

[Microfiche Frame 1]	[Microfiche Frame 2]	[Microfiche Frame 3]	[Microfiche Frame 4]	[Microfiche Frame 5]	[Microfiche Frame 6]	[Microfiche Frame 7]	[Microfiche Frame 8]	[Microfiche Frame 9]	[Microfiche Frame 10]	[Microfiche Frame 11]	[Microfiche Frame 12]	[Microfiche Frame 13]	[Microfiche Frame 14]
[Microfiche Frame 15]	[Microfiche Frame 16]	[Microfiche Frame 17]	[Microfiche Frame 18]	[Microfiche Frame 19]	[Microfiche Frame 20]	[Microfiche Frame 21]	[Microfiche Frame 22]	[Microfiche Frame 23]	[Microfiche Frame 24]	[Microfiche Frame 25]	[Microfiche Frame 26]	[Microfiche Frame 27]	[Microfiche Frame 28]
[Microfiche Frame 29]	[Microfiche Frame 30]	[Microfiche Frame 31]	[Microfiche Frame 32]	[Microfiche Frame 33]	[Microfiche Frame 34]	[Microfiche Frame 35]	[Microfiche Frame 36]	[Microfiche Frame 37]	[Microfiche Frame 38]	[Microfiche Frame 39]	[Microfiche Frame 40]	[Microfiche Frame 41]	[Microfiche Frame 42]
[Microfiche Frame 43]	[Microfiche Frame 44]	[Microfiche Frame 45]	[Microfiche Frame 46]	[Microfiche Frame 47]	[Microfiche Frame 48]	[Microfiche Frame 49]	[Microfiche Frame 50]	[Microfiche Frame 51]	[Microfiche Frame 52]	[Microfiche Frame 53]	[Microfiche Frame 54]	[Microfiche Frame 55]	[Microfiche Frame 56]
[Microfiche Frame 57]	[Microfiche Frame 58]	[Microfiche Frame 59]	[Microfiche Frame 60]	[Microfiche Frame 61]	[Microfiche Frame 62]	[Microfiche Frame 63]	[Microfiche Frame 64]	[Microfiche Frame 65]	[Microfiche Frame 66]	[Microfiche Frame 67]	[Microfiche Frame 68]	[Microfiche Frame 69]	[Microfiche Frame 70]
[Microfiche Frame 71]	[Microfiche Frame 72]	[Microfiche Frame 73]	[Microfiche Frame 74]	[Microfiche Frame 75]	[Microfiche Frame 76]	[Microfiche Frame 77]	[Microfiche Frame 78]	[Microfiche Frame 79]	[Microfiche Frame 80]	[Microfiche Frame 81]	[Microfiche Frame 82]	[Microfiche Frame 83]	[Microfiche Frame 84]
[Microfiche Frame 85]	[Microfiche Frame 86]	[Microfiche Frame 87]	[Microfiche Frame 88]	[Microfiche Frame 89]	[Microfiche Frame 90]	[Microfiche Frame 91]	[Microfiche Frame 92]	[Microfiche Frame 93]	[Microfiche Frame 94]	[Microfiche Frame 95]	[Microfiche Frame 96]	[Microfiche Frame 97]	[Microfiche Frame 98]
[Microfiche Frame 99]	[Microfiche Frame 100]	[Microfiche Frame 101]	[Microfiche Frame 102]	[Microfiche Frame 103]	[Microfiche Frame 104]	[Microfiche Frame 105]	[Microfiche Frame 106]	[Microfiche Frame 107]	[Microfiche Frame 108]	[Microfiche Frame 109]	[Microfiche Frame 110]	[Microfiche Frame 111]	[Microfiche Frame 112]
[Microfiche Frame 113]	[Microfiche Frame 114]	[Microfiche Frame 115]	[Microfiche Frame 116]	[Microfiche Frame 117]	[Microfiche Frame 118]	[Microfiche Frame 119]	[Microfiche Frame 120]	[Microfiche Frame 121]	[Microfiche Frame 122]	[Microfiche Frame 123]	[Microfiche Frame 124]	[Microfiche Frame 125]	[Microfiche Frame 126]
[Microfiche Frame 127]	[Microfiche Frame 128]	[Microfiche Frame 129]	[Microfiche Frame 130]	[Microfiche Frame 131]	[Microfiche Frame 132]	[Microfiche Frame 133]	[Microfiche Frame 134]	[Microfiche Frame 135]	[Microfiche Frame 136]	[Microfiche Frame 137]	[Microfiche Frame 138]	[Microfiche Frame 139]	[Microfiche Frame 140]
[Microfiche Frame 141]	[Microfiche Frame 142]	[Microfiche Frame 143]	[Microfiche Frame 144]	[Microfiche Frame 145]	[Microfiche Frame 146]	[Microfiche Frame 147]	[Microfiche Frame 148]	[Microfiche Frame 149]	[Microfiche Frame 150]	[Microfiche Frame 151]	[Microfiche Frame 152]	[Microfiche Frame 153]	[Microfiche Frame 154]

PDP11-45/55/70

FP11C DIAGNOSTIC PART 1
MD-11-DEFPA-A

EP DEFPA-A DL
COPYRIGHT 1976
FICHE 2 OF 2

JAN 1978
digital
MADE IN USA

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DEPPA-A-D
PRODUCT NAME: PDP11-45/55/70 FP11C DIAGNOSTIC PART 1
DATE CREATED: FEBRUARY 21, 1976
MAINTAINER: DIAGNOSTIC ENGINEERING
AUTHOR: DONALD W. MONPUE

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

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED TO THE PURCHASER UNDER A LICENSE FOR USE ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION OF DIGITAL'S COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.

