  172627  040200                    LDF     #↑040200,AC2    ;INIT AC2
6138  025640  172427  077600                    LDF     #↑077600,AC0    ;INIT AC0
6139  025644  170137  001364                    LDFPS   $FPS
6140  025650  176200                           STCFD   AC2,AC0         ;EXECUTE TEST
6141  025652  173402                           CMPF    AC2,AC0         ;RESULT OK?
6142  025654  170000                           CFCC
6143  025656  001404                           BEQ     16$             ;BRANCH IF YES
6144  025660  170104                           LDFPS   R4
6145  025662  174037  001204                  STF     AC0,$TMP2       ;SAVE RESULT
6146  025666  104024                           ERROR   24              ;RESULT WRONG
6147  025670                          16$:
6148  025670  012737  025676  001110           MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
6149  025676  170104                           LDFPS   R4
6150  025700  172727  077600                  LDF     #↑077600,AC3    ;INIT AC3
6151  025704  012704  000001                  MOV     #1,R4           ;INIT R4
6152  025710  170137  001364                  LDFPS   $FPS
6153  025714  176704                           LDEXP   R4,AC3          ;EXECUTE TEST
6154  025716  173702                           CMPF    AC2,AC3         ;RESULT OK?
6155  025720  170000                           CFCC
6156  025722  001406                           BEQ     17$             ;BRANCH IF YES
6157  025724  170104                           LDFPS   R4
6158  025726  174337  001204                  STF     AC3,$TMP2       ;SAVE RESULT
6159  025732  174237  001200                  STF     AC2,$TMP0
6160  025736  104024                           ERROR   24              ;RESULT WRONG
6161  025740                          17$:
6162  025740  012737  025746  001110           MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
6163  025746  170104                           LDFPS   R4
6164  025750  032737  000100  001364           BIT     #FL,$FPS        ;FL BIT ON?
6165  025756  001403                           BEQ     127$            ;BRANCH IF NO
6166  025760  172627  044200                  LDF     #↑044200,AC2
6167  025764  000402                           BR      126$
6168  025766  172627  040200          127$:    LDF     #↑040200,AC2
6169  025772  170403                  126$:    CLRF    AC3             ;INIT AC3
6170  025774  012703  000001                  MOV     #1,R3           ;INIT R3
6171  026000  170137  001364                  LDFPS   $FPS
6172  026004  177303                           LDCLF   R3,AC3          ;EXECUTE TEST
6173  026006  173702                           CMPF    AC2,AC3         ;RESULT OK?
6174  026010  170000                           CFCC
6175  026012  001406                           BEQ     20$             ;BRANCH IF YES
6176  026014  170104                           LDFPS   R4
6177  026016  174337  001204                  STF     AC3,$TMP2       ;SAVE RESULT
6178  026022  174237  001200                  STF     AC2,$TMP0
6179  026026  104024                           ERROR   24              ;RESULT WRONG
6180  026030                          20$:
6181  026030  012737  026036  001110           MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
6182  026036  170104                           LDFPS   R4
6183  026040  172527  140200                  LDF     #↑0140200,AC1   ;INIT AC1
6184  026044  005002                           CLR     R2              ;INIT R2
6185  026046  170137  001364                  LDFPS   $FPS
6186  026052  175502                           STCFI   AC1,R2          ;EXECUTE TEST
6187  026054  170104                           LDFPS   R4
6188  026056  022702  177777                  CMP     #-1,R2          ;RESULT OK?
6189  026062  001406                           BEQ     21$             ;BRANCH IF YES
6190  026064  012737  177777  001200           MOV     #-1,$TMP0
6191  026072  010237  001202                  MOV     R2,$TMP1
6192  026076  104162                           ERROR   162             ;RESULT WRONG
```

```
                                        21$:
026100                                          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
026100  012737  026106  001110                  CLRF    AC0             ;INIT AC0
026106  170400                                  LDFPS   $FPS
026110  170137  001364                          LDCFD   AC1,AC0         ;EXECUTE TEST
026114  177401                                  CMPF    AC1,AC0         ;RESULT OK?
026116  173401                                  CFCC
026120  170000                                  BEQ     22$             ;BRANCH IF YES
026122  001406                                  LDFPS   R4
026124  170104                                  STF     AC1,$TMP0
026126  174137  001200                          STF     AC0,$TMP2
026132  174037  001204                          ERROR   24              ;RESULT WRONG
026136  104024                          022737  CMP     #353,$FPS       ;DONE?
026140  022737  000353  001364  22$:           BEQ     97$             ;BRANCH IF YES
026146  001405                                  ADD     #BIT6,$FPS
026150  062737  000100  001364                  JMP     LOOP1           ;CONTINUE
026156  000137  024254
                                        ;;*****************************************************************
                                        ;NOT MODE 0 AND IMMEDIATE INSTRUCTIONS
026162  012737  000100  001364  97$:     MOV     #100,$FPS       ;INIT $FPS
026170                                  LABLE4:
026170  005004                                  CLR     R4
026172  170104                                  LDFPS   R4
026174  012737  030360  000010  LOOP2:         MOV     #63$,RESVEC     ;SETUP RESVEC
026202  012737  026210  001110                  MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
026210  170137  001364                          LDFPS   $FPS
026214  170127                                  LDFPS   (PC)+           ;EXECUTE TEST
026216  000017                                  .WORD   17
026220  000403                                  BR      1$
026222  104004                                  ERROR   4               ;ADX ROM FAILED
026224  104004                                  ERROR   4               ;       *
026226  104004                                  ERROR   4               ;       *
026230  170200                          1$:     STFPS   R0
026232  022700  000017                          CMP     #17,R0          ;DATA OK?
026236  001407                                  BEQ     2$              ;BRANCH IF YES
026240  170104                                  LDFPS   R4
026242  010037  001202                          MOV     R0,$TMP1
026246  012737  000017  001200                  MOV     #17,$TMP0
026254  104003                                  ERROR   3
026256  052737  007000  001364  2$:            BIS     #7000,$FPS      ;MAKE $FPS LOOK LIKE ILLEGAL INSTR
026264  012737  026272  001110                  MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
026272  005037  026304                          CLR     3$
026276  170137  001364                          LDFPS   $FPS
026302  170227                                  STFPS   (PC)+           ;EXECUTE TEST
026304  000000                          3$:     .WORD   0
026306  000403                                  BR      4$
026310  104004                                  ERROR   4               ;ADX ROM FAILED
026312  104004                                  ERROR   4               ;       *
026314  104004                                  ERROR   4               ;       *
026316  170104                          4$:     LDFPS   R4
026320  023737  001364  026304                  CMP     $FPS,3$         ;DATA OK?
026326  001407                                  BEQ     5$              ;BRANCH IF YES
026330  013737  026304  001202                  MOV     3$,$TMP1
026336  013737  001364  001200                  MOV     $FPS,$TMP0
026344  104003                                  ERROR   3               ;FPS BAD
026346  042737  007000  001364  5$:            BIC     #7000,$FPS
```

```
6249  026354  012737  026362  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
6250  026362  005037  026374                  CLR     6$
6251  026366  170137  001364                  LDFPS   $FPS
6252  026372  170327                          STST    (PC)+           ;EXECUTE TEST
6253  026374  000000                  6$:     .WORD   0
6254  026376  000403                          BR      7$
6255  026400  104004                          ERROR   4               ;ADX ROM FAILED
6256  026402  104004                          ERROR   4               ;       *
6257  026404  104004                          ERROR   4               ;       *
6258  026406  170104                  7$:     LDFPS   R4
6259  026410  022737  000016  026374          CMP     #16,6$          ;FEC OK?
6260  026416  001407                          BEQ     8$              ;BRANCH IF YES
6261  026420  012737  000016  001200          MOV     #16,$TMP0
6262  026426  013737  026374  001204          MOV     6$,$TMP2
6263  026434  104101                          ERROR   101             ;FEC WRONG
6264  026436                          8$:
6265  026436  012737  026444  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
6266  026444  012737  007000  026460          MOV     #7000,9$
6267  026452  170137  001364                  LDFPS   $FPS
6268  026456  170427                          CLRF    (PC)+           ;EXECUTE TEST
6269  026460  007000                  9$:     .WORD   7000
6270  026462  000403                          BR      10$
6271  026464  104004                          ERROR   4               ;ADX ROM FAILED
6272  026466  104004                          ERROR   4               ;       *
6273  026470  104004                          ERROR   4               ;       *
6274  026472  170104                  10$:    LDFPS   R4
6275  026474  005737  026460                  TST     9$              ;DATA OK?
6276  026500  001406                          BEQ     11$             ;BRANCH IF YES
6277  026502  005037  001200                  CLR     $TMP0
6278  026506  013737  026460  001202          MOV     9$,$TMP1
6279  026514  104162                          ERROR   162             ;DATA WRONG
6280  026516                          11$:
6281  026516  012737  026524  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
6282  026524  170137  001364                  LDFPS   $FPS
6283  026530  170527                          TSTF    (PC)+           ;EXECUTE TEST
6284  026532  106700                  12$:    .WORD   106700
6285  026534  000403                          BR      13$
6286  026536  104004                          ERROR   4               ;ADX ROM FAILED
6287  026540  104004                          ERROR   4               ;       *
6288  026542  104004                          ERROR   4               ;       *
6289  026544  170200                  13$:    STFPS   R0
6290  026546  170104                          LDFPS   R4
6291  026550  042700  177760                  BIC     #177760,R0
6292  026554  022700  000010                  CMP     #FN,R0          ;CC'S OK?
6293  026560  001406                          BEQ     14$             ;BRANCH IF YES
6294  026562  010037  001202                  MOV     R0,$TMP1
6295  026566  012737  000010  001200          MOV     #FN,$TMP0
6296  026574  104003                          ERROR   3               ;CC'S BAD
6297  026576                          14$:
6298  026576  012737  026604  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
6299  026604  012737  106400  026620          MOV     #106400,15$
6300  026612  170137  001364                  LDFPS   $FPS
6301  026616  170627                          ABSF    (PC)+           ;EXECUTE TEST
6302  026620  106400                  15$:    .WORD   106400
6303  026622  000403                          BR      16$
6304  026624  104004                          ERROR   4               ;ADX ROM FAILED
```

```
6305  026626  104004                          ERROR   4               ;       *
6306  026630  104004                          ERROR   4               ;       *
6307  026632  170104                  16$:    LDFPS   R4
6308  026634  022737  006400  026620          CMP     #6400,15$       ;DATA OK?
6309  026642  001407                          BEQ     17$             ;BRANCH IF YES
6310  026644  013737  026620  001202          MOV     15$,$TMP1
6311  026652  012737  006400  001200          MOV     #6400,$TMP0
6312  026660  104162                          ERROR   162             ;DATA BAD
6313  026662                          17$:
6314  026662  012737  026670  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
6315  026670  012737  006400  026704          MOV     #6400,18$
6316  026676  170137  001364                  LDFPS   $FPS
6317  026702  170727                          NEGF    (PC)+           ;EXECUTE TEST
6318  026704  006400                  18$:    .WORD   6400
6319  026706  000403                          BR      19$
6320  026710  104004                          ERROR   4               ;ADX ROM FAILED
6321  026712  104004                          ERROR   4               ;       *
6322  026714  104004                          ERROR   4               ;       *
6323  026716  170104                  19$:    LDFPS   R4
6324  026720  022737  106400  026704          CMP     #106400,18$     ;DATA OK?
6325  026726  001407                          BEQ     20$             ;BRANCH IF YES
6326  026730  012737  106400  001200          MOV     #106400,$TMP0
6327  026736  013737  026704  001202          MOV     18$,$TMP1
6328  026744  104162                          ERROR   162             ;DATA BAD
6329  026746                          20$:
6330  026746  012737  026754  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
6331  026754  012737  040200  001200          MOV     #40200,$TMP0
6332  026762  005037  001202                  CLR     $TMP1
6333  026766  172737  001200                  LDF     $TMP0,AC3
6334  026772  170137  001364                  LDFPS   $FPS
6335  026776  171327                          MULF    (PC)+,AC3       ;EXECUTE TEST
6336  027000  007000                          .WORD   7000
6337  027002  000403                          BR      21$
6338  027004  104004                          ERROR   4               ;ADX ROM FAILED
6339  027006  104004                          ERROR   4               ;       *
6340  027010  104004                          ERROR   4               ;       *
6341  027012  170104                  21$:    LDFPS   R4
6342  027014  173727  007000                  CMPF    #↑07000,AC3     ;DATA OK?
6343  027020  170000                          CFCC
6344  027022  001406                          BEQ     22$             ;BRANCH IF YES
6345  027024  174337  001204                  STF     AC3,$TMP2
6346  027030  012737  007000  001200          MOV     #7000,$TMP0
6347  027036  104024                          ERROR   24              ;DATA BAD
6348  027040                          22$:
6349  027040  012737  027046  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
6350  027046  172637  001200                  LDF     $TMP0,AC2       ;INIT AC2
6351  027052  170137  001364                  LDFPS   $FPS
6352  027056  171627                          MODF    (PC)+,AC2       ;EXECUTE TEST
6353  027060  106700                          .WORD   106700
6354  027062  000403                          BR      23$
6355  027064  104004                          ERROR   4               ;ADX ROM FAILED
6356  027066  104004                          ERROR   4               ;       *
6357  027070  104004                          ERROR   4               ;       *
6358  027072  170104                  23$:    LDFPS   R4
6359  027074  173627  106700                  CMPF    #↑0106700,AC2   ;RESULT OK?
6360  027100  170000                          CFCC
```

```
6361  027102  001406                        BEQ     24$                 ;BRANCH IF YES
6362  027104  012737  106700  001200        MOV     #106700,$TMP0
6363  027112  174237  001204                STF     AC2,$TMP2
6364  027116  104024                        ERROR   24                  ;DATA BAD
6365  027120                          24$:
6366  027120  012737  027126  001110        MOV     #.+6,@#$LPERR       ;SET ERROR LOOP
6367  027126  012737  106400  001200        MOV     #106400,$TMP0
6368  027134  172737  001200                LDF     $TMP0,AC3
6369  027140  170137  001364                LDFPS   $FPS
6370  027144  172327                        ADDF    (PC)+,AC3           ;EXECUTE TEST
6371  027146  106400                        .WORD   106400
6372  027150  000403                        BR      25$
6373  027152  104004                        ERROR   4                   ;ADX ROM FAILED
6374  027154  104004                        ERROR   4                   ;       *
6375  027156  104004                        ERROR   4                   ;       *
6376  027160  170104                  25$:  LDFPS   R4
6377  027162  173727  106600                CMPF    #^O106600,AC3       ;RESULT OK?
6378  027166  170000                        CFCC
6379  027170  001406                        BEQ     26$                 ;BRANCH IF YES
6380  027172  174337  001204                STF     AC3,$TMP2
6381  027176  012737  106600  001200        MOV     #106600,$TMP0
6382  027204  104024                        ERROR   24                  ;RESULT WRONG
6383  027206                          26$:
6384  027206  012737  027214  001110        MOV     #.+6,@#$LPERR       ;SET ERROR LOOP
6385  027214  170401                        CLRF    AC1                 ;INIT AC1
6386  027216  170137  001364                LDFPS   $FPS
6387  027222  172527                        LDF     (PC)+,AC1           ;EXECUTE TEST
6388  027224  107777                        .WORD   107777
6389  027226  000403                        BR      27$
6390  027230  104004                        ERROR   4                   ;ADX ROM FAILED
6391  027232  104004                        ERROR   4                   ;       *
6392  027234  104004                        ERROR   4                   ;       *
6393  027236  170104                  27$:  LDFPS   R4
6394  027240  173527  107777                CMPF    #^O107777,AC1       ;RESULT OK?
6395  027244  170000                        CFCC
6396  027246  001406                        BEQ     28$                 ;BRANCH IF YES
6397  027250  012737  107777  001200        MOV     #107777,$TMP0
6398  027256  174137  001204                STF     AC1,$TMP2
6399  027262  104024                        ERROR   24                  ;RESULT WRONG
6400  027264                          28$:
6401  027264  012737  027272  001110        MOV     #.+6,@#$LPERR       ;SET ERROR LOOP
6402  027272  012737  107100  001200        MOV     #107100,$TMP0
6403  027300  172637  001200                LDF     $TMP0,AC2
6404  027304  170137  001364                LDFPS   $FPS
6405  027310  173227                        SUBF    (PC)+,AC2           ;EXECUTE TEST
6406  027312  106700                        .WORD   106700
6407  027314  000403                        BR      29$
6408  027316  104004                        ERROR   4                   ;ADX ROM FAILED
6409  027320  104004                        ERROR   4                   ;       *
6410  027322  104004                        ERROR   4                   ;       *
6411  027324  170104                  29$:  LDFPS   R4
6412  027326  173627  106700                CMPF    #^O106700,AC2       ;RESULT OK?
6413  027332  170000                        CFCC
6414  027334  001406                        BEQ     30$                 ;BRANCH IF YES
6415  027336  174237  001204                STF     AC2,$TMP2
6416  027342  012737  106700  001200        MOV     #106700,$TMP0
```

```
6417  027350  104024                          ERROR   24              ;RESULT WRONG
6418  027352                          30$:
6419  027352  012737  027360  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
6420  027360  172427  107777                  LDF     #^O107777,AC0   ;INIT AC0
6421  027364  170137  001364                  LDFPS   $FPS
6422  027370  173427                          CMPF    (PC)+,AC0       ;EXECUTE TEST
6423  027372  107777                          .WORD   107777
6424  027374  000403                          BR      31$
6425  027376  104004                          ERROR   4               ;ADX ROM FAILED
6426  027400  104004                          ERROR   4               ;       *
6427  027402  104004                          ERROR   4               ;       *
6428  027404  170200                  31$:    STFPS   R0
6429  027406  170104                          LDFPS   R4
6430  027410  042700  177760                  BIC     #177760,R0
6431  027414  022700  000004                  CMP     #FZ,R0          ;CC'S OK?
6432  027420  001406                          BEQ     32$             ;BRANCH IF YES
6433  027422  010037  001202                  MOV     R0,$TMP1
6434  027426  012737  000004  001200          MOV     #FZ,$TMP0
6435  027434  104003                          ERROR   3               ;CC'S BAD
6436  027436                          32$:
6437  027436  012737  027444  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
6438  027444  005037  027462                  CLR     33$
6439  027450  172727  106700                  LDF     #^O106700,AC3   ;INIT AC3
6440  027454  170137  001364                  LDFPS   $FPS
6441  027460  174327                          STF     AC3,(PC)+       ;EXECUTE TEST
6442  027462  000000                  33$:    .WORD   0
6443  027464  000403                          BR      34$
6444  027466  104004                          ERROR   4               ;ADX ROM FAILED
6445  027470  104004                          ERROR   4               ;       *
6446  027472  104004                          ERROR   4               ;       *
6447  027474  170104                  34$:    LDFPS   R4
6448  027476  022737  106700  027462          CMP     #106700,33$     ;DATA OK?
6449  027504  001407                          BEQ     35$             ;BRANCH IF YES
6450  027506  012737  106700  001200          MOV     #106700,$TMP0
6451  027514  013737  027462  001202          MOV     33$,$TMP1
6452  027522  104162                          ERROR   162             ;DATA WRONG
6453  027524                          35$:
6454  027524  012737  027532  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
6455  027532  012737  106700  001200          MOV     #106700,$TMP0
6456  027540  172637  001200                  LDF     $TMP0,AC2       ;INIT AC2
6457  027544  170137  001364                  LDFPS   $FPS
6458  027550  174627                          DIVF    (PC)+,AC2       ;EXECUTE TEST
6459  027552  106700                          .WORD   106700
6460  027554  000403                          BR      36$
6461  027556  104004                          ERROR   4               ;ADX ROM FAILED
6462  027560  104004                          ERROR   4               ;       *
6463  027562  104004                          ERROR   4               ;       *
6464  027564  170104                  36$:    LDFPS   R4
6465  027566  173627  040200                  CMPF    #^O040200,AC2   ;RESULT OK?
6466  027572  170000                          CFCC
6467  027574  001406                          BEQ     37$             ;BRANCH IF YES
6468  027576  174237  001204                  STF     AC2,$TMP2
6469  027602  012737  040200  001200          MOV     #40200,$TMP0
6470  027610  104024                          ERROR   24              ;RESULT WRONG
6471  027612                          37$:
6472  027612  012737  027620  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
```

```
6473  027620  005037  027636                    CLR     38$
6474  027624  172727  043600                    LDF     #↑043600,AC3    ;INIT AC3
6475  027630  170137  001364                    LDFPS   $FPS
6476  027634  175327                            STEXP   AC3,(PC)+       ;EXECUTE TEST
6477  027636  000000                  38$:      .WORD   0
6478  027640  000403                            BR      39$
6479  027642  104004                            ERROR   4               ;ADX ROM FAILED
6480  027644  104004                            ERROR   4               ;       *
6481  027646  104004                            ERROR   4               ;       *
6482  027650  170104                  39$:      LDFPS   R4
6483  027652  022737  000017  027636            CMP     #17,38$         ;RESULT OK?
6484  027660  001407                            BEQ     40$             ;BRANCH IF YES
6485  027662  012737  000017  001200            MOV     #17,$TMP0
6486  027670  013737  027636  001202            MOV     38$,$TMP1
6487  027676  104162                            ERROR   162             ;RESULT WRONG
6488  027700                      40$:
6489  027700  012737  027706  001110            MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
6490  027706  032737  000100  001364            BIT     #FL,$FPS        ;FL BIT SET?
6491  027714  001403                            BEQ     127$            ;BRANCH IF NO
6492  027716  172527  045160                    LDF     #↑045160,AC1
6493  027722  000402                            BR      126$
6494  027724  172527  041160            127$:   LDF     #↑041160,AC1
6495  027730  005037  027742            126$:   CLR     41$
6496  027734  170137  001364                    LDFPS   $FPS
6497  027740  175527                            STCFI   AC1,(PC)+       ;EXECUTE TEST
6498  027742  000000                  41$:      .WORD   0
6499  027744  000403                            BR      42$
6500  027746  104004                            ERROR   4               ;ADX ROM FAILED
6501  027750  104004                            ERROR   4               ;       *
6502  027752  104004                            ERROR   4               ;       *
6503  027754  170104                  42$:      LDFPS   R4
6504  027756  022737  000017  027742            CMP     #17,41$         ;RESULT OK?
6505  027764  001407                            BEQ     43$             ;BRANCH IF YES
6506  027766  012737  000017  001200            MOV     #17,$TMP0
6507  027774  013737  027742  001202            MOV     41$,$TMP1
6508  030002  104162                            ERROR   162             ;RESULT WRONG
6509  030004                      43$:
6510  030004  012737  030012  001110            MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
6511  030012  172727  106700                    LDF     #↑0106700,AC3   ;INIT AC3
6512  030016  005037  030030                    CLR     44$
6513  030022  170137  001364                    LDFPS   $FPS
6514  030026  176327                            STCFD   AC3,(PC)+       ;EXECUTE TEST
6515  030030  000000                  44$:      .WORD   0
6516  030032  000403                            BR      45$
6517  030034  104004                            ERROR   4               ;ADX ROM FAILED
6518  030036  104004                            ERROR   4               ;       *
6519  030040  104004                            ERROR   4               ;       *
6520  030042  170104                  45$:      LDFPS   R4
6521  030044  022737  106700  030030            CMP     #106700,44$     ;RESULT OK?
6522  030052  001407                            BEQ     46$             ;BRANCH IF YES
6523  030054  012737  106700  001200            MOV     #106700,$TMP0
6524  030062  013737  030030  001202            MOV     44$,$TMP1
6525  030070  104162                            ERROR   162             ;RESULT WRONG
6526  030072                      46$:
6527  030072  012737  030100  001110            MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
6528  030100  170402                            CLRF    AC2             ;INIT AC2
```

```
 6529  030102  170137  001364                    LDFPS   $FPS
 6530  030106  176627                            LDEXP   (PC)+,AC2       ;EXECUTE TEST
 6531  030110  000017                            .WORD   17
 6532  030112  000403                            BR      47$
 6533  030114  104004                            ERROR   4               ;ADX ROM FAILED
 6534  030116  104004                            ERROR   4               ;       *
 6535  030120  104004                            ERROR   4               ;       *
 6536  030122  170104                    47$:    LDFPS   R4
 6537  030124  173627  043600                    CMPF    #^O043600,AC2   ;RESULT OK?
 6538  030130  170000                            CFCC
 6539  030132  001406                            BEQ     48$             ;BRANCH IF YES
 6540  030134  012737  043600  001200            MOV     #43600,$TMP0
 6541  030142  174237  001204                    STF     AC2,$TMP2
 6542  030146  104024                            ERROR   24              ;RESULT WRONG
 6543  030150                    48$:
 6544  030150  012737  030156  001110            MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
 6545  030156  170403                            CLRF    AC3             ;INIT AC3
 6546  030160  032737  000100  001364            BIT     #FL,$FPS        ;IL BIT ON?
 6547  030166  001403                            BEQ     125$            ;BRANCH IF NO
 6548  030170  172427  147746                    LDF     #^O0147746,AC0
 6549  030174  000402                            BR      124$
 6550  030176  172427  143746            125$:   LDF     #^O0143746,AC0
 6551  030202  170137  001364            124$:   LDFPS   $FPS
 6552  030206  177327                            LDCIF   (PC)+,AC3       ;EXECUTE TEST
 6553  030210  106400                            .WORD   106400
 6554  030212  000403                            BR      49$
 6555  030214  104004                            ERROR   4               ;ADX ROM FAILED
 6556  030216  104004                            ERROR   4               ;       *
 6557  030220  104004                            ERROR   4               ;       *
 6558  030222  170104                    49$:    LDFPS   R4
 6559  030224  173700                            CMPF    AC0,AC3         ;RESULT OK?
 6560  030226  170000                            CFCC
 6561  030230  001405                            BEQ     50$             ;BRANCH IF YES
 6562  030232  174037  001200                    STF     AC0,$TMP0
 6563  030236  174337  001204                    STF     AC3,$TMP2
 6564  030242  104024                            ERROR   24              ;RESULT WRONG
 6565  030244                    50$:
 6566  030244  012737  030252  001110            MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
 6567  030252  170401                            CLRF    AC1             ;INIT AC1
 6568  030254  170137  001364                    LDFPS   $FPS
 6569  030260  177527                            LDCFD   (PC)+,AC1       ;EXECUTE TEST
 6570  030262  107777                            .WORD   107777
 6571  030264  000403                            BR      51$
 6572  030266  104004                            ERROR   4               ;ADX ROM FAILED
 6573  030270  104004                            ERROR   4               ;       *
 6574  030272  104004                            ERROR   4               ;       *
 6575  030274  170104                    51$:    LDFPS   R4
 6576  030276  173527  107777                    CMPF    #^O0107777,AC1  ;RESULT OK?
 6577  030302  170000                            CFCC
 6578  030304  001406                            BEQ     52$             ;BRANCH IF YES
 6579  030306  012737  107777  001200            MOV     #107777,$TMP0
 6580  030314  174137  001204                    STF     AC1,$TMP2
 6581  030320  104024                            ERROR   24              ;RESULT WRONG
 6582  030322                    52$:
 6583  030322  012737  030330  001110            MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
 6584  030330  042737  177077  001364            BIC     #177077,$FPS
```

```
6585  030336  022737  000300  001364          CMP     #300,$FPS       ;DONE YET?
6586  030344  001412                          BEQ     96$             ;BRANCH IF YES
6587  030346  062737  000100  001364          ADD     #BIT6,$FPS
6588  030354  000137  026174                  JMP     LOOP2           ;CONTINUE
6589  030360  022626                  63$:    CMP     (SP)+,(SP)+     ;RESTORE THE SP
6590  030362  016637  177774  001200          MOV     -4(SP),$TMP0
6591  030370  104163                          ERROR   163             ;ADX ROM FAILED
6592
6593                                  ;;**************************************************************
6594                                  ;NOT MODE 0*NOT IMMEDIATE INSTRUCTIONS
6595  030372  012737  000017  001364  96$:    MOV     #17,$FPS        ;INIT $FPS
6596  030400                          LABLE5:
6597  030400  012737  047000  001200          MOV     #47000,$TMP0    ;INIT $TMP0
6598  030406  005037  001202                  CLR     $TMP1
6599  030412  005037  001204                  CLR     $TMP2
6600  030416  005037  001206                  CLR     $TMP3
6601  030422  012737  001204  001160          MOV     #$TMP2,$REG0    ;PUT ADR OF $TMP2 IN $REG0
6602  030430  012737  001210  001162          MOV     #$TMP4,$REG1
6603  030436  012737  001214  001164          MOV     #$TMP6,$REG2
6604  030444  012737  001220  001166          MOV     #$TMP7+2,$REG3
6605  030452  012737  001212  001170          MOV     #$TMP5,$REG4
6606  030460  012737  001214  001172          MOV     #$TMP6,$REG5
6607  030466  005005                          CLR     R5              ;INIT R5
6608  030470  012703  000002                  MOV     #2,R3
6609  030474  005001                          CLR     R1              ;INIT R1
6610  030476                          LOOP3:
6611  030476  012737  030504  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
6612  030504  012700  001204                  MOV     #$TMP2,R0
6613  030510  013702  001364                  MOV     $FPS,R2
6614  030514  170102                          LDFPS   R2
6615  030516  170120                          LDFPS   (R0)+           ;EXECUTE TEST
6616  030520  170102                          LDFPS   R2
6617  030522  170220                          STFPS   (R0)+           ;EXECUTE TEST
6618  030524  170104                          LDFPS   R4
6619  030526  022700  001210                  CMP     #$TMP4,R0       ;ADX ROM OK?
6620  030532  001401                          BEQ     1$              ;BRANCH IF YES
6621  030534  104004                          ERROR   4               ;ADX ROM FAILED
6622  030536                          1$:
6623  030536  012737  030544  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
6624  030544  012700  001210                  MOV     #$TMP4,R0
6625  030550  170137  001364                  LDFPS   $FPS
6626  030554  170320                          STST    (R0)+           ;EXECUTE TEST
6627  030556  170104                          LDFPS   R4
6628  030560  022700  001214                  CMP     #$TMP6,R0       ;ADX ROM OK?
6629  030564  001401                          BEQ     2$              ;BRANCH IF YES
6630  030566  104004                          ERROR   4               ;ADX ROM FAILED
6631  030570  022737  000016  001210  2$:     CMP     #16,$TMP4       ;DATA OK?
6632  030576  001407                          BEQ     3$              ;BRANCH IF YES
6633  030600  012737  000016  001200          MOV     #16,$TMP0
6634  030606  013737  001210  001204          MOV     $TMP4,$TMP2
6635  030614  104101                          ERROR   101             ;FEC WRONG
6636  030616                          3$:
6637  030616  012737  030624  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
6638  030624  012737  040000  001210          MOV     #40000,$TMP4
6639  030632  012700  001210                  MOV     #$TMP4,R0
6640  030636  170137  001364                  LDFPS   $FPS
```

```
6641  030642  170420                          CLRF    (R0)+           ;EXECUTE TEST
6642  030644  026100  001164                  CMP     $REG2(R1),R0    ;ADX ROM OK?
6643  030650  001401                          BEQ     4$              ;BRANCH IF YES
6644  030652  104004                          ERROR   4               ;ADX ROM FAILED
6645  030654  170104                  4$:     LDFPS   R4
6646  030656  005737  001210                  TST     $TMP4           ;RESULT OK?
6647  030662  001401                          BEQ     5$              ;BRANCH IF YES
6648  030664  104061                          ERROR   61              ;RESULT WRONG
6649  030666                          5$:
6650  030666  012737  030674  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
6651  030674  012700  001200                  MOV     #$TMP0,R0
6652  030700  170137  001364                  LDFPS   $FPS
6653  030704  170520                          TSTF    (R0)+           ;EXECUTE TEST
6654  030706  170202                          STFPS   R2
6655  030710  026100  001160                  CMP     $REG0(R1),R0    ;ADX ROM OK?
6656  030714  001401                          BEQ     6$              ;BRANCH IF YES
6657  030716  104004                          ERROR   4               ;ADX ROM FAILED
6658  030720  170104                  6$:     LDFPS   R4
6659  030722  042702  177760                  BIC     #177760,R2
6660  030726  001405                          BEQ     7$              ;BRANCH IF CC'S OK
6661  030730  010237  001202                  MOV     R2,$TMP1
6662  030734  005037  001200                  CLR     $TMP0
6663  030740  104003                          ERROR   3               ;CC'S BAD
6664  030742                          7$:
6665  030742  012737  030750  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
6666  030750  012700  001210                  MOV     #$TMP4,R0
6667  030754  012710  140000                  MOV     #140000,(R0)
6668  030760  170137  001364                  LDFPS   $FPS
6669  030764  170620                          ABSF    (R0)+           ;EXECUTE TEST
6670  030766  170104                          LDFPS   R4
6671  030770  026100  001164                  CMP     $REG2(R1),R0    ;ADX ROM OK?
6672  030774  001401                          BEQ     8$              ;BRANCH IF YES
6673  030776  104004                          ERROR   4               ;ADX ROM FAILED
6674  031000  022737  040000  001210  8$:     CMP     #40000,$TMP4    ;DATA OK?
6675  031006  001401                          BEQ     9$              ;BRANCH IF YES
6676  031010  104061                          ERROR   61              ;DATA BAD
6677  031012                          9$:
6678  031012  012737  031020  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
6679  031020  170104                          LDFPS   R4
6680  031022  012700  001210                  MOV     #$TMP4,R0
6681  031026  012710  140000                  MOV     #140000,(R0)
6682  031032  170137  001364                  LDFPS   $FPS
6683  031036  170720                          NEGF    (R0)+           ;EXECUTE TEST
6684  031040  170104                          LDFPS   R4
6685  031042  026100  001164                  CMP     $REG2(R1),R0    ;ADX ROM OK?
6686  031046  001401                          BEQ     10$             ;BRANCH IF YES
6687  031050  104004                          ERROR   4               ;ADX ROM FAILED
6688  031052  022737  040000  001210  10$:    CMP     #40000,$TMP4    ;RESULT OK?
6689  031060  001401                          BEQ     11$             ;BRANCH IF YES
6690  031062  104061                          ERROR   61              ;RESULT WRONG
6691  031064                          11$:
6692  031064  012737  031072  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
6693  031072  012737  040400  001200          MOV     #40400,$TMP0
6694  031100  012700  001200                  MOV     #$TMP0,R0
6695  031104  172727  040400                  LDF     #^O040400,AC3
6696  031110  170137  001364                  LDFPS   $FPS
```