COPYRIGHT (C) 1976 BY DIGITAL EQUIPMENT CORPORATION

CONTENTS

1. ABSTRACT
2. REQUIREMENTS
 - 2.1 EQUIPMENT
 - 2.2 STORAGE
 - 2.3 PRELIMINARY PROGRAMS
3. LOADING PROCEDURE
 - 3.1 METHOD
4. STARTING PROCEDURE
 - 4.1 CONTROL SWITCH SETTINGS
 - 4.2 STARTING ADDRESS
 - 4.3 PROGRAM AND OPERATOR ACTION
5. OPERATING PROCEDURE
 - 5.1 OPERATIONAL SWITCH SETTINGS
 - 5.2 SUBROUTINE ABSTRACTS
 - 5.3 OPERATOR ACTION
6. ERRORS
 - 6.1 ERROR HALTS AND DESCRIPTION
 - 6.2 ERROR RECOVERY
7. RESTRICTIONS
 - 7.1 STARTING RESTRICTIONS
 - 7.2 OPERATING RESTRICTIONS
8. MISCELLANEOUS
 - 8.1 EXECUTION TIME
 - 8.2 STACK POINTER
 - 8.3 PASS COUNT
 - 8.4 ITERATIONS
 - 8.5 SPECIAL REGISTERS
 - 8.6 T BIT TRAPPING
 - 8.7 OSCILLOSCOPE SYNC POINTS
 - 8.8 A-BRANCH, NO-MEM BRANCH, AND ADX PCM TESTS
 - 8.9 ACT11 AND APT11 COMPATABILITY
9. PROGRAM DESCRIPTION
10. LISTINGS

1. **ABSTRACT**

DEPPA/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**

DEPPA 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**

DEPPB 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 OPERATIONAL SWITCH SETTINGS

THE SWITCH SETTINGS ARE:

SW<15>=1 ... HALT ON ERPOP
SW<14>=1 ... LOOP ON CURRENT TEST
SW<13>=1 ... INHIBIT ERROR TYPE OUIS
SW<12>=1 ... SKIP MEMORY MANAGEMENT TFSTS (DEFPP ONLY)
SW<11>=1 ... INHIBIT ITERATIONS
SW<10>=1 ... RING BELL ON ERROR
SW<9>=1 ... LOOP ON ERPOP (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 TOT 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 ITERATION, 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 (ERPPC), 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 VVY 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 YVVVVY ZZZZZZ

WHERE U = SIGN BIT
W = 8-BIT EXPONENT (RIGHT JUSTIFIED)
V = 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 DEPPA THIS ROUTINE SETS UP THE LINE CLOCK TO INTERRUPT THE CENTRAL PROCESSOR EXACTLY ONE CLOCK PERIOD AFTER THE ROUTINE IS EXITED.

IN DEPPP THIS ROUTINE HANDLES THE CLOCK INTERRUPTS AND CONTROLS THE NUMBER OF SECONDS THAT THE ROM TEST IS LOOPEL 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 SERPTR 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

NONE

8. MISCELLANEOUS

8.1 EXECUTION TIME

A. DEPPA

THE FIRST PASS TAKES APPROXIMATELY 5 SECONDS.
ALL SUBSEQUENT PASSES TAKE APPROXIMATELY 7 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 (DR2 ON SLOT 5)

SYNCE CERTAIN TESTS USE THE MICRO-TRAP FEATURE FOR FAULT ISOLATION, THE MICRO-WATCH 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 DEPPB 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 "SENDAD" FOR THE ACT11 SEQUENCE TABLE.

LOCATION 44 CONTAINS THE ADDRESS OF "SAPTHDR" 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

PDP 11/45/55/70...FP11-C DIAGNOSTIC PART 1

COPYRIGHT 1976
DIGITAL EQUIPMENT CORPORATION
MAYNARD, MASS. 01754

TABLE OF CONTENTS

46	OPERATIONAL SWITCH SETTINGS
59	BASIC DEFINITIONS
196	CACHE REGISTER DEFINITIONS
197	CPU REGISTER DEFINITIONS
211	MEMORY MANAGEMENT DEFINITIONS
360	UNIRUS MAP REGISTER DEFINITIONS
447	FLOATING POINT REGISTER DEFINITIONS
461	FLOATING POINT STATUS BIT DEFINITIONS
477	FLOATING POINT EXCEPTION CODE DEFINITIONS
486	FLOATING POINT VECTOR DEFINITION
491	TPAP CATCHER
501	STARTING ADDRESS(ES)
504	ACT11 HOOKS
515	COMMON TAGS
576	APT MAILBOX-ETABLE
684	ERROR POINTER TABLE
1976	APT PARAMETER BLOCK
3347	
3348	FOUR POP STATES
3749	
4581	
4582	FIVE POP STATES

TABLE OF CONTENTS

4583	
8150	END OF PASS ROUTINE
8199	SCOPE HANDLER ROUTINE
8277	ERROR HANDLER ROUTINE
8340	CONVERT FLOATING BINARY TO OCTAL ASCII
8413	CONVERT FLOATING DOUBLE BINARY TO OCTAL ASCII
8466	ROUTINE TO START THE LINE CLOCK
8497	SAVE AND RESTORE R0-R5 ROUTINES
8543	TYPE ROUTINE
8623	APT COMMUNICATIONS ROUTINE
8682	ERROR MESSAGE TIMEOUT ROUTINE
8805	BINARY TO OCTAL (ASCII) AND TYPE
8883	CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
8951	DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
8991	UNEXPECTED TRAP TO 4 ROUTINE
9005	UNEXPECTED TRAP TO 114 ROUTINE
9020	UNEXPECTED TRAP TO 244 ROUTINE
9028	TRAP DECODER
9045	TRAP TABLE
9063	POWER DOWN AND UP ROUTINES

21 COPYRIGHT (C) FEBRUARY 21, 1976
DIGITAL EQUIPMENT CORP.
MAYNARD, MASS. 01754

PROGRAM BY DONALD W. MOWROE

THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
PACKAGE (MAINDEC-11-DZQAC-B7), NOV 21, 1975.