```
6697  031114  171320                          MULF    (R0)+,AC3       ;EXECUTE TEST
6698  031116  170104                          LDFPS   R4
6699  031120  026100  001160                  CMP     $REGO(R1),R0    ;ADX ROM OK?
6700  031124  001401                          BEQ     12$             ;BRANCH IF YES
6701  031126  104004                          ERROR   4               ;ADX ROM FAILED
6702  031130  173727  040600          12$:    CMPF    #^O040600,AC3   ;RESULT OK?
6703  031134  170000                          CFCC
6704  031136  001406                          BEQ     13$             ;BRANCH IF YES
6705  031140  012737  040600  001200          MOV     #40600,$TMP0
6706  031146  174337  001210                  STF     AC3,$TMP4
6707  031152  104061                          ERROR   61              ;RESULT WRONG
6708  031154                          13$:
6709  031154  012737  031162  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
6710  031162  012700  001200                  MOV     #$TMP0,R0
6711  031166  005037  001206                  CLR     $TMP3
6712  031172  172427  040400                  LDF     #^O040400,AC0   ;INIT AC0
6713  031176  170137  001364                  LDFPS   $FPS
6714  031202  171420                          MODF    (R0)+,AC0       ;EXECUTE TEST
6715  031204  026100  001160                  CMP     $REGO(R1),R0    ;ADX ROM OK?
6716  031210  001401                          BEQ     14$             ;BRANCH IF YES
6717  031212  104004                          ERROR   4               ;ADX ROM FAILED
6718  031214  170500                  14$:    TSTF    AC0             ;RESULT OK?
6719  031216  170000                          CFCC
6720  031220  001405                          BEQ     15$             ;BRANCH IF YES
6721  031222  005037  001200                  CLR     $TMP0
6722  031226  174037  001210                  STF     AC0,$TMP4
6723  031232  104061                          ERROR   61              ;RESULT WRONG
6724  031234                          15$:
6725  031234  012737  031242  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
6726  031242  012737  040200  001200          MOV     #40200,$TMP0
6727  031250  012700  001200                  MOV     #$TMP0,R0
6728  031254  172627  040400                  LDF     #^O040400,AC2   ;INIT AC2
6729  031260  170137  001364                  LDFPS   $FPS
6730  031264  173220                          SUBF    (R0)+,AC2       ;EXECUTE TEST
6731  031266  026100  001160                  CMP     $REGO(R1),R0    ;ADX ROM OK?
6732  031272  001401                          BEQ     16$             ;BRANCH IF YES
6733  031274  104004                          ERROR   4               ;ADX ROM FAILED
6734  031276  173627  040200          16$:    CMPF    #^O040200,AC2   ;RESULT OK?
6735  031302  170000                          CFCC
6736  031304  001403                          BEQ     17$             ;BRANCH IF YES
6737  031306  174237  001210                  STF     AC2,$TMP4
6738  031312  104061                          ERROR   61              ;RESULT WRONG
6739  031314                          17$:
6740  031314  012737  031322  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
6741  031322  012700  001200                  MOV     #$TMP0,R0
6742  031326  172737  001200                  LDF     $TMP0,AC3
6743  031332  170137  001364                  LDFPS   $FPS
6744  031336  173720                          CMPF    (R0)+,AC3       ;EXECUTE TEST
6745  031340  026100  001160                  CMP     $REGO(R1),R0    ;ADX ROM OK?
6746  031344  001401                          BEQ     18$             ;BRANCH IF YES
6747  031346  104004                          ERROR   4               ;ADX ROM FAILED
6748  031350  170200                  18$:    STFPS   R0
6749  031352  042700  177760                  BIC     #177760,R0
6750  031356  022700  000004                  CMP     #FZ,R0          ;CC'S OK?
6751  031362  001406                          BEQ     19$             ;BRANCH IF YES
6752  031364  010037  001202                  MOV     R0,$TMP1
```

```
 6753  031370  012737  000004  001200          MOV     #FZ,$TMP0
 6754  031376  104003                          ERROR   3               ;CC'S BAD
 6755  031400                          19$:
 6756  031400  012737  031406  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
 6757  031406  012700  001210                  MOV     #$TMP4,R0
 6758  031412  005010                          CLR     (R0)
 6759  031414  172637  001200                  LDF     $TMP0,AC2       ;INIT AC2
 6760  031420  170137  001364                  LDFPS   $FPS
 6761  031424  174220                          STF     AC2,(R0)+       ;EXECUTE TEST
 6762  031426  026100  001164                  CMP     $REG2(R1),R0    ;ADX ROM OK?
 6763  031432  001401                          BEQ     20$             ;BRANCH IF YES
 6764  031434  104004                          ERROR   4               ;ADX ROM FAILED
 6765  031436  173637  001210          20$:    CMPF    $TMP4,AC2       ;RESULT OK?
 6766  031442  170000                          CFCC
 6767  031444  001401                          BEQ     21$             ;BRANCH IF YES
 6768  031446  104061                          ERROR   61              ;RESULT WRONG
 6769  031450                          21$:
 6770  031450  012737  031456  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
 6771  031456  012700  001200                  MOV     #$TMP0,R0
 6772  031462  012710  040400                  MOV     #40400,(R0)
 6773  031466  172727  040200                  LDF     #^O040200,AC3
 6774  031472  170137  001364                  LDFPS   $FPS
 6775  031476  174720                          DIVF    (R0)+,AC3       ;EXECUTE TEST
 6776  031500  026100  001160                  CMP     $REG0(R1),R0    ;ADX ROM OK?
 6777  031504  001401                          BEQ     22$             ;BRANCH IF YES
 6778  031506  104004                          ERROR   4               ;ADX ROM FAILED
 6779  031510  173727  040000          22$:    CMPF    #^O040000,AC3   ;RESULT OK?
 6780  031514  170000                          CFCC
 6781  031516  001406                          BEQ     23$             ;BRANCH IF YES
 6782  031520  174337  001210                  STF     AC3,$TMP4
 6783  031524  012737  040000  001200          MOV     #40000,$TMP0
 6784  031532  104061                          ERROR   61              ;RESULT WRONG
 6785  031534                          23$:
 6786  031534  012737  031542  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
 6787  031542  012700  001210                  MOV     #$TMP4,R0
 6788  031546  172727  077777                  LDF     #^O077777,AC3
 6789  031552  170137  001364                  LDFPS   $FPS
 6790  031556  175320                          STEXP   AC3,(R0)+       ;EXECUTE TEST
 6791  031560  022700  001212                  CMP     #$TMP5,R0       ;ADX ROM OK?
 6792  031564  001401                          BEQ     24$             ;BRANCH IF YES
 6793  031566  104004                          ERROR   4               ;ADX ROM FAILED
 6794  031570  022737  000177  001210  24$:    CMP     #177,$TMP4      ;RESULT OK?
 6795  031576  001407                          BEQ     25$             ;BRANCH IF YES
 6796  031600  012737  000177  001200          MOV     #177,$TMP0
 6797  031606  013737  001210  001202          MOV     $TMP4,$TMP1
 6798  031614  104162                          ERROR   162             ;RESULT WRONG
 6799  031616                          25$:
 6800  031616  012737  031624  001110          MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
 6801  031624  012700  001210                  MOV     #$TMP4,R0
 6802  031630  005010                          CLR     (R0)
 6803  031632  005060  000002                  CLR     2(R0)
 6804  031636  172527  041777                  LDF     #^O041777,AC1   ;INIT AC1
 6805  031642  170137  001364                  LDFPS   $FPS
 6806  031646  175520                          STCFI   AC1,(R0)+       ;EXECUTE TEST
 6807  031650  026500  001170                  CMP     $REG4(R5),R0    ;ADX ROM OK?
 6808  031654  001401                          BEQ     26$             ;BRANCH IF YES
```

```
6809  031656  104004                                    ERROR   4               ;ADX ROM FAILED
6810  031660  022765  000177  001210  26$:              CMP     #177,$TMP4(R5)  ;RESULT OK?
6811  031666  001410                                    BEQ     27$             ;BRANCH IF YES
6812  031670  005037  001202                            CLR     $TMP1
6813  031674  005037  001200                            CLR     $TMP0
6814  031700  012765  000177  001200                    MOV     #177,$TMP0(R5)
6815  031706  104161                                    ERROR   161             ;RESULT WRONG
6816  031710                                27$:
6817  031710  012737  031716  001110                    MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
6818  031716  012700  001210                            MOV     #$TMP4,R0
6819  031722  172627  040200                            LDF     #^040200,AC2    ;INIT AC2
6820  031726  005010                                    CLR     (R0)
6821  031730  170137  001364                            LDFPS   $FPS
6822  031734  176220                                    STCFD   AC2,(R0)+       ;EXECUTE TEST
6823  031736  026300  001164                            CMP     $REG2(R3),R0    ;ADX ROM OK?
6824  031742  001401                                    BEQ     28$             ;BRANCH IF YES
6825  031744  104004                                    ERROR   4               ;ADX ROM FAILED
6826  031746  022737  040200  001210  28$:              CMP     #40200,$TMP4    ;RESULT OK?
6827  031754  001401                                    BEQ     29$             ;BRANCH IF YES
6828  031756  104061                                    ERROR   61              ;RESULT WRONG
6829  031760                                29$:
6830  031760  012737  031766  001110                    MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
6831  031766  012700  001210                            MOV     #$TMP4,R0
6832  031772  012710  000077                            MOV     #77,(R0)
6833  031776  170401                                    CLRF    AC1
6834  032000  170137  001364                            LDFPS   $FPS
6835  032004  176520                                    LDEXP   (R0)+,AC1       ;EXECUTE TEST
6836  032006  022700  001212                            CMP     #$TMP5,R0       ;ADX ROM OK?
6837  032012  001401                                    BEQ     30$             ;BRANCH IF YES
6838  032014  104004                                    ERROR   4               ;ADX ROM FAILED
6839  032016  173527  057600                30$:       CMPF    #^057600,AC1    ;RESULT OK?
6840  032022  170000                                    CFCC
6841  032024  001406                                    BEQ     31$             ;BRANCH IF YES
6842  032026  174137  001210                            STF     AC1,$TMP4
6843  032032  012737  057600  001200                    MOV     #5760[illegible] $TMP0
6844  032040  104061                                    ERROR   61              ;RESLT WRONG
6845  032042                                31$:
6846  032042  012737  032050  001110                    MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
6847  032050  012700  001210                            MOV     #$TMP4,R0
6848  032054  005037  001210                            CLR     $TMP4
6849  032060  012765  000017  001210                    MOV     #17,$TMP4(R5)   ;
6850  032066  170403                                    CLRF    AC3
6851  032070  170137  001364                            LDFPS   $FPS
6852  032074  177320                                    LDCIF   (R0)+,AC3       ;EXECUTE TEST
6853  032076  005065  001210                            CLR     $TMP4(R5)
6854  032102  026500  001170                            CMP     $REG4(R5),R0    ;ADX ROM OK?
6855  032106  001401                                    BEQ     32$             ;BRANCH IF YES
6856  032110  104004                                    ERROR   4               ;ADX ROM FAILED
6857  032112  173727  041160                32$:       CMPF    #^041160,AC3    ;RESULT OK?
6858  032116  170000                                    CFCC
6859  032120  001406                                    BEQ     33$             ;BRANCH IF YES
6860  032122  174337  001210                            STF     AC3,$TMP4
6861  032126  012737  041160  001200                    MOV     #41160,$TMP0
6862  032134  104061                                    ERROR   61              ;RESULT WRONG
6863  032136                                33$:
6864  032136  012737  032144  001110                    MOV     #.+6,@#$LPERR   ;SET ERROR LOOP
```

```
 6865  032144  012700  001210                        MOV     #$TMP4,R0
 6866  032150  170402                                CLRF    AC2
 6867  032152  012710  040000                        MOV     #40000,(R0)
 6868  032156  170137  001364                        LDFPS   $FPS
 6869  032162  177620                                LDCFD   (R0)+,AC2       ;EXECUTE TEST
 6870  032164  026300  001164                        CMP     $REG2(R3),R0    ;ADX ROM OK?
 6871  032170  001401                                BEQ     34$             ;BRANCH IF YES
 6872  032172  104004                                ERROR   4               ;ADX ROM FAILED
 6873  032174  173637  001210            34$:        CMPF    $TMP4,AC2       ;RESULT OK?
 6874  032200  170000                                CFCC
 6875  032202  001406                                BEQ     35$             ;BRANCH IF YES
 6876  032204  174237  001210                        STF     AC2,$TMP4
 6877  032210  012737  040000  001200                MOV     #40000,$TMP0
 6878  032216  104061                                ERROR   61              ;RESULT WRONG
 6879  032220  013700  001364            35$:        MOV     $FPS,R0
 6880  032224  042700  177477                        BIC     #177477,R0
 6881  032230  022700  000300                        CMP     #300,R0         ;DONE YET?
 6882  032234  001426                                BEQ     STARTCL         ;BRANCH IF YES
 6883  032236  062737  000100  001364                ADD     #BIT6,$FPS
 6884  032244  005005                                CLR     R5
 6885  032246  032737  000100  001364                BIT     #BIT6,$FPS
 6886  032254  001402                                BEQ     36$
 6887  032256  012705  000002                        MOV     #2,R5
 6888  032262  005001            36$:        CLR     R1
 6889  032264  012703  000002                        MOV     #2,R3
 6890  032270  032737  000200  001364                BIT     #BIT7,$FPS
 6891  032276  001403                                BEQ     37$
 6892  032300  012701  000002                        MOV     #2,R1
 6893  032304  005003                                CLR     R3
 6894  032306  000137  030476            37$:        JMP     LOOP3
 6895
 6896                                    ;;**********************************************************************
 6897                                    .SBTTL  CLOCK HANDLER
 6898                                    ;*      THIS CODE CONTROLS THE CLOCK AND THE LOOPING ON THE
 6899                                    ;*      PREVIOUS TESTS WHILE THE CLOCK IS RUNNING.
 6900                                    ;*      A TICK COUNT IS KEPT IN THE LOCATION CALLED "TICKS".
 6901                                    ;*      THE LOCATION CALLED "TIME" CONTROLS THE NUMBER OF SECONDS
 6902                                    ;*      BEFORE END OF PASS IS REPORTED.
 6903                                    ;*
 6904                                    ;*      WHILE THIS TEST IS BEING PERFORMED, THE HIGH BYTE OF THE
 6905                                    ;*      DATA LIGHTS WILL INCREMENT APPROXIMATELY ONCE PER SECOND.
 6906                                    ;;**********************************************************************
 6907
 6908  032312  012737  000054  001102  STARTCL:MOV     #54,$TSTNM      ;RESET THE TEST NUMBER
 6909  032320  005737  001366                        TST     TICKS           ;STARTED CLOCK YET?
 6910  032324  001402                                BEQ     1$              ;BRANCH IF NO
 6911  032326  000137  023630                        JMP     LOOP0           ;CONTINUE TEST
 6912  032332  012737  032352  000004  1$:         MOV     #2$,ERRVEC      ;SET ERROR VEC
 6913  032340  005737  172540                        TST     PCLKST          ;P CLOCK AVAILABLE?
 6914  032344  005237  001366                        INC     TICKS
 6915  032350  000430                                BR      PCLK            ;BRANCH IF YES
 6916  032352  012737  032376  000004  2$:         MOV     #3$,ERRVEC
 6917  032360  012706  001100                        MOV     #STACK,SP
 6918  032364  005737  177546                        TST     LKSTAT          ;LINE CLOCK AVAILABLE?
 6919  032370  005237  001366                        INC     TICKS
 6920  032374  000402                                BR      LCLK            ;BRANCH IF YES
```

```
032376  000137  032712          3$:     JMP     $EOP            ;END TEST

                                ;;*****************************************************************
                                ;THIS ROUTINE STARS THE LINE CLOCK
032402  000230                  LCLK:   SPL     0
032404  012737  032476  000100          MOV     #LKSRV,LKVEC    ;SET INTERRUPT VECTOR
032412  012737  000340  000102          MOV     #PR7,LKVEC+2
032420  112737  000100  177546          MOVB    #BIT6,LKSTAT    ;START CLOCK
032426  000137  023630                  JMP     LOOPD           ;START TEST

                                ;;*****************************************************************
                                ;THIS ROUTINE STARTS THE PROGRAMMABLE CLOCK
032432  000230                  PCLK:   SPL     0
032434  012737  032610  000104          MOV     #PKSRV,PKVEC    ;SET INTERRUPT VECTOR
032442  012737  000340  000106          MOV     #PR7,PKVEC+2
032450  012737  000004  172542          MOV     #4,COSETB       ;SET COUNT BUFFER
032456  012737  000004  172544          MOV     #4,COUNTER      ;INIT THE COUNTER
032464  012737  000111  172540          MOV     #111,PCLKST     ;START THE CLOCK
032472  000137  023630                  JMP     LOOPD           ;STARS TEST

                                ;;*****************************************************************
                                ;THIS ROUTINE SERVICES THE LINE CLOCK INTERRUPT
032476  005237  001366          LKSRV:  INC     TICKS
032502  022737  000075  001366          CMP     #61.,TICKS      ;ONE SECOND YET?
032510  001032                          BNE     3$              ;BRANCH IF NO
032512  012737  000001  001366          MOV     #1,TICKS        ;INIT TICKS
032520  105237  032607                  INCB    OUT+1           ;INCREMENT DISPLAY
032524  123737  001370  032607          CMPB    TIMES,OUT+1     ;DONE?
032532  003010                          BGT     1$              ;BRANCH IF NO
032534  005037  177546                  CLR     LKSTAT          ;TURN CLOCK OFF
032540  005037  032606                  CLR     OUT
032544  005037  001366                  CLR     TICKS
032550  000137  032712                  JMP     $EOP            ;END TEST
032554  113737  001102  032606  1$:     MOVB    $TSTNM,OUT
032562  105737  001103                  TSTB    $ERFLG          ;ANY ERRORS?
032566  001003                          BNE     3$              ;BRANCH IF YES
032570  013777  032606  146340          MOV     OUT,@SWR        ;DISPLAY SECONDS
032576  012737  000100  177546  3$:     MOV     #BIT6,LKSTAT    ;START CLOCK
032604  000002                          RTI                     ;CONTINUE TEST
032606  000000                  OUT:    .WORD

                                ;;*****************************************************************
                                ;THIS ROUTINE SERVICES THE PROGRAMMABLE CLOCK INTERRUPT
032610  005237  001366          PKSRV:  INC     TICKS
032614  022737  060650  001366          CMP     #25000.,TICKS   ;ONE SECOND YET?
032622  001032                          BNE     2$              ;BRANCH IF NO
032624  012737  000001  001366          MOV     #1,TICKS        ;INIT TICKS
032632  105237  032607                  INCB    OUT+1           ;INCREMENT SECOND DISPLAY
032636  123737  001370  032607          CMPB    TIMES,OUT+1     ;DONE YET?
032644  003010                          BGT     1$              ;BRANCH IF NO
032646  005037  172540                  CLR     PCLKST          ;TURN OFF CLOCK
032652  005037  032606                  CLR     OUT
032656  005037  001366                  CLR     TICKS
032662  000137  032712                  JMP     $EOP            ;END TEST
032666  113737  001102  032606  1$:     MOVB    $TSTNM,OUT
032674  105737  001103                  TSTB    $ERFLG          ;ANY ERRORS?
```

```
                                        BNE     2$                      ;BRANCH IF YES
                                        MOV     OUT,@SWR                ;DISPLAY SECONDS
                                2$:     RTI                             ;CONTINUE TEST

                                .SBTTL  END OF PASS ROUTINE

                                ;;**********************************************************************
                                ;*INCREMENT THE PASS NUMBER ($PASS)
                                ;*TYPE "END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYY"
                                ;*WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS
                                ;*IF THERES A MONITOR GO TO IT
                                ;*IF THERE ISN'T JUMP TO LOOP

                                $EOP:
                                        SCOPE
                                        CLR     $TSTNM                  ;;ZERO THE TEST NUMBER
                                        CLR     $TIMES                  ;;ZERO THE NUMBER OF ITERATIONS
                                        INC     $PASS                   ;;INCREMENT THE PASS NUMBER
                                        BIC     #100000,$PASS           ;;DON'T ALLOW A NEG. NUMBER
                                        DEC     (PC)+                   ;;LOOP?
                                $EOPCT: .WORD   1
                                        BGT     $DOAGN                  ;;YES
                                        MOV     (PC)+,@(PC)+            ;;RESTORE COUNTER
                                $ENDCT: .WORD   1
                                        $EOPCT
                                        TYPE    ,65$                    ;;TYPE ASCIZ STRING
                                        BR      64$                     ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <12><15>/END PASS #/
                                64$:
                                        MOV     $PASS,-(SP)             ;;SAVE $PASS FOR TYPEOUT
                                                                        ;;TYPE PASS NUMBER
                                        TYPDS                           ;;GO TYPE--DECIMAL ASCII WITH SIGN
                                        TYPE    ,67$                    ;;TYPE ASCIZ STRING
                                        BR      66$                     ;;GET OVER THE ASCIZ
                                ;;67$:  .ASCIZ  /  TOTAL ERRORS SINCE LAST REPORT /
                                66$:
                                        MOV     $ERTTL,-(SP)            ;;SAVE $ERTTL FOR TYPEOUT
                                                                        ;;TOTAL NUMBER OF ERRORS
                                        TYPDS                           ;;GO TYPE--DECIMAL ASCII WITH SIGN
                                        TYPE    ,$CRLF                  ;;TYPE CARRIAGE RETURN, LINE FEED
                                        CLR     $ERTTL                  ;;CLEAR ERROR TOTAL
                                $GET42: MOV     @#42,R0                 ;;GET MONITOR ADDRESS
                                        BEQ     $DOAGN                  ;;BRANCH IF NO MONITOR
                                        RESET                           ;;CLEAR THE WORLD
                                $ENDAD: JSR     PC,(R0)                 ;;GO TO MONITOR
                                        NOP                             ;;SAVE ROOM
                                        NOP                             ;;FOR
                                        NOP                             ;;ACT11
                                $DOAGN:
                                        JMP     @#LOOP                  ;;RETURN
                                $ENULL: .BYTE   -1,-1,0                 ;;NULL CHARACTER STRING
                                        .EVEN

                                .SBTTL  SCOPE HANDLER ROUTINE

                                ;;**********************************************************************
```

```
                                        ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
                                        ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
                                        ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
                                        ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                        ;*SW14=1         LOOP ON TEST
                                        ;*SW11=1         INHIBIT ITERATIONS
                                        ;*SW09=1         LOOP ON ERROR
                                        ;*SW08=1         LOOP ON TEST IN SWR<7:0>
                                        ;*CALL
                                        ;*      SCOPE            ;;SCOPE=IOT

033122                                  $SCOPE:
033122  170127  000000                          LDFPS   #0
033126  032777  000400  146002                  BIT     #SW8,@SWR       ;SWITCH 8 ON?
033134  001013                                  BNE     1$              ;BRANCH IF YES
033136  010346                                  MOV     R3,-(SP)        ;SAVE R3
033140  117703  145772                          MOVB    @SWR,R3         ;GET  LOWER SWITCHES
033144  170003                                  LDUB                    ;PUT IN MICRO-BREAK REGISTER
033146  012603                                  MOV     (SP)+,R3        ;RESTORE  R3
033150  012737  037156  000004                  MOV     #CPSPUR,ERRVEC
033156  012737  037276  000244                  MOV     #FPSPUR,FPPVEC
033164  032777  040000  145744  1$:             BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
033172  001114                                  BNE     $OVER           ;;YES IF SW14=1
                                        ;*****START OF CODE FOR THE XOR TESTER*****
033174  000416                        $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                                                        ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
033176  013746  000004                          MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
033202  012737  033222  000004                  MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
033210  005737  177060                          TST     @#177060        ;;TIME OUT ON XOR?
033214  012637  000004                          MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
033220  000463                                  BR      $SVLAD          ;;GO TO THE NEXT TEST
033222  022626                          5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
033224  012637  000004                          MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
033230  000423                                  BR      7$              ;;LOOP ON THE PRESENT TEST
033232                          6$:;*****END OF CODE FOR THE XOR TESTER*****
033232  032777  000400  145676                  BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
033240  001404                                  BEQ     2$              ;;BR IF NO
033242  127737  145670  001102                  CMPB    @SWR,$TSTNM     ;;ON THE RIGHT TEST?    SWR<7:0>
033250  001465                                  BEQ     $OVER           ;;BR IF YES
033252  105737  001103                  2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
033256  001421                                  BEQ     3$              ;;BR IF NO
033260  123737  001115  001103                  CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
033266  101015                                  BHI     3$              ;;BR IF NO
033270  032777  001000  145640                  BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
033276  001404                                  BEQ     4$              ;;BR IF NO
033300  013737  001110  001106  7$:             MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
033306  000446                                  BR      $OVER
033310  105037  001103                  4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
033314  005037  001220                          CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
033320  000415                                  BR      1$              ;;ESCAPE TO THE NEXT TEST
033322  032777  004000  145606  3$:             BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
033330  001011                                  BNE     1$              ;;BR IF YES
033332  005737  001242                          TST     $PASS           ;;IF FIRST PASS OF PROGRAM
033336  001406                                  BEQ     1$              ;;      INHIBIT ITERATIONS
033340  005237  001104                          INC     $ICNT           ;;INCREMENT ITERATION COUNT
033344  023737  001220  001104                  CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
```

```
7089  033352  002024                          BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
7090  033354  012737  000001  001104  1$:     MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
7091  033362  013737  033440  001220          MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
7092  033370  105237  001102          $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
7093  033374  113737  001102  001240          MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
7094  033402  011637  001106                  MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
7095  033406  011637  001110                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
7096  033412  005037  001222                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
7097  033416  112737  000001  001115          MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
7098  033424  013777  001102  145506  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
7099  033432  013716  001106                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
7100  033436  000002                          RTI                     ;;FIXES PS
7101  033440  003720                  $MXCNT: 2000.                   ;;MAX. NUMBER OF ITERATIONS
7102
7103                                   .SBTTL  ERROR HANDLER ROUTINE
7104
7105                                   ;;**************************************************************
7106                                   ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
7107                                   ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
7108                                   ;*AND GO TO $ERRTYP ON ERROR
7109                                   ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
7110                                   ;*SW15=1        HALT ON ERROR
7111                                   ;*SW13=1        INHIBIT ERROR TYPEOUTS
7112                                   ;*SW10=1        BELL ON ERROR
7113                                   ;*SW09=1        LOOP ON ERROR
7114                                   ;*CALL
7115                                   ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
7116
7117  033442                   $ERROR:
7118  033442  032777  000400  145466          BIT     #SW8,@SWR       ;SWITCH 8 ON?
7119  033450  001007                          BNE     7$              ;BRANCH IF YES
7120  033452  010346                          MOV     R3,-(SP)        ;SAVE R3
7121  033454  005003                          CLR     R3
7122  033456  170103                          LDFPS   R3              ;CLEAR THE FPS REGISTER
7123  033460  117703  145452                  MOVB    @SWR,R3         ;GET LOWER  SWITCHES
7124  033464  170003                          LDUB                    ;PUT IN MICRO-BREAK REG
7125  033466  012603                          MOV     (SP)+,R3        ;RESTORE  R3
7126  033470  105237  001103          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
7127  033474  001775                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
7128  033476  013777  001102  145434          MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
7129  033504  032777  002000  145424          BIT     #BIT10,@SWR     ;;BELL ON ERROR?
7130  033512  001402                          BEQ     1$              ;;NO - SKIP
7131  033514  104400  001224                  TYPE    ,$BELL          ;;RING BELL
7132  033520  005237  001112          1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
7133  033524  011637  001116                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
7134  033530  162737  000002  001116          SUB     #2,$ERRPC
7135  033536  117737  145354  001114          MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
7136  033544  032777  020000  145364          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
7137  033552  001004                          BNE     20$             ;;SKIP TYPEOUTS
7138  033554  004737  035604                  JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
7139  033560  104400  001231                  TYPE    ,$CRLF
7140  033564                           20$:
7141  033564  122737  000001  001254          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
7142  033572  001007                          BNE     2$              ;;NO,SKIP APT ERROR REPORT
7143  033574  113737  001114  033606          MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
7144  033602  004737  035354                  JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
```

```
7145  033606  000                     21$:    .BYTE   0
7146  033607  000                             .BYTE   0
7147  033610  000777                  22$:    BR      22$             ;;APT ERROR LOOP
7148  033612  005777  145320          2$:     TST     @SWR            ;;HALT ON ERROR
7149  033616  100001                          BPL     3$              ;;SKIP IF CONTINUE
7150  033620  000000                          HALT                    ;;HALT ON ERROR!
7151  033622  032777  001000  145306  3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
7152  033630  001402                          BEQ     4$              ;;BR IF NO
7153  033632  013716  001110                  MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
7154  033636  005737  001222          4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
7155  033642  001402                          BEQ     5$              ;;BR IF NONE
7156  033644  013716  001222                  MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
7157  033650                          5$:
7158  033650  022737  033102  000042          CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
7159  033656  001001                          BNE     6$              ;;BRANCH IF NO
7160  033660  000000                          HALT                    ;;YES
7161  033662                          6$:
7162  033662  000002                          RTI                     ;;RETURN
7163
7164                                  ;;**************************************************************
7165                                  .SBTTL  CONVERT FLOATING BINARY TO OCTAL ASCIZ
7166                                  ;*
7167                                  ;*THIS ROUTINE CONVERTS A 32 BIT FLOATING NUMBER TO AN OCTAL
7168                                  ;*ASCIZ STRING IN THE FOLLOWING FORMAT:
7169                                  ;*
7170                                  ;*              W  XXX  YYY ZZZZZZ
7171                                  ;*
7172                                  ;*      WHERE   W = SIGN BIT
7173                                  ;*              X = 8-BIT EXPONENT (RIGHT JUSTIFIED)
7174                                  ;*              Y = FRACTION BITS <57:51> (RIGHT JUSTIFIED)
7175                                  ;*              Z = FRACTION BITS <50:35>
7176                                  ;*
7177                                  ;*IT IS ENTERED BY A TRAP CALL WITH THE ADDRESS OF THE FLOATING
7178                                  ;*NUMBER IN THE WORD FOLLOWING THE CALL.
7179                                  ;*IT RETURNS WITH THE ADDRESS OF THE ASCIZ STRING ON THE STACK.
7180                                  ;**************************************************************
7181  033664  104405                  $FL20:  SAVREG
7182  033666  017600  000000                  MOV     @(SP),R0        ;GET ADDRESS OF DATA
7183  033672  062716  000002                  ADD     #2,(SP)         ;ADJUST RETURN PC
7184  033676  016001  000002                  MOV     2(R0),R1        ;PUT SECOND DATA WORD IN R1
7185  033702  011000                          MOV     (R0),R0         ;PUT FIRST DATA WORD IN R0
7186  033704  012704  001333                  MOV     #$FLBUFF+23,R4  ;GET ADDRESS OF BUFFER END IN R4
7187  033710  112744  000000                  MOVB    #0,-(R4)        ;PUT TERMINATOR IN BUFFER
7188  033714  012705  000005                  MOV     #5,R5           ;SET SOB COUNT FOR FRACTION DIGITS
7189  033720  010103                  1$:     MOV     R1,R3           ;GET LSB'S OF FRACTION
7190  033722  042703  177770                  BIC     #^C7,R3         ;SAVE LS 3 BITS
7191  033726  062703  000060                  ADD     #60,R3          ;MAKE THEM ASCII
7192  033732  110344                          MOVB    R3,-(R4)        ;STORE IN BUFFER
7193  033734  073027  177775                  ASHC    #-3,R0          ;SHIFT NUMBER TO NEXT 3 BITS
7194  033740  077511                          SOB     R5,1$           ;CONTINUE FOR 7 DIGITS
7195  033742  010103                          MOV     R1,R3           ;GET NEXT DIGITS
7196  033744  042703  177776                  BIC     #^C1,R3         ;ONLY WANT 1 BIT
7197  033750  062703  000060                  ADD     #60,R3          ;MAKE THEM ASCII
7198  033754  110344                          MOVB    R3,-(R4)        ;STORE IN BUFFER
7199  033756  112744  000040                  MOVB    #40,-(R4)       ;PUT SPACE IN BUFFER
7200  033762  073027  177777                  ASHC    #-1,R0
```

```
 7201  033766  012705  000002                MOV     #2,R5           ;SET SOB COUNT
 7202  033772  010103              3$:       MOV     R1,R3           ;GET LOW WORD
 7203  033774  042703  177770                BIC     #^C7,R3         ;MASK 3 BITS
 7204  034000  062703  000060                ADD     #60,R3          ;MAKE THEM ASCII
 7205  034004  110344                        MOVB    R3,-(R4)        ;PUT IN BUFFER
 7206  034006  073027  177775                ASHC    #-3,R0          ;GET NEXT 3 BITS
 7207  034012  077511                        SOB     R5,3$           ;CONVERT THEM
 7208  034014  010103                        MOV     R1,R3           ;
 7209  034016  042703  177776                BIC     #^C1,R3         ;ONLY WANT 1 BIT
 7210  034022  062703  000060                ADD     #60,R3          ;MAKE IT ASCII
 7211  034026  110344                        MOVB    R3,-(R4)        ;PUT IN BUFFER
 7212  034030  112744  000040                MOVB    #40,-(R4)       ;PUT SPACE IN BUFFER
 7213  034034  112744  000040                MOVB    #40,-(R4)
 7214  034040  072127  177777                ASH     #-1,R1          ;GET FIRST 3 BITS OF EXPONENT
 7215  034044  012705  000002                MOV     #2,R5           ;SET SOB COUNT FOR 2 DIGITS
 7216  034050  010103              2$:       MOV     R1,R3           ;GET LSB'S OF EXPONENT
 7217  034052  042703  177770                BIC     #^C7,R3         ;SAVE 3 BITS
 7218  034056  062703  000060                ADD     #60,R3          ;MAKE THEM ASCII
 7219  034062  110344                        MOVB    R3,-(R4)        ;STORE IN BUFFER
 7220  034064  072127  177775                ASH     #-3,R1          ;GET NEXT 3 BITS
 7221  034070  077511                        SOB     R5,2$           ;CONTINUE
 7222  034072  010103                        MOV     R1,R3           ;GET LAST 2 BITS OF EXPONENT
 7223  034074  042703  177774                BIC     #^C3,R3         ;MAKE SURE ONLY 2 BITS
 7224  034100  062703  000060                ADD     #60,R3          ;MAKE THEM ASCII
 7225  034104  110344                        MOVB    R3,-(R4)        ;STORE IN BUFFER
 7226  034106  112744  000040                MOVB    #40,-(R4)       ;PUT SPACE IN BUFFER
 7227  034112  112744  000040                MOVB    #40,-(R4)
 7228  034116  042700  177776                BIC     #^C1,R0         ;GET SIGN BIT (IT WAS EXTENDED)
 7229  034122  062700  000060                ADD     #60,R0          ;MAKE IT ASCII
 7230  034126  110044                        MOVB    R0,-(R4)        ;PUT IT IN THE BUFFER
 7231  034130  104406                        RESREG
 7232  034132  011646                        MOV     (SP),-(SP)      ;SAVE RETURN PC
 7233  034134  016666  000004  000002        MOV     4(SP),2(SP)     ;AND RETURN PSW
 7234  034142  012766  001310  000004        MOV     #$FLBUFF,4(SP)  ;PUT BUFFER ADDRESS ON STACK
 7235  034150  000006                        RTT                     ;RETURN
 7236                          ;;*****************************************************************
 7237                          .SBTTL  CONVERT FLOATING DOUBLE BINARY TO OCTAL ASCIZ
 7238                          ;*
 7239                          ;*THIS ROUTINE CONVERTS A 64 BIT FLOATING NUMBER TO AN OCTAL
 7240                          ;*ASCIZ STRING IN THE FOLLOWING FORMAT:
 7241                          ;*
 7242                          ;*              U  VVV  WWW XXXXXX YYYYYY ZZZZZZ
 7243                          ;*
 7244                          ;*      WHERE   U = SIGN BIT
 7245                          ;*              V = 8-BIT EXPONENT (RIGHT JUSTIFIED)
 7246                          ;*              W = FRACTION BITS <57:51> (RIGHT JUSTIFIED)
 7247                          ;*              X = FRACTION BITS <50:35>
 7248                          ;*              Y = FRACTION BITS <34:19>
 7249                          ;*              Z = FRACTION BITS <18:03>
 7250                          ;*
 7251                          ;*IT IS ENTERED BY A TRAP CALL WITH THE ADDRESS OF THE FLOATING
 7252                          ;*NUMBER IN THE WORD FOLLOWING THE CALL.
 7253                          ;*IT RETURNS WITH THE ADDRESS OF THE ASCIZ STRING ON THE STACK.
 7254                          ;;*****************************************************************
 7255  034152  104405          $FLD20: SAVREG
 7256  034154  017637  000000  034170        MOV     @(SP),1$        ;GET ADDRESS OF DATA TO CONVERT
```

```
7257  034162  062716  000002                    ADD     #2,(SP)         ;ADJUST RETURN PC
7258  034166  104407                            FL2O                    ;CONVERT MS 32 BITS
7259  034170  000000                    1$:     .WORD
7260  034172  012600                            MOV     (SP)+,R0        ;GET ADDRESS OF CONVERTED DATA
7261  034174  010037  001352                    MOV     R0,$BUFF        ;SAVE IT
7262  034200  062700  000041                    ADD     #41,R0          ;ADJUST TO END OF BUFFER
7263  034204  105040                            CLRB    -(R0)           ;PUT TERMINATOR IN BUFFER
7264  034206  013701  034170                    MOV     1$,R1           ;GET ADDRESS OF DATA TO CONVERT
7265  034212  062701  000004                    ADD     #4,R1           ;ADJUST TO LOWER 32 BITS
7266  034216  012102                            MOV     (R1)+,R2        ;SAVE THE DATA
7267  034220  012103                            MOV     (R1)+,R3
7268  034222  012701  000002                    MOV     #2,R1           ;SET LOOP COUNT
7269  034226  012704  000005            3$:     MOV     #5,R4           ;SET LOOP COUNT
7270  034232  010305                    4$:     MOV     R3,R5           ;GET LS 32 BITS OF DATA
7271  034234  042705  177770                    BIC     #^C7,R5         ;MASK 3 BITS
7272  034240  062705  000060                    ADD     #60,R5          ;MAKE THEM ASCII
7273  034244  110540                            MOVB    R5,-(R0)        ;PUT IN BUFFER
7274  034246  073227  177775                    ASHC    #-3,R2          ;GET NEXT 3 BITS
7275  034252  077411                            SOB     R4,4$           ;CONTINUE
7276  034254  010305                            MOV     R3,R5           ;GET LS 32 BITS
7277  034256  042705  177776                    BIC     #^C1,R5         ;ONLY WANT 1 BIT
7278  034262  062705  000060                    ADD     #60,R5          ;MAKE IT ASCII
7279  034266  110540                            MOVB    R5,-(R0)        ;PUT IN TABLE
7280  034270  112740  000040                    MOVB    #40,-(R0)       ;PUT SPACE IN TABLE
7281  034274  073227  177777                    ASHC    #-1,R2
7282  034300  077126                            SOB     R1,3$           ;CONVERT NEXT 16 BITS
7283  034302  104406                            RESREG
7284  034304  011646                            MOV     (SP),-(SP)      ;ADJUST STACK
7285  034306  016666  000004  000002            MOV     4(SP),2(SP)     ;TO RETURN WITH ADDRESS
7286  034314  013766  001352  000004            MOV     $BUFF,4(SP)     ;OF BUFFER ON STACK
7287  034322  000006                            RTT                     ;RETURN
7288                                    ;;*****************************************************************
7289                                    .SBTTL  ROUTINE TO START THE LINE CLOCK
7290                                    ;*THIS ROUTINE SETS UP THE LINE CLOCK TO GUARANTEE THE
7291                                    ;*MAXIMUM DELAY BEFORE IT WILL INTERRUPT.
7292                                    ;;*****************************************************************
7293  034324  012737  034340  000004    GETTIK: MOV     #2$,ERRVEC
7294  034332  005737  177546                    TST     LKSTAT          ;LINE CLOCK THERE?
7295  034336  000441                            BR      3$              ;BRANCH IF YES
7296  034340  012706  001100            2$:     MOV     #STACK,SP
7297  034344  005737  001242                    TST     $PASS           ;FIRST PASS?
7298  034350  001032                            BNE     4$              ;BRANCH IF NO
7299  034352  005227  177777                    INC     #-1
7300  034356  001027                            BNE     4$
7301  034360  104400  034366                    TYPE    ,65$            ;;TYPE ASCIZ STRING
7302  034364  000424                            BR      64$             ;;GET OVER THE ASCIZ
7303                                    ;;65$:  .ASCIZ  "SHOULD HAVE LINE CLOCK INSTALLED!!!!!!!"
7304  034436                            64$:
7305  034436  000177  144560            4$:     JMP     @$ESCAPE
7306  034442  013746  000100            3$:     MOV     LKVEC,-(SP)     ;SAVE CONTENTS OF VECTOR
7307  034446  013746  177776                    MOV     PSW,-(SP)       ;SAVE PSW
7308  034452  005037  177776                    CLR     PSW
7309  034456  012737  034474  000100            MOV     #1$,LKVEC       ;SETUP THE VECTOR
7310  034464  012737  000100  177546            MOV     #BIT6,LKSTAT    ;START THE INTERRUPT
7311  034472  000001                            WAIT                    ;WAIT FOR IT TO HAPPEN
7312  034474  022626                    1$:     CMP     (SP)+,(SP)+     ;RESTORE THE STACK
```

```
7313  034476  012637  177776              MOV     (SP)+,PSW       ;RESTORE THE PSW
7314  034502  012637  000100              MOV     (SP)+,LKVEC     ;RESTORE THE VECTOR
7315  034506  012737  000100  177546      MOV     #BIT6,LKSTAT    ;START THE CLOCK AGAIN
7316  034514  000207                      RTS     PC              ;RETURN
7317
7318                                  ;;*****************************************************************
7319                                  .SBTTL  MUL SHIFT ENCODER ROM CONVERT ROUTINE
7320                                  ;*
7321                                  ;*      THIS ROUTINE TAKES THE CONTENTS OF $REG0 AND $REG1 AS A
7322                                  ;*      MULTIPLIER AND GENERATES THE ROM ADDRESSES AND OUTPUT
7323                                  ;*      DATA OF THE MUL SHIFT ENCODER ROM FOR THE ENTIRE MULTI-
7324                                  ;*      PLICATION.
7325                                  ;;*****************************************************************
7326  034516  012737  054532  001376  MULSHF: MOV     #OUTTBL,OUTPTR  ;INITIALIZE ROMOUT TABLE POINTER
7327  034524  012737  052532  001374          MOV     #ADRTBL,ADRPTR  ;INITIALIZE ADDRESS TABLE POINTER
7328  034532  012705  000030                  MOV     #30,R5          ;SET LOOP COUNT
7329  034536  105737  001276                  TSTB    $FD             ;MULD ERROR?
7330  034542  001406                          BEQ     5$              ;BRANCH IF NO
7331  034544  013700  001164                  MOV     $REG2,R0        ;GET THE
7332  034550  013701  001166                  MOV     $REG3,R1        ;MULTIPLIER
7333  034554  005004                          CLR     R4              ;INITIALIZE STRING
7334  034556  000411                          BR      4$
7335  034560  013700  001160          5$:     MOV     $REG0,R0        ;GET THE
7336  034564  013701  001162                  MOV     $REG1,R1        ;MULTIPLIER
7337  034570  005004                          CLR     R4              ;INITIALIZE STRING
7338  034572  042700  177600                  BIC     #177600,R0      ;GET RID OF EXPONENT
7339  034576  052700  000200                  BIS     #BIT7,R0        ;INSERT HIDDEN BIT
7340  034602  010102                  4$:     MOV     R1,R2           ;GET LOW ORDER WORD
7341  034604  042702  177600                  BIC     #177600,R2      ;GET CURRENT ROM ADDRESS
7342  034610  005704                          TST     R4              ;STRING SET?
7343  034612  001002                          BNE     1$              ;BRANCH IF YES
7344  034614  052702  000200                  BIS     #BIT7,R2        ;SET ADDRESS BIT7
7345  034620  110277  144550          1$:     MOVB    R2,@ADRPTR      ;PUT ADDRESS IN TABLE
7346  034624  005237  001374                  INC     ADRPTR          ;INCREMENT POINTER
7347  034630  062702  001400                  ADD     #MULTBL,R2      ;GENERATE POINTER TO SHIFT COUNT
7348  034634  111202                          MOVB    (R2),R2         ;GET SHIFT COUNT
7349  034636  032702  000010                  BIT     #BIT3,R2        ;STRING ON?
7350  034642  001402                          BEQ     2$              ;BRANCH IF NO
7351  034644  010204                          MOV     R2,R4           ;SET STRING
7352  034646  000401                          BR      3$
7353  034650  005004                  2$:     CLR     R4              ;CLEAR STRING
7354  034652  110277  144520          3$:     MOVB    R2,@OUTPTR      ;PUT ROMOUT DATA IN TABLE
7355  034656  142777  000360  144512          BICB    #360,@OUTPTR    ;GET RID OF UPPER BITS
7356  034664  005237  001376                  INC     OUTPTR          ;INCREMENT TABLE PTR
7357  034670  052702  000010                  BIS     #BIT3,R2        ;MAKE R2 THE SHIFT COUNT (2'S COMPLIMENT)
7358  034674  073002                          ASHC    R2,R0           ;SHIFT THE MULTIPLIER BY THE SHIFT COUNT
7359  034676  052702  177400                  BIS     #177400,R2      ;SIGN EXTEND R2
7360  034702  060205                          ADD     R2,R5           ;DECREMENT LOOP COUNT BY SHIFT COUNT
7361  034704  100401                          BMI     6$
7362  034706  001335                          BNE     4$              ;CONTINUE
7363  034710  105077  144462          6$:     CLRB    @OUTPTR         ;TERMINATE THE ROMOUT TABLE
7364  034714  000207                          RTS     PC
7365
7366                                  ;;*****************************************************************
7367                                  ;.SBTTL ROUTINE TO PUT 2 INTO FEC REGISTER
7368                                  ;*      THIS ROUTINE DOES A MAINTENANCE TRAP TO LOAD THE FEC
```

```
                                        ;*      REGISTER WITH 2.
                                        ;;*****************************************************************
034716  013700  000244          CHGFEC: MOV     FPPVEC,R0
034722  012737  034750  000244          MOV     #TEMP,FPPVEC
034730  012703  000010                  MOV     #10,R3          ;SET  MICRO-TRAP TO PUT
034734  170003                          LDUB                    ;KNOWN CONDITIONS IN FEC & FEA
034736  170127  000020                  LDFPS   #FMM
034742  176427  000000          $$FEA:  LDEXP   #0,AC0          ;TRAP ON THIS INSTRUCTION
034746  170000                          CFCC                    ;WAIT FOR TRAP
034750  022626                  TEMP:   CMP     (SP)+,(SP)+     ;RESTORE THE SP
034752  010037  000244                  MOV     R0,FPPVEC
034756  000207                          RTS     PC              ;RETURN

                                        .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES

                                        ;;*****************************************************************
                                        ;*SAVE R0-R5
                                        ;*CALL:
                                        ;*      SAVREG
                                        ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
                                        ;*
                                        ;*TOP---(+16)
                                        ;* +2---(+18)
                                        ;* +4---R5
                                        ;* +6---R4
                                        ;* +8---R3
                                        ;*+10---R2
                                        ;*+12---R1
                                        ;*+14---R0

034760                          $SAVREG:
034760  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
034762  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
034764  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
034766  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
034770  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
034772  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
034774  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
035000  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
035004  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
035010  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
035014  000002                          RTI

                                        ;*RESTORE R0-R5
                                        ;*CALL:
                                        ;*      RESREG
035016                          $RESREG:
035016  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
035022  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
035026  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
035032  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
035036  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
035040  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
035042  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
035044  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
035046  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
```

```
                        MOV     (SP)+,R0        ;;POP STACK INTO R0
                        RTI

                .SBTTL  TYPE ROUTINE

                ;;*****************************************************************
                ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
                ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
                ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
                ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
                ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
                ;*
                ;*CALL:
                ;*1) USING A TRAP INSTRUCTION
                ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
                ;*OR
                ;*      TYPE
                ;*      MESADR
                ;*

000001  $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
                BPL     1$              ;;BR IF YES
                HALT                    ;;HALT HERE IF NO TERMINAL
                BR      3$              ;;LEAVE
000002  1$:     MOV     R0,-(SP)        ;;SAVE R0
                MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
000001  001254  CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
                BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
000100  001255  BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
                BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
035124          MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
035344          JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
000000  61$:    .WORD   0               ;;MESSAGE ADDRESS
000040  001255  62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
                BNE     60$             ;;YES,SKIP TYPE OUT
112046  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
                BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
                TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
012600  60$:    MOV     (SP)+,R0        ;;RESTORE R0
000002  3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
000002          RTI                     ;;RETURN
000011  4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
                BEQ     8$
000200          CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
                BNE     5$
                TST     (SP)+           ;;POP  <CR><LF> EQUIV
                TYPE                    ;;TYPE A CR AND LF
                $CRLF
035332          CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
                BR      2$              ;;GET NEXT CHARACTER
035266  5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
001154  6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
                BNE     2$              ;;IF NO GO GET NEXT CHAR.
001152          MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                        ;;AND THE NULL CHAR.
000001  7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
```

```
7481  035226  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
7482  035230  004737  035266                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
7483  035234  105337  035332                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
7484  035240  000770                          BR      7$              ;;LOOP
7485
7486                                  ;HORIZONTAL TAB PROCESSOR
7487
7488  035242  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
7489  035246  004737  035266          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
7490  035252  132737  000007  035332          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
7491  035260  001372                          BNE     9$              ;;TAB STOP
7492  035262  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
7493  035264  000724                          BR      2$              ;;GET NEXT CHARACTER
7494  035266  105777  143654          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
7495  035272  100375                          BPL     $TYPEC
7496  035274  116677  000002  143646          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
7497  035302  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
7498  035310  001003                          BNE     1$              ;;BRANCH IF NO
7499  035312  105037  035332                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
7500  035316  000406                          BR      $TYPEX          ;;EXIT
7501  035320  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
7502  035326  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
7503  035330  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
7504  035332  000000          $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
7505  035334  000207          $TYPEX: RTS     PC
7506
7507
7508                                  .SBTTL  APT COMMUNICATIONS ROUTINE
7509
7510                                  ;;**************************************************************************
7511  035336  112737  000001  035602  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
7512  035344  112737  000001  035600  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
7513  035352  000403                          BR      $ATYC
7514  035354  112737  000001  035602  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
7515  035362                          $ATYC:
7516  035362  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
7517  035364  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
7518  035366  105737  035600                  TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
7519  035372  001450                          BEQ     5$              ;;IF NOT:  BR
7520  035374  122737  000001  001254          CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
7521  035402  001031                          BNE     3$              ;;IF NOT:  BR
7522  035404  132737  000100  001255          BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
7523  035412  001425                          BEQ     3$              ;;IF NOT:  BR
7524  035414  017600  000004                  MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
7525  035420  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
7526  035426  005737  001234          1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
7527  035432  001375                          BNE     1$              ;;IF NOT:  WAIT
7528  035434  010037  001250                  MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
7529  035440  105720                  2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
7530  035442  001376                          BNE     2$
7531  035444  163700  001250                  SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
7532  035450  006200                          ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
7533  035452  010037  001252                  MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
7534  035456  012737  000004  001234          MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
7535  035464  000413                          BR      5$
7536  035466  017637  000004  035512  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
```

```
7537  035474  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
7538  035502  013746  177776                  MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
7539  035506  004737  035054                  JSR     PC,$TYPE        ;;CALL TYPE MACRO
7540  035512  000000                  4$:     .WORD   0
7541  035514                          5$:
7542  035514  105737  035602          10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
7543  035520  001416                          BEQ     12$             ;;IF NOT:  BR
7544  035522  005737  001254                  TST     $ENV            ;;RUNNING UNDER APT?
7545  035526  001413                          BEQ     12$             ;;IF NOT:  BR
7546  035530  005737  001234          11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
7547  035534  001375                          BNE     11$             ;;IF NOT:  WAIT
7548  035536  017637  000004  001236          MOV     @4(SP),$FATAL   ;;GET ERROR #
7549  035544  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
7550  035552  005237  001234                  INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
7551  035556  105037  035602          12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
7552  035562  105037  035601                  CLRB    $LFLG           ;;CLEAR LOG FLAG
7553  035566  105037  035600                  CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
7554  035572  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
7555  035574  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
7556  035576  000207                          RTS     PC              ;;RETURN
7557  035600     000          $MFLG:  .BYTE   0               ;;MESSG. FLAG
7558  035601     000          $LFLG:  .BYTE   0               ;;LOG FLAG
7559  035602     000          $FFLG:  .BYTE   0               ;;FATAL FLAG
7560          035604                  .EVEN
7561          000200          APTSIZE=200
7562          000001          APTENV=001
7563          000100          APTSPOOL=100
7564          000040          APTCSUP=040
7565                          ;;**************************************************************************
7566
7567                          .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
7568
7569                          ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
7570                          ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
7571                          ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
7572
7573  035604  113737  001102  001266  $ERRTYP:MOVB    $TSTNM,$$TSTNM
7574  035612  104400  001231                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
7575  035616  010046                          MOV     R0,-(SP)        ;;SAVE R0
7576  035620  005000                          CLR     R0              ;;PICKUP THE ITEM INDEX
7577  035622  153700  001114                  BISB    @#$ITEMB,R0
7578  035626  001005                          BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
7579                                                                  ;;TYPE THE PC OF THE ERROR
7580  035630  013746  001116                  MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
7581                                                                  ;;ERROR ADDRESS
7582  035634  104401                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
7583  035636  000137  036346                  JMP     10$             ;;GET OUT
7584  035642  005300          1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
7585  035644  006300                  ASL     R0              ;;      WORK FOR THE ERROR TABLE
7586  035646  006300                  ASL     R0
7587  035650  006300                  ASL     R0
7588  035652  062700  002172                  ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
7589  035656  012037  035666                  MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
7590  035662  001404                          BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
7591  035664  104400                          TYPE                    ;;TYPE THE "ERROR MESSAGE"
7592  035666  000000          2$:     .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
```

```
7593  035670  104400  001231                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
7594  035674  012037  035752          3$:     MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
7595  035700  001427                          BEQ     5$              ;;SKIP TYPEOUT IF 0
7596  035702  010146                          MOV     R1,-(SP)        ;SAVE R1
7597  035704  016001  000002                  MOV     2(R0),R1        ;GET DATA FORMAT POINTER
7598  035710  001416                          BEQ     127$            ;BRANCH IF ZERO
7599  035712  105711                          TSTB    (R1)            ;IS FORMAT TYPE ZERO?
7600  035714  001414                          BEQ     127$            ;BRANCH IF YES
7601  035716  104400  037614                  TYPE    ,MSG1
7602  035722  013746  001116                  MOV     $ERRPC,-(SP)    ;PUT ERROR PC ON STACK
7603  035726  104401                          TYPOC                   ;TYPE IT
7604  035730  104400  036360                  TYPE    ,11$            ;TYPE 2 SPACES
7605  035734  013746  001266                  MOV     $$TSTNM,-(SP)   ;PUT TEST NUMBER ON STACK
7606  035740  104401                          TYPOC                   ;TYPE IT
7607  035742  104400  001231                  TYPE    ,$CRLF
7608  035746  012601                  127$:   MOV     (SP)+,R1        ;RESTORE R1
7609  035750  104400                          TYPE                    ;;TYPE THE "DATA HEADER"
7610  035752  000000                  4$:     .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
7611  035754  104400  001231                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
7612  035760  010146                  5$:     MOV     R1,-(SP)        ;;SAVE R1
7613  035762  012001                          MOV     (R0)+,R1        ;;PICKUP "DATA TABLE" POINTER
7614  035764  001563                          BEQ     9$              ;;BR IF NO DATA TO BE TYPED
7615  035766  012000                          MOV     (R0)+,R0        ;;PICKUP "DATA FORMAT" POINTER
7616  035770  005700                  6$:     TST     R0              ;IS FORMAT POINTER 0?
7617  035772  001410                          BEQ     12$             ;BRANCH IF YES
7618  035774  122710  000001                  CMPB    #1,(R0)         ;IS IT 0, 1, OR >1?
7619  036000  003004                          BGT     20$             ;BRANCH IF ZERO
7620  036002  001407                          BEQ     30$             ;BRANCH IF 1
7621  036004  122710  000003                  CMPB    #3,(R0)         ;IS IT 2, 3, OR >3?
7622  036010  003016                          BGT     40$             ;BRANCH IF 2
7623
7624                                  ;DATA FORMAT 0-6 DIGIT OCTAL
7625  036012  005200                  20$:    INC     R0              ;ADJUST FORMAT POINTER
7626  036014                          12$:
7627  036014  013146                          MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
7628  036016  104401                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
7629  036020  000537                          BR      8$
7630
7631                                  ;DATA FORMAT 1-TYPE A FLOATING NUMBER
7632  036022  005200                  30$:    INC     R0              ;ADJUST FORMAT POINTER
7633  036024  012137  036032                  MOV     (R1)+,31$       ;GET ADDRESS OF NUMBER
7634  036030  104407                          FL2O                    ;CONVERT IT TO OCTAL
7635  036032  000000                  31$:    .WORD
7636  036034  012637  036042                  MOV     (SP)+,32$       ;GET ADDRESS OF ASCIZ
7637  036040  104400                          TYPE                    ;TYPE THE NUMBER
7638  036042  000000                  32$:    .WORD
7639  036044  000525                          BR      8$
7640
7641                                  ;DATA FORMAT 2-TYPE A FLOATING DOUBLE NUMBER
7642  036046  005200                  40$:    INC     R0              ;ADJUST FORMAT POINTER
7643  036050  012137  036056                  MOV     (R1)+,41$       ;GET ADDRESS OF NUMBER
7644  036054  104410                          FLD2O                   ;CONVERT IT TO ASCII
7645  036056  000000                  41$:    .WORD
7646  036060  012637  036066                  MOV     (SP)+,42$       ;GET ADDRESS OF ASCII
7647  036064  104400                          TYPE                    ;TYPE THE NUMBER
7648  036066  000000                  42$:    .WORD
```

```
                                        BR      8$

                                ;ERROR 25 HANDLER
                        60$:    TYPE    ,$CRLF          ;TYPE CARRIAGE RETURN LINE FEED
                                TYPE    ,EM25A          ;TYPE "MULTIPLIER"
                                TYPE    ,11$            ;TYPE TWO SPACES
                                TSTB    $FD
                                BEQ     66$
                                FLD20   ,$REGO
                                BR      67$
                        66$:    FL20    ,$REGO          ;CONVERT MULTIPLIER TO ASCIZ
                        67$:    MOV     (SP)+,61$
                                TYPE                    ;TYPE THE MULTIPLIER
                        61$:    .WORD
                                TYPE    ,$CRLF
                                TYPE    ,EM25B
                                SAVREG
                                JSR     PC,MULSHF       ;GO CONVERT MULTIPLIER
                                RESREG
                                MOV     #ADRTBL,R0      ;GET ADDRESS OF ADDRESS TABLE
                                MOV     #OUTTBL+1,R1    ;GET ADDRESS OF ROMOUT TABLE
                                CLR     $TMP0           ;INITIALIZE
                                CLR     $TMP1           ;DATA BUFFER
                        62$:    TYPE    ,11$            ;TYPE TWO SPACES
                                MOVB    (R0)+,$TMP0     ;GET ADDRESS
                                MOV     #$TMP0,-(SP)
                                JSR     PC,$DB20        ;CONVERT TO ASCIZ
                                ADD     #10,(SP)        ;ONLY WANT 3 DIGITS
                                MOV     (SP)+,63$       ;GET ADDRESS OF ASCIZ
                                TYPE                    ;TYPE THE ROM ADDRESS
                        63$:    .WORD                   ;
                                TSTB    (R1)+           ;DONE?
                                BNE     62$             ;BRANCH IF NO
                                TYPE    ,$CRLF
                                TYPE    ,EM25C
                                MOV     #OUTTBL,R0      ;GET ADDRESS OF ROMOUT DATA
                        64$:    TYPE    ,11$            ;TYPE TWO SPACES
                                MOVB    (R0)+,$TMP0     ;GET DATA
                                MOV     #$TMP0,-(SP)
                                JSR     PC,$DB20        ;CONVERT TO ASCIZ
                                ADD     #10,(SP)        ;ONLY WANT 3 DIGITS
                                MOV     (SP)+,65$
                                TYPE                    ;TYPE THE ROM OUT DATA
                        65$:    .WORD
                                TSTB    (R0)            ;DONE?
                                BNE     64$             ;BRANCH IF NO
                                TYPE    ,$CRLF
                                BR      100$            ;EXIT
                        8$:     TST     (R1)            ;;IS THERE ANOTHER NUMBER?
                                BEQ     9$              ;;BR IF NO
                                TYPE    ,11$            ;;TYPE TWO(2) SPACES
                                JMP     6$              ;;LOOP

                        9$:     CMPB    #25,$ITEMB      ;ERROR 25?
                                BEQ     60$             ;BRANCH IF YES
                        100$:   MOV     (SP)+,R1        ;;RESTORE R1
```

```
 7705  036346  012600                  10$:    MOV     (SP)+,R0        ;;RESTORE R0
 7706  036350  104400  001231                  TYPE    $CRLF           ;;"CARRIAGE RETURN" & "LINE FEED"
 7707  036354  000207                          RTS     PC              ;;RETURN
 7708  036356  000040                  13$:    .ASCIZ  / /
 7709  036360  020040      000         11$:    .ASCIZ  /  /            ;;TWO(2) SPACES
 7710          036364                          .EVEN
 7711
 7712                                  .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
 7713
 7714                                  ;;**************************************************************
 7715                                  ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
 7716                                  ;*OCTAL (ASCII) NUMBER AND TYPE IT.
 7717                                  ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
 7718                                  ;*CALL:
 7719                                  ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
 7720                                  ;*      TYPOS                   ;;CALL FOR TYPEOUT
 7721                                  ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
 7722                                  ;*      .BYTE   M               ;;M=1 OR 0
 7723                                  ;*                                      ;;1=TYPE LEADING ZEROS
 7724                                  ;*                                      ;;0=SUPPRESS LEADING ZEROS
 7725                                  ;*
 7726                                  ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
 7727                                  ;*$TYPOS OR $TYPOC
 7728                                  ;*CALL:
 7729                                  ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
 7730                                  ;*      TYPON                   ;;CALL FOR TYPEOUT
 7731                                  ;*
 7732                                  ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
 7733                                  ;*CALL:
 7734                                  ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
 7735                                  ;*      TYPOC                   ;;CALL FOR TYPEOUT
 7736
 7737  036364  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
 7738  036370  116637  000001  036607          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
 7739  036376  112637  036611                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
 7740  036402  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
 7741  036406  000406                          BR      $TYPON
 7742  036410  112737  000001  036607  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
 7743  036416  112737  000006  036611          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
 7744  036424  112737  000005  036606  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
 7745  036432  010346                          MOV     R3,-(SP)        ;;SAVE R3
 7746  036434  010446                          MOV     R4,-(SP)        ;;SAVE R4
 7747  036436  010546                          MOV     R5,-(SP)        ;;SAVE R5
 7748  036440  113704  036611                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
 7749  036444  005404                          NEG     R4
 7750  036446  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
 7751  036452  110437  036610                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
 7752  036456  113704  036607                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
 7753  036462  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
 7754  036466  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
 7755  036470  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
 7756  036472  000404                          BR      3$              ;;GO DO MSB
 7757  036474  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
 7758  036476  006105                          ROL     R5
 7759  036500  006105                          ROL     R5
 7760  036502  010503                          MOV     R5,R3
```

```
7761  036504  006103                    3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
7762  036506  105337  036610                    DECB    $OMODE          ;;TYPE THIS DIGIT?
7763  036512  100016                            BPL     7$              ;;BR IF NO
7764  036514  042703  177770                    BIC     #177770,R3      ;;GET RID OF JUNK
7765  036520  001002                            BNE     4$              ;;TEST FOR 0
7766  036522  005704                            TST     R4              ;;SUPPRESS THIS 0?
7767  036524  001403                            BEQ     5$              ;;BR IF YES
7768  036526  005204                    4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
7769  036530  052703  000060                    BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
7770  036534  052703  000040            5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
7771  036540  110337  036604                    MOVB    R3,8$           ;;SAVE FOR TYPING
7772  036544  104400  036604                    TYPE    ,8$             ;;GO TYPE THIS DIGIT
7773  036550  105337  036606            7$:     DECB    $OCNT           ;;COUNT BY 1
7774  036554  003347                            BGT     2$              ;;BR IF MORE TO DO
7775  036556  002402                            BLT     6$              ;;BR IF DONE
7776  036560  005204                            INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
7777  036562  000744                            BR      2$              ;;GO DO THE LAST DIGIT
7778  036564  012605                    6$:     MOV     (SP)+,R5        ;;RESTORE R5
7779  036566  012604                            MOV     (SP)+,R4        ;;RESTORE R4
7780  036570  012603                            MOV     (SP)+,R3        ;;RESTORE R3
7781  036572  016666  000002  000004           MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
7782  036600  012616                            MOV     (SP)+,(SP)
7783  036602  000002                            RTI                     ;;RETURN
7784  036604     000            8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
7785  036605     000                    .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
7786  036606     000            $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
7787  036607     000            $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
7788  036610  000000            $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
7789
7790                                    .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
7791
7792                                    ;;**************************************************************
7793                                    ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
7794                                    ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
7795                                    ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
7796                                    ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
7797                                    ;*REPLACED WITH SPACES.
7798                                    ;*CALL:
7799                                    ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
7800                                    ;*      TYPDS                   ;;GO TO THE ROUTINE
7801
7802  036612                            $TYPDS:
7803  036612  010046                            MOV     R0,-(SP)        ;;PUSH R0 ON STACK
7804  036614  010146                            MOV     R1,-(SP)        ;;PUSH R1 ON STACK
7805  036616  010246                            MOV     R2,-(SP)        ;;PUSH R2 ON STACK
7806  036620  010346                            MOV     R3,-(SP)        ;;PUSH R3 ON STACK
7807  036622  010546                            MOV     R5,-(SP)        ;;PUSH R5 ON STACK
7808  036624  012746  020200                    MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
7809  036630  016605  000020                    MOV     20(SP),R5       ;;GET THE INPUT NUMBER
7810  036634  100004                            BPL     1$              ;;BR IF INPUT IS POS.
7811  036636  005405                            NEG     R5              ;;MAKE THE BINARY NUMBER POS.
7812  036640  112766  000055  000001           MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
7813  036646  005000                    1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
7814  036650  012703  037026                    MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
7815  036654  112723  000040                    MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
7816  036660  005002                    2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
```

```
7817  036662  016001  037016                MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
7818  036666  160105                3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
7819  036670  002402                        BLT     4$              ;;BR IF DONE
7820  036672  005202                        INC     R2              ;;INCREASE THE BCD DIGIT BY 1
7821  036674  000774                        BR      3$
7822  036676  060105                4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
7823  036700  005702                        TST     R2              ;;CHECK IF BCD DIGIT=0
7824  036702  001002                        BNE     5$              ;;FALL THROUGH IF 0
7825  036704  105716                        TSTB    (SP)            ;;STILL DOING LEADING 0'S?
7826  036706  100407                        BMI     7$              ;;BR IF YES
7827  036710  106316                5$:     ASLB    (SP)            ;;MSD?
7828  036712  103003                        BCC     6$              ;;BR IF NO
7829  036714  116663  000001  177777        MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
7830  036722  052702  000060        6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
7831  036726  052702  000040        7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
7832  036732  110223                        MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
7833  036734  005720                        TST     (R0)+           ;;JUST INCREMENTING
7834  036736  020027  000010                CMP     R0,#10          ;;CHECK THE TABLE INDEX
7835  036742  002746                        BLT     2$              ;;GO DO THE NEXT DIGIT
7836  036744  003002                        BGT     8$              ;;GO TO EXIT
7837  036746  010502                        MOV     R5,R2           ;;GET THE LSD
7838  036750  000764                        BR      6$              ;;GO CHANGE TO ASCII
7839  036752  105726                8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
7840  036754  100003                        BPL     9$              ;;BR IF NO
7841  036756  116663  177777  177776        MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
7842  036764  105013                9$:     CLRB    (R3)            ;;SET THE TERMINATOR
7843  036766  012605                        MOV     (SP)+,R5        ;;POP STACK INTO R5
7844  036770  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
7845  036772  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
7846  036774  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
7847  036776  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
7848  037000  104400  037026                TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
7849  037004  016666  000002  000004        MOV     2(SP),4(SP)     ;;ADJUST THE STACK
7850  037012  012616                        MOV     (SP)+,(SP)
7851  037014  000002                        RTI                     ;;RETURN TO USER
7852  037016  023420                $DTBL:  10000.
7853  037020  001750                        1000.
7854  037022  000144                        100.
7855  037024  000012                        10.
7856  037026  000004                $DBLK:  .BLKW   4
7857
7858                                        .SBTTL  DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
7859
7860                                        ;;**************************************************************
7861                                        ;*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
7862                                        ;*UNSIGNED OCTAL ASCIZ NUMBER.
7863                                        ;*CALL
7864                                        ;*      MOV     #PNTR,-(SP)     ;;POINTER TO LOW WORD OF BINARY NUMBER
7865                                        ;*      JSR     PC,@#$DB20      ;;CALL THE ROUTINE
7866                                        ;*      RETURN                  ;;THE ADDRESS OF THE FIRST ASCIZ CHAR. IS ON THE STACK
7867
7868
7869  037036  104405                $DB20:  SAVREG                  ;;SAVE ALL REGISTERS
7870  037040  016601  000002                MOV     2(SP),R1        ;;PICKUP THE POINTER TO LOW WORD
7871  037044  012705  037155                MOV     #$OCTVL+13.,R5  ;;POINTER TO DATA TABLE
7872  037050  012704  000014                MOV     #12.,R4         ;;DO ELEVEN CHARACTERS
```

```
7873  037054  012703  177770                MOV     #^C7,R3         ;;MASK
7874  037060  012100                        MOV     (R1)+,R0        ;;LOWER WORD
7875  037062  012101                        MOV     (R1)+,R1        ;;HIGH WORD
7876  037064  005002                        CLR     R2              ;;TERMINATOR
7877  037066  110245              1$:       MOVB    R2,-(R5)        ;;PUT CHARACTER IN DATA TABLE
7878  037070  010002                        MOV     R0,R2           ;;GET THIS DIGIT
7879  037072  005304                        DEC     R4              ;;COUNT THIS CHARACTER
7880  037074  003007                        BGT     3$              ;;BR IF NOT THE LAST DIGIT
7881  037076  001405                        BEQ     2$              ;;BR IF IT IS THE LAST DIGIT
7882  037100  005205                        INC     R5              ;;ALL DIGITS DONE-ADJUST POINTER FOR FIRST
7883  037102  010566  000002                MOV     R5,2(SP)        ;;ASCIZ CHAR. & PUT IT ON THE STACK
7884  037106  104406                        RESREG                  ;;RESTORE ALL REGISTERS
7885  037110  000207                        RTS     PC              ;;RETURN TO USER
7886  037112  006203              2$:       ASR     R3              ;;POSITION THE MASK FOR THE LAST DIGIT
7887  037114  006001              3$:       ROR     R1              ;;POSITION THE BINARY NUMBER FOR
7888  037116  006000                        ROR     R0              ;;      THE NEXT OCTAL DIGIT
7889  037120  006001                        ROR     R1
7890  037122  006000                        ROR     R0
7891  037124  006001                        ROR     R1
7892  037126  006000                        ROR     R0
7893  037130  040302                        BIC     R3,R2           ;;MASK OUT ALL JUNK
7894  037132  062702  000060                ADD     #'0,R2          ;;MAKE THIS CHAR. ASCII
7895  037136  000753                        BR      1$              ;;GO PUT IT IN THE DATA TABLE
7896  037140  000016              $OCTVL: .BLKB   14.              ;;RESERVE DATA TABLE
7897                                ;;**************************************************************
7898                                .SBTTL  UNEXPECTED TRAP TO 4 ROUTINE
7899
7900  037156  011637  001200      CPSPUR: MOV     (SP),$TMP0      ;SAVE PC OF TRAP
7901  037162  012706  001100              MOV     #STACK,SP       ;RESTORE THE SP
7902  037166  005737  001372              TST     $CPUER          ;11/70?
7903  037172  001410                      BEQ     1$              ;BRANCH IF NO
7904  037174  013737  177766  001202      MOV     CPUERR,$TMP1    ;SAVE ERROR REG
7905  037202  005037  177766              CLR     CPUERR          ;CLEAR THE ERROR
7906  037206  104154                      ERROR   154             ;UNEXPECTED TRAP TO 4
7907  037210  000137  004456              JMP     LOOP            ;RESTART
7908  037214  104155              1$:     ERROR   155             ;UNEXPECTED TRAP TO 4
7909  037216  000137  004456              JMP     LOOP            ;RESTART
7910
7911                                ;;**************************************************************
7912                                .SBTTL  UNEXPECTED TRAP TO 114 ROUTINE
7913
7914  037222  011637  001200      CACHSPU:MOV     (SP),$TMP0      ;SAVE PC OF TRAP
7915  037226  012706  001100              MOV     #STACK,SP       ;INIT THE SP
7916  037232  005737  001372              TST     $CPUERR         ;11/70?
7917  037236  001414                      BEQ     1$              ;BRANCH IF NO
7918  037240  013737  177744  001202      MOV     MEMERR,$TMP1    ;SAVE ERROR REGISTER
7919  037246  013737  177740  001204      MOV     LOADRS,$TMP2    ;SAVE ERROR
7920  037254  013737  177742  001206      MOV     HIADRS,$TMP3    ;ADDRESS REGISTER
7921  037262  104156                      ERROR   156             ;UNEXPECTED TRAP TO 114
7922  037264  000137  004456              JMP     LOOP            ;RESTART
7923  037270  104157              1$:     ERROR   157             ;UNEXPECTED TRAP TO 114
7924  037272  000137  004456              JMP     LOOP            ;RESTART
7925
7926                                ;;**************************************************************
7927                                .SBTTL  UNEXPECTED TRAP TO 244 ROUTINE
7928
```