46

OPERATIONAL SWITCH SETTINGS

47

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

59

BASIC DEFINITIONS

- 61 INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
- 76 MISCELLANEOUS DEFINITIONS
- 82 GENERAL PURPOSE REGISTER DEFINITIONS
- 103 PRIORITY LEVEL DEFINITIONS
- 113 "SWITCH REGISTER" SWITCH DEFINITIONS
- 141 DATA BIT DEFINITIONS (BIT00 TO BIT15)
- 169 BASIC "CPU" TRAP VECTOR ADDRESSES

196 *****
CACHE REGISTER DEFINITIONS

197 *****
CPU REGISTER DEFINITIONS

211 *****
MEMORY MANAGEMENT DEFINITIONS

214 MEMORY MANAGEMENT STATUS REGISTER ADDRESSES

225 USER "I" PAGE DESCRIPTOR REGISTERS

236 USER "D" PAGE DESCRIPTOR REGISTERS

247 USER "I" PAGE ADDRESS REGISTERS

258 USER "D" PAGE ADDRESS REGISTERS

269 SUPERVISOR "I" PAGE DESCRIPTOR REGISTERS

280 SUPERVISOR "D" PAGE DESCRIPTOR REGISTERS

291 SUPERVISOR "I" PAGE ADDRESS REGISTERS

302 SUPERVISOR "D" PAGE ADDRESS REGISTERS

313 KERNEL "I" PAGE DESCRIPTOR REGISTERS

324 KERNEL "D" PAGE DESCRIPTOR REGISTERS

335 KERNEL "I" PAGE ADDRESS REGISTERS

346 KERNEL "D" PAGE ADDRESS REGISTERS

360 *****
UNIBUS MAP REGISTER DEFINITIONS

363 THE LOWER 16 BITS OF THE MAP REGISTERS ARE LABELED "MAPLXX"
THE UPPER 6 BITS OF THE MAP REGISTERS ARE LABELED "MAPHXX"

447 *****
FLOATING POINT REGISTER DEFINITIONS

461 *****
FLOATING POINT STATUS BIT DEFINITIONS

477 *****
FLOATING POINT EXCEPTION CODE DEFINITIONS

496 *****
FLOATING POINT VECTOR DEFINITION

491 *****
TRAP CATCHER

494 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

501 *****
STARTING ADDRESS(ES)

504 *****
ACT11 HOOKS

515 *****
COMMON TAGS

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

576

 APT MAILBOX-ETABLE

595

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

684

 ERROR POINTER TABLE

686

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

692

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

1876

 APT PARAMETER BLOCK

1967

TEST 1 LDFPS*M0

POSSIBLE CPU FAILURES

A FORK

IF PACK A7 RAR00 DOES NOT GO LOW A TRAP TO LOCATION 10
 WILL OCCUR.

CFORK

IF IRCC F/CLASS DOES NOT GO LOW EXECUTION WILL GO TO STATE 200
 WHICH WILL CAUSE A TRAP THRU LOCATION 24.
 IF IRCC F/CLASS DOES NOT GET TO E40(A0) OR IF E(40) IS BAD EXECUTION
 WILL GO TO STATE 201 (JSR.10). THIS WILL CAUSE R0 TO BE PUSHED
 ONTO THE STACK.
 IF IRCC F/CLASS DOES NOT GET TO IRCC E21 OR E21 IS BAD, EXECUTIN WILL GO TO
 STATE 210 (NEG.90). THIS WILL CAUSE THE SR+1 TO BE PUT IN THE DR.
 IF IRCC DSTM0 DOES NOT GO LOW ON [SM0*IP(8:6)*PCLASS] EXECUTION WILL GO TO
 STATE WAT.00, CAUSING THE PROCESSOR TO HANG UNTIL THE LINE CLOCK INTERRUPTS.

BEN 17

IF PACK E64(A0) DOES NOT GO HIGH EXECUTION WILL GO TO STATE 153.
 THIS WILL CAUSE A TRAP THRU LOCATIN 24. THE STACKED PC WILL BE
 THE PC OF THE LDFPS INSTRUCTION.
 IF PACK RTP+FP SYNC DOES NOT GO LOW EXECUTION WILL LOOP ON STATE 133.
 THIS WILL HANG THE PROCESSOR UNTIL THE BRQ (LINE CLOCK INTERRUPT) OCCURS.

BEN 7

IF PACK BE75 DOES NOT GET FP PEG WT AS A LOW EXECUTION WILL GO TO

STATE 333. THIS WILL CAUSE THE DESTINATION REGISTER TO BE Clobbered.

CPU ROM FLOW - 101,314,211,133,173
FPP ROM FLOW -30,60

2037 TEST 2 INTERFACE DATA PATH

THIS TEST TESTS THE INTERFACE DATA PATH BY FLOATING A ONE AND THEN A ZERO THROUGH THE DESTINATION REGISTER. A PIR TRAP IS USED TO YANK THE CPU OUT OF THE FPP SERVICE FLOW CAUSING THE DESTINATION REGISTER TO BE RESTORED FROM THE FDR REGISTER.

NOTE: A BIT FAILURE IN THIS TEST COULD BE CAUSED BY ONE OF THREE DATA PATHS: 1) CPU TO FDR; 2) FDR TO DOMUX; OR 3) DOMUX TO CPU.

CPU FAILURES

C FORK

IF IRCC DSTMO L DOES NOT GO HIGH EXECUTION WILL GO TO STATE FOP.00. THIS WILL HANG THE PROCESSOR IN A LOOP BETWEEN ROM ADDRESSES 101 AND 314.

BEN 15

IF IRCB FJ CLASS DOES NOT GO LOW EXECUTION WILL GO TO STATE D12.10. THIS WILL FINISH THE BUS OPERATION AND THEN THE B FORK WILL FAIL TO STATE S13.01. THIS WILL HANG THE PROCESSOR IN A LOOP AROUND ROM STATES 111,175,27, AND 27.