```
7929  037276  011637  001200          FPSPUR: MOV     (SP),$TMP0      ;SAVE ADDRESS OF TRAP
7930  037302  012706  001100                  MOV     #STACK,SP       ;RESTORE THE SP
7931  037306  170337  001202                  STST    $TMP1           ;GET FEC AND FEA
7932  037312  104160                          ERROR   160             ;UNEXPECTED TRAP TO 244
7933  037314  000137  004456                  JMP     LOOP            ;RESTART
7934
7935                                  .SBTTL  TRAP DECODER
7936
7937                                  ;;**********************************************************************
7938                                  ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
7939                                  ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
7940                                  ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
7941                                  ;*GO TO THAT ROUTINE.
7942
7943  037320  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
7944  037322  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
7945  037326  005740                          TST     -(R0)           ;;BACKUP BY 2
7946  037330  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
7947  037332  006300                          ASL     R0              ;;POSITION FOR INDEXING
7948  037334  016000  037342                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
7949  037340  000200                          RTS     R0              ;;GO TO ROUTINE
7950
7951
7952                                  .SBTTL  TRAP TABLE
7953
7954                                  ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
7955                                  ;*BY THE "TRAP" INSTRUCTION.
7956
7957                                  ;       ROUTINE
7958                                  ;       -------
7959  037342                          $TRPAD:
7960  037342  035054                          $TYPE   ;;CALL=TYPE     TRAP+0(104400)  TTY TYPEOUT ROUTINE
7961  037344  036410                          $TYPOC  ;;CALL=TYPOC    TRAP+1(104401)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
7962  037346  036364                          $TYPOS  ;;CALL=TYPOS    TRAP+2(104402)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
7963  037350  036424                          $TYPON  ;;CALL=TYPON    TRAP+3(104403)  TYPE OCTAL NUMBER (AS PER LAST CALL)
7964  037352  036612                          $TYPDS  ;;CALL=TYPDS    TRAP+4(104404)  TYPE DECIMAL NUMBER (WITH SIGN)
7965  037354  034760                          $SAVREG ;;CALL=SAVREG   TRAP+5(104405)  SAVE R0-R5 ROUTINE
7966  037356  035016                          $RESREG ;;CALL=RESREG   TRAP+6(104406)  RESTORE R0-R5 ROUTINE
7967  037360  033664                          $FL20   ;;CALL=FL20     TRAP+7(104407)
7968  037362  034152                          $FLD20  ;;CALL=FLD20    TRAP+10(104410)
7969
7970                                  .SBTTL  POWER DOWN AND UP ROUTINES
7971
7972                                  ;;**********************************************************************
7973                                  ;POWER DOWN ROUTINE
7974  037364  012737  037576  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
7975  037372  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
7976  037400  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
7977  037402  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
7978  037404  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
7979  037406  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
7980  037410  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
7981  037412  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
7982  037414  170200                          STFPS   R0              ;GET FPS
7983  037416  170127  000200                  LDFPS   #FD
7984  037422  010046                          MOV     R0,-(SP)        ;PUSH ON STACK
```

```
 7985  037424  174046                          STD     AC0,-(SP)
 7986  037426  174146                          STD     AC1,-(SP)
 7987  037430  174246                          STD     AC2,-(SP)
 7988  037432  174346                          STD     AC3,-(SP)
 7989  037434  172404                          LDD     AC4,AC0
 7990  037436  174046                          STD     AC0,-(SP)
 7991  037440  172405                          LDD     AC5,AC0
 7992  037442  174046                          STD     AC0,-(SP)
 7993  037444  010637  037602                  MOV     SP,$SAVR6       ;;SAVE SP
 7994  037450  012737  037462  000024          MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
 7995  037456  000000                          HALT
 7996  037460  000776                          BR      .-2             ;;HANG UP
 7997
 7998                                  ;;*****************************************************************
 7999                                  ;POWER UP ROUTINE
 8000  037462  012737  037576  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
 8001  037470  013706  037602                  MOV     $SAVR6,SP       ;;GET SP
 8002  037474  005037  037602                  CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
 8003  037500  005237  037602          1$:     INC     $SAVR6          ;;WAIT FOR THE INC
 8004  037504  001375                          BNE     1$              ;;OF  WORD
 8005  037506  170127  000200                  LDFPS   #FD
 8006  037512  172426                          LDD     (SP)+,AC0
 8007  037514  174005                          STD     AC0,AC5
 8008  037516  172426                          LDD     (SP)+,AC0
 8009  037520  174004                          STD     AC0,AC4
 8010  037522  172726                          LDD     (SP)+,AC3
 8011  037524  172626                          LDD     (SP)+,AC2
 8012  037526  172526                          LDD     (SP)+,AC1
 8013  037530  172426                          LDD     (SP)+,AC0
 8014  037532  170126                          LDFPS   (SP)+
 8015  037534  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
 8016  037536  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
 8017  037540  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
 8018  037542  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
 8019  037544  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
 8020  037546  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
 8021  037550  012737  037364  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
 8022  037556  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
 8023  037564  104400                          TYPE                    ;REPORT THE POWER FAILURE
 8024  037566  037604                  $PWRMG: .WORD   $POWER          ;;POWER FAIL MESSAGE POINTER
 8025  037570  013716                          MOV     @(PC)+,(SP)     ;;RESTART AT $ESCAPE
 8026  037572  001222                  $PWRAD: .WORD   $ESCAPE         ;;RESTART ADDRESS
 8027  037574  000002                          RTI
 8028  037576  000000                  $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
 8029  037600  000776                          BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
 8030  037602  000000                  $SAVR6: 0                       ;;PUT THE SP HERE
 8031  037604  005015  047520  042527  $POWER: .ASCIZ  <15><12>"POWER"
 8032  037612  000122
 8033                                          .EVEN
 8034
 8035  037614  051105  050122  004503  MSG1:   .ASCIZ  /ERRPC  TEST NO/<CRLF>
 8036  037622  042524  052123  047040
 8037  037630  100117     000
 8038  037633     104  052123  042040  EM1:    .ASCIZ  /DST DID NOT CLEAR ON MULF/
 8039  037640  042111  047040  052117
 8040  037646  041440  042514  051101
```

```
8041  037654  047440  020116  052515
8042  037662  043114     000
8043  037665     105  051122  041520  DH1:    .ASCII  /ERRPC                   DATA                    TEST NO/<CRLF>
8044  037672  004411  020040  020040
8045  037700  020040  020040  040504
8046  037706  040524  004411  052011
8047  037714  051505  020124  047516
8048  037722     200
8049  037723     011  020040  020040          .ASCIZ  /          EXPECT                 ACTUAL/
8050  037730  054105  042520  052103
8051  037736  004411  020040  040440
8052  037744  052103  040525  000114
8053                                          .EVEN
8054  037752  001116  001200  001204  DT1:    .WORD   $ERRPC,$TMP0,$TMP2,$$TSTNM,0
8055  037760  001266  000000
8056  037764     000     001     001  DF1:    .BYTE   0,1,1,0,0
8057  037767     000     000
8058  037771     106  050130  020103  EM2:    .ASCIZ  /FXPC FIRB08(1) NOT GETTING TO FRMA AS A LOW OR NOT GETTING TO RAD02 AS
8059  037776  044506  041122  034060
8060  040004  030450  020051  047516
8061  040012  020124  042507  052124
8062  040020  047111  020107  047524
8063  040026  043040  046522  020101
8064  040034  051501  040440  046040
8065  040042  053517  047440  020122
8066  040050  047516  020124  042507
8067  040056  052124  047111  020107
8068  040064  047524  051040  042101
8069  040072  031060  040440  020123
8070  040100  020101  000110
8071  040104  047503  042116  052111  EM3:    .ASCIZ  /CONDITION CODES BAD/
8072  040112  047511  020116  047503
8073  040120  042504  020123  040502
8074  040126  000104
8075  040130  051105  050122  004503  DH3:    .ASCII  /ERRPC        FPS       TEST NO/<CRLF>
8076  040136  020040  020040  020040
8077  040144  050106  004523  042524
8078  040152  052123  047040  100117
8079  040160  042411  050130  041505          .ASCIZ  /       EXPECT  ACTUAL/
8080  040166  004524  041501  052524
8081  040174  046101     000
8082          040200                          .EVEN
8083  040200  001116  001200  001202  DT3:    .WORD   $ERRPC,$TMP0,$TMP1,$$TSTNM,0
8084  040206  001266  000000
8085  040212  042101  020130  047522  EM4:    .ASCIZ  /ADX ROM FAILED/
8086  040220  020115  040506  046111
8087  040226  042105     000
8088  040231     105  051122  041520  DH4:    .ASCIZ  /ERRPC  TEST NO/
8089  040236  052011  051505  020124
8090  040244  047516     000
8091          040250                          .EVEN
8092  040250  001116  001266  000000  DT4:    .WORD   $ERRPC,$$TSTNM,0
8093  040256  051106  045510  046440  EM5:    .ASCIZ  /FRHK MUL SWR STUCK LOW/
8094  040264  046125  051440  051127
8095  040272  051440  052524  045503
8096  040300  046040  053517     000
```

```
8097  040305  000106  050130  020120  EM6:    .ASCIZ  /FXPP SCOO NOT GETTING TO FRMA AS A HIGH/
8098  040312  041523  030060  047040
8099  040320  052117  043440  052105
8100  040326  044524  043516  052040
8101  040334  020117  051106  040515
8102  040342  040440  020123  020101
8103  040350  044510  044107  000000
8104  040355  000106  046522  020105  EM7:    .ASCIZ  /FRME SHFC 0(0)H NOT GOING HIGH WHEN 1(0)H IS LOW/
8105  040362  044123  041506  030040
8106  040370  030050  044051  047040
8107  040376  052117  043440  044517
8108  040404  043516  044040  043511
8109  040412  020110  044127  047105
8110  040420  030440  030050  044051
8111  040426  044440  020123  047514
8112  040434  000127
8113                          .EVEN
8114  040436  001116  001200  001210  DT7:    .WORD   $ERRPC,$TMP0,$TMP4,$$TSTNM,0
8115  040444  001266  000000
8116  040450  051121  042040  052101  EM10:   .ASCIZ  /QR DATA BAD, CHECK MUL SHIFT ENCODER ROM OUT IN STATE 233/
8117  040456  020101  040502  026104
8118  040464  041440  042510  045503
8119  040472  046440  046125  051440
8120  040500  044510  052106  042440
8121  040506  041516  042117  051105
8122  040514  051040  046517  047440
8123  040522  052125  044440  020116
8124  040530  052123  052101  020105
8125  040536  031462  000063
8126  040544  051105  050122  004503  DH10:   .ASCII  /ERRPC                   DATA                                    ROMOUT  TEST NO/
8127  040550  004411  040504  040524
8128  040556  004411  004411  047522
8129  040564  047515  052125  052011
8130  040572  051505  020124  047516
8131  040600  000200
8132  040601  000011  042411  050130          .ASCIZ  /               EXPECT          ACTUAL/
8133  040606  041505  004524  040411
8134  040614  052103  040525  000114
8135                          .EVEN
8136  040622  001116  001200  001210  DT10:   .WORD   $ERRPC,$TMP0,$TMP4,$$TSTNM,0
8137  040630  001266  000000
8138  040634  051106  046110  046440  EM11:   .ASCIZ  /FRHL MPY SIGN EXTEND NOT GOING LOW/
8139  040642  054520  051440  043511
8140  040650  020116  054105  042524
8141  040656  042116  047040  052117
8142  040664  043440  044517  043516
8143  040672  046040  053517  000000
8144  040677  000106  044122  020114  EM12:   .ASCII  "FRHL MPY/DIV ADD(1)L NOT GOING LOW OR NOT CAUSING"<CRLF>
8145  040704  050115  027531  044504
8146  040712  020126  042101  024104
8147  040720  024461  020114  047516
8148  040726  020124  047507  047111
8149  040734  020107  047514  020127
8150  040742  051117  047040  052117
8151  040750  041440  052501  044523
8152  040756  043516  000200
```

```
8153  040761    106  046522  020110          .ASCIZ  "FRMH FORCE MPY/DIV ADD TO FORCE THE ADD"
8154  040766 047506  041522  020105
8155  040774 050115  027531  044504
8156  041002 020126  042101  020104
8157  041010 047524  043040  051117
8158  041016 042503  052040  042510
8159  041024 040440  042104    000
8160  041031    101  020122  040504  EM13:   .ASCIZ  /AR DATA BAD/
8161  041036 040524  041040  042101
8162  041044    000
8163         041046                          .EVEN
8164  041046 001116  001204  001214  DT13:   .WORD   $ERRPC,$TMP2,$TMP6,$$TSTNM,0
8165  041054 001266  000000
8166  041060 052515  020114  044123  EM14:   .ASCIZ  /MUL SHIFT ENCODER ROMOUT BAD/
8167  041066 043111  020124  047105
8168  041074 047503  042504  020122
8169  041102 047522  047515  052125
8170  041110 041040  042101    000
8171  041115    105  051122  041520  DH14:   .ASCIZ  /ERRPC  ROMOUT  TEST NO/
8172  041122 051011  046517  052517
8173  041130 004524  042524  052123
8174  041136 047040  000117
8175                                  .EVEN
8176  041142 001116  001172  001266  DT14:   .WORD   $ERRPC,$REG5,$$TSTNM,0
8177  041150 000000
8178  041152 051106  045510  046440  EM15:   .ASCII  "FRHK MPY/DIV ADD(1)L NOT GOING HIGH OR" <CRLF>
8179  041160 054520  042057  053111
8180  041166 040440  042104  030450
8181  041174 046051  047040  052117
8182  041202 043440  044517  043516
8183  041210 044040  043511  020110
8184  041216 051117    200
8185  041221    106  046522  020110          .ASCIZ  "FRMH FORCE MPY/DIV SUB NOT GOING LOW"
8186  041226 047506  041522  020105
8187  041234 050115  027531  044504
8188  041242 020126  052523  020102
8189  041250 047516  020124  047507
8190  041256 047111  020107  047514
8191  041264 000127
8192  041266 051106  045510  046440  EM16:   .ASCII  /FRHK MU. SWR NOT GOING LOW OR MUL SHF ENCODER ROM/<CRLF>
8193  041274 046125  051440  051127
8194  041302 047040  052117  043440
8195  041310 044517  043516  046040
8196  041316 053517  047440  020122
8197  041324 052515  020114  044123
8198  041332 020106  047105  047503
8199  041340 042504  020122  047522
8200  041346 100115
8201  041350 047516  020124  052517          .ASCIZ  /NOT OUTPUTTING 12 IN STATE 112/
8202  041356 050124  052125  044524
8203  041364 043516  030440  020062
8204  041372 047111  051440  040524
8205  041400 042524  030440  031061
8206  041406    000
8207  041407    106  044122  020113  EM17:   .ASCIZ  /FRHK MPY SIGN EXTEND NOT GOING HIGH/
8208  041414 050115  020131  044523
```

```
8209  041422  047107  042440  052130
8210  041430  047105  020104  047516
8211  041436  020124  047507  047111
8212  041444  020107  044510  044107
8213  041452     000
8214  041453     106  044122  020113  EM20:   .ASCIZ  /FRHK E115(4) NOT GOING LOW/
8215  041460  030505  032461  032050
8216  041466  020051  047516  020124
8217  041474  047507  047111  020107
8218  041502  047514  000127
8219  041506  051106  045510  042440  EM21:   .ASCIZ  /FRHK E115(4) NOT GOING HIGH/
8220  041514  030461  024065  024464
8221  041522  047040  052117  043440
8222  041530  044517  043516  044040
8223  041536  043511  000110
8224  041542  051106  045510  042440  EM22:   .ASCIZ  /FRHK E107(12) NOT GOING LOW/
8225  041550  030061  024067  031061
8226  041556  020051  047516  020124
8227  041564  047507  047111  020107
8228  041572  047514  000127
8229  041576  051106  045510  046440  EM23:   .ASCIZ  /FRHK MUL SWR DID NOT ENABLE FRMH FORCE A/
8230  041604  046125  051440  051127
8231  041612  042040  042111  047040
8232  041620  052117  042440  040516
8233  041626  046102  020105  051106
8234  041634  044115  043040  051117
8235  041642  042503  040440     000
8236  041647     122  051505  046125  EM24:   .ASCIZ  /RESULT WRONG/
8237  041654  020124  051127  047117
8238  041662  000107
8239  041664  051106  045510  047440  EM25:   .ASCIZ  /FRHK OR FRLH MUL SHIFT ENCODER ROM FAILED/
8240  041672  020122  051106  044114
8241  041700  046440  046125  051440
8242  041706  044510  052106  042440
8243  041714  041516  042117  051105
8244  041722  051040  046517  043040
8245  041730  044501  042514  000104
8246  041736  052515  052114  050111  EM25A:  .ASCIZ  /MULTIPLIER/
8247  041744  044514  051105     000
8248  041751     122  046517  040440  EM25B:  .ASCIZ  /ROM ADDRESS/
8249  041756  042104  042522  051523
8250  041764     000
8251  041765     122  046517  052517  EM25C:  .ASCIZ  /ROMOUT DATA/
8252  041772  020124  040504  040524
8253  042000     000
8254  042001     106  044122  020113  EM26:   .ASCIZ  /FRHK MUL SWR NOT GOING HIGH WITH PIN 9 & 10 ON A HIGH/
8255  042006  052515  020114  053523
8256  042014  020122  047516  020124
8257  042022  047507  047111  020107
8258  042030  044510  044107  053440
8259  042036  052111  020110  044520
8260  042044  020116  020071  020046
8261  042052  030061  047440  020116
8262  042060  020101  044510  044107
8263  042066     000
8264  042067     011  004411  042011  DH26:   .ASCII  /                                DATA/<CRLF>
```

```
8265  042074  052101  100101
8266  042100  020011  020040  054105          .ASCIZ  /          EXPECT                                    ACTUAL/
8267  042106  042520  052103  004411
8268  042114  020011  020040  020011
8269  042122  020040  041501  052524
8270  042130  046101    000
8271          042134                          .EVEN
8272  042134  001200  001210  000000  DT26:   .WORD   $TMP0,$TMP4,0
8273  042142     002     002          DF26:   .BYTE   2,2
8274  042144  051106  044114  042440  EM27:   .ASCIZ  /FRLH E1(11) DID NOT GO LOW WITH H,L INPUT/
8275  042152  024061  030461  020051
8276  042160  044504  020104  047516
8277  042166  020124  047507  046040
8278  042174  053517  053440  052111
8279  042202  020110  026110  020114
8280  042210  047111  052520  000124
8281  042216  051106  045510  042440  EM30:   .ASCIZ  /FRHK E107 NOT GOING LOW WITH MUL SHF 0 ON A H/
8282  042224  030061  020067  047516
8283  042232  020124  047507  047111
8284  042240  020107  047514  020127
8285  042246  044527  044124  046440
8286  042254  046125  051440  043110
8287  042262  030040  047440  020116
8288  042270  020101  000110
8289  042274  051106  044114  042440  EM31:   .ASCIZ  /FRLH E1(11) DID NOT GO HIGH WITH BOTH INPUTS ON A HIGH/
8290  042302  024061  030461  020051
8291  042310  044504  020104  047516
8292  042316  020124  047507  044040
8293  042324  043511  020110  044527
8294  042332  044124  041040  052117
8295  042340  020110  047111  052520
8296  042346  051524  047440  020116
8297  042354  020101  044510  044107
8298  042362     000
8299  042363     106  044122  020113  EM32:   .ASCIZ  /FRHK E107(6) NOT GOING LOW WITH H,L,H INPUT/
8300  042370  030505  033460  033050
8301  042376  020051  047516  020124
8302  042404  047507  047111  020107
8303  042412  047514  020127  044527
8304  042420  044124  044040  046054
8305  042426  044054  044440  050116
8306  042434  052125    000
8307  042437     106  044122  020113  EM33:   .ASCIZ  /FRHK E107(12) NOT GOING LOW WITH H,H,L INPUT/
8308  042444  030505  033460  030450
8309  042452  024462  047040  052117
8310  042460  043440  044517  043516
8311  042466  046040  053517  053440
8312  042474  052111  020110  026110
8313  042502  026110  020114  047111
8314  042510  052520  000124
8315
8316  042514  052123  052101  020105  EM34:   .ASCIZ  /STATE 355 DID NOT CLEAR DST+1 OR FXPC FIRB08 NOT GETTING TO FRMA AS A H
8317  042522  032463  020065  044504
8318  042530  020104  047516  020124
8319  042536  046103  040505  020122
8320  042544  051504  025524  020061
```

```
8321  042552  051117  043040  050130
8322  042560  020103  044506  041122
8323  042566  034060  047040  052117
8324  042574  043440  052105  044524
8325  042602  043516  052040  020117
8326  042610  051106  040515  040440
8327  042616  020123  020101  044510
8328  042624  044107     000
8329  042627     123  040524  042524  EM35:   .ASCIZ  /STATE 305 DID NOT CLEAR DST+1/
8330  042634  031440  032460  042040
8331  042642  042111  047040  052117
8332  042650  041440  042514  051101
8333  042656  042040  052123  030453
8334  042664     000
8335  042665     106  040522  052103  EM36:   .ASCIZ  /FRACTION IS WRONG ON MODF*INT=0/
8336  042672  047511  020116  051511
8337  042700  053440  047522  043516
8338  042706  047440  020116  047515
8339  042714  043104  044452  052116
8340  042722  030075     000
8341  042725     123  040524  042524  EM37:   .ASCIZ  /STATE 266 DID NOT SET BN/
8342  042732  031040  033066  042040
8343  042740  042111  047040  052117
8344  042746  051440  052105  041040
8345  042754  000116
8346  042756  051106  041501  044524  EM40:   .ASCIZ  /FRACTION ACC DID NOT CLEAR IN STATE 341/
8347  042764  047117  040440  041503
8348  042772  042040  042111  047040
8349  043000  052117  041440  042514
8350  043006  051101  044440  020116
8351  043014  052123  052101  020105
8352  043022  032063  000061
8353  043026  047111  042524  042507  EM41:   .ASCIZ  /INTEGER PORTION WRONG IN STATE 7/
8354  043034  020122  047520  052122
8355  043042  047511  020116  051127
8356  043050  047117  020107  047111
8357  043056  051440  040524  042524
8358  043064  033440     000
8359  043067     106  040522  052103  EM42:   .ASCIZ  /FRACTION WRONG ON MOD/
8360  043074  047511  020116  051127
8361  043102  047117  020107  047117
8362  043110  046440  042117     000
8363  043115     111  052116  043505  EM43:   .ASCIZ  /INTEGER WRONG ON MOD/
8364  043122  051105  053440  047522
8365  043130  043516  047440  020116
8366  043136  047515  000104
8367  043142  047111  042524  042507  EM44:   .ASCIZ  /INTEGER DID NOT CLEAR ON OVERFLOW/
8368  043150  020122  044504  020104
8369  043156  047516  020124  046103
8370  043164  040505  020122  047117
8371  043172  047440  042526  043122
8372  043200  047514  000127
8373  043204  052123  052101  020105  EM45:   .ASCIZ  /STATE 105 NOT DETECTING ZERO FRACTION/
8374  043212  030061  020065  047516
8375  043220  020124  042504  042524
8376  043226  052103  047111  020107
```

```
8377  043234  042532  047522  043040
8378  043242  040522  052103  047511
8379  043250  000116
8380  043252  047514  044507  040503  EM46:   .ASCIZ  /LOGICAL "AND" FAILED ON FRACTION/
8381  043260  020114  040442  042116
8382  043266  020042  040506  046111
8383  043274  042105  047440  020116
8384  043302  051106  041501  044524
8385  043310  047117     000
8386  043313     114  043517  041511  EM47:   .ASCIZ  /LOGICAL "AND" FAILED ON INTEGER/
8387  043320  046101  021040  047101
8388  043326  021104  043040  044501
8389  043334  042514  020104  047117
8390  043342  044440  052116  043505
8391  043350  051105     000
8392          043354                  .EVEN
8393  043354  001170  001210  000000  DT47:   .WORD   $REG4,$TMP4,0
8394  043362  044504  043126  042052  EM50:   .ASCIZ  /DIVF*DST=0 DID NOT CLR DST/
8395  043370  052123  030075  042040
8396  043376  042111  047040  052117
8397  043404  041440  051114  042040
8398  043412  052123     000
8399  043415     106  044122  020103  EM51:   .ASCIZ  /FRHC FALU59 NOT GETTING TO FRLF AS A HIGH/
8400  043422  040506  052514  034465
8401  043430  047040  052117  043440
8402  043436  052105  044524  043516
8403  043444  052040  020117  051106
8404  043452  043114  040440  020123
8405  043460  020101  044510  044107
8406  043466     000
8407  043467     106  044122  020113  EM52:   .ASCII  /FRHK DIV DONE STUCK LOW OR FRHC FALU59 NOT/<CRLF>
8408  043474  044504  020126  047504
8409  043502  042516  051440  052524
8410  043510  045503  046040  053517
8411  043516  047440  020122  051106
8412  043524  041510  043040  046101
8413  043532  032525  020071  047516
8414  043540  100124
8415  043542  042507  052124  047111          .ASCIZ  /GETTING TO FRHL AS A HIGH/
8416  043550  020107  047524  043040
8417  043556  044122  020114  051501
8418  043564  040440  044040  043511
8419  043572  000110
8420  043574  051121  042040  052101  EM53:   .ASCIZ  /QR DATA BAD/
8421  043602  020101  040502  000104
8422  043610  051106  044114  044040  EM54:   .ASCIZ  /FRLH HI QUOT GEN BIT DID NOT GO LOW/
8423  043616  020111  052521  052117
8424  043624  043440  047105  041040
8425  043632  052111  042040  042111
8426  043640  047040  052117  043440
8427  043646  020117  047514  000127
8428  043654  051106  041510  043040  EM55:   .ASCIZ  /FRHC FALU59 NOT GETTING TO FRLF AS A LOW/
8429  043662  046101  032525  020071
8430  043670  047516  020124  042507
8431  043676  052124  047111  020107
8432  043704  047524  043040  046122
```

```
8433  043712  020106  051501  040440
8434  043720  046040  053517     000
8435  043725     106  044122  020103  EM56:   .ASCIZ  /FRHC FALU59 NOT GETTING TO FRHL DIV NORM MUX AS A HIGH/
8436  043732  040506  052514  034465
8437  043740  047040  052117  043440
8438  043746  052105  044524  043516
8439  043754  052040  020117  051106
8440  043762  046110  042040  053111
8441  043770  047040  051117  020115
8442  043776  052515  020130  051501
8443  044004  040440  044040  043511
8444  044012  000110
8445  044014  051106  046110  046440  EM60:   .ASCIZ  "FRHL MPY/DIV ADD(1)L DID NOT GO LOW WITH FALU59 L ON A LOW"
8446  044022  054520  042057  053111
8447  044030  040440  042104  030450
8448  044036  046051  042040  042111
8449  044044  047040  052117  043440
8450  044052  020117  047514  020127
8451  044060  044527  044124  043040
8452  044066  046101  032525  020071
8453  044074  020114  047117  040440
8454  044102  046040  053517     000
8455  044107     106  046122  020106  EM62:   .ASCIZ  /FRLF HI QUOT GEN BIT NOT GOING HIGH/
8456  044114  044510  050440  047525
8457  044122  020124  042507  020116
8458  044130  044502  020124  047516
8459  044136  020124  047507  047111
8460  044144  020107  044510  044107
8461  044152     000
8462  044153     106  044122  020114  EM63:   .ASCIZ  "FRHL MPY/DIV ADD(1)L NOT GOING HIGH WITH FALU59 L ON A HIGH"
8463  044160  050115  027531  044504
8464  044166  020126  042101  024104
8465  044174  024461  020114  047516
8466  044202  020124  047507  047111
8467  044210  020107  044510  044107
8468  044216  053440  052111  020110
8469  044224  040506  052514  034465
8470  044232  046040  047440  020116
8471  044240  020101  044510  044107
8472  044246     000
8473  044247     106  044122  020113  EM64:   .ASCIZ  /FRHK NORM NEG ENCODER FAILED/
8474  044254  047516  046522  047040
8475  044262  043505  042440  041516
8476  044270  042117  051105  043040
8477  044276  044501  042514  000104
8478  044304  004411  040504  040524  DH64:   .ASCII  /                DATA                    FALU<58:51>/<CRLF>
8479  044312  004411  043011  046101
8480  044320  036125  034065  032472
8481  044326  037061     200
8482  044331     040  020040  020040          .ASCIZ  /        EXPECT            ACTUAL/
8483  044336  020040  042440  050130
8484  044344  041505  004524  020011
8485  044352  020040  041501  052524
8486  044360  046101     000
8487          044364          .EVEN
8488  044364  001200  001204  001172  DT64:   .WORD   $TMP0,$TMP2,$REG5,0
```

```
8489  044372  000000
8490  044374     001      001      000  DF64:   .BYTE   1,1,0,0
8491  044377     000
8492  044400  042522  052523  052114  EM65:   .ASCIZ  /RESULT WRONG ON DIV/
8493  044406  053440  047522  043516
8494  044414  047440  020116  044504
8495  044422  000126
8496  044424  051106  045510  047040  EM66:   .ASCIZ  /FRHK NORM SHIFT COUNT LOOKUP ROM FAILED/
8497  044432  051117  020115  044123
8498  044440  043111  020124  047503
8499  044446  047125  020124  047514
8500  044454  045517  050125  051040
8501  044462  046517  043040  044501
8502  044470  042514  000104
8503  044474  004411  042040  052101  DH66:   .ASCII  /                DATA                    NORM ROM/<CRLF>
8504  044502  004501  004411  020040
8505  044510  047040  051117  020115
8506  044516  047522  100115
8507  044522  020040  020040  020040          .ASCIZ  /      EXPECT            ACTUAL          ADDRESS ROMOUT/
8508  044530  054105  042520  052103
8509  044536  004411  020040  041501
8510  044544  052524  046101  020011
8511  044552  020040  042101  051104
8512  044560  051505  020123  047522
8513  044566  047515  052125     000
8514          044574          .EVEN
8515  044574  001200  001204  001172  DT66:   .WORD   $TMP0,$TMP2,$REG5,$REG6,0
8516  044602  001174  000000
8517  044606  051106  043114  042040  EM67:   .ASCIZ  /FRLF DLE PREC QT LO DID NOT GO LOW/
8518  044614  042514  050040  042522
8519  044622  020103  052121  046040
8520  044630  020117  044504  020104
8521  044636  047516  020124  047507
8522  044644  046040  053517     000
8523  044651     106  046122  020110  EM70:   .ASCIZ  /FRLH DLE PREC QT HI B(1) NOT GOING HIGH/
8524  044656  046104  020105  051120
8525  044664  041505  050440  020124
8526  044672  044510  041040  030450
8527  044700  020051  047516  020124
8528  044706  047507  047111  020107
8529  044714  044510  044107     000
8530  044721     106  046122  020106  EM71:   .ASCIZ  /FRLF E21(12) NOT GOING HIGH/
8531  044726  031105  024061  031061
8532  044734  020051  047516  020124
8533  044742  047507  047111  020107
8534  044750  044510  044107     000
8535  044755     106  046122  020110  EM72:   .ASCIZ  /FRLH DLE PREC QT HI NOT GOING LOW/
8536  044762  046104  020105  051120
8537  044770  041505  050440  020124
8538  044776  044510  047040  052117
8539  045004  043440  044517  043516
8540  045012  046040  053517     000
8541  045017     106  046122  020110  EM73:   .ASCIZ  /FRLH DLE PREC QT LO DID NOT GO HI/
8542  045024  046104  020105  051120
8543  045032  041505  050440  020124
8544  045040  047514  042040  042111
```

```
8545  045046  047040  052117  043440
8546  045054  020117  044510     000
8547  045061     106  046122  020112  EM75:   .ASCIZ  /FRLJ QSI-1 DID NOT GO LOW WHEN SELECTING C1 INPUT/
8548  045066  051521  026511  020061
8549  045074  044504  020104  047516
8550  045102  020124  047507  046040
8551  045110  053517  053440  042510
8552  045116  020116  042523  042514
8553  045124  052103  047111  020107
8554  045132  030503  044440  050116
8555  045140  052125     000
8556  045143     106  046122  020112  EM76:   .ASCIZ  /FRLJ QSI-1 DID NOT GO LOW WHEN SELECTING A1 INPUT/
8557  045150  051521  026511  020061
8558  045156  044504  020104  047516
8559  045164  020124  047507  046040
8560  045172  053517  053440  042510
8561  045200  020116  042523  042514
8562  045206  052103  047111  020107
8563  045214  030501  044440  050116
8564  045222  052125     000
8565  045225     106  046122  020112  EM77:   .ASCIZ  /FRLJ QSI-1 DID NOT GO HIGH WHEN SELECTING A1 INPUT/
8566  045232  051521  026511  020061
8567  045240  044504  020104  047516
8568  045246  020124  047507  044040
8569  045254  043511  020110  044127
8570  045262  047105  051440  046105
8571  045270  041505  044524  043516
8572  045276  040440  020061  047111
8573  045304  052520  000124
8574  045310  047522  020115  052123  EM100:  .ASCIZ  /ROM STATE 246 FAILED/
8575  045316  052101  020105  032062
8576  045324  020066  040506  046111
8577  045332  042105     000
8578  045335     106  041505  053440  EM101:  .ASCIZ  /FEC WRONG ON STST/
8579  045342  047522  043516  047440
8580  045350  020116  052123  052123
8581  045356     000
8582  045357     106  040505  053440  EM102:  .ASCIZ  /FEA WRONG ON STST/
8583  045364  047522  043516  047440
8584  045372  020116  052123  052123
8585  045400     000
8586  045401     105  051122  041520  DH101:  .ASCII  /ERRPC        FEC       TEST NO/<CRLF>
8587  045406  020011  020040  020040
8588  045414  043040  041505  052011
8589  045422  051505  020124  047516
8590  045430     200
8591  045431     011  054105  042520          .ASCIZ  /       EXPECT  ACTUAL/
8592  045436  052103  040411  052103
8593  045444  040525  000114
8594                          .EVEN
8595  045450  001116  001200  001204  DT101:  .WORD   $ERRPC,$TMP0,$TMP2,$$TSTNM,0
8596  045456  001266  000000
8597  045462  051105  050122  004503  DH102:  .ASCII  /ERRPC        FEA       TEST NO/<CRLF>
8598  045470  020040  020040  020040
8599  045476  042506  004501  042524
8600  045504  052123  047040  100117
```