B FORK

IF IRCB BO PAB01 DOES NOT GO LOW EXECUTION WILL GO TO STATE JSR.00. THIS WILL CAUSE THE PC TO BE PUSHED ON THE STACK AND THE CONTENTS OF THE DESTINATION TO BE PUT IN THE PC.

IF IRCB BO PAB03 DOES NOT GO LOW EXECUTION WILL GO TO STATE S67.00. THIS WILL CAUSE A DESTINATION MODE 6 TO START EXECUTING. THIS FAILURE WILL BE FOUND BY

2064

BEN 17

MAKING THE INDEX ODD, SO AN ODD ADDRESS TPAP WILL OCCUR.

IF PACK BRCAB05 DOES NOT GO LOW (ON BRQ*(T+CONF)) EXECUTION WILL GO TO STATE 327. THIS WILL CAUSE THE FPU TO COMPLETE THE INSTRUCTION BEFORE THE INTERRUPT OCCURS.

CPU ROM FLOW-101,314,111,135,36,347,150,245

2150 TEST 3 STFPS*MO

THIS TEST STARTS OUT JUST VERIFYING THAT LOAD FPS AND STFPS GET DECODED PROPERLY IN THE FPU.

CPU FAILURES

BEN 7

IF PACK BE75 DOES NOT GET FP REG WT AS A HIGH EXECUTION WILL GO TO STATE PET.06. THIS WILL CAUSE THE DESTINATION REGISTER TO REMAIN UNCHANGED.

IT THEN FLOATS A ONE AND A ZERO THROUGH THE FPS REGISTER.

CPU ROM FLOW 101, 214, 211, 133, 173, 333, 365
FPU ROM FLOW 62, 23

2225 TEST 4 SET MODES

THIS TEST ENSURES THAT THE FOUR INSTRUCTIONS "SETP", "SETD", "SETL", AND "SETI" FUNCTION.

FPU ROM FLOW - 30, 33

2295 TEST 5 MICRO-BREAK TRAP

THIS TEST ENSURES THAT THE MICRO-BREAK TRAP FUNCTIONS PROPERLY IT IS TESTED HERE SO THAT IT CAN BE USED TO DEBUG ANY INSTRUCTION FAILURES THAT MAY OCCUR LATER ON.

IF THE NO-MEM BRANCH FAILS EXECUTION WOULD GO TO ONE OF TWO STATES: SET MODES, OR ILLEGAL OP CODE. SET MODES WOULD BE CAUSED BY FXPB FIRA00 OR 01 FAILING AND ILLEGAL OP CODE WOULD BE CAUSED BY FXPB FIRA 02 OR 03 FAILING.

IF THE TRAP FAILS, THE FAILURE COULD BE IN THE FPP ON THE BACKPLANE OR IN THE CPU ON TMC(8135). THE TRAP VECTOR COULD FAIL TO 4 INSTEAD OF 244. THIS WOULD BE CAUSED TMCB FPTRAP L OR DAPE TV05*07.

FPU ROM FLOW -30, 33, 2, 6

2339 TEST 6 CPCC

THIS TEST ENSURES THAT THE PSW CONDITION CODES GET LOADED FROM THE FPP CONDITION CODES

IF IT FAILS, EITHER FXPC PCLD EN IS NOT GOING LOW OR IT IS NOT GETTING TO IRCE OR ICF CC<BR IS NOT GOING HIGH.

2359 TEST 7 LDF*-MO*IMM

CPU FAILURES

BEN 10

IF FP CLASS DOES NOT GET TO RACK EXECUTION WILL GO TO STATE FSV.30. THIS WILL CAUSE THE PROCESSOR TO HANG IN POP STATE 265 WAITING FOR FP SYNC.

BEN 7

IF PRMF FPREQ DOES NOT GET TO RACK AS A LOW THE CPU WILL CONTINUE TO DO PUS OPERATIONS UNTIL THE ADDRESS TIMES OUT.

IF THE DESTINATION REGISTER (R7) FAILS TO AUTO-INCREMENT THE WORD AT 35 WILL TRY TO BE EXECUTED. THIS WILL CAUSE A RESERVED INSTRUCTION TRAP.

FPU FAILURES

E2

ONLY THOSE FAILURES WHICH CAN BE DETECTED IN THIS TEST ARE DISCUSSED HERE. OTHER POSSIBLE FAILURES ARE DEFERRED TO THE NEXT TEST.

A BRANCH

IF FXPB FIRA10(0) H IS NOT GOING LOW OR NOT GETTING THRU THE ROM ADDRESS MUX, EXECUTION WILL GO TO STATE 25. THIS WILL CAUSE THE DESTINATION TO BE CLEARED.
IF FXPB A BRANCH MUX(A3) DOES NOT GO HIGH, EXECUTION WILL GO TO STATE 13. THIS WILL CAUSE A STX*-NO TO BE EXECUTED.

CPU ROM FLOW-101,314,111,135,36,327,367,362,307,237
FPU ROM FLOW-12,133,16

2424 TEST 10 STF*-NO*IMM

CPU FAILURES

THE CPU SHOULD NOT FAIL

FPU FAILURES

LDF A BRANCH

IF FXPB FIRA08(0) H IS STUCK HIGH EXECUTION WILL GO TO THE LDAC6*FD(0) FLOWS.

IF FXPB FD(1) H IS STUCK HIGH EXECUTION WILL GO TO LDD*-NO. THIS WILL BE TESTED LATER.

IF FXPB NO IS STUCK HIGH EXECUTION WILL GO TO NO*-ST SPEC.

LDF IMM BRANCH

IF FXPB IMMEDIATE DOES NOT GET TO FPMF AS A LOW, TWO WORDS WILL BE LOADED INSTEAD OF ONE. THIS FAILURE WILL BE DETECTED ON THE STF INSTRUCTION.

STF A BRANCH

IF FXPB FIRA11(0) H IS STUCK HIGH EXECUTION WILL GO TO STATE ZERO (0).

IF FXPB FIRA10(0) H IS STUCK LOW A STCFI WILL OCCUR.

IF FXPB FIRA08(0) H IS STUCK LOW EXECUTION WILL GO TO LDAC6*FD(0) FLOWS.

STF IMM BRANCH

IF THIS BRANCH FAILS (SEE ABOVE) THE WORD FOLLOWING THE DATA WORD WILL BE OVERWRITTEN WITH DATA, THAT WILL EXECUTE AS A NOP.

IF AD2 DOES NOT ACCEPT THE DATA WORD FOLLOWING THE INSTRUCTION WILL BE EXECUTED.

FPU ROM FLOW-13,203

2489 TEST 11 LDF*STF*NO

A BRANCH

IF FXPB NO DOES NOT GO LOW EXECUTION WILL GO TO THE -NO FLOW THIS WOULD CAUSE THE FPU TO HANG IN ROM STATE 132.

NO MEM BRANCH

IF FXPB INMX2 DOES NOT GET TO THE NO-MEM MUX AS A HIGH, EXECUTION WILL GO TO ROM STATE 0, ON THE LDF.

IF FXPB IRMX3 DOES NOT GET TO THE NO-MEM MUX AS A HIGH OR
 IF FXPB IR(11:6)0 L DOES NOT GO HIGH
 EXECUTION WILL GO TO ROM STATE 0 ON THE STF.
 IF FXPB IR(11:9)0 DOES NOT GET TO THE NO-MEM MUX AS A LOW,
 EXECUTION WILL GO TO ROM STATE 56 ON THE LDF AND 0 ON
 THE STF.

FPU ROM FLOW - LDF: 30,42,16
 STF: 30,43,262

2600 TEST 12 LDF*STF*-MO*-IMM

FPU FAILURES
LDF IMM BRANCH

IF FRMA IMMEDIATE L DOES NOT GET TO FRMF PPREQ AS A HIGH THE
 THE CPU WILL HANG IN ROM STAT2 327 ON THE STF INSTRUCTION.

STF PD BRANCH

IF FRMJ PD(0)H DOES NOT GET TO THE PAD00 MUX (ON FRMA) AS A HIGH EXECUTION
 WILL GO TO STATE 206. THIS WILL CAUSE THE FPU TO HANG.

IF FRMJ PD(1)H DOES NOT GET TO THE AD1,AD2 ROM (ON FXPA) AS A LOW,
 THE DESTINATION REGISTER WILL AUTO-INCREMENT BY 10 INSTEAD OF 4.

IF FXPE AD1 DOES NOT GET TO IRCD AS A HIGH THE DESTINATION REG
 WILL INCREMENT BY 10.

IF FXPE AD2 DOES NOT GO LOW OR DOES NOT GET IRCD E68(1) THE
 DESTINATION REGISTER WON'T INCREMENT.

IF IRCD DSTCON<4 DOES NOT GO HIGH (ON AD1) THE DEST REG WON'T INCREMENT.

IF IRCD DSTCON<10 DOES NOT GO LOW (ON AD1 H) THE DESTINATION REG.
 WILL INCREMENT BY 14.

IF FRMJ PD(1)H DOES NOT GET TO FXP AS A LOW A LDD AND STD WILL BE EXECUTED.

2626

FPU ROM FLOW-LDF:12,132,42,16
 STF:13,203,202

2676 TEST 13 HIGH ORDER DATA PATH

THIS TEST FLOATS A ONE AND A ZERO THROUGH QUADRANTS 2 AND 3
 OF ACO TO VERIFY THE HIGH ORDER DATA PATH.

2734 TEST 14 HIGH ORDER SCRATCH PAD DUAL ADDRESSING

THIS TEST ENSURES THAT THE HIGH ORDER SCRATCH PAD ACCUMULATOR ADDRESS LINES
 ARE FUNCTIONAL. THIS IS PERFORMED BY WRITTING THE ADDRESS OF

- 2738 THE ACCUMULATORS INTO THE ACCUMULATORS AND THEN READING THEM BACK AND CHECKING THEM. THE ADDRESS OF THE ACCUMULATOR IS WRITTEN INTO EACH CHIP OF THE ACCUMULATOR.
- 2786 TEST 15 LDPPS/STPPS*-M0
 THE A BRANCH SHOULD NOT FAIL. IF THE INSTRUCTIONS DON'T WORK, THE THE CONTROL STOPE IS PROBABLY BAD OR THE ADX ROM.
 FPU ROM FLOW-LDPPS:22
 STPPS:62,23
- 2820 TEST 16 CLRFP*M0
 IF PIPA OR(O)H DOES NOT GET TO THE A BRANCH NOW MUX AS A LOW, EXECUTION WILL GO TO P0M STATE 70 AND THEN TO ROM STATE 0 ON THE NO-MEM BRANCH. ALSO, IF PIPOR(O)H DOES NOT FORCE FXPB IR(11:6)0 L TO GO HIGH, EXECUTION WILL GO TO ROM STATE 3 ON THE NO-MEM BRANCH. BOTH THESE FAILURES WOULD CAUSE THE DESTINATION NOT TO BE CLEARED.
 FPU ROM FLOW-30,55
- 2909 TEST 17 TSTP*M0
 THE ONLY WAY THE A AND NO MEM BRANCHES SHOULD FAIL IS IF THE ROM(S) FAIL.
 A ONE (1) IS FLOATED THRU THE EXPONET TO VERIFY THAT THE SIGNAL FXPL ACMX EQ 0 IS UNACERTED FOR ALL * INPUTS.
 FPU ROM FLOW-30,56
- 2947 TEST 20 CLRFP*-M0*IMM
 THE ONLY WAY THIS SHOULD FAIL, IS IF THE CONTROL STORE FAILS.
 FPU ROM FLOW-25,211
- 2987 TEST 21 CLRFP*-M0*-IMM
 THE ONLY WAY THIS TEST SHOULD FAIL, IS IF THE CONTROL STOPE, (ROM STATE 216) FAILS.
 FPU ROM FLOW-25,211,216
- 3009 TEST 22 ABSF*M0*-(EXP=0)
 THE A BRANCH SHOULD NOT FAIL.
 THE NO-MEM BRANCH SHOULD NOT FAIL.
 ABSX BRANCH
 IF PIPB06(1)H DOES NOT GO LOW OR DOES NOT GET TO P0MA RADP03 AS A HIGH, EXECUTION WILL GO TO ROM STATE 134. THIS WILL CAUSE THE SIGN BIT TO BE COMPLIMENTED.