```
8601  045512  042411  050130  041505          .ASCIZ  /       EXPECT  ACTUAL/
8602  045520  004524  041501  052524
8603  045526  046101     000
8604          045532          .EVEN
8605  045532  001116  001202  001206  DT102:  .WORD   $ERRPC,$TMP1,$TMP3,$$TSTNM,0
8606  045540  001266  000000
8607  045544  046506  020060  051124  EM103:  .ASCIZ  /FMO TRAP FAILED-CHECK FRMB FMO/
8608  045552  050101  043040  044501
8609  045560  042514  026504  044103
8610  045566  041505  020113  051106
8611  045574  041115  043040  030115
8612  045602     000
8613  045603     106  041505  053440  EM104:  .ASCIZ  /FEC WRONG ON FMO TRAP-CHECK ROM STATE 1/
8614  045610  047522  043516  047440
8615  045616  020116  046506  020060
8616  045624  051124  050101  041455
8617  045632  042510  045503  051040
8618  045640  046517  051440  040524
8619  045646  042524  030440     000
8620  045653     106  040505  053440  EM105:  .ASCIZ  /FEA WRONG ON FMO TRAP /
8621  045660  047522  043516  047440
8622  045666  020116  046506  020060
8623  045674  051124  050101  000040
8624  045702  046506  020060  040506  EM106:  .ASCIZ  /FMO FAILED-CHECK APPROPRIATE ROM STATE/
8625  045710  046111  042105  041455
8626  045716  042510  045503  040440
8627  045724  050120  047522  051120
8628  045732  040511  042524  051040
8629  045740  046517  051440  040524
8630  045746  042524     000
8631  045751     125  042116  051105  EM107:  .ASCIZ  /UNDERFLOW DID NOT TRAP-CHECK FRMA FIU(0)/
8632  045756  046106  053517  042040
8633  045764  042111  047040  052117
8634  045772  052040  040522  026520
8635  046000  044103  041505  020113
8636  046006  051106  040515  043040
8637  046014  052511  030050  000051
8638  046022  042506  020103  051127  EM110:  .ASCIZ  /FEC WRONG ON UNDERFLOW-CHECK ROM STATE 205/
8639  046030  047117  020107  047117
8640  046036  052440  042116  051105
8641  046044  046106  053517  041455
8642  046052  042510  045503  051040
8643  046060  046517  051440  040524
8644  046066  042524  031040  032460
8645  046074     000
8646  046075     106  040505  053440  EM111:  .ASCIZ  /FEA WRONG ON UNDERFLOW/
8647  046102  047522  043516  047440
8648  046110  020116  047125  042504
8649  046116  043122  047514  000127
8650  046124  054105  047520  042516  EM112:  .ASCIZ  /EXPONENT WRONG ON UNDERFLOW/
8651  046132  052116  053440  047522
8652  046140  043516  047440  020116
8653  046146  047125  042504  043122
8654  046154  047514  000127
8655  046160  053117  051105  046106  EM113:  .ASCIZ  /OVERFLOW DID NOT TRAP-CHECK FRLN FVINT GETTING TO FRMA/
8656  046166  053517  042040  042111
```

```
 8657  046174  047040  052117  052040
 8658  046202  040522  026520  044103
 8659  046210  041505  020113  051106
 8660  046216  047114  043040  044526
 8661  046224  052116  043440  052105
 8662  046232  044524  043516  052040
 8663  046240  020117  051106  040515
 8664  046246     000
 8665  046247     105  050130  047117  EM114:  .ASCIZ  /EXPONENT WRONG ON OVERFLOW/
 8666  046254  047105  020124  051127
 8667  046262  047117  020107  047117
 8668  046270  047440  042526  043122
 8669  046276  047514  000127
 8670  046302  042506  020103  051127  EM115:  .ASCIZ  /FEC WRONG ON OVERFLOW-CHECK STATE 171/
 8671  046310  047117  020107  047117
 8672  046316  047440  042526  043122
 8673  046324  047514  026527  044103
 8674  046332  041505  020113  052123
 8675  046340  052101  020105  033461
 8676  046346  000061
 8677  046350  042506  020101  051127  EM116:  .ASCIZ  /FEA WRONG ON OVERFLOW/
 8678  046356  047117  020107  047117
 8679  046364  047440  042526  043122
 8680  046372  047514  000127
 8681  046376  042506  020103  051127  EM117:  .ASCIZ  /FEC WRONG ON OVERFLOW-CHECK STATE 105/
 8682  046404  047117  020107  047117
 8683  046412  047440  042526  043122
 8684  046420  047514  026527  044103
 8685  046426  041505  020113  052123
 8686  046434  052101  020105  030061
 8687  046442  000065
 8688  046444  047503  053116  051105  EM120:  .ASCIZ  /CONVERSION ERROR DID NOT TRAP-CHECK FRLN FCINT GETTING TO FRMA/
 8689  046452  044523  047117  042440
 8690  046460  051122  051117  042040
 8691  046466  042111  047040  052117
 8692  046474  052040  040522  026520
 8693  046502  044103  041505  020113
 8694  046510  051106  047114  043040
 8695  046516  044503  052116  043440
 8696  046524  052105  044524  043516
 8697  046532  052040  020117  051106
 8698  046540  040515     000
 8699  046543     106  041505  053440  EM121:  .ASCIZ  /FEC WRONG ON CONVERSION ERROR/
 8700  046550  047522  043516  047440
 8701  046556  020116  047503  053116
 8702  046564  051105  044523  047117
 8703  046572  042440  051122  051117
 8704  046600     000
 8705  046601     106  040505  053440  EM122:  .ASCIZ  /FEA WRONG ON CONVERSION ERROR/
 8706  046606  047522  043516  047440
 8707  046614  020116  047503  053116
 8708  046622  051105  044523  047117
 8709  046630  042440  051122  051117
 8710  046636     000
 8711  046637     111  046114  043505  EM123:  .ASCIZ  /ILLEGAL OP-CODE DID NOT TRAP-CHECK NO-MEM ROM/
 8712  046644  046101  047440  026520
```

```
8713  046652  047503  042504  042040
8714  046660  042111  047040  052117
8715  046666  052040  040522  026520
8716  046674  044103  041505  020113
8717  046702  047516  046455  046505
8718  046710  051040  046517     000
8719  046715     105  051122  041520  DH123:  .ASCIZ  /ERRPC  OPCODE  TEST NO/
8720  046722  047411  041520  042117
8721  046730  004505  042524  052123
8722  046736  047040  000117
8723                                  .EVEN
8724  046742  001116  001200  001266  DT123:  .WORD   $ERRPC,$TMPO,$STSTNM,0
8725  046750  000000
8726  046752  042506  020103  051127  EM124:  .ASCIZ  /FEC WRONG ON ILLEGAL OP-CODE/
8727  046760  047117  020107  047117
8728  046766  044440  046114  043505
8729  046774  046101  047440  026520
8730  047002  047503  042504     000
8731  047007     106  040505  053440  EM125:  .ASCIZ  /FEA WRONG ON ILLEGAL OP CODE/
8732  047014  047522  043516  047440
8733  047022  020116  046111  042514
8734  047030  040507  020114  050117
8735  047036  041440  042117  000105
8736  047044  042514  040507  020114  EM126:  .ASCIZ  /LEGAL OP CODE FAILED (LDCIF*R6)/
8737  047052  050117  041440  042117
8738  047060  020105  040506  046111
8739  047066  042105  024040  042114
8740  047074  044503  025106  033122
8741  047102  000051
8742  047104  051105  050122  004503  DH126:  .ASCII  /ERRPC          DATA    TEST NO/<CRLF>
8743  047112  042011  052101  004501
8744  047120  042524  052123  047040
8745  047126  100117
8746  047130  042411  050130  041505          .ASCIZ  /       EXPECT  ACTUAL/
8747  047136  004524  041501  052524
8748  047144  046101     000
8749  047147     114  043505  046101  EM127:  .ASCIZ  /LEGAL OP-CODE TRAPPED (LDCIF*R6)(NO-MEM ROM)/
8750  047154  047440  026520  047503
8751  047162  042504  052040  040522
8752  047170  050120  042105  024040
8753  047176  042114  044503  025106
8754  047204  033122  024051  047516
8755  047212  046455  046505  051040
8756  047220  046517  000051
8757  047224  051105  050122  004503  DH127:  .ASCIZ  /ERRPC  OPCODE  TEST NO/
8758  047232  050117  047503  042504
8759  047240  052011  051505  020124
8760  047246  047516     000
8761  047251     122  046517  051440  EM130:  .ASCIZ  /ROM STATE 32 DID NOT LOAD FEC/
8762  047256  040524  042524  031440
8763  047264  020062  044504  020104
8764  047272  047516  020124  047514
8765  047300  042101  043040  041505
8766  047306     000
8767  047307     124  041515  020101  EM131:  .ASCIZ  /TMCA HONOR PIR7 DID NOT GO HIGH WHEN FP TRAP PRESENT/
8768  047314  047510  047516  020122
```

```
8769  047322  044520  033522  042040
8770  047330  042111  047040  052117
8771  047336  043440  020117  044510
9772  047344  044107  053440  042510
8773  047352  020116  050106  052040
8774  047360  040522  020120  051120
8775  047366  051505  047105  000124
8776  047374  046524  040503  044040  EM132:  .ASCIZ  /TMCA HONOR FP TRAP DID NOT GO HIGH WHEN SL YELLOW PRESENT/
8777  047402  047117  051117  043040
8778  047410  020120  051124  050101
8779  047416  042040  042111  047040
8780  047424  052117  043440  020117
8781  047432  044510  044107  053440
8782  047440  042510  020116  046123
8783  047446  054440  046105  047514
8784  047454  020127  051120  051505
8785  047462  047105  000124
8786  047466  044505  044124  051105  EM133:  .ASCIZ  /EITHER SAPK I SPACEA(0) NOT GOING LOW OR "M8138-YA" NOT INSTALLED/
8787  047474  051440  050101  020113
8788  047502  020111  050123  041501
8789  047510  040505  030050  020051
8790  047516  047516  020124  047507
8791  047524  047111  020107  047514
8792  047532  020127  051117  021040
8793  047540  034115  031461  026470
8794  047546  040531  020042  047516
8795  047554  020124  047111  052123
8796  047562  046101  042514  000104
8797  047570  042506  020103  051127  EM134:  .ASCIZ  /FEC WRONG IN STATE 356 (DIV BY ZERO)/
8798  047576  047117  020107  047111
8799  047604  051440  040524  042524
8800  047612  031440  033065  024040
8801  047620  044504  020126  054502
8802  047626  055040  051105  024517
8803  047634     000
8804  047635     106  040505  053440  EM135:  .ASCIZ  /FEA WRONG ON DIV BY ZERO/
8805  047642  047522  043516  047440
8806  047650  020116  044504  020126
8807  047656  054502  055040  051105
8808  047664  000117
8809  047666  042506  020103  051127  EM136:  .ASCIZ  /FEC WRONG IN STATE 346 (DIV BY ZERO)/
8810  047674  047117  020107  047111
8811  047702  051440  040524  042524
8812  047710  031440  033064  024040
8813  047716  044504  020126  054502
8814  047724  055040  051105  024517
8815  047732     000
8816  047733     124  041515  020101  EM137:  .ASCIZ  /TMCA HONOR SEGT NOT DISABLING FP TRAP/
8817  047740  047510  047516  020122
8818  047746  042523  052107  047040
8819  047754  052117  042040  051511
8820  047762  041101  044514  043516
8821  047770  043040  020120  051124
8822  047776  050101     000
8823  050001     116  043505  020106  EM140:  .ASCIZ  /NEGF AND FMD FAILED/
8824  050006  047101  020104  046506
```

```
8825  050014  020060  040506  046111
8826  050022  042105     000
8827  050025     105  051122  041520  DH140:  .ASCII  /ERRPC       DATA       TEST NO/<CRLF>
8828  050032  020011  020040  020040
8829  050040  040504  040524  052011
8830  050046  051505  020124  047516
8831  050054     200
8832  050055     011  054105  042520          .ASCIZ  /       EXPECT  ACTUAL/
8833  050062  052103  020040  041501
8834  050070  052524  046101     000
8835          050076                  .EVEN
8836  050076  001116  001200  001160  DT140:  .WORD   $ERRPC,$TMPO,$REGO,$$TSTNM,0
8837  050104  001266  000000
8838  050110  051106  041115  044440  EM141:  .ASCIZ  /FRMB IMMEDIATE*REG WT NOT GOING LOW/
8839  050116  046515  042105  040511
8840  050124  042524  051052  043505
8841  050132  053440  020124  047516
8842  050140  020124  047507  047111
8843  050146  020107  047514  000127
8844  050154  044124  020105  050123  EM142:  .ASCIZ  /THE SP IS WRONG/
8845  050162  044440  020123  051127
8846  050170  047117  000107
8847  050174  051105  050122  004503  DH142:  .ASCII  /ERRPC         SP         TEST NO/<CRLF>
8848  050202  020040  020040  020040
8849  050210  050123  052011  051505
8850  050216  020124  047516     200
8851  050223     011  054105  042520          .ASCIZ  /       EXPECT  ACTUAL/
8852  050230  052103  020040  041501
8853  050236  052524  046101     000
8854  050243     131  046105  047514  EM143:  .ASCIZ  /YELLOW ZONE AND FP TRAP DID NOT OCCUR IN CORRECT SEQUENCE/
8855  050250  020127  047532  042516
8856  050256  040440  042116  043040
8857  050264  020120  051124  050101
8858  050272  042040  042111  047040
8859  050300  052117  047440  041503
8860  050306  051125  044440  020116
8861  050314  047503  051122  041505
8862  050322  020124  042523  052521
8863  050330  047105  042503     000
8864  050335     115  046507  020124  EM144:  .ASCIZ  /MGMT AND FP TRAP OUT OF SEQUENCE/
8865  050342  047101  020104  050106
8866  050350  052040  040522  020120
8867  050356  052517  020124  043117
8868  050364  051440  050505  042525
8869  050372  041516  000105
8870  050376  044502  020124  052123  EM145:  .ASCIZ  /BIT STUCK IN FPA OR FEA/
8871  050404  041525  020113  047111
8872  050412  043040  040520  047440
8873  050420  020122  042506  000101
8874  050426  047515  042504  030440  EM146:  .ASCIZ  /MODE 1 DID NOT CAUSE FXPB IMMEDIATE TO GO TRUE/
8875  050434  042040  042111  047040
8876  050442  052117  041440  052501
8877  050450  042523  043040  050130
8878  050456  020102  046511  042515
8879  050464  044504  052101  020105
8880  050472  047524  043440  020117
```

```
8881   050500  051124  042525     000
8882   050505     115  042117  020105  EM147:  .ASCIZ  /MODE 4 DID NOT CAUSE FXPB IMMEDIATE TO GO TRUE/
8883   050512  020064  044504  020104
8884   050520  047516  020124  040503
8885   050526  051525  020105  054106
8886   050534  041120  044440  046515
8887   050542  042105  040511  042524
8888   050550  052040  020117  047507
8889   050556  052040  052522  000105
8890   050564  042120  042122  050040  EM150:  .ASCIZ  /PDRD PS14 NOT DISABLING FXPN FMM(1)H/
8891   050572  030523  020064  047516
8892   050600  020124  044504  040523
8893   050606  046102  047111  020107
8894   050614  054106  047120  043040
8895   050622  046515  030450  044051
8896   050630     000
8897   050631     106  050130  020116  EM151:  .ASCIZ  /FXPN ACS0 DID NOT GO HIGH WITH ACF0(1)H*FIRB06(1)H/
8898   050636  041501  030123  042040
8899   050644  042111  047040  052117
8900   050652  043440  020117  044510
8901   050660  044107  053440  052111
8902   050666  020110  041501  030106
8903   050674  030450  044051  043052
8904   050702  051111  030102  024066
8905   050710  024461  000110
8906   050714  054106  047120  040440  EM152:  .ASCIZ  /FXPN ACS1 DID NOT GO HIGH WITH ACF0(1)H*FIRB07(1)H/
8907   050722  051503  020061  044504
8908   050730  020104  047516  020124
8909   050736  047507  044040  043511
8910   050744  020110  044527  044124
8911   050752  040440  043103  024060
8912   050760  024461  025110  044506
8913   050766  041122  033460  030450
8914   050774  044051     000
8915   050777     123  051103  052101  EM153:  .ASCIZ  /SCRATCH PAD CHIP FAILED/
8916   051004  044103  050040  042101
8917   051012  041440  044510  020120
8918   051020  040506  046111  042105
8919   051026     000
8920   051027     011  004411  020040  DH153:  .ASCII  /                                DATA                                ACC#
8921   051034  020040  020040  042040
8922   051042  052101  004501  004411
8923   051050  020011  020040  040440
8924   051056  041503  100043
8925   051062  004411  054105  042520          .ASCIZ  /                EXPECT                  ACTUAL/
8926   051070  052103  004411  040411
8927   051076  052103  040525  000114
8928                                    .EVEN
8929   051104  001200  001210  001172  DT153:  .WORD   $TMP0,$TMP4,$REG5,0
8930   051112  000000
8931   051114  047125  054105  042520  EM154:  .ASCIZ  /UNEXPECTED TRAP TO 4/
8932   051122  052103  042105  052040
8933   051130  040522  020120  047524
8934   051136  032040     000
8935   051141     105  051122  041520  DH154:  .ASCIZ  /ERRPC  PCOFTRP ERRREG  TEST NO/
8936   051146  050011  047503  052106
```

```
 8937  051154  050122  042411  051122
 8938  051162  042522  004507  042524
 8939  051170  052123  047040  000117
 8940                                          .EVEN
 8941  051176  001116  001200  001202  DT154:  .WORD   $ERRPC,$TMP0,$TMP1,$STSTNM,0
 8942  051204  001266  000000
 8943  051210  051105  050122  004503  DH155:  .ASCIZ  /ERRPC  PCOFTRP TEST NO/
 8944  051216  041520  043117  051124
 8945  051224  004520  042524  052123
 8946  051232  047040  000117
 8947                                          .EVEN
 8948  051236  001116  001200  001266  DT155:  .WORD   $ERRPC,$TMP0,$STSTNM,0
 8949  051244  000000
 8950  051246  047125  054105  042520  EM156:  .ASCIZ  /UNEXPECTED TRAP TO 114/
 8951  051254  052103  042105  052040
 8952  051262  040522  020120  047524
 8953  051270  030440  032061     000
 8954  051275     105  051122  041520  DH156:  .ASCIZ  /ERRPC  PCOFTRP ERRREG  LOADRS  HIADRS  TEST NO/
 8955  051302  050011  047503  052106
 8956  051310  050122  042411  051122
 8957  051316  042522  004507  047514
 8958  051324  042101  051522  044011
 8959  051332  040511  051104  004523
 8960  051340  042524  052123  047040
 8961  051346  000117
 8962                                          .EVEN
 8963  051350  001116  001200  001202  DT156:  .WORD   $ERRPC,$TMP0,$TMP1,$TMP2,$TMP3,$STSTNM,0
 8964  051356  001204  001206  001266
 8965  051364  000000
 8966  051366  047125  054105  042520  EM160:  .ASCIZ  /UNEXPECTED TRAP TO 244/
 8967  051374  052103  042105  052040
 8968  051402  040522  020120  047524
 8969  051410  031040  032064     000
 8970  051415     105  051122  041520  DH160:  .ASCIZ  /ERRPC  PCOFTRP   FEC     FEA   TEST NO/
 8971  051422  050011  047503  052106
 8972  051430  050122  020011  043040
 8973  051436  041505  020011  043040
 8974  051444  040505  052011  051505
 8975  051452  020124  047516     000
 8976          051460                          .EVEN
 8977  051460  001116  001200  001202  DT160:  .WORD   $ERRPC,$TMP0,$TMP1,$TMP2,$STSTNM,0
 8978  051466  001204  001266  000000
 8979  051474  051105  050122  004503  DH161:  .ASCII  /ERRPC                DATA                TEST NO/<CRLF>
 8980  051502  020011  020040  020040
 8981  051510  040504  040524  004411
 8982  051516  042524  052123  047040
 8983  051524  100117
 8984  051526  020011  020040  054105          .ASCIZ  /          EXPECT          ACTUAL/
 8985  051534  042520  052103  020011
 8986  051542  020040  041501  052524
 8987  051550  046101     000
 8988          051554                          .EVEN
 8989  051554  001116  001200  001202  DT161:  .WORD   $ERRPC,$TMP0,$TMP1,$TMP4,$TMP5,$STSTNM,0
 8990  051562  001210  001212  001266
 8991  051570  000000
 8992  051572  051105  050122  004503  DH162:  .ASCII  /ERRPC       DATA       TEST NO/<CRLF>
```

```
 8993  051600  020040  020040  042040
 8994  051606  052101  004501  042524
 8995  051614  052123  047040  100117
 8996  051622  042411  050130  041505          .ASCIZ  /      EXPECT  ACTUAL/
 8997  051630  004524  041501  052524
 8998  051636  046101     000
 8999          051641                    X14=.
 9000                                    ;;**********************************************************************
 9001                                    ;;**********************************************************************
 9002                                    ;;**********************************************************************
 9003                                    ;THIS WORD IS USED TO TEST THE FPC AND FEA REGISTERS.
 9004                                    ;IT MUST BE ASSEMBLED PRIOR TO THE PREVIOUS CODE REACHING
 9005                                    ;ADDRESS 52524. IF NOT, AN ERROR MESSAGE IS TYPED DURING
 9006                                    ;THE ASSEMBLY.
 9007
 9008
 9009
 9010          052524                    .=52524
 9011  052524  170016                            .WORD   170016
 9012  052526  000240                            NOP
 9013  052530  000240                            NOP
 9014
 9015
 9016
 9017                                    ;;**********************************************************************
 9018                                    ;;**********************************************************************
 9019                                    ;;**********************************************************************
 9020  052532  001000                    ADRTBL: .BLKW   1000
 9021  054532  000000                    OUTTBL: .WORD
 9022          000001                            .END
```

DEFPBA.CMB CROSS REFERENCE TABLE -- USER SYMBOLS

```
ABASE = 000000          2020
ACDW1 = 000000          2020
ACDW2 = 000000          2020
ACPUOP= 000000          2020  2035
AC0   =%000000          1885#  3011*  3015*  3017   3020   3048*  3052*  3057   3061   3081*  3083*  3090   3093
                        3138*  3140*  3159*  3160*  3169   3174   3232*  3233*  3243   3268*  3269*  3282   3322*
                        3323*  3327   3341*  3342*  3346   3356   3374*  3375*  3381   3394*  3395*  3399   3418*
                        3420*  3422   3423   3446*  3448*  3451   3486*  3487*  3488   3505*  3506*  3507   3547*
                        3549*  [illegible]  3571*  3572*  3576   3593*  3594*  3598   3616*  3617*  3621   3639*  3640*
                        3644   [illegible]  3665*  3669   3707*  3708*  3709   3726*  3727*  3728   3751*  3754*  3781*
                        3784*  [illegible]  3795   3820*  3828*  3830   3833   3896*  3898*  3935*  3936*  3937   3940
                        3967*  [illegible]  3978   3981   4007*  4008*  4022   4025   4058*  4059*  4060   4063   4103*
                        4104*  4110   4113   4147*  4150*  4155   4191*  4192*  4196   4209   4231*  4232*  4238
                        4245   4281*  4282*  4288   4331*  4332*  4340   4413*  4414*  4415   4418   4428*  4429*
                        4430   4448   4474*  4475*  4482   4504*  4505*  4511   4544*  4545*  4551   4572*  4573*
                        4579   4596*  4597*  4603   4630*  4631*  4638   4641   4727*  4760*  4772*  4784*  4796*
                        4884*  4886*  4901   4919*  4921*  4927   4945*  4947*  4953   4971*  4973*  4984*  4986*
                        5006*  5008*  5014   5040*  5042*  5058*  5060*  5073*  5077   5087*  5091   5102*  5106
                        5125*  5127   5148   5160   5176*  5179*  5194*  5197*  5201*  5260   5319*  5320   5333*
                        5335   5346*  5347   5589*  5590*  5591   5592   5602*  5605*  5608   5609   5674*  5675
                        5686*  5687   5688*  5689   5693*  5695   5707*  5708   5711*  5712   5727*  5728   5758*
                        5759   5760*  5761   5773*  5774   5804*  5817   5983*  5985*  5987   5990   6085*  6088
                        6138*  6140*  6141   6145   6195*  6197*  6198   6203   6420*  6422   6548*  6550*  6559
                        6562   6712*  6714*  6718   6722   7376*  7985   7989*  7990   7991*  7992   8006*  8007
                        8008*  8009   8013*
AC1   =%000001          1886#  3012*  3013   3023   3026   3752*  3755   3758   3782*  3788   3836   3839   3895*
                        3900   3903   3969*  3983   3986   4009   4016   4065   4068   4105   4108   4723*  4724
                        4725   4741   4744   4747   5077*  5648*  5692*  5698   5805*  5817*  5903*  5904*  5912
                        5916   5995*  5997*  5999   6002   6023*  6025   6060*  6063*  6065   6068   6099*  6102*
                        6103   6107   6183*  6186   6197   6198   6202   6385*  6387*  6394   6398   6492*  6494*
                        6497   6567*  6569*  6576   6580   6804*  6806   6833*  6835*  6839   6842   7986   8012*
AC2   =%000002          1887#  3013*  3015   3753*  3754   3868*  3869*  3870   3873   5647*  5691*  5701   5806*
                        5823*  5824   5825   5861*  5862*  5888*  5889   5970*  5972   6010*  6012*  6014   6017
                        6022*  6025*  6027   6030   6072*  6075*  6077   6080   6100*  6102   6103   6112*  6113
                        6115*  6117   6120   6137*  6140   6141   6154   6159   6166*  6168*  6173   6178   6350*
                        6352*  6359   6363   6403*  6405*  6412   6415   6456*  6458*  6465   6468   6528*  6530*
                        6537   6541   6728*  6730*  6734   6737   6759*  6761   6765   6819*  6822   6866*  6869*
                        6873   6876   7987   8011*
AC3   =%000003          1888#  3876   3879   4724*  4744   5646*  5651   5690*  5704   5807*  5824*  5825*  5826
                        5829   5872*  5873   5958*  5960*  5962   5966   6007*  6012   6058*  6061*  6063   6073*
                        6075   6084*  6085   6088   6113*  6115   6125*  6127   6150*  6153*  6154   6158   6169*
                        6172*  6173   6177   6333*  6335*  6342   6345   6368*  6370*  6377   6380   6439*  6441
                        6474*  6476   6511*  6514   6545*  6552*  6559   6563   6695*  6697*  6702   6706   6742*
                        6744   6773*  6775*  6779   6782   6788*  6790   6850*  6852*  6857   6860   7988   8010*
AC4   =%000004          1889#  5689*  5707   5808*  7989   8009*
AC5   =%000005          1890#  4725*  4747   5687*  5711   5809*  7991   8007*
AC6   =%000006          1891#  5260*
AC7   =%000007          1892#
ADDW0 = 000000          2020
ADDW1 = 000000          2020
ADDW10= 000000          2020
ADDW11= 000000          2020
ADDW12= 000000          2020
ADDW13= 000000          2020
ADDW14= 000000          2020
ADDW15= 000000          2020
```

```
ADDW2 = 000000           2020
ADDW3 = 000000           2020
ADDW4 = 000000           2020
ADDW5 = 000000           2020
ADDW6 = 000000           2020
ADDW7 = 000000           2020
ADDW8 = 000000           2020
ADDW9 = 000000           2020
ADEVCT= 000000           2020    2026
ADEVM = 000000           2020
ADRPTR  001374           2058#   7327*   7345*   7346*
ADRTBL  052532           2058    7327    7668    9020#
AENV  = 000000           2020    2031
AENVM = 000000           2020    2032
AFATAL= 000000           2020    2023
AMADR1= 000000           2020
AMADR2= 000000           2020
AMADR3= 000000           2020
AMADR4= 000000           2020
AMAMS1= 000000           2020
AMAMS2= 000000           2020
AMAMS3= 000000           2020
AMAMS4= 000000           2020
AMSGAD= 000000           2020    2028
AMSGLG= 000000           2020    2029
AMSGTY= 000000           2020    2022
AMTYP1= 000000           2020
AMTYP2= 000000           2020
AMTYP3= 000000           2020
AMTYP4= 000000           2020
APASS = 000000           2020    2025
APRIOR= 000000           2020
APTCSU= 000040           7458    7564#
APTENV= 000001           7141    7451    7520    7562#
APTSIZ= 000200           2973    7561#
APTSPO= 000100           7453    7522    7563#
ASWREG= 000000           2020    2033
ATESTN= 000000           2020    2024
AUNIT = 000000           2020    2027
AUSWR = 000000           2020    2034
AVECT1= 000000           2020
AVECT2= 000000           2020
BIT0  = 000001           1604#   1899    4375    4377    5466
BIT00 = 000001           1594#   1604
BIT01 = 000002           1593#   1603
BIT02 = 000004           1592#   1602
BIT03 = 000010           1591#   1601
BIT04 = 000020           1590#   1600
BIT05 = 000040           1589#   1599
BIT06 = 000100           1588#   1598
BIT07 = 000200           1587#   1597
BIT08 = 000400           1586#   1596    7068
BIT09 = 001000           1585#   1595    7076    7151
BIT1  = 000002           1603#   1900    1915    3368
BIT10 = 002000           1584#   1909    7129
BIT11 = 004000           1583#   1910    3329    3348    3535    3558    7083
```

DEFPBA.CMB CROSS REFERENCE TABLE -- USER SYMBOLS

```
BIT12 = 010000          1582#   3627
BIT13 = 020000          1581#   7136
BIT14 = 040000          1580#   1911    3515    3623    3625    3736    5732    5779    7054
BIT15 = 100000          1579#   1912    3170    3175    3184    3244    3283    3285    3288    3328    3347    3357    3450
                        3557    3577    3599    3622    3645    3670    3867    4383    5374
BIT2  = 000004          1602#   1901    1916    5467    5508    6086
BIT3  = 000010          1601#   1902    4199    4200    7349    7357
BIT4  = 000020          1600#   1903    3600    3603
BIT5  = 000040          1599#   1904    3317    3539    3578    3580    3658    3671    3674
BIT6  = 000100          1598#   1905    4078    5607    6207    6587    6883    6885    6928    6958    7310    7315
BIT7  = 000200          1597#   1906    3514    3649    3735    4212    4354    4366    4368    4423    4441    6890    7339
                        7344
BIT8  = 000400          1596#   1907
BIT9  = 001000          1595#   1908    3382    3646    3648    5716
BPTVEC= 000014          1611#
CACHSP  037222          2993    7914#
CACHVE= 000114          1619#   2993*   2994*
CHGFEC  034716          4660    4679    4697    4722    4758    4770    4782    4794    4806    4818    4835    4855    4883
                        4918    4944    4970    4983    5004    5039    5057    5075    5089    5104    5123    5146    5158
                        5178    5200    5222    7371#
CONTRL= 177746          1629#
COSETB= 172542          1928#   6936*
COUNT = 001160          1894#
COUNTE= 172544          1929#   6937*
CPSPUR  037156          2991    7052    7900#
CPUERR= 177766          1642#   2987    2988    7904    7905*
CR    = 000015          1516#   7497    7507
CRLF  = 000200          1517#   2985    7468    7507    8035    8043    8075    8126    8144    8178    8192    8264    8407
                        8478    8503    8586    8597    8742    8827    8847    8920    8979    8992
DDISP = 177570          1510#   1982    2962
DF1     037764          2223    2229    2260    2267    2273    2279    2285    2297    2309    2333    2339    2388    2394
                        2400    2406    2412    2418    2424    2430    2436    2460    2466    2472    2478    2484    2490
                        2496    2502    2508    2514    2520    2526    2538    2665    2677    2833    2839    8056#
DF26    042142          2352    2358    2364    2370    2376    2382    2448    2454    2551    2557    2563    2569    2575
                        2581    2587    2593    2599    2605    2863    8273#
DF64    044374          2532    2545    8490#
DH1     037665          2220    2227    2258    2265    2271    2277    2283    2295    2307    2331    2337    2386    2392
                        2398    2404    2410    2416    2422    2428    2434    2458    2464    2470    2476    2482    2488
                        2494    2500    2506    2512    2518    2524    2536    2663    2675    2831    2837    8043#
DH10    040542          8126#
DH101   045401          2609    2627    2651    2681    2693    2705    2723    2747    2771    2783    8586#
DH102   045462          2615    2633    2657    2687    2711    2729    2777    2825    8597#
DH123   046715          2717    8719#
DH126   047104          2735    2849    2855    8742#
DH127   047224          2741    8757#
DH14    041115          2289    8171#
DH140   050025          2795    8827#
DH142   050174          2807    8847#
DH153   051027          2861    8920#
DH154   051141          2867    8935#
DH155   051210          2873    2885    2909    8943#
DH156   051275          2879    8954#
DH160   051415          2891    8970#
DH161   051474          2897    8979#
DH162   051572          2903    8992#
DH26    042067          2349    2356    2362    2368    2374    2380    2446    2452    2549    2555    2561    2567    2573
```

```
                2579    2585    2591    2597    2603    8264#
DH3     040130  2233    2843    8075#
DH4     040231  2240    2246    2252    2301    2313    2319    2325    2343    2440    2621    2639    2645    2669
                2699    2753    2759    2765    2789    2801    2813    2819    8088#
DH64    044204  2530    8478#
DH66    044474  2542    8503#
DISPLA  001140  1982#   2962*   2968*   7098*   7128*
DISPRE  000174  1939#   2968
DSWR    177570  1509#   1981    2961
DT1     037752  2222    2228    2338    2387    2393    2399    2405    2411    2417    2423    2429    2435    2459
                2465    2477    2537    2664    2676    2832    2838    8054#
DT10    040622  2266    8136#
DT101   045450  2610    2628    2652    2682    2694    2706    2724    2748    2772    2784    8595#
DT102   045532  2616    2634    2658    2688    2712    2730    2778    2826    8605#
DT123   046742  2718    2742    8724#
DT13    041046  2284    2296    2308    2332    2507    2513    2525    8164#
DT14    041142  2290    8176#
DT140   050076  2796    8836#
DT153   051104  2862    8929#
DT154   051176  2868    8941#
DT155   051236  2874    2886    2910    8948#
DT156   051350  2880    8963#
DT160   051460  2892    8977#
DT161   051554  2898    8989#
DT26    042134  2351    2357    2363    2369    2375    2381    2447    2550    2556    2562    2568    2574    2580
                2586    2592    2598    2604    8272#
DT3     040200  2235    2736    2808    2844    2850    2856    2904    8083#
DT4     040250  2241    2247    2253    2302    2314    2320    2326    2344    2441    2622    2640    2646    2670
                2700    2754    2760    2766    2790    2802    2814    2820    8092#
DT47    043354  2453    8393#
DT64    044364  2531    8488#
DT66    044574  2544    8515#
DT7     040436  2259    2272    2278    2471    2483    2489    2495    2501    2519    8114#
EMTVEC= 000030  1614#   2945*   2946*
EM1     037633  2219    8038#
EM10    040450  2263    8116#
EM100   045310  2602    8574#
EM101   045335  2608    8578#
EM102   045357  2614    8582#
EM103   045544  2620    8607#
EM104   045603  2626    8613#
EM105   045653  2632    8620#
EM106   045702  2638    8624#
EM107   045751  2644    8631#
EM11    040634  2270    8138#
EM110   046022  2650    8638#
EM111   046075  2656    8646#
EM112   046124  2662    8650#
EM113   046160  2668    8655#
EM114   046247  2674    8665#
EM115   046302  2680    8670#
EM116   046350  2686    8677#
EM117   046376  2692    8681#
EM12    040677  2276    8144#
EM120   046444  2698    8688#
EM121   046543  2704    8699#
```