BZ BRANCH

3018 IF FRMB B7(0)H DOES NOT GO HIGH OR DOES NOT GET TO FRMA RAD00 AS A LOW, EXECUTION WILL GO TO STATE 106. THIS WILL CAUSE THE EXPONENT AND FRACTION TO BE CLEARED.

FPU ROM FLOW-30,57,124

3080 TEST 23 NEG*MO*-(EXP=0)

THE ONLY POSSIBLE FAILURE SHOULD BE THE ARSY BRANCH. IF FIRB06(1) DOES NOT GO HIGH OR DOES NOT GET TO FRMA RAD03 AS A LOW, EXECUTION WILL GO TO STATE 124. THIS WILL CAUSE AN ARSY TO BE EXECUTED.

FPU ROM FLOW-30,57,134

3130 TEST 24 ADD*MO*(SRC=0)*-(DST=0)

NEITHER THE A NOR THE NO-MEM BRANCHES SHOULD FAIL.

6P1 BRANCH

IF FRMJ EXPB0(1)H DOES NOT GO HIGH OR DOES NOT GET TO FRMJ RAD02 AS A LOW EXECUTION WILL GO TO STATE 240. SINCE THIS FAILURE WOULD NOT BE DETECTED BY THE ANSWR, THE MICRO-BREAK REGISTER IS SETUP TO TRAP ON STATE 240.

IF FRMJ EXPA0(1)H DOES NOT GO LOW OR DOES NOT GET TO FRMA RAD03 AS A HIGH, EXECUTION WILL GO TO STATE 254. THIS WOULD CAUSE THE DESTINATION TO BE CLEARED.

NOTE: SYNC POINT NOT SELECTIBLE--DEFAULT=240

NOTE: SYNC POINT NOT SELECTABLE. DEFAULT=240

FPU ROM FLOW-30,65,244

3180 TEST 25 ADD*MO*(SRC=0)*(DST=0)

6P1 BRANCH

IF FRMJ EXPA0(1)H DOES NOT GO HIGH OR DOES NOT GET TO FRMA RAD03, EXECUTION WILL GO TO STATE 244. THIS WILL CAUSE THE DESTINATION FRACTION TO REMAIN UNCHANGED.

3186

FPU ROM FLOW-30,65,254

3208 TEST 26 SUB*MO*(SRC=0)*(DST=0)

THE ONLY POSSIBLE FAILURE WOULD BE A BAD CELL IN EITHER THE A BRANCH ROM OR THE NO MEM ROM.

FPU ROM FLOW-30,65,254

3255 TEST 27 CMPF*MO*(SRC=0)*-(DST=0)

THE ONLY THING THAT SHOULD FAIL IS THE CONTROL STORE FOR ROM STATE 374, OR THE A OR NO-MEM ROMS.

FPU ROM FLOW-30,67,324

3303 TEST 30 CMPF*MO*-(SRC=0)*(DST=0)

THE ONLY POSSIBLE FAILURE IS THE CONTROL STORE SIGNALS IN ROM STATE 330 OR THE SD MUX.

FPU ROM FLOW-30,67,330

3323 TEST 31 CMPF*MO*(SRC=0)*(DST=0)

THE ONLY POSSIBLE FAILURE IS THE CONTROL STORE SIGNALS IN ROM STATE 334.

FPU ROM FLOW-30,67,334

3347

3348 FOUR ROM STATES

3349

3352 TEST 32 LDEXP*-MO*(EXP=+)

3354 THE A AND NO-MEM BRANCHES SHOULD NOT FAIL.

BZ-PN BRANCH

IF PRMB BN(0)H DOES NOT GO HIGH OR DOES NOT GET TO PAD01 AS A LOW, EXECUTION WILL GO TO STATE 360. A MICRO-BREAK TRAP WILL BE SET TO CATCH THIS FAILURE.

THE BZ SIGNAL SHOULD NOT FAIL.

A ZERO AND ONE ARE THEN FLOATED THROUGH BITS <6:0> TO CHECK THE STEP COUNTER.

FPU ROM FLOW-15,10,260,362

3447 TEST 33 ABSF*MO*(SRC=0)

BZ BRANCH

IF PRMB BZ(0)H DOES NOT GO LOW OR DOES NOT GET TO PRMB RAD00 AS A HIGH, EXECUTION WILL GO TO ROM STATE 3. THIS WILL CAUSE THE FRACTION NOT TO BE CLEARED.

FPU ROM FLOW-30,57,124,106

3473 TEST 34 NEG^F*M0*(SRC=0)

THE ONLY POSSIBLE FAILURE WOULD BE THE CONTROL STORE IN ROM STATES 134 OR 106.

FPU ROM FLOW-30,57,134,106

3492 TEST 35 ADD^F*M0*-(SRC=0)*(DST=0)

THE A AND NO-MEM BRANCHES SHOULD NOT FAIL.

6F1 BRANCH

IF FRMJ EXPB0(1)H DOES NOT GO LOW OR DOES NOT GET TO FRMA MAD02 AS A HIGH, EXECUTION WILL GO TO STATE 254. THIS WILL CAUSE THE DESTINATION PRAC. TO BE CLEARED.

FPU ROM FLOW-30,65,250,306

3536 TEST 36 SUB^F*M0*-(SRC=0)*(DST=0)

THE ONLY POSSIBLE FAILURE IN THIS TEST IS THE SIGNAL FRMF SS(0)H XOR SUB.