```
EM122   046601          2710    8705#
EM123   046637          2716    8711#
EM124   046752          2722    8726#
EM125   047007          2728    8731#
EM126   047044          2734    8736#
EM127   047147          2740    8749#
EM13    041031          2282    8160#
EM130   047251          2746    8761#
EM131   047307          2752    8767#
EM132   047374          2758    8776#
EM133   047466          2764    8786#
EM134   047570          2770    8797#
EM135   047635          2776    8804#
EM136   047666          2782    8809#
EM137   047733          2788    8816#
EM14    041060          2288    8166#
EM140   050001          2794    8823#
EM141   050110          2800    8838#
EM142   050154          2806    8844#
EM143   050243          2812    8854#
EM144   050335          2818    8864#
EM145   050376          2824    8870#
EM146   050426          2830    8874#
EM147   050505          2836    8882#
EM15    041152          2294    8178#
EM150   050564          2842    8890#
EM151   050631          2848    8897#
EM152   050714          2854    8906#
EM153   050777          2860    8915#
EM154   051114          2866    2872    8931#
EM156   051246          2878    2884    8950#
EM16    041266          2300    8192#
EM160   051366          2890    8966#
EM17    041407          2306    8207#
EM2     037771          2226    8058#
EM20    041453          2312    8214#
EM21    041506          2318    8219#
EM22    041542          2324    8224#
EM23    041576          2330    8229#
EM24    041647          2336    2896    2902    8236#
EM25    041664          2342    8239#
EM25A   041736          7653    8246#
EM25B   041751          7664    8248#
EM25C   041765          7683    8251#
EM26    042001          2348    8254#
EM27    042144          2355    8274#
EM3     040104          2232    8071#
EM30    042216          2361    8281#
EM31    042274          2367    8289#
EM32    042363          2373    8299#
EM33    042437          2379    8307#
EM34    042514          2385    8316#
EM35    042627          2391    8329#
EM36    042665          2397    8335#
EM37    042725          2403    8341#
EM4     040212          2239    2908    8085#
```

DEFPBA.CMB CROSS REFERENCE TABLE -- USER SYMBOLS

```
EM40    042756          2409    8346#
EM41    043026          2415    8353#
EM42    043067          2421    8359#
EM43    043115          2427    8363#
EM44    043142          2433    8367#
EM45    043204          2439    8373#
EM46    043252          2445    8380#
EM47    043313          2451    8386#
EM5     040256          2245    8093#
EM50    043362          2457    8394#
EM51    043415          2463    8399#
EM52    043467          2469    8407#
EM53    043574          2475    2499    2578    8420#
EM54    043610          2481    8422#
EM55    043654          2487    8428#
EM56    043725          2493    8435#
EM6     040305          2251    8097#
EM60    044014          2505    8445#
EM62    044107          2517    8455#
EM63    044153          2523    8462#
EM64    044247          2529    8473#
EM65    044400          2535    8492#
EM66    044424          2541    8496#
EM67    044606          2548    8517#
EM7     040355          2257    8104#
EM70    044651          2554    8523#
EM71    044721          2560    8530#
EM72    044755          2566    8535#
EM73    045017          2572    8541#
EM75    045061          2584    8547#
EM76    045143          2590    8556#
EM77    045225          2596    8565#
ERRVEC= 000004          1607#   2959    2960*   2971*   2986*   2991*   2992*   4623*   5389*   5427*   5497*   5542*   6912*
                        6916*   7052*   7059    7060*   7062*   7065*   7293*

FC    = 000001          1899#
FD    = 000200          1906#   3546    3570    3592    3615    3638    3663    3706    4006    4473    4503    4543    4571
                        4595    4629    4771    4783    5076    5090    5105    5926    7983    8005

FDZ   = 000004          1916#
FER   = 100000          1912#   4907    4933    4959    5030    5032
FIC   = 000400          1907#   5126    5147    5159
FICE  = 000006          1917#   5133    5135
FID   = 040000          1911#
FIU   = 002000          1909#   4885    4907    4920    4933    4946    4959    4972    4985
FIUV  = 004000          1910#   4726    4759    4771    4783    4795    4807    4820    4837    4857    5193    5510
FIV   = 001000          1908#   5007    5029    5041    5059    5076    5090    5105
FL    = 000100          1905#   5147    5926    6164    6490    6546
FLD20 = 104410          7644    7657    7968#
FL20  = 104407          7258    7634    7659    7967#
FMM   = 000020          1903#   3139    3151    3163    3224    3236    3261    3272    3321    3340    3373    3393    3546
                        3570    3592    3615    3638    3663    3778    3827    3844    3934    4190    4230    4280    4330
                        4473    4503    4543    4571    4595    5816    7375
FN    = 000010          1902#   3455    3457    4960    4963    6292    6295
FO    = 000010          1918#   5021    5023    5048    5050
FOCE  = 000002          1915#
FPPVEC= 000244          1924#   2989*   2990*   3137*   3152*   3221*   3258*   3315*   3336*   3369*   3389*   3542*   3566*
                        3588*   3611*   3634*   3659*   3779*   3818*   3930*   4186*   4226*   4276*   4326*   4468*   4499*
```

```
                        4542*  4567*  4591*  4720*  4756*  4768*  4780*  4792*  4804*  4816*  4833*  4853*  4881*
                        4916*  4942*  4968*  4981*  5005*  5037*  5055*  5068*  5085*  51C0*  5124*  5144*  5156*
                        5175*  5192*  5198*  5223*  5241*  5258*  5279*  5369*  5388*  55C3*  5541*  5566*  5813*
                        7053*  7371   7372*  7379*
FPSPUR  037276          2989   7053   7929#
FT    = 000040          1904#
FU    = 000012          1919#  4893   4895
FUV   = 000014          1920#  4733   4735
FV    = 000002          1900#  3906   3909   5030   5032
FZ    = 000004          1901#  3031   3033   3064   3066   3097   3099   3845   3848   3906   3909   4162   4165
                        4934   4937   6094   6431   6434   6750   6753
GETTIK  034324          5604   7293#
GNS   = ****** U        1938   2984   7004   7011   7303   7960   7961   7962   7963   7964   7965   7966   7967
                        7968
HIADRS= 177742          1627#  7920
HITMIS= 177752          1631#
HT    = 000011          1514#  7466   7507
ILLEND  002112          2193#  5252
ILLOP   002000          2156#  5242
IOTVEC= 000020          1612#  2943*  2944*
KDPAR0= 172360          1785#
KDPAR1= 172362          1786#
KDPAR2= 172364          1787#
KDPAR3= 172366          1788#
KDPAR4= 172370          1789#
KDPAR5= 172372          1790#
KDPAR6= 172374          1791#
KDPAR7= 172376          1792#
KDPDR0= 172320          1763#  5461*
KDPDR1= 172322          1764#
KDPDR2= 172324          1765#
KDPDR3= 172326          1766#  5506*
KDPDR4= 172330          1767#
KDPDR5= 172332          1768#
KDPDR6= 172334          1769#
KDPDR7= 172336          1770#  5462*
KERSTK= 001100          1500#
KIPAR0= 172340          1774#  5435
KIPAR1= 172342          1775#
KIPAR2= 172344          1776#
KIPAR3= 172346          1777#
KIPAR4= 172350          1778#
KIPAR5= 172352          1779#
KIPAR6= 172354          1780#
KIPAR7= 172356          1781#
KIPDR0= 172300          1752#  5452
KIPDR1= 172302          1753#
KIPDR2= 172304          1754#
KIPDR3= 172306          1755#
KIPDR4= 172310          1756#
KIPDR5= 172312          1757#
KIPDR6= 172314          1758#
KIPDR7= 172316          1759#
KSP   =%000006          1535#
LABLE1  023700          5835#
LABLE2  023744          5850#
```

```
LABLE3  024254          5922#
LABLE4  026170          6213#
LABLE5  030400          6596#
LCLK    032402          6920    6925#
LF    = 000012          1515#   7501    7507
LKS   = 177546          1511#
LKSRV   032476          6926    6943#
LKSTAT= 177546          1926#   5607*   6918    6928*   6950*   6958*   7294    7310*   7315*
LKVEC = 000100          1618#   1925#   5599*   6926*   6927*   7306    7309*   7314*
LOADRS= 177740          1626#   7919
LOOP    004456          2986    2989#   7026    7907    7909    7922    7924    7933
LOOPO   023630          5820#   6911    6929    6939
LOOP1   024254          5923#   6208
LOOP2   026174          6216#   6588
LOOP3   030476          6610#   6894
MAINT = 177750          1630#
MAPHO = 170202          1869#
MAPHOO= 170202          1805#   1869
M3PHO1= 170206          1807#   1871
MAPHO2= 170212          1809#   1873
MAPHO3= 170216          1811#   1875
MAPHO4= 170222          1813#   1877
MAPHO5= 170226          1815#   1879
MAPHO6= 170232          1817#   1881
MAPHO7= 170236          1819#   1883
MAPH1 = 170206          1871#
MAPH10= 170242          1821#
MAPH11= 170246          1823#
MAPH12= 170252          1825#
MAPH13= 170256          1827#
MAPH14= 170262          1829#
MAPH15= 170266          1831#
MAPH16= 170272          1833#
MAPH17= 170276          1835#
MAPH2 = 170212          1873#
MAPH20= 170302          1837#
MAPH21= 170306          1839#
MAPH22= 170312          1841#
MAPH23= 170316          1843#
MAPH24= 170320          1845#
MAPH25= 170326          1847#
MAPH26= 170332          1849#
MAPH27= 170336          1851#
MAPH3 = 170216          1875#
MAPH30= 170342          1853#
MAPH31= 170346          1855#
MAPH32= 170352          1857#
MAPH33= 170356          1859#
MAPH34= 170362          1861#
MAPH35= 170366          1863#
MAPH36= 170372          1865#
MAPH37= 170376          1867#
MAPH4 = 170222          1877#
MAPH5 = 170226          1879#
MAPH6 = 170232          1881#
MAPH7 = 170236          1883#
```

DEFPBA.CMB CROSS REFERENCE TABLE -- USER SYMBOLS

```
MAPL0 = 170200          1868#
MAPL00= 170200          1804#   1868
MAPL01= 170204          1806#   1870
MAPL02= 170210          1808#   1872
MAPL03= 170214          1810#   1874
MAPL04= 170220          1812#   1876
MAPL05= 170224          1814#   1878
MAPL06= 170230          1816#   1880
MAPL07= 170234          1818#   1882
MAPL1 = 170204          1870#
MAPL10= 170240          1820#
MAPL11= 170244          1822#
MAPL12= 170250          1824#
MAPL13= 170254          1826#
MAPL14= 170260          1828#
MAPL15= 170264          1830#
MAPL16= 170270          1832#
MAPL17= 170274          1834#
MAPL2 = 170210          1872#
MAPL20= 170300          1836#
MAPL21= 170304          1838#
MAPL22= 170310          1840#
MAPL23= 170314          1842#
MAPL24= 170320          1844#
MAPL25= 170324          1846#
MAPL26= 170330          1848#
MAPL27= 170334          1850#
MAPL3 = 170214          1874#
MAPL30= 170340          1852#
MAPL31= 170344          1854#
MAPL32= 170350          1856#
MAPL33= 170354          1858#
MAPL34= 170360          1860#
MAPL35= 170364          1862#
MAPL36= 170370          1864#
MAPL37= 170374          1866#
MAPL4 = 170220          1876#
MAPL5 = 170224          1878#
MAPL6 = 170230          1880#
MAPL7 = 170234          1882#
MEMERR= 177744          1628#   7918
MMR0  = 177572          1653#   1657    5428    5466*   5473*   5476*   5498    5509*   5515*   5518*   5523*
MMR1  = 177574          1654#   1658
MMR2  = 177576          1655#   1659
MMR3  = 172516          1656#   1660    5467*   5472*   5475*   5508*   5519*
MMVEC = 000250          1621#   5463*   5502*
MSG1    037614          7601    8035#
MSN   = 170004          1893#   3172    3241    3280    3354    4207    4236    4241    4243    4286    4336    4338    4509
                        4549    4577    4601
MT    = 000016          1921#   4666    4668    4687    4700
MULSHF  034516          7326#   7666
MULTBL  001400          2060#   7347
OFFSET= 001164          1896#
OUT     032606          5812*   6947*   6948    6951*   6954*   6957    6960#   6968*   6969    6972*   6975*   6978
OUTPTR  001376          2059#   7326*   7354*   7355*   7356*   7363*
OUTTBL  054532          2059    7326    7669    7684    9021#
```

DEFPBA.CMB CROSS REFERENCE TABLE -- USER SYMBOLS

```
PC     =%000007          1538#  3083   4631   4660*  4679*  4681*  4697*  4722*  4758*  4770*  4782*  4794*  4806*
                         4818*  4835*  4838*  4855*  4858*  4883*  4918*  4944*  4970*  4983*  5004*  5039*  5057*
                         5075*  5089*  5091*  5104*  5123*  5146*  5158*  5178*  5200*  5222*  5468*  5555   5559
                         5562   5590   5604*  5605   5852   5862   5873*  5889*  5904   6219   6236*  6252*  6268*
                         6283   6301*  6317*  6335   6352   6370   6387   6405   6422   6441*  6458   6476*  6497*
                         6514*  6530   6552   6569   6996*  6999*  7021*  7138*  7144*  7316*  7364*  7380*  7456*
                         7475*  7482*  7489*  7503*  7505*  7539*  7556*  7666*  7675*  7688*  7707*  7885*  8025
PCLK     032432          6915   6933#
PCLKST=  172540          1927#  6913   6938*  6971*
PIRQ   = 177772          1508#  5374*  5381*  5387*
PIRQVE=  000240          1620#  5370*
PKSRV    032610          6934   6964#
PKVEC  = 000104          1930#  6934*  6935*
PR0    = 000000          1541#
PR1    = 000040          1542#
PR2    = 000100          1543#
PR3    = 000140          1544#
PR4    = 000200          1545#
PR5    = 000240          1546#
PR6    = 000300          1547#
PR7    = 000340          1548#  2990   2992   2994   6927   6935
PS     = 177776          1505#  1506
PSW    = 177776          1506#  5625*  5628*  7307   7308*  7313*
PWRVEC=  000024          1613#  2949*  2950*  7974*  7975*  7994*  8000*  8021*  8022*
RESREG=  104406          7231   7283   7667   7884   7966#
RESVEC=  000010          1608#  4677*  5849*  5860*  5870*  5886*  5902*  6216*
R0     =%000000          1520#  1528   3050*  3052   3054   3162*  3163   3235*  3236   3271*  3272   3421*  3427
                         3430   3449*  3455   3458   3548*  3549   3552   3783*  3784   3785   3829*  3844*  3845
                         3847   3899*  3906   3908   3968*  3970   3971   4087*  4088*  4089*  4090*  4091*  4092*
                         4093*  4094*  4095*  4119*  4120*  4121*  4122*  4123*  4124*  4125*  4126*  4127*  4128*
                         4129*  4148*  4150   4152   4324*  4358   4375*  4376*  4377*  4425*  4446*  4472*  4475
                         4478   4661*  4662*  4663   4698*  4699*  4700   4702   4891*  4907*  4908   4911   4926*
                         4933*  4934   4936   4952*  4959*  4960   4962   5127*  5148*  5242*  5244   5251*  5252
                         5435*  5436*  5437*  5438*  5439*  5440*  5441*  5442*  5443*  5444*  5445*  5446*  5447*
                         5448*  5449*  5450*  5451*  5452*  5455*  5459*  5562*  5564   5650*  5651*  5652   5655
                         5656   5671*  5675*  5677*  5678   5679*  5680*  5681*  5682*  5685*  5686   5688   5690
                         5691   5692   5693   5695   5698   5701   5704   5708   5712   5720*  5721   5723*  5725
                         5729   5731*  5732*  5734*  5735*  5736*  5737*  5743   5746*  5747*  5748*  5749   5752
                         5754*  5755   5757   5760   5764*  5765   5769*  5770*  5771*  5772*  5773   5774*  5776*
                         5777   5778*  5779*  5834*  5837   5839   5935*  5936   5945*  5948*  5949   5951   5973*
                         5974*  5976   5979   6091*  6092*  6093   6124*  6127*  6128   6131   6225*  6226   6229
                         6289*  6291*  6292   6294   6428*  6430*  6431   6433   6612*  6615   6617*  6619   6624*
                         6626*  6628   6639*  6641*  6642   6651*  6653   6655   6666*  6667*  6669*  6671   6680*
                         6681*  6683*  6685   6694*  6697   6699   6710*  6714   6715   6727*  6730   6731   6741*
                         6744   6745   6748*  6749*  6750   6752   6757*  6758*  6761*  6762   6771*  6772*  6775
                         6776   6787*  6790*  6791   6801*  6802*  6803*  6806*  6807   6818*  6820*  6822*  6823
                         6831*  6832*  6835   6836   6847*  6852   6854   6865*  6867*  6869   6870   6879*  6880*
                         6881   7018*  7021   7182*  7184   7185*  7193*  7200*  7206*  7228*  7229*  7230   7260*
                         7261   7262*  7263*  7273*  7279*  7280*  7331*  7335*  7338*  7339*  7358*  7371*  7379
                         7400   7425*  7449   7450*  7455   7460   7463*  7516   7524*  7528   7529   7531*  7532*
                         7533   7555*  7575   7576*  7577*  7584*  7585*  7586*  7587*  7588*  7589   7594   7597
                         7613   7615*  7616   7618   7621   7625*  7632*  7642*  7668*  7673   7684*  7686   7693
                         7705*  7803   7813*  7817   7833   7834   7847*  7874*  7878   7888*  7890*  7892*  7943
                         7944*  7945   7946*  7947*  7948*  7949*  7976   7982*  7984   8020*
R1     =%000001          1521#  1529   4100*  4123   4128   4151*  4162   4164   4325*  4379*  4385*  4423*  4438
                         4444*  5453*  5455   5458*  5459   5672*  5676*  5768*  5775*  5937*  5938   5940   6609*
```

DEFPBA.CMB CROSS REFERENCE TABLE -- USER SYMBOLS

```
                    6642    6655    6671    6685    6699    6715    6731    6745    6762    6776    6888*   6892*   7184*
                    7189    7195    7202    7208    7214*   7216    7220*   7222    7264*   7265*   7266    7267    7268*
                    7282*   7332*   7336*   7340    7401    7424*   7517    7554*   7596    7597*   7599    7608*   7612
                    7613*   7627    7633    7643    7669*   7680    7697    7704*   7804    7817*   7818    7822    7846*
                    7870*   7874    7875*   7887*   7889*   7891*   7977    8019*
R10     =%000000    1528#
R11     =%000001    1529#
R12     =%000002    1530#
R13     =%000003    1531#
R14     =%000004    1532#
R15     =%000005    1533#
R2      =%000002    1522#   1530    4323*   4387*   5454*   5456*   5457*   5460*   5678*   5721    5727    5728*   5777*
                    6184*   6186*   6188    6191    6613*   6614    6616    6654*   6659*   6661    7266*   7274*   7281*
                    7340*   7341*   7344*   7345    7347*   7348*   7349    7351    7354    7357*   7358    7359*   7360
                    7402    7423*   7805    7816*   7820*   7823    7830*   7831*   7832    7837*   7845*   7876*   7877
                    7878*   7893*   7894*   7978    8018*
R3      =%000003    1523#   1531    3135*   3149*   3222*   3259*   3319*   3338*   3371*   3391*   3544*   3568*   3590*
                    3613*   3636*   3661*   3776*   3816*   3932*   4188*   4228*   4278*   4328*   4470*   4501*   4540*
                    4569*   4593*   5743*   5814*   6170*   6172    6608*   6823    6870    6889*   6893*   7048    7049*
                    7051*   7120    7121*   7122    7123*   7125*   7189*   7190*   7191*   7192    7195*   7196*   7197*
                    7198    7202*   7203*   7204*   7205    7208*   7209*   7210*   7211    7216*   7217*   7218*   7219
                    7222*   7223*   7224*   7225    7267*   7270    7276    7373*   7403    7422*   7745    7754*   7760*
                    7761*   7764*   7769*   7770*   7771    7780*   7806    7814*   7815*   7829*   7832*   7841*   7842*
                    7844*   7873*   7886*   7893    7979    8017*
R4      =%000004    1524#   1532    3171*   3240*   3279*   3353*   4206*   4235*   4240*   4242*   4285*   4335*   4337*
                    4508*   4548*   4576*   4600*   5923*   5946    5956    5961    5975    5986    5998    6013    6026
                    6064    6076    6098    6106    6111    6116    6136    6144    6149    6151*   6153    6157    6163
                    6176    6182    6187    6201    6214*   6215    6228    6242    6258    6274    6290    6307    6323
                    6341    6358    6376    6393    6411    6429    6447    6464    6482    6503    6520    6536    6558
                    6575    6618    6627    6645    6658    6670    6679    6684    6698    7186*   7187*   7192*   7198*
                    7199*   7205*   7211*   7212*   7213*   7219*   7225*   7226*   7227*   7230*   7269*   7275*   7333*
                    7337*   7342    7351*   7353*   7404    7421*   7746    7748*   7749*   7750*   7751    7752*   7766
                    7768*   7776*   7779*   7872*   7879*   7980    8016*
R5      =%000005    1525#   1533    6607*   6807    6810    6814*   6849*   6853*   6854    6884*   6887*   7188*   7194*
                    7201*   7207*   7215*   7221*   7270*   7271*   7272*   7273    7276*   7277*   7278*   7279    7328*
                    7360*   7405    7420*   7747    7753*   7755*   7757*   7758*   7759*   7760    7778*   7807    7809*
                    7811*   7818*   7822*   7837    7843*   7871*   7877*   7882*   7883    7981    8015*
R6      =%000006    1526#   1534    2937*   2938*   2939    5283    5295*   5296    5298    5307*   5308    5309*   5320*
                    5323    5333    5346
R7      =%000007    1527#   1538
SAVREG= 104405      7181    7255    7665    7869    7965#
SDPAR0= 172260      1741#
SDPAR1= 172262      1742#
SDPAR2= 172264      1743#
SDPAR3= 172266      1744#
SDPAR4= 172270      1745#
SDPAR5= 172272      1746#
SDPAR6= 172274      1747#
SDPAR7= 172276      1748#
SDPDR0= 172220      1719#
SDPDR1= 172222      1720#
SDPDR2= 172224      1721#
SDPDR3= 172226      1722#
SDPDR4= 172230      1723#
SDPDR5= 172232      1724#
SDPDR6= 172234      1725#
```

```
SDPDR7= 172236          1726#
SIPAR0= 172240          1730#
SIPAR1= 172242          1731#
SIPAR2= 172244          1732#
SIPAR3= 172246          1733#
SIPAR4= 172250          1734#
SIPAR5= 172252          1735#
SIPAR6= 172254          1736#
SIPAR7= 172256          1737#
SIPDR0= 172200          1708#
SIPDR1= 172202          1709#
SIPDR2= 172204          1710#
SIPDR3= 172206          1711#
SIPDR4= 172210          1712#
SIPDR5= 172212          1713#
SIPDR6= 172214          1714#
SIPDR7= 172216          1715#
SIZEHI= 177762          1639#
SIZELO= 177760          1637#
SP     =%000006         1534#  1535   1536   1537   2941*  2959*  2969*  2971   3143   3167   3239   3278   3325
                        3344   3379   3397   3551*  3574*  3596*  3619*  3642*  3667*  3800   3850   3943   4194
                        4234   4284   4334   4477   4507   4547   4575   4599   4643   4690   4731   4764   4776
                        4788   4800   4812   4825   4844   4863   4890   4925   4951   4977   4990   5012   5046
                        5064   5081   5096   5110   5131   5152   5164   5181   5196   5203   5229   5248   5255*
                        5262   5282*  5300*  5321   5325*  5332*  5334*  5345*  5352*  5353*  5354   5355*  5371*
                        5378   5380*  5386*  5391*  5393*  5395   5397   5399*  5402   5404*  5406*  5430*  5477
                        5500   5504*  5516   5520   5522*  5525*  5551   5557   5569   5606   5819*  5858   5868
                        5879   5895   5910   6589   6590   6917*  7006*  7013*  7048*  7051   7059*  7062   7064
                        7065   7094   7095   7099*  7120*  7125   7133   7153*  7156*  7182   7183*  7232*  7233*
                        7234*  7256   7257*  7260   7284*  7285*  7286*  7296*  7306*  7307*  7312   7313   7314
                        7378   7400*  7401*  7402*  7403*  7404*  7405*  7406*  7407*  7408*  7409*  7415*  7417*
                        7418*  7419*  7420   7421   7422   7423   7424   7425   7449*  7450   7460*  7462   7463
                        7464*  7466   7468   7470   7476   7478*  7480*  7488*  7492   7496   7497   7501   7516*
                        7517*  7524   7525*  7536   7537*  7538*  7548   7549*  7554   7555   7575*  7580*  7596*
                        7602*  7605*  7608   7612*  7627*  7636   7646   7660   7674*  7676*  7677   7687*  7689*
                        7690   7704   7705   7737*  7738   7739   7740*  7745*  7746*  7747*  7753   7778   7779
                        7780   7781*  7782*  7803*  7804*  7805*  7806*  7807*  7808*  7809   7812*  7825   7827*
                        7829   7839   7841   7843   7844   7845   7846   7847   7849*  7850*  7870   7883*  7900
                        7901*  7914   7915*  7929   7930*  7943*  7944   7976*  7977*  7978*  7979*  7980*  7981*
                        7984*  7985*  7986*  7987*  7988*  7990*  7992*  7993   8001*  8006   8008   8010   8011
                        8012   8013   8014   8015   8016   8017   8018   8019   8020   8025*
SR0    = 177572         1657#
SR1    = 177574         1658#
SR2    = 177576         1659#
SR3    = 172516         1660#
SSP    =%000006         1536#
STACK  = 001100         1499#  1500   1501   1502   2941   5282   5300   5309   5325   5334   5345   5355   5399
                        5404   5406   5430   5504   5522   5525   5819   6917   7296   7901   7915   7930
START    004036         1943   2935#
STARTC   032312         6882   6908#
STEP   = 001162         1895#
STKLMT= 177774          1507#
SUPSTK= 000700          1501#
SWR      001136         1981#  2961*  2963   2967*  2975*  3147   3219   3256   3491   3510   3712   3731   3821
                        4070   4115   4344   4433   5432   5495   5716   6957*  6978*  7046   7049   7054   7068
                        7070   7076   7083   7118   7123   7129   7136   7148   7151
```

```
SWREG   000176          1940#   2967
SW0   = 000001          1576#
SW00  = 000001          1566#   1576
SW01  = 000002          1565#   1575
SW02  = 000004          1564#   1574
SW03  = 000010          1563#   1573
SW04  = 000020          1562#   1572
SW05  = 000040          1561#   1571
SW06  = 000100          1560#   1570
SW07  = 000200          1559#   1569
SW08  = 000400          1558#   1568
SW09  = 001000          1557#   1567
SW1   = 000002          1575#
SW10  = 002000          1556#
SW11  = 004000          1555#
SW12  = 010000          1554#   5432    5495
SW13  = 020000          1553#
SW14  = 040000          1552#
SW15  = 100000          1551#
SW2   = 000004          1574#
SW3   = 000010          1573#
SW4   = 000020          1572#
SW5   = 000040          1571#
SW6   = 000100          1570#
SW7   = 000200          1569#
SW8   = 000400          1568#   7046    7118
SW9   = 001000          1567#   3147    3219    3256    3491    3510    3712    3731    3821    4070    4115    4344    4433
SYSTID= 177764          1641#
TBITVE= 000014          1609#
TEMP    034750          7372    7378#
TICKS   001366          2055#   5811*   6909    6914*   6919*   6943*   6944    6946*   6952*   6964*   6965    6967*   6973*
TIMES   001370          2056#   6948    6969
TKVEC = 000060          1616#
TPVEC = 000064          1617#
TRAPVE= 000034          1615#   2947*   2948*
TRTVEC= 000014          1610#
TST1    004522          3008#
TST10   006622          3415    3428    3441#
TST11   006734          3443    3456    3478#
TST12   007144          3480    3531#
TST13   010166          3533    3672    3697#
TST14   010402          3748#
TST15   010454          3750    3757    3771#
TST16   010632          3773    3794    3811#
TST17   011052          3813    3846    3861#
TST2    004652          3010    3032    3045#
TST20   011174          3863    3878    3891#
TST21   011306          3893    3907    3925#
TST22   011422          3927    3939    3954#
TST23   011636          3956    3985    3998#
TST24   012034          4000    4024    4041#
TST25   012512          4043    4140#
TST26   012646          4142    4163    4177#
TST27   013442          4179    4293    4315#
TST3    004770          3047    3065    3078#
TST30   014120          4317    4389    4407#
```

```
TST31   014362          4447    4461#
TST32   014724          4463    4515    4535#
TST33   015320          4537    4606    4620#
TST34   015450          4622    4640    4653#
TST35   015666          4655    4701    4714#
TST36   016542          4716    4865    4876#
TST37   017312          4878    4999#
TST4    005076          3080    3098    3129#
TST40   020036          5001    5118#
TST41   020244          5120    5172#
TST42   020464          5174    5206    5219#
TST43   020736          5221    5265    5276#
TST44   021304          5278    5349    5365#
TST45   021532          5367    5408    5424#
TST46   022046          5426    5431    5433    5474    5492#
TST47   022252          5494    5496    5501    5538#
TST5    005452          3131    3192    3215#
TST50   022504          5540    5572    5582#
TST51   022660          5585    5611    5621#
TST52   022740          5623    5630    5640#
TST53   023034          5642    5653    5667#
TST54   023524          5669    5766    5798#
TST6    006060          3217    3289    3311#
TST7    006520          3401    3413#
TYPDS = 104404          7008    7015    7964#
TYPE  = 104400          2982    7002    7009    7016    7131    7139    7301    7471    7574    7591    7593    7601    7604
                        7607    7609    7611    7637    7647    7652    7653    7654    7661    7663    7664    7672    7678
                        7682    7683    7685    7691    7695    7699    7706    7772    7848    7960#   8023
TYPOC = 104401          7582    7603    7606    7628    7961#
TYPON = 104403          7963#
TYPOS = 104402          7962#
UDPAR0= 177660          1697#
UDPAR1= 177662          1698#
UDPAR2= 177664          1699#
UDPAR3= 177666          1700#
UDPAR4= 177670          1701#
UDPAR5= 177672          1702#
UDPAR6= 177674          1703#
UDPAR7= 177676          1704#
UDPDR0= 177620          1675#
UDPDR1= 177622          1676#
UDPDR2= 177624          1677#
UDPDR3= 177626          1678#
UDPDR4= 177630          1679#
UDPDR5= 177632          1680#
UDPDR6= 177634          1681#
UDPDR7= 177636          1682#
UIPAR0= 177640          1686#
UIPAR1= 177642          1687#
UIPAR2= 177644          1688#
UIPAR3= 177646          1689#
UIPAR4= 177650          1690#
UIPAR5= 177652          1691#
UIPAR6= 177654          1692#
UIPAR7= 177656          1693#
UIPDR0= 177600          1664#
```

```
UIPDR1= 177602          1665#
UIPDR2= 177604          1666#
UIPDR3= 177606          1667#
UIPDR4= 177610          1668#
UIPDR5= 177612          1669#
UIPDR6= 177614          1670#
UIPDR7= 177616          1671#
USESTK= 000600          1502#
USP   =%000006          1537#
WALKBU  002112          2194#   5671    5677    5685    5720    5746    5764    5776
X14   = 051641          6033#   6034    8999#   9000
$ACO    002162          2195#   5725
$APTHD  004022          2922    2928#
$ASTAT= ****** U        7542    7557
$ATYC   035362          7513    7515#
$ATY1   035336          7511#
$ATY3   035344          7456    7512#
$ATY4   035354          7144    7514#
$BDADR  001122          1975#
$BDDAT  001126          1977#   2939
$BELL   001224          2011#   7131    7163
$BUFF   001352          2053#   7261#   7286
$CHARC  035332          7473#   7483#   7490    7499#   7504#
$CMTAG  001100          1963#   2936    2937    2945    2951    2952    2953
$CM1  = 000010          1993#   1994#   1995#   1996#   1997#   1998#   1999#   2000#   2001#
$CM2  = 000020          1993#   1994#   1995#   1996#   1997#   1998#   1999#   2000#   2001#
$CM3  = 000010          1991#   1993
$CM4  = 000010          2001#   2002#   2003#   2004#   2005#   2006#   2007#   2008#   2009#
$CPUER  001372          2057#   2988#   7902    7916
$CPUOP  001262          2035#
$CRLF   001231          2013#   7016    7139    7163    7472    7507    7574    7593    7607    7611    7652    7663    7682
                        7695    7706
$DBLK   037026          7814    7848    7856#
$DB20   037036          7675    7688    7869#
$DEVCT  001244          2026#
$DOAGN  033112          6998    7019    7025#
$DTBL   037016          7817    7852#
$ENDAD  033102          1951    2980    7021#   7158
$ENDCT  032746          2951    7000#
$ENULL  033116          7027#
$ENV    001254          2031#   7141    7451    7520    7544
$ENVM   001255          2032#   2973    7453    7458    7522
$EOP    032712          5802    6921    6953    6974    6990#
$EOPCT  032740          2951#   6997#   7001
$ERFLG  001103          1966#   3154    3226    3263    3493    3512    3714    3733    3823    4072    4117    4346    4435
                        5249    5718    6955    6976    7035    7072    7074    7080#   7102    7126#   7163
$ERMAX  001115          1972#   2954#   7074    7097#   7102
$ERPSW  001264          2044#
$ERROR  033442          2945    7117#
$ERRPC  001116          1973#   7133#   7134#   7135    7163    7580    7602    8054    8083    8092    8114    8136    8164
                        8176    8595    8605    8724    8836    8941    8948    8963    8977    8989
$ERRTB  002172          2214#   7588
$ERRTY  035604          7138    7573#
$ERTTL  001112          1970#   7013    7017#   7132#   7163
$ESCAP  001222          2010#   2953#   3010#   3047#   3080#   3131#   3217#   3415#   3443#   3480#   3533#   3750#   3773#
                        3813#   3863#   3893#   3927#   3956#   4000#   4043#   4142#   4179#   4317#   4463#   4537#   4622#
```

```
                        4655*   4716*   4878*   5001*   5120*   5174*   5221*   5278*   5367*   5426*   5494*   5540*   5585*
                        5623*   5642*   5669*   5802*   7096*   7154    7156    7163    7305    8026
$ETABL  001254          2030#
$ETEND  001264          2042#   2934
$FATAL  001236          2023#   7548*
$FD     001276          2050#   3482*   3699*   7329    7655
$FEA    001272          2047#
$FEC    001270          2046#
$FFLG   035602          7511*   7514*   7542    7551*   7559#
$FILLC  001154          1989#   7476    7507
$FILLS  001153          1988#   7507
$FL     001275          2049#
$FLBUF  001310          2052#   7186    7234
$FLD20  034152          7255#   7968
$FLT    001300          2051#
$FL20   033664          7181#   7967
$FPS    001364          2054#   5921*   5924*   5925    5928    5935    5938    5941    5947    5959    5971    5984    5996
                        6011    6024    6062    6074    6086*   6087    6101    6114    6126    6139    6152    6164    6171
                        6185    6196    6205    6207*   6212*   6218    6232*   6235    6243    6246    6248*   6251    6267
                        6282    6300    6316    6334    6351    6369    6386    6404    6421    6440    6457    6475    6490
                        6496    6513    6529    6546    6551    6568    6584*   6585    6587*   6595*   6613    6625    6640
                        6652    6668    6682    6696    6713    6729    6743    6760    6774    6789    6805    6821    6834
                        6851    6868    6879    6883*   6885    6890
$FT     001274          2048#
$GDADR  001120          1974#
$GDDAT  001124          1976#
$GET42  033072          7018#
$HD   = 000000          1466
$HIBTS  004022          2929#
$ICNT   001104          1967#   7087*   7088    7090*   7101
$ILLUP  037576          7974    8000    8028#
$ITEMB  001114          1971#   7135*   7143    7163    7577    7702
$LF     001232          2014#   7163    7507
$LFLG   035601          7552*   7558#
$LPADR  001106          1968#   2955*   5464*   7078*   7094*   7099    7101
$LPERR  001110          1969#   2956*   3158*   3231*   3267*   3318*   3337*   3370*   3390*   3485*   3504*   3543*   3567*
                        3589*   3612*   3635*   3660*   3705*   3724*   3780*   3819*   3894*   3931*   3965*   4005*   4056*
                        4101*   4146*   4187*   4227*   4277*   4327*   4412*   4427*   4469*   4500*   4568*   4592*   4628*
                        4659*   4678*   4696*   4721*   4757*   4769*   4781*   4793*   4805*   4817*   4834*   4854*   4882*
                        4917*   4943*   4969*   4982*   5038*   5056*   5074*   5088*   5103*   5145*   5157*   5177*   5199*
                        5243*   5259*   5293*   5306*   5318*   5331*   5344*   5372*   5390*   5465*   5507*   5567*   5588*
                        5601*   5649*   5683*   5927*   5944*   5955*   5969*   5982*   5994*   6006*   6021*   6056*   6071*
                        6083*   6097*   6110*   6123*   6135*   6148*   6162*   6181*   6194*   6217*   6233*   6249*   6265*
                        6281*   6298*   6314*   6330*   6349*   6366*   6384*   6401*   6419*   6437*   6454*   6472*   6489*
                        6510*   6527*   6544*   6566*   6583*   6611*   6623*   6637*   6650*   6665*   6678*   6692*   6709*
                        6725*   6740*   6756*   6770*   6786*   6800*   6817*   6830*   6846*   6864*   7078    7095*   7101
                        7153
$MAIL   001234          2021#   2930    2934    2972    3009    3046    3079    3130    3216    3312    3414    3442    3479
                        3532    3698    3749    3772    3812    3862    3892    3926    3955    3999    4042    4141    4178
                        4316    4408    4462    4536    4621    4654    4715    4877    5000    5119    5173    5220    5277
                        5366    5425    5493    5539    5584    5622    5641    5668    5800    7093    7141    7451
$MBADR  004024          2930#
$MFLG   035600          7512*   7518    7553*   7557#
$MSGAD  001250          2028#   7528*   7531
$MSGLG  001252          2029#   7533*
$MSGTY  001234          2022#   7526    7534*   7546    7550*
```