FPU ROM FLOW-30,65,250,306

3566 TEST 37 CMP^F*M0*(SRC>DST)

THE A, NO-MEM, AND 6F1 BRANCHES SHOULD NOT FAIL.

3F2 BRANCH

THE BZ BIT SHOULD NOT FAIL.

IF FXPL BCN L DOES NOT GO HIGH WITH EVALU09 ON A LOW, EXECUTION WILL GO TO STATE 301. THIS WILL CAUSE THE N BIT TO BE WRONG.

A ONE(1) IS THEN FLOATED THROUGH EXPONENT A TO VERIFY THAT FXPL EXPA EQ 0 GOES LOW FOR ALL 8 BITS.

FPU ROM FLOW-30,67,320,303

3647 TEST 40 CMP^F*M0*(SRC<DST)

3F2 BRANCH

IF FXPL BCN L DOES NOT GO LOW WITH EVALU09 ON A HIGH, EXECUTION WILL GO TO STATE 303. THIS WILL CAUSE THE N BIT TO BE WRONG.

A ONE(1) IS THEN FLOATED THROUGH EXPONENT B TO VERIFY THAT FXPL EXPB EQ 0 GOES LOW FOR ALL 8 BITS.

FPU ROM FLOW-30,67,320,301

3704 TEST 41 LDD/STD*-M0*-IMM

IF FRMJ PD(1) DOES NOT GO HIGH, ONLY 2 WORDS WILL BE LOADED AND STORED.
IF THE ADX POP FAILS, THE DESTINATION REGISTER WILL AUTO-INCREMENT BY 4 INSTEAD OF 10. THIS WILL ALSO OCCUR IF AD1 & AD2 DO NOT GET TO IRCD AS A LOW.

IF FRMF PPREG DOES NOT STAY ASCERTED FOR 4 CYCLES THE CPU & FP WILL GET OUT OF SYNC.

LDD

FPU ROM FLOW-21,142,165,201,42,16

STD

FPU ROM FLOW-13,203,202,206,212

3759 TEST 42 LOW ORDER DATA PATH

THIS TEST FLOATS A ONE AND A ZERO THROUGH QUADRANTS 0 AND 1 OF AC0 TO VERIFY THE LOW ORDER DATA PATH.

3810 TEST 43 LOW ORDER SCRATCH PAD DUAL ADDRESSING

THIS TEST ENSURES THAT THE LOW ORDER SCRATCH PAD ACCUMULATOR ADDRESS LINES ARE FUNCTIONAL. THIS IS PERFORMED BY WRITING THE ADDRESS OF THE ACCUMULATORS INTO THE ACCUMULATORS AND THEN READING THEM BACK AND CHECKING THEM. THE ADDRESS OF THE ACCUMULATOR IS WRITTEN INTO EACH CHYP OF THE ACCUMULATOR.

3894 TEST 44 STAO*STQO

THE STAO AND STQO INSTRUCTIONS ARE TESTED BY DOING A CMPD INSTRUCTION & TRAPPING OUT AT ROM STATE CMP.25. THIS LEAVES THE SRC & DST DATA IN THE AR AND QR RESPECTIVELY. THE AR WILL CONTAIN THE -ABS VAL OF THE DIFFERENCE IN EXPONENTS.

THIS TEST ALSO VERIFIES THAT THE LDAC6*PD(1) FLOW BINGS IN ALL FOUR WORDS.

NOTE: SYNC POINT ONLY SELECTABLE ON STAO AND STQO

CMPD FPU ROM FLOW-20,140,270,150,67,320
STAO FPU ROM FLOW-30,36
STQO FPU ROM FLOW-30,53

3968 TEST 45 QR DATA PATH

3970

THIS TEST ENSURES THAT THERE ARE NO BITS STUCK IN THE QNX AND QR. THIS IS DONE BY EXECUTING A CMPD INSTRUCTION AND TRAPPING ON STATE 67 WHICH LEAVES THE DST ACC IN THE QR. A STQO IS THEN EXECUTED TO GET THE QR DATA BACK. THE DATA IS COMPOSED OF A FLOATING "ONE" AND THEN A FLOATING "ZERO".

THE ONLY SYNC POINT AVAILABLE DURING THE CMPD IS STATE 67. THE QR GETS LOADED IN THIS STATE

NOTE: SYNC POINT ONLY SELECTABLE ON STQO INSTRUCTION.

4091 TEST 46 DISABLE LOW FMX

THIS TEST INSURES THAT FRND AND FRLC DISABLE LOW FMX AND FRND DISABLE POUND FMX AND FRLC DISABLE FMX19 ARE NOT STUCK LOW. THIS IS DONE BY MOVING ALL ONE'S FROM AC1 TO AC0 AND TRAPPING ON STATE 65 AND EXAMINING THE AR.

NOTE: SYNC POINT NOT SELECTABLE. DEFAULT=65

4137 TEST 47 DISABLE LOW QMX

4138

THIS TEST INSURES THAT FRND AND FRLA DISABLE LOW QMX IS FUNCTIONING PROPERLY. THIS IS DONE BY TRAPPING ON STATE 67 OF A CMPF AND ENSURING THAT THE LOW QR IS ZERO.

NOTE: SYNC POINT NOT SELECTABLE. DEFAULT=67

4179 TEST 50 ACDC3,0)COND_ACMX

THIS TEST INSURES THAT FRMH WRITE ACD1 GOES HIGH WITH FD(0) AND A CONDITIONAL WRITE.

4211 TEST 51 LDD*-M0*INM

THE A AND IMMEDIATE BRANCHES SHOULD NOT FAIL. IF THE ADX ROM FAILS, THE PC WILL AUTO INCREMENT BY SOMETHING OTHER THAN 2.

FPU ROM FLOW - 21, 143, 16

4262 TEST 52 LDCPD*M0*-(EXP=0)

NONE OF THE MICRO BRANCHES SHOULD FAIL. IF FRMH CLR AR<34:00> DOES NOT GO LOW, OR GET TO ALL THE CHIPS OF THE LOWER AR, THE AR WON'T CLEAR.