```
$MXCNT  033440          7091    7101#
$NULL   001152          1987#   7478    7507
$NWTST= 000001          2998#   3000    3037#   3039    3070#   3072    3103#   3105    3203#   3205    3296#   3298    3405#
                        3407    3434#   3436    3462#   3464    3519#   3521    3681#   3683    3739#   3741    3762#   3764
                        3803#   3805    3853#   3855    3883#   3885    3912#   3914    3946#   3948    3990#   3992    4028#
                        4030    4132#   4134    4168#   4170    4299#   4301    4392#   4394    4452#   4454    4524#   4526
                        4612#   4614    4646#   4648    4705#   4707    4870#   4872    4992#   4994    5112#   5114    5166#
                        5168    5210#   5212    5269#   5271    5358#   5360    5411#   5413    5482#   5484    5527#   5529
                        5576#   5578    5615#   5617    5634#   5636    5661#   5663    5795#
$OCNT   036606          7744*   7773*   7786#
$OCTVL  037140          7871    7896#
$OMODE  036610          7739*   7743*   7748    7751*   7762*   7788#
$OVER   033424          7055    7071    7079    7089    7098#
$PASS   001242          2025#   2972*   6994*   6995*   7006    7027    7085    7102    7297
$PASTM  0040[illegible] 2932#
$POWER  037604          8024    8031#
$PWRAD  037572          8026#
$PWRDN  037364          2949    7974#   8021
$PWRMG  037566          8024#
$PWRUP  037462          7994    8000#
$QUES   001230          2012#   7163    7507
$RDCHR= ****** U        7965
$RDDEC= ****** U        7965
$RDLIN= ****** U        7965
$RDOCT= ****** U        7965
$REGAD  001156          1991#
$REGO   001160          1894    1993#   3133*   3140    3157*   3160    3229*   3233    3269    3316*   3323    3342    3375
                        3395    3483*   3487    3488    3502*   3506    3507    3538*   3548    3572    3594    3617    3640
                        3665    3701*   3708    3709    3727    3728    3774*   3782    3783    3814*   3828    3866*   3869
                        3928*   3936    3957*   3967    4001*   4008    4044*   4058    4077*   4078*   4087    4103    4144*
                        4148    4184*   4192    4225*   4232    4275*   4282    4318*   4332    4348*   4349    4355    4358
                        4382*   4464*   4472    4498*   4505    4719*   4727    4760    4772    4784    4796    4808    4819*
                        4821*   4826    4848*   4866*   5069*   5073    5087    5102    6601*   6655    6699    6715    6731
                        6745    6776    7335    7657    7659    8836
$REG1   001162          1895    1994#   3134*   3230*   3317*   3335*   3368*   3388*   3484*   3495*   3503*   3514*   3515
                        3539*   3657*   3702*   3775*   3815*   3867*   3929*   3958*   4002*   4045*   4076*   4145*   4185*
                        4319*   4383*   4465*   5070*   6602*   7336
$REG2   001164          1896    1995#   3540*   3703*   3785    3959*   4003*   4046*   4075*   4152    4466*   5071*   6603*
                        6642    6671    6685    6762    6823    6870    7331
$REG3   001166          1996#   3541*   3565*   3587*   3610*   3633*   3658*   3704*   3716*   3723*   3735*   3736    3960*
                        4004*   4047*   4074*   4467*   6604*   7332
$REG4   001170          1997#   3552    3961*   3968    4052*   4105    4478    6605*   6807    6854    8393
$REG5   001172          1998#   3218*   3962*   4053*   4320*   4378*   4386*   4426*   4437*   5749*   5750*   5751*   6606*
                        8176    8488    8515    8929
$REG6   001174          1999#   3963*   4054*   4424*   4445*   8515
$REG7   001176          2000#   3964*   3971    4055*   4119
$RESRE  035016          7415#   7966
$R2A  = ****** U        7967
$SAVRE  034760          7399#   7965
$SAVR6  037602          7993*   8001    8002*   8003*   8030#
$SCOPE  033122          2943    7044#
$SETUP= 000037          2215#   2942    2943    2945    2947    2949    2951    2952    2953    2955    2980    6992    7158
$STUP = 177777          2215#
$SVLAD  033370          7063    7092#
$SVPC = 000204          1949#   1954
$SWR  = 167777          1437#   1466    1486    1487    1488    1489    1490    1491    1492    1493    2009    2010    2011
```

```
                    2952    2953    2955    2956    3009    3046    3079    3130    3216    3312    3414    3442    3479
                    3532    3698    3749    3772    3812    3862    3892    3926    3955    3999    4042    4141    4178
                    4316    4408    4462    4536    4621    4654    4715    4877    5000    5119    5173    5220    5277
                    5366    5425    5493    5539    5583    5622    5641    5668    5799    6987    6993    7020    7026
                    7027    7036    7037    7038    7039    7040    7054    7066    7068    7069    7072    7073    7074
                    7081    7082    7083    7095    7098    7101    7109    7110    7111    7112    7113    7129    7136
                    7148    7151    7163    8027
$SWREG  001256      2033#   2975
$SWRMK= 000000      1493    1494    7040    7041    7070
$TESTN  001240      2024#   3009*   3046*   3079*   3130*   3216*   3312*   3414*   3442*   3479*   3532*   3698*   3749*
                    3772*   3812*   3862*   3892*   3926*   3955*   3999*   4042*   4141*   4178*   4316*   4408*   4462*
                    4536*   4621*   4654*   4715*   4877*   5000*   5119*   5173*   5220*   5277*   5366*   5425*   5493*
                    5539*   5584*   5622*   5641*   5668*   5800*   7093*
$TIMES  001220      2009#   2952*   5583*   5799*   6993*   7081*   7088    7091*   7101
$TKB    001144      1984#
$TKS    001142      1983#
$TMPO   001200      2001#   3021*   3027*   3033*   3060*   3066*   3094*   3099*   3176*   3416*   3420    3423    3429*
                    3444*   3448    3450*   3451    3457*   3534*   3582*   3602*   3626*   3651*   3673*   3759*   3791*
                    3796*   3834*   3840*   3848*   3874*   3880*   3904*   3909*   3941*   3974*   3978    3987*   4012*
                    4018*   4022    4048*   4060    4083*   4096*   4110    4160*   4165*   4198*   4249*   4322*   4409*
                    4414    4421*   4428    4430    4438*   4439    4441*   4442*   4484*   4553*   4624*   4638    4666*
                    4735*   4828*   4847*   4867*   4893*   4904*   4910*   4930*   4937*   4956*   4963*   5017*   5023*
                    5032*   5050*   5135*   5160*   5185*   5207*   5227*   5233*   5256*   5266*   5287*   5299*   5312*
                    5324*   5338*   5350*   5354*   5403*   5564*   5573*   5586*   5592    5598*   5609    5624*   5654*
                    5761*   5820*   5826    5834    5844*   5883*   5899*   5915*   5925*   5926*   5932    5941*   5952*
                    5957*   5965*   5978*   5991*   6003*   6008*   6010    6018*   6023    6031*   6094*   6117    6130*
                    6159*   6178*   6190*   6202*   6230*   6246*   6261*   6277*   6295*   6311*   6326*   6331*   6333
                    6346*   6350    6362*   6367*   6368    6381*   6397*   6402*   6403    6416*   6434*   6450*   6455*
                    6456    6469*   6485*   6506*   6523*   6540*   6562*   6579*   6590*   6597*   6633*   6651    6662*
                    6693*   6694    6705*   6710    6721*   6726*   6727    6741    6742    6753*   6759    6771    6783*
                    6796*   6813*   6814*   6843*   6861*   6877*   7670*   7673*   7674    7686*   7687    7900*   7914*
                    7929*   8054    8083    8114    8136    8272    8488    8515    8595    8724    8836    8929    8941
                    8948    8963    8977    8989
$TMP1   001202      2002#   3016*   3028*   3031    3053*   3064    3089*   3097    3177*   3417*   3430*   3445*   3458*
                    3535*   3580*   3603*   3625*   3648*   3674*   3797*   3841*   3847*   3865*   3908*   3975*   4013*
                    4019*   4049*   4082*   4097*   4164*   4199*   4212*   4250*   4321*   4342    4351*   4352    4354*
                    4357*   4360    4362    4364    4366*   4367*   4368*   4370*   4372*   4374*   4384*   4411*   4422*
                    4443*   4485*   4554*   4625*   4667*   4739*   4894*   4905*   4911*   4931*   4936*   4957*   4962*
                    5013*   5018*   5027*   5029*   5030    5139*   5189*   5237*   5284*   5285    5298*   5308*   5310
                    5323*   5335*   5336    5347*   5348    5402*   5587*   5600*   5627*   5629    5655*   5821*   5838*
                    5842    5884*   5900*   5931*   5932    5940*   5979*   6009*   6093*   6131*   6191*   6229*   6245*
                    6278*   6294*   6310*   6327*   6332*   6433*   6451*   6486*   6507*   6524*   6598*   6661*   6752*
                    6797*   6812*   7671*   7904*   7918*   7931*   8083    8605    8941    8963    8977    8989
$TMP2   001204      2003#   3020*   3026*   3061*   3093*   3178*   3245*   3284*   3362*   3422*   3536*   3581*   3604*
                    3627*   3649*   3675*   3758*   3788*   3789    3795*   3833*   3839*   3873*   3879*   3903*   3940*
                    3976*   4014*   4020*   4050*   4081*   4098*   4155*   4156    4196*   4197*   4202    4265*   4291*
                    4340*   4341*   4418*   4448*   4486*   4555*   4626*   4661    4668    4702*   4732*   4733    4892*
                    4895    4901*   4902    4927*   4928    4953*   4954    5014*   5015    5020*   5021    5047*   5048
                    5106*   5132*   5133    5182*   5183    5204*   5205    5230*   5231    5263*   5264    5558*   5570*
                    5591*   5608*   5829*   5839    5916*   5951*   5966*   5990*   6002*   6017*   6030*   6068*   6080*
                    6107*   6120*   6145*   6158*   6177*   6203*   6262*   6345*   6363*   6380*   6398*   6415*   6468*
                    6541*   6563*   6580*   6599*   6601    6612    6634*   7919*   8054    8164    8488    8515    8595
                    8963    8977
$TMP3   001206      2004#   3179*   3246*   3285*   3537*   3650*   3676*   3977*   4015*   4021*   4051*   4080*   4099*
                    4158    4200    4216    4266*   4290*   4342    4487*   4513*   4556*   4566*   4590*   4627*   4671
                    4737    4898    5025    5137    5187    5235    5559    5571    6600*   6711*   7920*   8605    8963
```

```
$TMP4   001210          2005#  3049*  3050   3169*  3170*  3180   3327*  3328*  3346*  3347*  3555*  3557*  3576*
                        3577*  3598*  3599*  3621*  3622*  3644*  3645*  3669*  3670*  3981*  3986*  4016*  4025*
                        4063*  4068*  4108*  4113*  4209*  4210   4245*  4246*  4482*  4483*  4511*  4512*  4551*
                        4552*  4579*  4580*  4603*  4604*  4641*  4663   5759*  6602   6619   6624   6631   6634
                        6638*  6639   6646   6666   6674   6680   6688   6706*  6722*  6737*  6757   6765   6782*
                        6787   6794   6797   6801   6810   6818   6826   6831   6842*  6647   6848*  6849*  6853*
                        6860*  6865   6873   6876*  8114   8136   8272   8393   8929   8989
$TMP5   001212          2006#  3182   3184   3329   3348   3558   3578   3600   3623   3646   3671   4247   4251
                        4254   4257   4494   6605   6791   6836   8989
$TMP6   001214          2007#  3054   3174*  3175*  3189   3193   3196   3243*  3244*  3247   3249   3282*  3283*
                        3286   3290   3356*  3357*  3358   3381*  3399*  4238*  4239*  4263   4267   4288*  4289*
                        4292   4294   6603   6606   6628   8164
$TMP7   001216          2008#  3191   3288   3360   3382   3400   4488   4490   4514   4516   4519   4557   4559
                        4581   4583   4605   4607   6604
$TN   = 000055          1437#  1466   2998   3009#  3010   3032   3037   3046#  3047   3065   3070   3079#  3080
                        3098   3103   3130#  3131   3192   3203   3216#  3217   3289   3296   3312#  3401   3405
                        3414#  3415   3428   3434   3442#  3443   3456   3462   3479#  3480   3519   3532#  3533
                        3672   3681   3698#  3739   3749#  3750   3757   3762   3772#  3773   3794   3803   3812#
                        3813   3846   3853   3862#  3863   3878   3883   3892#  3893   3907   3912   3926#  3927
                        3939   3946   3955#  3956   3985   3990   3999#  4000   4024   4028   4042#  4043   4132
                        4141#  4142   4163   4168   4178#  4179   4293   4299   4316#  4317   4389   4392   4408#
                        4447   4452   4462#  4463   4515   4524   4536#  4537   4606   4612   4621#  4622   4640
                        4646   4654#  4655   4701   4705   4715#  4716   4865   4870   4877#  4878   4992   5000#
                        5001   5112   5119#  5120   5166   5173#  5174   5206   5210   5220#  5221   5265   5269
                        5277#  5278   5349   5358   5366#  5367   5408   5411   5425#  5426   5431   5433   5474
                        5482   5493#  5494   5496   5501   5527   5539#  5540   5572   5576   5583#  5584   5585
                        5611   5615   5622#  5623   5630   5634   5641#  5642   5653   5661   5668#  5669   5766
                        5795   5799#  5800
$TPB    001150          1986#  7496*  7507
$TPFLG  001155          1990#  7445   7507
$TPS    001146          1985#  7494   7507
$TRAP   037320          2947   7943#
$TRP  = 000011          7951#  7961#  7962#  7963#  7964#  7965#  7966#  7967#  7968#  7969#
$TRPAD  037342          7948   7959#
$TSTM   004026          2931#
$TSTNM  001102          1965#  6908*  6954   6975   6992*  7035   7070   7092*  7093   7098   7102   7128   7163
                        7573
$TYPBN= ****** U        7965
$TYPDS  036612          7802#  7964
$TYPE   035054          7445#  7539   7951   7960
$TYPEC  035266          7475   7482   7489   7494#  7495
$TYPEX  035334          7500   7502   7505#
$TYPOC  036410          7742#  7961
$TYPON  036424          7741   7744#  7963
$TYPOS  036364          7737#  7962
$UNIT   001246          2027#
$UNITM  004032          2933#
$USWR   001260          2034#
$XTSTR  033174          7057#
$$FEA   034742          4667   4671   7376#
$$GET4= 000000          7020#
$$TSTN  001266          2045#  7573*  7605   8054   8083   8092   8114   8136   8164   8176   8595   8605   8724
                        8836   8941   8948   8963   8977   8989
$OFILL  036607          7738*  7742*  7752   7787#
$200    001600          2108#
$300    001700          2132#
```

DEFPBA.CMB CROSS REFERENCE TABLE -- USER SYMBOLS

```
$40CAT= ****** U        7054    7138
.     = 054534          1934#   1938#   1949    1950#   1952#   1954#   1962#   2015    2051#   2052#   2053#   2194#   2918
                        2919#   2921#   2923#   2940    2955    2956    2985#   3158    3231    3267    3318    3337    3370
                        3390    3485    3504    3543    3567    3589    3612    3635    3660    3705    3724    3780    3819
                        3894    3931    3965    4005    4056    4101    4146    4187    4217    4227    4277    4327    4412
                        4427    4469    4500    4568    4592    4628    4659    4678    4696    4721    4757    4769    4781
                        4793    4805    4817    4834    4854    4882    4917    4943    4969    4982    5038    5056    5074
                        5088    5103    5145    5157    5177    5199    5243    5259    5293    5306    5318    5331    5344
                        5372    5390    5465    5507    5567    5588    5601    5649    5683    5927    5944    5955    5969
                        5982    5994    6006    6021    6033    6046#   6056    6071    6083    6097    6110    6123    6135
                        6148    6162    6181    6194    6217    6233    6249    6265    6281    6298    6314    6330    6349
                        6366    6384    6401    6419    6437    6454    6472    6489    6510    6527    6544    6566    6583
                        6611    6623    6637    6650    6665    6678    6692    6709    6725    6740    6756    6770    6786
                        6800    6817    6830    6846    6864    7005#   7027    7028#   7101    7102    7163    7507    7560#
                        7710#   7856#   7896#   7996    8029    8082#   8091#   8163#   8271#   8392#   8487#   8514#   8604#
                        8835#   8976#   8988#   8999    9010#   9020#
.$ASTA= ****** U        7512    7515
.$X   = 004022          2918#   2923
```

DEFPBA.CMB CROSS REFERENCE TABLE -- MACRO NAMES

```
COMMEN      1#   1884#
COM1     2997#   3000
COM10    3433#   3436
COM11    3461#   3464
COM12    3518#   3521
COM13    3680#   3683
COM14    3738#   3741
COM15    3761#   3764
COM16    3802#   3805
COM17    3852#   3855
COM2     3036#   3039
COM20    3882#   3885
COM21    3911#   3914
COM22    3945#   3948
COM23    3989#   3992
COM24    4027#   4030
COM25    4131#   4134
COM26    4167#   4170
COM27    4298#   4301
COM3     3069#   3072
COM30    4391#   4394
COM31    4451#   4454
COM32    4523#   4526
COM33    4611#   4614
COM34    4645#   4648
COM35    4704#   4707
COM36    4869#   4872
COM37    4991#   4994
COM4     3102#   3105
COM40    5111#   5114
COM40A   5165#   5168
COM41    5209#   5212
COM42    5268#   5271
COM43    5357#   5360
COM44    5410#   5413
COM45    5481#   5484
COM46    5526#   5529
COM47    5575#   5578
COM5     3202#   3205
COM50    5614#   5617
COM51    5633#   5636
COM52    5660#   5663
COM6     3295#   3298
COM7     3404#   3407
ENDCOM      1#   1884#
ERRINS   7102#   7118
ERROR    1503#   3022   3029   3034   3056   3062   3067   3086   3087   3088   3095   3100   3144   3165   3186
         3187    3195   3198   3200   3238   3251   3253   3274   3292   3293   3331   3350   3363   3377   3384
         3402    3426   3431   3454   3459   3499   3554   3560   3583   3605   3628   3652   3677   3720   3760
         3787    3792   3798   3801   3835   3842   3849   3851   3875   3881   3905   3910   3942   3944   3973
         3982    3988   4017   4026   4064   4069   4109   4114   4154   4161   4166   4204   4213   4218   4219
         4253    4256   4259   4261   4269   4270   4296   4297   4390   4419   4449   4480   4492   4493   4496
         4518    4521   4522   4561   4562   4585   4586   4609   4610   4635   4636   4637   4642   4644   4665
         4670    4673   4684   4685   4686   4689   4691   4703   4730   4736   4740   4750   4751   4752   4763
         4775    4787   4799   4811   4824   4829   4843   4849   4862   4868   4889   4897   4900   4906   4912
         4924    4932   4938   4950   4958   4964   4976   4989   5011   5019   5024   5028   5033   5045   5051
```

DEFPBA.CMB CROSS REFERENCE TABLE -- MACRO NAMES

```
                5063    5080    5095    5109    5130    5136    5140    5151    5163    5186    5190    5208    5228    5234    5238
                5257    5267    5288    5301    5313    5326    5339    5351    5356    5382    5400    5405    5409    5478    5517
                5524    5565    5574    5595    5612    5631    5658    5659    5762    5830    5841    5845    5855    5856    5857
                5859    5865    5866    5867    5869    5876    5877    5878    5880    5885    5892    5893    5894    5896    5901
                5907    5908    5909    5911    5917    5934    5942    5953    5967    5980    5992    6004    6019    6032    6069
                6081    6095    6108    6121    6132    6146    6160    6179    6192    6204    6222    6223    6224    6231    6239
                6240    6241    6247    6255    6256    6257    6263    6271    6272    6273    6279    6286    6287    6288    6296
                6304    6305    6306    6312    6320    6321    6322    6328    6338    6339    6340    6347    6355    6356    6357
                6364    6373    6374    6375    6382    6390    6391    6392    6399    6408    6409    6410    6417    6425    6426
                6427    6435    6444    6445    6446    6452    6461    6462    6463    6470    6479    6480    6481    6487    6500
                6501    6502    6508    6517    6518    6519    6525    6533    6534    6535    6542    6555    6556    6557    6564
                6572    6573    6574    6581    6591    6621    6630    6635    6644    6648    6657    6663    6673    6676    6687
                6690    6701    6707    6717    6723    6733    6738    6747    6754    6764    6768    6778    6784    6793    6798
                6809    6815    6825    6828    6838    6844    6856    6862    6872    6878    7906    7908    7921    7923    7932
ESCAPE             1#  1884#   3010    3047    3080    3131    3217    3415    3443    3480    3533    3750    3773    3813    3863
                3893    3927    3956    4000    4043    4142    4179    4317    4463    4537    4622    4655    4716    4878    5001
                5120    5174    5221    5278    5367    5426    5494    5540    5585    5623    5642    5669    5802
FLOTST          1441#
FPDWN           7969#   7982
FPUP            7969#   8005
GETPRI             1#  1884#
MULT               1#  1884#
MYTAGS          1441#   2044
NEWTST             1#  1884#
POP                1#  1884#   7420    7554    7555    7843    8015
PUSH               1#  1884#   7400    7515    7517    7538    7802    7976
REPORT             1#  1884#
SCOINS          7029#   7045
SCOPE           1504#   3008    3045    3078    3129    3215    3311    3413    3441    3478    3531    3697    3748    3771    3811
                3861    3891    3925    3954    3998    4041    4140    4177    4315    4407    4461    4535    4620    4653    4714
                4876    4999    5118    5172    5219    5276    5365    5424    5492    5538    5582    5621    5640    5667    5798
                6991
SETLOO          1441#   3158    3231    3266    3318    3337    3370    3390    3485    3504    3543    3567    3589    3612    3635
                3660    3705    3724    3780    3819    3894    3931    3965    4005    4056    4101    4146    4187    4227    4277
                4327    4412    4427    4469    4500    4568    4592    4628    4659    4678    4695    4721    4757    4769    4781
                4793    4805    4817    4834    4854    4882    4917    4943    4969    4982    5038    5056    5074    5088    5103
                5145    5157    5177    5199    5243    5259    5292    5305    5317    5330    5343    5372    5390    5465    5507
                5567    5588    5601    5649    5683    5927    5943    5954    5968    5981    5993    6005    6020    6055    6070
                6082    6096    6109    6122    6134    6147    6161    6180    6193    6217    6233    6249    6264    6280    6297
                6313    6329    6348    6365    6383    6400    6415    6436    6453    6471    6488    6509    6526    6543    6565
                6582    6610    6622    6636    6649    6664    6677    6691    6708    6724    6739    6755    6769    6785    6799
                6816    6829    6845    6863
SETPRI             1#  1884#
SETTRA          7951#   7961    7962    7963    7964    7965    7966    7967    7968
SETUP              1#  1884#   2935
SKIP               1#  1884#   3032    3065    3098    3192    3289    3401    3428    3456    3672    3757    3794    3846    3878
                3907    3939    3985    4024    4163    4293    4389    4447    4515    4606    4640    4701    4865    5206    5265
                5349    5408    5431    5433    5474    5496    5501    5572    5611    5630    5653    5766
SLASH              1#  1884#
SPACE           1884#
STARS              1#  1884#   1947    1958    2015    2019    2915    2917    2924    2998    3007    3037    3044    3070    3077
                3103    3128    3145    3203    3214    3254    3296    3310    3332    3365    3385    3405    3412    3434    3440
                3462    3477    3498    3500    3519    3530    3562    3584    3607    3630    3654    3681    3696    3719    3721
                3739    3747    3762    3770    3803    3810    3853    3860    3883    3890    3912    3924    3946    3953    3990
                3997    4028    4040    4085    4132    4139    4143    4168    4176    4180    4183    4220    4224    4271    4274
                4299    4314    4392    4406    4420    4452    4460    4524    4534    4563    4587    4612    4619    4646    4652
```

```
                4656    4674    4692    4705    4713    4753    4765    4777    4789    4801    4813    4830    4850    4870    4875
                4913    4939    4965    4978    4992    4998    5034    5052    5065    5082    5097    5112    5117    5141    5153
                5166    5171    5191    5210    5218    5239    5269    5275    5289    5302    5314    5327    5340    5358    5364
                5411    5423    5482    5491    5527    5537    5576    5581    5615    5620    5634    5639    5661    5666    5739
                5763    5782    5792    5795    5797    5832    5847    5919    6034    6035    6036    6037    6053    6054    6133
                6210    6593    6896    6906    6923    6931    6941    6962    6983    7032    7105    7154    7180    7236    7254
                7288    7292    7318    7325    7366    7370    7384    7430    7510    7565    7714    7792    7860    7897    7911
                7926    7937    7972    7998    9000    9001    9002    9017    9018    9019
SWRSU             1#    1884#   2957#
TRMTRP          7951#
TYPBIN            1#    1884#
TYPDEC            1#    1884#   7006    7013
TYPNAM            1#    1884#   2977
TYPNUM            1#    1884#
TYPOCS            1#    1884#
TYPOCT            1#    1884#   7580    7626
TYPTXT            1#    1884#   7002    7009    7301
$MSN            1441#   3171    3240    3279    3352    4205    4235    4240    4242    4285    4335    4337    4508    4548    4576
                4600
$NEWTS          1438#   2998    3037    3070    3103    3203    3296    3405    3434    3462    3519    3681    3739    3762    3803
                3853    3883    3912    3946    3990    4028    4132    4168    4299    4392    4452    4524    4612    4646    4705
                4870    4992    5112    5166    5210    5269    5358    5411    5482    5527    5576    5615    5634    5661    5795
$$CMRE          1955#   1993    1994    1995    1996    1997    1998    1999    2000
$$CMTM          1955#   2001    2002    2003    2004    2005    2006    2007    2008
$$ESCA            1#    1884#   3010    3047    3080    3131    3217    3415    3443    3480    3533    3750    3773    3813    3863
                3893    3927    3956    4000    4043    4142    4179    4317    4463    4537    4622    4655    4716    4878    5001
                5120    5174    5221    5278    5367    5426    5494    5540    5585    5623    5642    5669
$$NEWT            1#    1440#   1884#   2998    3037    3070    3103    3203    3296    3405    3434    3462    3519    3681    3739
                3762    3803    3853    3883    3912    3946    3990    4028    4132    4168    4299    4392    4452    4524    4612
                4646    4705    4870    4992    5112    5166    5210    5269    5358    5411    5482    5527    5576    5615    5634
                5661    5795
$$SET           7951#   7961    7962    7963    7964    7965    7966    7967    7968
$$SKIP            1#    1884#   3032    3065    3098    3192    3289    3401    3428    3456    3672    3757    3794    3846    3878
                3907    3939    3985    4024    4163    4293    4389    4447    4515    4606    4640    4701    4865    5206    5265
                5349    5408    5431    5433    5474    5496    5501    5572    5611    5630    5653    5766
.EQUAT            1#
.FP11C          1441#   1884
.HEADE            1#    1437#   1456
.KT11             1#
.SETUP            1#    1437#   2215
.SWRHI            1#    1437#   1481
.SWRLO          1494#
.$ACTI            1#    1437#   1944
.$APTB            1#    2016#
.$APTH            1#    1437#   2912
.$APTY            1#    1437#   7507
.$ASTA            1#
.$CATC            1#    1437#   1931
.$CMTA            1#    1437#   1955
.$DB2D            1#
.$DB20            1#    1437#   7857
.$DIV             1#
.$EOP             1#    1437#   6980
.$ERRO            1#    1437#   7102
.$ERRT            1#    1441#   7565
.$MULT            1#
```

.$POWE	1#	1438#	7969
.$RAND	1#		
.$RDDE	1#		
.$RDOC	1#		
.$READ	1#		
.$R2AZ	1#		
.$SAVE	1#	1437#	7381
.$SB2D	1#		
.$SB2O	1#		
.$SCOP	1#	1437#	7029
.$SIZE	1#		
.$SUPR	1#		
.$TRAP	1#	1437#	7934
.$TYPB	1#		
.$TYPD	1#	1437#	7789
.$TYPE	1#	1437#	7427
.$TYPO	1#	1437#	7711
.$40CA	1#		
.1170	1#	1437#	1495

DEFPBA.CMB CROSS REFERENCE TABLE -- PERMANENT SYMBOLS

```
            5864    5875    5891    5906    6167    6221    6238    6254    6270    6285    6303    6319    6337    6354    6372
            6389    6407    6424    6443    6460    6478    6493    6499    6516    6532    6549    6554    6571    6915    6920
            7003    7010    7057    7063    7066    7079    7082    7147    7295    7302    7334    7352    7448    7474    7484
            7493    7500    7513    7535    7629    7639    7649    7658    7696    7741    7756    7777    7821    7838    7895
            7996    8029
CFCC        3018    3024    3058    3091    3324    3343    3376    3396    3424    3452    3489    3508    3550    3573    3595
            3618    3641    3666    3710    3729    3756    3831    3837    3871    3877    3901    3938    3979    3984    4010
            4023    4061    4066    4106    4111    4193    4233    4283    4333    4416    4431    4476    4506    4546    4574
            4598    4639    4728    4742    4745    4748    4761    4773    4785    4797    4809    4822    4840    4842    4860
            4887    4922    4948    4974    4987    5009    5043    5061    5078    5093    5107    5128    5149    5161    5180
            5195    5202    5261    5593    5610    5696    5699    5702    5705    5709    5713    5818    5827    5913    5963
            5988    6000    6015    6028    6066    6078    6089    6104    6118    6142    6155    6174    6199    6343    6360
            6378    6395    6413    6466    6538    6560    6577    6703    6719    6735    6766    6780    6840    6858    6874
            7377
CLC         4079
CLR         2938    2952    2953    2972    3028    3177    3179    3246    3335    3417    3429    3484    3534    3536    3537
            3540    3541    3582    3602    3626    3650    3651    3657    3673    3675    3676    3702    3703    3704    3775
            3797    3815    3841    3865    3929    3962    3963    3975    3976    4013    4014    4015    4045    4046    4049
            4050    4053    4054    4055    4089    4090    4093    4094    4096    4097    4098    4099    4100    4145    4185
            4198    4249    4266    4290    4291    4319    4322    4411    4422    4465    4466    4467    4484    4485    4486
            4553    4554    4555    4680    4698    4828    4847    4867    4905    4910    4931    4957    5018    5381    5387
            5393    5436    5444    5472    5473    5475    5476    5515    5518    5519    5523    5587    5600    5628    5650
            5811    5812    5821    5844    5871    5887    5923    5945    5957    5978    6009    6031    6124    6184    6214
            6234    6250    6277    6332    6438    6473    6495    6512    6598    6599    6600    6607    6609    6662    6711
            6721    6758    6802    6803    6812    6813    6820    6848    6853    6884    6888    6893    6950    6951    6952
            6971    6972    6973    6992    6993    7017    7081    7096    7121    7308    7333    7337    7353    7576    7670
            7671    7754    7813    7816    7876    7905    8002
CLRB        3482    7080    7263    7363    7473    7499    7551    7552    7553    7842
CLRD        3987    5731    5778    5804    5805    5806    5807    5808    5809    6058
CLRF        3011    3021    3049    3060    3081    3094    3362    3751    3753    3759    3791    3834    3904    3941    4147
            4160    5589    5602    5903    5960    5965    6060    6099    6169    6195    6268    6385    6528    6545    6567
            6641    6833    6850    6866
CMP         2939    2963    2980    3031    3054    3064    3097    3143    3167    3180    3182    3184    3189    3193    3239
            3247    3249    3278    3286    3288    3290    3325    3329    3344    3348    3379    3382    3397    3400    3455
            3552    3558    3578    3600    3623    3646    3671    3785    3800    3845    3850    3906    3943    3971    4152
            4162    4194    4200    4202    4210    4234    4247    4251    4257    4263    4267    4284    4294    4334    4342
            4349    4352    4355    4360    4362    4364    4477    4478    4488    4490    4507    4514    4516    4547    4557
            4559    4575    4581    4583    4599    4605    4643    4663    4668    4671    4687    4690    4700    4731    4733
            4737    4764    4776    4788    4800    4812    4825    4844    4863    4890    4895    4898    4902    4925    4928
            4934    4951    4954    4960    4977    4990    5012    5015    5021    5025    5030    5046    5048    5064    5081
            5096    5110    5131    5133    5137    5152    5164    5181    5183    5187    5196    5203    5205    5229    5231
            5235    5248    5252    5262    5264    5285    5296    5310    5321    5336    5348    5378    5395    5397    5407
            5477    5500    5516    5520    5551    5557    5559    5569    5571    5606    5629    5652    5721    5725    5752
            5765    5839    5858    5868    5879    5881    5895    5897    5910    5932    5938    5949    6128    6188    6205
            6226    6243    6259    6292    6308    6324    6431    6448    6483    6504    6521    6585    6589    6619    6628
            6631    6642    6655    6671    6674    6685    6689    6699    6715    6731    6745    6750    6762    6776    6791
            6794    6807    6810    6823    6826    6836    6854    6870    6881    6944    6965    7064    7088    7158    7312
            7378    7834
CMPB        4358    6948    6969    7070    7074    7141    7451    7466    7468    7476    7497    7501    7520    7618    7621
            7702
CMPD        3709    3728    3978    3983    4009    4022    4060    4065    4105    4110    4638    5695    5698    5701    5704
            5708    5712
CMPF        3023    3057    3090    3423    3451    3488    3507    3755    3793    3830    3836    3870    3876    3900    3937
            4415    4430    4741    4744    4747    5592    5609    5826    5912    6014    6065    6077    6088    6103    6117
            6141    6154    6173    6198    6342    6359    6377    6394    6412    6422    6465    6537    6559    6576    6702
            6734    6744    6765    6779    6839    6857    6873
```

DEFPBA.CMB CROSS REFERENCE TABLE -- PERMANENT SYMBOLS

```
DEC     4378  4379  4387  4445  6996  7584  7879
DECB    7480  7483  7762  7773
DIVD    4475  4505  4545  4573  4597  4631
DIVF    4150  4192  4232  4282  4332  4414  4429  5179  5201  5825  6115  6458  6775
EMT     1503
HALT    1938  7150  7160  7447  7995  8028
INC     2978  4348  4437  6914  6919  6943  6964  6994  7087  7132  7299  7346  7356  7550  7625
        7632  7642  7768  7776  7820  7882  8003
INCB    3495  3716  6947  6968  7092  7126  7503
IOT     1504
JMP     1943  5555  5568  5780  6208  6588  6894  6911  6921  6929  6939  6953  6974  7026  7305
        7583  7700  7907  7909  7922  7924  7933
JSR     4660  4679  4697  4722  4758  4770  4782  4794  4806  4818  4835  4855  4883  4918  4944
        4970  4983  5004  5039  5057  5075  5089  5104  5123  5146  5158  5178  5200  5222  5604
        7021  7138  7144  7456  7475  7482  7489  7539  7666  7675  7688
LDCDF   4796  5904
LDCFD   6197  6569  6869
LDCIF   5319  5346  5888  6552  6852
LDCLF   6172
LDD     3547  3571  3593  3616  3639  3664  3707  3726  3967  3969  4007  4058  4103  4474  4504
        4544  4572  4596  4630  4772  4784  5073  5087  5102  5674  5686  5688  5690  5691  5692
        5693  5707  5711  5727  5758  5760  5773  7989  7991  8006  8008  8010  8011  8012  8013
LDEXP   4886  4921  4947  5008  5333  6153  6530  6835  7376
LDF     3012  3013  3048  3138  3159  3232  3268  3322  3341  3374  3394  3418  3446  3486  3505
        3752  3781  3782  3820  3868  3895  3896  3935  4191  4231  4281  4331  4413  4428  4723
        4724  4760  4884  4919  4945  4971  4984  5006  5040  5058  5125  5176  5194  5197  5590
        5605  5646  5647  5648  5817  5823  5824  5861  5872  5959  5970  5983  5995  6007  6010
        6022  6023  6061  6063  6072  6073  6084  6085  6100  6112  6113  6125  6137  6138  6150
        6166  6168  6183  6333  6350  6368  6387  6403  6420  6439  6456  6474  6492  6494  6511
        6548  6550  6695  6712  6728  6742  6759  6773  6788  6804  6819
LDFPS   3014  3051  3082  3139  3151  3156  3224  3228  3261  3265  3321  3340  3373  3393  3419
        3447  3546  3570  3592  3615  3638  3663  3706  3778  3825  3827  3864  3897  3934  4006
        4149  4190  4230  4280  4330  4410  4473  4503  4543  4571  4595  4629  4726  4753  4771
        4783  4795  4807  4820  4837  4857  4885  4920  4946  4972  4985  5007  5041  5059  5076
        5090  5105  5126  5147  5159  5193  5283  5294  5510  5626  5816  5822  5836  5851  5928
        5936  5946  5947  5956  5959  5961  5971  5975  5984  5986  5996  5998  6011  6013  6024
        6026  6062  6064  6074  6076  6087  6098  6101  6106  6111  6114  6116  6126  6136  6139
        6144  6149  6152  6157  6163  6171  6176  6182  6185  6187  6196  6201  6215  6218  6219
        6228  6235  6242  6251  6258  6267  6274  6282  6290  6300  6307  6316  6323  6334  6341
        6351  6358  6369  6376  6386  6393  6404  6411  6421  6429  6440  6447  6457  6464  6475
        6482  6496  6503  6513  6520  6529  6536  6551  6558  6568  6575  6614  6615  6616  6618
        6625  6627  6640  6645  6652  6658  6668  6670  6679  6682  6684  6696  6698  6713  6729
        6743  6760  6774  6789  6805  6821  6834  6851  6868  7045  7122  7375  7983  8005  8014
LDUB    3136  3150  3223  3260  3320  3339  3372  3392  3545  3569  3591  3614  3637  3662  3777
        3817  3933  4189  4229  4279  4329  4471  4502  4541  4570  4594  5815  7050  7124  7374
MODD    3970  4008  4059  4104
MODF    3754  3784  3828  3869  3898  3936  5042  5862  6025  6352  6714
MOV     2937  2941  2943  2944  2945  2946  2947  2948  2949  2950  2951  2955  2956  2959  2960
        2961  2962  2967  2968  2969  2971  2975  2986  2988  2989  2990  2991  2992  2993  2994
        3009  3010  3027  3033  3046  3047  3050  3066  3079  3080  3099  3130  3131  3133  3134
        3135  3137  3149  3152  3157  3158  3171  3176  3178  3216  3217  3218  3221  3222  3229
        3230  3231  3240  3245  3258  3259  3267  3279  3284  3285  3312  3315  3316  3317  3318
        3319  3336  3337  3338  3353  3368  3369  3370  3371  3388  3389  3390  3391  3414  3415
        3416  3430  3442  3443  3444  3445  3457  3458  3479  3480  3483  3485  3502  3503  3504
        3532  3533  3535  3538  3539  3542  3543  3544  3548  3551  3565  3566  3567  3568  3574
        3580  3581  3587  3588  3589  3590  3596  3603  3604  3610  3611  3612  3613  3619  3625
```

```
3627  3633  3634  3635  3636  3642  3648  3649  3658  3659  3660  3661  3667  3674  3698
3701  3705  3723  3724  3749  3750  3772  3773  3774  3776  3779  3780  3783  3796  3812
3813  3814  3816  3818  3819  3840  3847  3848  3862  3863  3866  3867  3874  3880  3892
3893  3894  3908  3909  3926  3927  3928  3930  3931  3932  3955  3956  3957  3958  3959
3960  3961  3964  3965  3968  3974  3977  3999  4000  4001  4002  4003  4004  4005  4012
4018  4019  4020  4021  4042  4043  4044  4047  4048  4051  4052  4056  4087  4088  4091
4092  4095  4101  4119  4141  4142  4144  4146  4148  4164  4165  4178  4179  4184  4186
4187  4188  4199  4206  4212  4225  4226  4227  4228  4235  4240  4242  4250  4265  4275
4276  4277  4278  4285  4316  4317  4318  4320  4321  4323  4324  4325  4326  4327  4328
4335  4337  4351  4357  4370  4372  4374  4382  4383  4384  4385  4386  4408  4409  4412
4421  4423  4424  4425  4426  4427  4462  4463  4464  4468  4469  4470  4472  4487  4498
4499  4500  4501  4508  4513  4536  4537  4540  4542  4548  4556  4566  4567  4568  4569
4576  4590  4591  4592  4593  4600  4621  4622  4623  4624  4625  4626  4627  4628  4654
4655  4659  4661  4666  4667  4677  4678  4696  4702  4715  4716  4719  4720  4721  4735
4739  4756  4757  4768  4769  4780  4781  4792  4793  4804  4805  4816  4817  4819  4833
4834  4836  4848  4853  4854  4856  4866  4877  4878  4881  4882  4893  4894  4904  4911
4916  4917  4930  4936  4937  4942  4943  4956  4962  4963  4968  4969  4981  4982  5000
5001  5005  5017  5023  5027  5032  5037  5038  5050  5055  5056  5068  5069  5070  5071
5074  5085  5088  5100  5103  5119  5120  5124  5135  5139  5144  5145  5156  5157  5173
5174  5175  5177  5185  5189  5192  5198  5199  5207  5220  5221  5223  5227  5233  5237
5241  5242  5243  5244  5255  5256  5258  5259  5266  5277  5278  5279  5282  5287  5293
5298  5299  5300  5306  5308  5309  5312  5318  5323  5324  5325  5331  5332  5334  5338
5344  5345  5350  5352  5354  5355  5366  5367  5369  5370  5371  5372  5374  5380  5386
5388  5389  5390  5391  5399  5402  5403  5404  5406  5425  5426  5427  5430  5435  5437
5438  5439  5440  5441  5442  5443  5445  5446  5447  5448  5449  5450  5451  5452  5453
5454  5455  5457  5458  5459  5461  5462  5463  5464  5465  5466  5467  5493  5494  5497
5502  5503  5504  5505  5506  5507  5509  5522  5525  5539  5540  5541  5542  5544  5545
5546  5547  5548  5549  5552  5553  5554  5562  5564  5566  5567  5573  5583  5584  5585
5586  5588  5598  5599  5601  5622  5623  5624  5625  5641  5642  5649  5654  5655  5668
5669  5671  5672  5677  5678  5679  5680  5681  5682  5683  5685  5720  5734  5735  5736
5737  5743  5749  5764  5768  5769  5770  5771  5772  5776  5777  5799  5800  5802  5813
5814  5819  5820  5834  5849  5860  5870  5883  5884  5886  5899  5900  5902  5915  5921
5925  5927  5935  5940  5941  5944  5951  5952  5955  5963  5979  5982  5991  5994  6003
6006  6008  6018  6021  6056  6071  6083  6093  6094  6097  6110  6123  6130  6131  6135
6148  6151  6162  6170  6181  6190  6191  6194  6212  6216  6217  6229  6230  6233  6245
6246  6249  6261  6262  6265  6266  6278  6281  6294  6295  6298  6299  6310  6311  6314
6315  6326  6327  6330  6331  6346  6349  6362  6366  6367  6381  6384  6397  6401  6402
6416  6419  6433  6434  6437  6450  6451  6454  6455  6469  6472  6485  6486  6489  6506
6507  6510  6523  6524  6527  6540  6544  6566  6579  6583  6590  6595  6597  6601  6602
6603  6604  6605  6606  6608  6611  6612  6613  6623  6624  6633  6634  6637  6638  6639
6650  6651  6661  6665  6666  6667  6678  6680  6681  6692  6693  6694  6705  6709  6710
6725  6726  6727  6740  6741  6752  6753  6756  6757  6770  6771  6772  6783  6786  6787
6796  6797  6800  6801  6814  6817  6818  6830  6831  6832  6843  6846  6847  6849  6861
6864  6865  6867  6877  6879  6887  6889  6892  6908  6912  6916  6917  6926  6927  6934
6935  6936  6937  6938  6946  6957  6958  6967  6978  6999  7006  7013  7018  7048  7051
7052  7053  7059  7060  7062  7065  7078  7090  7091  7094  7095  7098  7099  7120  7125
7128  7133  7153  7156  7182  7184  7185  7186  7188  7189  7195  7201  7202  7208  7215
7216  7222  7232  7233  7234  7256  7260  7261  7264  7266  7267  7268  7269  7270  7276
7284  7285  7286  7293  7296  7306  7307  7309  7310  7313  7314  7315  7326  7327  7328
7331  7332  7335  7336  7340  7351  7371  7372  7373  7379  7400  7401  7402  7403  7404
7405  7406  7407  7408  7409  7416  7417  7418  7419  7420  7421  7422  7423  7424  7425
7449  7450  7455  7463  7478  7516  7517  7524  7528  7533  7534  7536  7538  7548  7554
7555  7575  7580  7589  7594  7596  7597  7602  7605  7608  7612  7613  7615  7627  7633
7636  7643  7646  7660  7668  7669  7674  7677  7684  7687  7690  7704  7705  7737  7745
7746  7747  7753  7760  7778  7779  7780  7781  7782  7803  7804  7805  7806  7807  7808
7809  7814  7817  7837  7843  7844  7845  7846  7847  7849  7850  7870  7871  7872  7873
```

DEFPBA.CMB CROSS REFERENCE TABLE -- PERMANENT SYMBOLS

```
        7874  7875  7878  7883  7900  7901  7904  7914  7915  7918  7919  7920  7929  7930  7943
        7944  7948  7974  7975  7976  7977  7978  7979  7980  7981  7984  7993  7994  8000  8001
        8015  8016  8017  8018  8019  8020  8021  8022  8025
MOVB    2954  3699  6928  6954  6975  7049  7093  7097  7123  7135  7143  7187  7192  7198  7199
        7205  7211  7212  7213  7219  7225  7226  7227  7230  7273  7279  7280  7345  7348  7354
        7460  7488  7496  7511  7512  7514  7573  7673  7686  7738  7739  7742  7743  7744  7748
        7751  7752  7771  7812  7815  7829  7832  7841  7877  7946
MULD    3549  3572  3594  3617  3640  3665  3708  3727
MULF    3015  3052  3083  3140  3160  3233  3269  3323  3342  3375  3395  3420  3448  3487  3506
        4973  6012  6335  6697
NEG     5747  7749  7811
NEGF    4821  4838  5468  5997  6317  6683
NOP     5225  5246  5377  5394  5470  5471  5513  5514  6048  6049  7022  7023  7024  9012  9013
RESET   7020
ROL     4074  4075  4076  4080  4081  4082  4120  4121  4122  4125  4126  4127  7755  7757  7758
        7759  7761
ROLB    4077  4083  4124  4129
ROR     4443  4444  7887  7888  7889  7890  7891  7892
RORB    4442
RTI     6959  6979  7100  7162  7410  7426  7465  7783  7851  8027
RTS     7316  7364  7380  7505  7556  7707  7885  7949
RTT     2970  7235  7287
SETD    3725  3966  4057  4102  5072  5086  5101  5673  5803  6057
SETF    5810  5929  6059
SETI    5930
SOB     4446  5456  5460  5676  5775  7194  7207  7221  7275  7282
SPL     5373  5376  5603  6925  6933
STAD    3173  3242  3281  3355  4237  4287
STCDF   5077  5091  5106
STCFD   5260  6140  6514  6822
STCFI   5127  5148  5160  5320  5347  5889  6186  6497  6806
STD     3556  3576  3598  3621  3644  3669  3981  3986  4016  4025  4063  4068  4108  4113  4482
        4511  4551  4579  4603  4641  5608  5675  5687  5689  5728  5759  5761  5774  7985  7986
        7987  7988  7990  7992  8007  8009
STEXP   5335  5651  5873  6127  6476  6790
STF     3020  3026  3061  3093  3169  3174  3243  3282  3327  3346  3356  3381  3399  3422  3758
        3788  3795  3833  3839  3873  3879  3903  3940  4155  4196  4209  4238  4245  4288  4340
        4418  4448  4725  4901  4927  4953  5014  5591  5829  5916  5966  5990  6002  6017  6030
        6068  6080  6102  6107  6120  6145  6158  6159  6177  6178  6202  6203  6345  6363  6380
        6398  6415  6441  6468  6541  6562  6563  6580  6706  6722  6737  6761  6782  6842  6860
        6876
STFPS   3016  3053  3089  3162  3235  3271  3421  3449  3829  3899  4151  4891  4926  4952  5013
        5284  5295  5627  5838  5931  5937  5973  6091  6225  6236  6289  6428  6617  6654  6748
        7982
STQD    3168  3326  3345  3380  3398  3555  3575  3597  3620  3643  3668  4195  4208  4244  4339
        4481  4510  4550  4578  4602
STST    4662  4681  4699  4732  4892  5020  5047  5132  5182  5204  5230  5263  5307  5558  5570
        5948  6252  6626  7931
SUB     5353  5746  7134  7531  7818
SUBF    4986  6075  6405  6730
SWAB    5750
TRAP    7951  7961  7962  7963  7964  7965  7966  7967  7968
TST     2966  2987  3191  3196  3358  3360  3427  3789  4156  4158  4216  4254  4292  4494  4519
        4607  4826  4845  4864  4908  5428  5498  5543  5656  5729  5842  5976  6275  6646  6909
        6913  6918  7061  7085  7148  7154  7294  7297  7342  7462  7470  7492  7526  7544  7546
        7616  7697  7766  7823  7833  7902  7916  7945
```

DEFPBA.CMB CROSS REFERENCE TABLE -- PERMANENT SYMBOLS

```
TSTB    3154  3226  3263  3493  3512  3714  3733  3823  4072  4117  4346  4435  4439  5249  5718
        6955  6976  7072  7329  7445  7494  7518  7529  7542  7599  7655  7680  7693  7825  7839
TSTF    3017  4808  5511  5837  5852  5962  5972  5987  5999  6027  6283  6653  6718
WAIT    7311
XOR     4123  4128
.ABS    1437
.ASCII  2012  2013  8043  8075  8126  8144  8178  8192  8264  8407  8478  8503  8586  8597  8742
        8827  8847  8920  8979  8992
.ASCIZ  2011  2014  2985  7005  7012  7304  7708  7709  8031  8035  8038  8049  8058  8071  8079
        8085  8088  8093  8097  8104  8116  8132  8138  8153  8160  8166  8171  8185  8201  8207
        8214  8219  8224  8229  8236  8239  8246  8248  8251  8254  8266  8274  8281  8289  8299
        8307  8316  8329  8335  8341  8346  8353  8359  8363  8367  8373  8380  8386  8394  8399
        8415  8420  8422  8428  8435  8445  8455  8462  8473  8482  8492  8496  8507  8517  8523
        8530  8535  8541  8547  8556  8565  8574  8578  8582  8591  8601  8607  8613  8620  8624
        8631  8638  8646  8650  8655  8665  8670  8677  8681  8688  8699  8705  8711  8719  8726
        8731  8736  8746  8749  8757  8761  8767  8776  8786  8797  8804  8809  8816  8823  8832
        8838  8844  8851  8854  8864  8870  8874  8882  8890  8897  8906  8915  8925  8931  8935
        8943  8950  8954  8966  8970  8984  8996
.BLKB    2052  7896
.BLKW    2051  2053  2194  7856  9020
.BYTE    1965  1966  1971  1972  1987  1988  1989  1990  2031  2032  2048  2049  2060  2063  2066
        2069  2072  2075  2078  2081  2084  2087  2090  2093  2096  2099  2102  2105  2108  2111
        2114  2117  2120  2123  2126  2129  2132  2135  2138  2141  2144  2147  2150  2153  7027
        7145  7146  7557  7558  7559  7784  7785  7786  7787  8056  8273  8490
.DSABL   1437
.ENABL      1  1437
.END    9022
.ENDC   1461  1490  1492  1493  1494  1925  1944  1948  1952  1954  1959  1963  1965  1991  2001
        2009  2010  2011  2012  2016  2020  2044  2199  2215  2916  2918  2925  2941  2942  2945
        2947  2949  2951  2952  2953  2955  2957  2977  2980  2982  2985  2999  3000  3007  3008
        3009  3010  3011  3033  3038  3039  3044  3045  3046  3047  3048  3066  3071  3072  3077
        3078  3079  3080  3081  3099  3104  3105  3128  3129  3130  3131  3132  3146  3172  3193
        3204  3205  3214  3215  3216  3217  3218  3241  3255  3280  3290  3297  3298  3310  3311
        3312  3313  3333  3354  3366  3386  3402  3406  3407  3412  3413  3414  3415  3416  3429
        3435  3436  3440  3441  3442  3443  3444  3457  3463  3464  3477  3478  3479  3480  3481
        3499  3501  3520  3521  3530  3531  3532  3533  3534  3563  3585  3608  3631  3655  3673
        3682  3683  3696  3697  3698  3699  3720  3722  3740  3741  3747  3748  3749  3750  3751
        3758  3763  3764  3770  3771  3772  3773  3774  3795  3804  3805  3810  3811  3812  3813
        3814  3847  3854  3855  3860  3861  3862  3863  3864  3879  3884  3885  3890  3891  3892
        3893  3894  3908  3913  3914  3924  3925  3926  3927  3928  3940  3947  3948  3953  3954
        3955  3956  3957  3986  3991  3992  3997  3998  3999  4000  4001  4025  4029  4030  4040
        4041  4042  4043  4044  4086  4133  4134  4139  4140  4141  4142  4143  4144  4164  4169
        4170  4176  4177  4178  4179  4180  4181  4184  4207  4221  4225  4236  4241  4243  4272
        4275  4286  4294  4300  4301  4314  4315  4316  4317  4318  4336  4338  4390  4393  4394
        4406  4407  4408  4409  4421  4448  4453  4454  4460  4461  4462  4463  4464  4509  4516
        4525  4526  4534  4535  4536  4537  4538  4549  4564  4577  4588  4601  4607  4613  4614
        4619  4620  4621  4622  4623  4641  4647  4648  4652  4653  4654  4655  4656  4657  4675
        4693  4702  4706  4707  4713  4714  4715  4716  4717  4754  4766  4778  4790  4802  4814
        4831  4851  4866  4871  4872  4875  4876  4877  4878  4879  4914  4940  4966  4973  4993
        4994  4998  4999  5000  5001  5002  5035  5053  5066  5083  5098  5113  5114  5117  5118
        5119  5120  5121  5142  5154  5167  5168  5171  5172  5173  5174  5175  5192  5207  5211
        5212  5218  5219  5220  5221  5222  5240  5266  5270  5271  5275  5276  5277  5278  5279
        5290  5303  5315  5328  5341  5350  5359  5360  5364  5365  5366  5367  5368  5409  5412
        5413  5423  5424  5425  5426  5427  5432  5434  5475  5483  5484  5491  5492  5493  5494
        5495  5497  5502  5528  5529  5537  5538  5539  5540  5541  5573  5577  5578  5581  5582
        5583  5584  5585  5586  5612  5616  5617  5620  5621  5622  5623  5624  5631  5635  5636
```

```
          5639  5640  5641  5642  5643  5654  5662  5663  5666  5667  5668  5669  5670  5740  5764
          5767  5783  5793  5796  5797  5798  5799  5800  5801  5803  5833  5848  5920  6034  6035
          6036  6037  6038  6054  6055  6134  6211  6594  6897  6907  6924  6932  6942  6963  6984
          6985  6987  6989  6992  6998  7001  7002  7005  7012  7018  7020  7026  7027  7028  7033
          7036  7041  7054  7056  7067  7070  7071  7072  7074  7076  7083  7087  7092  7094  7098
          7101  7102  7106  7109  7126  7133  7138  7139  7140  7148  7158  7162  7163  7165  7181
          7237  7255  7289  7293  7304  7319  7326  7367  7371  7385  7431  7460  7511  7512  7515
          7542  7557  7566  7584  7711  7715  7793  7861  7898  7912  7927  7938  7944  7947  7960
          7961  7962  7963  7964  7965  7966  7967  7968  7973  7982  7993  7999  8015  8025  8027
          8034  9000  9001  9002  9003  9018  9019  9020
.EQUIV    1503  1504  1506  1528  1529  1530  1531  1532  1533  1534  1535  1536  1537  1538  1567
          1568  1569  1570  1571  1572  1573  1574  1575  1576  1595  1596  1597  1598  1599  1600
          1601  1602  1603  1604  1657  1658  1659  1660  1868  1869  1870  1871  1872  1873  1874
          1875  1876  1877  1878  1879  1880  1881  1882  1883  1899  1900  1901  1902  1903  1904
          1905  1906  1907  1908  1909  1910  1911  1912  1915  1916
.EVEN     2020  2985  7005  7012  7028  7304  7560  7710  8033  8053  8082  8091  8113  8135  8163
          8175  8271  8392  8487  8514  8594  8604  8723  8835  8928  8940  8947  8962  8976  8988
.IF       1457  1489  1491  1492  1493  1494  1884  1941  1947  1950  1952  1958  1962  1964  1991
          2001  2009  2010  2011  2015  2016  2019  2042  2044  2215  2915  2917  2924  2936  2941
          2943  2945  2947  2949  2951  2952  2953  2955  2972  2979  2980  2981  2984  2998  3000
          3007  3009  3010  3032  3037  3039  3044  3046  3047  3065  3070  3072  3077  3079  3080
          3098  3103  3105  3128  3130  3131  3145  3171  3192  3203  3205  3214  3216  3217  3240
          3254  3279  3289  3296  3298  3310  3312  3332  3353  3365  3385  3401  3405  3407  3412
          3414  3415  3428  3434  3436  3440  3442  3443  3456  3462  3464  3477  3479  3480  3498
          3500  3519  3521  3530  3532  3533  3562  3584  3607  3630  3654  3672  3681  3683  3696
          3698  3719  3721  3739  3741  3747  3749  3750  3757  3762  3764  3770  3772  3773  3794
          3803  3805  3810  3812  3813  3846  3853  3855  3860  3862  3863  3878  3883  3885  3890
          3892  3893  3907  3912  3914  3924  3926  3927  3939  3946  3948  3953  3955  3956  3985
          3990  3992  3997  3999  4000  4024  4028  4030  4040  4042  4043  4085  4132  4134  4139
          4141  4142  4143  4163  4168  4170  4176  4178  4179  4180  4183  4206  4220  4224  4235
          4240  4242  4271  4274  4285  4293  4299  4301  4314  4316  4317  4335  4337  4389  4392
          4394  4406  4408  4420  4447  4452  4454  4460  4462  4463  4508  4515  4524  4526  4534
          4536  4537  4548  4563  4576  4587  4600  4606  4612  4614  4619  4621  4622  4640  4646
          4648  4652  4654  4655  4656  4674  4692  4701  4705  4707  4713  4715  4716  4753  4765
          4777  4789  4801  4813  4830  4850  4865  4870  4872  4875  4877  4878  4913  4939  4965
          4978  4992  4994  4998  5000  5001  5034  5052  5065  5082  5097  5112  5114  5117  5119
          5120  5141  5153  5166  5168  5171  5173  5174  5191  5206  5210  5212  5218  5220  5221
          5239  5265  5269  5271  5275  5277  5278  5289  5302  5314  5327  5340  5349  5358  5360
          5364  5366  5367  5408  5411  5413  5423  5425  5426  5431  5433  5474  5482  5484  5491
          5493  5494  5496  5501  5527  5529  5537  5539  5540  5572  5576  5578  5581  5583  5594
          5585  5611  5615  5617  5620  5622  5623  5630  5634  5636  5639  5641  5642  5653  5661
          5663  5666  5668  5669  5739  5763  5766  5782  5792  5795  5797  5799  5800  5802  5832
          5847  5919  6034  6035  6036  6037  6053  6054  6133  6210  6593  6896  6906  6923  6931
          6941  6962  6983  6984  6985  6987  6988  6989  6991  6997  7000  7002  7004  7011  7020
          7026  7027  7032  7035  7040  7045  7054  7066  7068  7069  7070  7072  7073  7074  7083
          7085  7093  7095  7100  7101  7102  7105  7108  7118  7129  7136  7138  7139  7141  7148
          7151  7158  7162  7163  7164  7180  7236  7254  7288  7292  7303  7318  7325  7366  7370
          7384  7430  7451  7510  7512  7515  7542  7557  7565  7583  7612  7714  7792  7860  7897
          7911  7926  7937  7943  7947  7951  7961  7962  7963  7964  7965  7966  7967  7968  7972
          7982  7998  8005  8015  8023  8025  8027  8031  9000  9001  9002  9017  9018  9019
.IFF      1489  1492  1493  1494  1948  1952  1954  1959  1962  1965  1991  2016  2020  2916  2918
          2925  2941  2979  2981  2999  3000  3008  3009  3011  3033  3038  3039  3045  3046  3048
          3066  3071  3072  3078  3079  3081  3099  3104  3105  3129  3130  3132  3146  3172  3193
          3204  3205  3215  3216  3218  3241  3255  3280  3290  3297  3298  3311  3312  3333  3354
          3366  3386  3402  3406  3407  3413  3414  3416  3429  3435  3436  3441  3442  3444  3457
          3463  3464  3478  3479  3481  3499  3501  3520  3521  3531  3532  3534  3563  3585  3608
```

```
                3631  3655  3673  3682  3683  3697  3698  3720  3722  3740  3741  3748  3749  3751  3758
                3763  3764  3771  3772  3774  3795  3804  3805  3811  3812  3814  3847  3854  3855  3861
                3862  3864  3879  3884  3885  3891  3892  3894  3908  3913  3914  3925  3926  3928  3940
                3947  3948  3954  3955  3957  3986  3991  3992  3998  3999  4001  4025  4029  4030  4041
                4042  4044  4086  4133  4134  4140  4141  4143  4144  4164  4169  4170  4177  4178  4180
                4181  4184  4207  4221  4225  4236  4241  4243  4272  4275  4286  4294  4300  4301  4315
                4316  4318  4336  4338  4390  4393  4394  4407  4408  4421  4448  4453  4454  4461  4462
                4464  4509  4516  4525  4526  4535  4536  4538  4549  4564  4577  4588  4601  4607  4613
                4614  4620  4621  4623  4641  4647  4648  4653  4654  4656  4657  4675  4693  4702  4706
                4707  4714  4715  4717  4754  4766  4778  4790  4802  4814  4831  4851  4866  4871  4872
                4876  4877  4879  4914  4940  4966  4979  4993  4994  4999  5000  5002  5035  5053  5066
                5083  5098  5113  5114  5118  5119  5121  5142  5154  5167  5168  5172  5173  5175  5192
                5207  5211  5212  5219  5220  5222  5240  5266  5270  5271  5276  5277  5279  5290  5303
                5315  5328  5341  5350  5359  5360  5365  5366  5368  5409  5412  5413  5424  5425  5427
                5432  5434  5475  5483  5484  5492  5493  5495  5497  5502  5528  5529  5538  5539  5541
                5573  5577  5578  5582  5583  5586  5612  5616  5617  5621  5622  5624  5631  5635  5636
                5640  5641  5643  5654  5662  5663  5667  5668  5670  5740  5764  5767  5783  5793  5796
                5797  5798  5799  5800  5802  5833  5848  5920  6035  6036  6037  6038  6054  6055  6134
                6211  6594  6897  6907  6924  6932  6942  6963  6984  6988  6992  6998  7001  7027  7033
                7067  7070  7071  7074  7101  7102  7106  7108  7129  7158  7163  7165  7181  7237  7255
                7289  7293  7319  7326  7367  7371  7385  7431  7511  7566  7583  7612  7715  7793  7861
                7898  7912  7927  7938  7944  7973  7999  8025  9001  9002  9003  9018  9019  9020
.IFT            2985  7005  7012  7082  7139  7304
.IFTF           2985  7005  7012  7080  7138  7304
.IIF            1456  1461  1466  1486  1487  1488  1490  1493  1494  1938  2015  2020  2942  2945  2951
                2952  2953  2955  2956  2980  6985  6992  6993  7007  7014  7027  7028  7036  7037  7038
                7039  7040  7041  7081  7082  7098  7101  7102  7109  7110  7111  7112  7113  7158  7163
                7507  7581  7628  7960  7961  7962  7963  7964  7965  7966  7967  7968
.IRP            2044  2215  2998  3037  3070  3103  3203  3296  3405  3434  3462  3519  3681  3739  3762
                3603  3853  3883  3912  3946  3990  4028  4132  4168  4299  4392  4452  4524  4612  4646
                4705  4870  4992  5112  5166  5210  5269  5358  5411  5482  5527  5576  5615  5634  5661
                5795  7045  7118  7400  7420  7516  7517  7538  7554  7555  7803  7843  7976  8015
.LIST              1  1437  1438  1439  1440  1441  1493  1884  1938  1991  1993  1994  1995  1996  1997
                1998  1999  2000  2001  2002  2003  2004  2005  2006  2007  2008  2009  2016  2020  2215
                2957  2980  2985  2998  3009  3037  3046  3070  3079  3103  3130  3203  3216  3296  3312
                3405  3414  3434  3442  3462  3479  3519  3532  3681  3698  3739  3749  3762  3772  3803
                3812  3853  3862  3883  3892  3912  3926  3946  3955  3990  3999  4028  4042  4132  4141
                4168  4178  4299  4316  4392  4408  4452  4462  4524  4536  4612  4621  4646  4654  4705
                4715  4870  4877  4992  5000  5112  5119  5166  5173  5210  5220  5269  5277  5358  5366
                5411  5425  5482  5493  5527  5539  5576  5583  5615  5622  5634  5641  5661  5668  5795
                5799  6992  7005  7012  7020  7040  7158  7304  7951  7960  7961  7962  7963  7964  7965
                7966  7967  7968  7969
.MACRO             1  1438  1440  1441  1494  1955  2997  3036  3069  3102  3202  3295  3404  3433  3461
                3518  3680  3738  3761  3802  3852  3882  3911  3945  3989  4027  4131  4167  4298  4391
                4451  4523  4611  4645  4704  4869  4991  5111  5165  5209  5268  5357  5410  5481  5526
                5575  5614  5633  5660  7029  7102  7951  7969
.MCALL          1437  1884  2016  2957
.MEXIT          2043
.NLIST             1  1437  1438  1439  1440  1441  1493  1884  1938  1991  1993  1994  1995  1996  1997
                1998  1999  2000  2001  2002  2003  2004  2005  2006  2007  2008  2009  2016  2020  2215
                2957  2980  2985  2998  3009  3037  3046  3070  3079  3103  3130  3203  3216  3296  3312
                3405  3414  3434  3442  3462  3479  3519  3532  3681  3698  3739  3749  3762  3772  3803
                3812  3853  3862  3883  3892  3912  3926  3946  3955  3990  3999  4028  4042  4132  4141
                4168  4178  4299  4316  4392  4408  4452  4462  4524  4536  4612  4621  4646  4654  4705
                4715  4870  4877  4992  5000  5112  5119  5166  5173  5210  5220  5269  5277  5358  5366
                5411  5425  5482  5493  5527  5539  5576  5583  5615  5622  5634  5641  5661  5668  5795
```

DEFPBA.CMB CROSS REFERENCE TABLE -- PERMANENT SYMBOLS

```
            5799    6992    7005    7012    7020    7040    7158    7304    7951    7960    7961    7962    7963    7964    7965
            7966    7967    7968    7969
.PAGE       1466    1495    1955    2015    2217
.REM           1
.REPT       1938    1993    2001
.SBTTL      1482    1496    1623    1634    1648    1797    1884    1898    1914    1923    1932    1942    1945    1956    2017
            2200    2913    2998    3037    3070    3103    3203    3296    3405    3434    3462    3519    3681    3739    3762
            3803    3853    3883    3912    3946    3990    4028    4132    4168    4299    4392    4452    4524    4612    4646
            4705    4870    4992    5112    5166    5210    5269    5358    5411    5482    5527    5576    5615    5634    5661
            5781    5783    5793    5795    6897    6981    7030    7103    7165    7237    7289    7319    7382    7428    7508
            7567    7712    7790    7858    7898    7912    7927    7935    7952    7970
.TITLE      1456
.WORD       1938    1939    1940    1953    1964    1967    1968    1969    1970    1973    1974    1975    1976    1977    1978
            1979    1980    1981    1982    1991    1993    1994    1995    1996    1997    1998    1999    2000    2001    2002
            2003    2004    2005    2006    2007    2008    2022    2023    2024    2025    2026    2027    2028    2029    2033
            2034    2035    2044    2045    2046    2047    2050    2054    2055    2056    2057    2058    2059    2156    2157
            2158    2159    2160    2161    2162    2163    2164    2165    2166    2167    2168    2169    2170    2171    2172
            2173    2174    2175    2176    2177    2178    2179    2180    2181    2182    2183    2184    2185    2186    2187
            2188    2189    2190    2191    2192    2195    2196    2197    2198    2929    2930    2931    2932    2933    2934
            3084    4632    4633    4682    4839    4859    5092    5224    5245    5375    5392    5469    5556    5560    5563
            5853    5863    5974    5890    5905    6047    6220    6237    6253    6269    6284    6302    6318    6336    6353
            6371    6388    6406    6423    6442    6459    6477    6498    6515    6531    6553    6570    6960    6997    7000
            7259    7457    7504    7540    7592    7610    7635    7638    7645    7648    7662    7679    7692    7788    8024
            8026    8054    8083    8092    8114    8136    8164    8176    8272    8393    8488    8515    8595    8605    8724
            8936    8929    8941    8948    8963    8977    8989    9011    9021
```

```
ERRORS DETECTED:  0
DEFAULT GLOBALS GENERATED:  0

*,DEFPBA.SEQ/SOL/CRF/PAGNUM/NL:TOC/DS:ERFZ=DEFPBA.SML,DEFPBA.CMB
RUN-TIME:  63 94 15 SECONDS
RUN-TIME RATIO:  361/173=2.0
CORE USED:  39K  (77 PAGES)
```