FPU ROM FLOW - 30, 46, 136

4308 TEST 53 LDCFD*M0*(EXP=0)

THE ONLY THING THAT SHOULD FAIL IS ROM STATE 276.

FPU ROM FLOW - 30, 46, 136, 276

4352 TEST 54 LDCDF*M0*(EXP=0)

THE ONLY THING THAT SHOULD FAIL IS ROM STATE 230.

FPU ROM FLOW - 30, 46, 137, 230

4390 TEST 55 DISABLE LOW & HI FMX ON ROUND

THIS TEST INSURES THAT PRHD DISABLE LOW FMX AND DISABLE HI FMX AND PRLC DISABLE LOW FMX GO HIGH WHEN ROUND IS ENABLED. THIS IS DONE BY LOADING THE QR WITH ALL ONE'S AND THEN EXECUTING A LDCDF, TRAPPING ON STATE 137, AND EXAMINING THE AP.

NOTE: SYNC POINT NOT SELECTABLE. DEFAULT=67

4450 TEST 56 LDCDF*M0*--(EXP=0)*--AR59*--ROU

THE ONLY THING THAT SHOULD FAIL IS THE *F2J BRANCH. IF PRHE AP59(1) DOES NOT GO HIGH OR DOES NOT GET TO PRMA RAD01 AS A LOW, EXECUTION WILL GO TO STATE 161.

A MICRO-TRAP WILL BE SET TO CATCH THIS FAILURE.

IF PRMA BOU+BZ DOES NOT GO HIGH, EXECUTION WILL GO TO STATE 162.

THIS WILL CAUSE THE DESTINATION TO BE CLEARED.

IF PRHC FMX 35 DOES NOT GO LOW, THE DESTINATION WILL HAVE A "1" IN THE LSR.

FPU ROM FLOW - 30, 46, 137, 231

4510 TEST 57 LDCFD*-M0*IMM*--(EXP=0)

THE ONLY POSSIBLE FAILURES ARE ROM STATES 17 OR 221 OR THE ADX ROM.

FPU ROM FLOW - 17, 221, 136

4550 TEST 60 LDCFD*-M0*-IMM*--(EXP=0)

THE ONLY POSSIBLE FAILURE IN THIS TEST SHOULD BE ROM STATE 220.

FPU ROM FLOW - 17, 220, 46, 136

4581

4582

FIVE ROM STATES

4583

4586 TEST 61 LDEXP*NO*(EXP=+)

THE ONLY THING THAT SHOULD FAIL IS THE A BRANCH ROM,
THE NO-MEM ROM, OR ROM STATE 45.

FPU ROM FLOW - 30, 45, 10, 260, 362

4606 TEST 62 STEP

THE A BRANCH SHOULD ONLY FAIL, IF THE ROM FAILS. THE EXPONENT BEING WRONG
COULD BE A FUNCTION OF THE BALU CARRY LOGIC OR ANY OF THE CONTROL
STORE SIGNALS.

IF THE SIGN IS WRONG, THEN FXPP SC09 IS NOT GETTING TO THE SD MUX OR
THE SD MUX IS BAD.

FPU ROM FLOW - 50, 64, 147, 63, 152

4677 TEST 63 CLRD*-NO*-IMM

THE ONLY THING THAT SHOULD FAIL IS ROM STATE 222 OR THE ADX ROM.

FPU ROM FLOW - 25, 211, 216, 227, 212

4711 TEST 64 ARSF*-NO*-(EXP=0)

IF FRMB FP CLASS DOES NOT GO LOW OR DOES NOT GET TO RACK AS
A LOW, THE CPU AND FP WILL GET OUT OF SYNC.

THE BRANCHES IN THE FPU SHOULD NOT FAIL.

CPU ROM FLOW - 101, 314, 111, 135, 36, 327, 367, 347, 265, 225, 362, 307, 237
FPU ROM FLOW - 27, 217, 224, 177, 212

4743 TEST 65 NEGF*-NO*-(EXP=0)

THE ONLY THING THAT SHOULD FAIL IS ROM STATE 234.

FPU ROM FLOW - 27, 217, 234, 177, 212

4771 TEST 66 ADDP*-M0*--(SRC=0)*(DST=0)

THE ONLY THING THAT SHOULD FAIL IS THE LDAC6 FLOWS.

FPU ROM FLOW - 11, 130, 65, 250, 306

4805 TEST 67 ADDP*-M0*IMM*--(SRC=0)*(DST=0)

THE ONLY THING THAT SHOULD FAIL IS ROM STATE 131 OR THE ADX ROM

FPU ROM FLOW - 11, 131, 65, 250, 306

4834 TEST 70 SUBP*-M0*--(SRC=0)*(DST=0)

THE ONLY THING THAT SHOULD FAIL IS THE A BRANCH OR ADX PCW.

FPU ROM FLOW - 11, 130, 65, 250, 306

4868 TEST 71 SUBP*-M0*IMM*--(SRC=0)*(DST=0)

THE ONLY THING THAT SHOULD FAIL IS THE ADX ROM

4887 TEST 72 STCFD*M0*--(EXP=0)

6F2J BRANCH

IF PKLP PVINT DOES NOT GO HIGH OR DOES NOT GET TO PRMA PAD01 AS A LOW, EXECUTION WILL GO TO STATE 171. THIS WILL CAUSE AN INTERRUPT.

THE ONLY OTHER POSSIBLE FAILURES SHOULD BE FROM SIGNALS IN THE CONTROL STORE.

NOTE: SYNC POINT NOT SELECTABLE. DEFAULT=171

FPU ROM FLOW - 44, 35, 167, 173, 157

4949 TEST 73 STCFI*M0*(INT=0)

5F1 BRANCH

THIS BRANCH WILL BE TESTED LATER

BN BRANCH

A MICRO-BREAK TRAP WILL BE SET ON STATE 373 IN-CASE THE EXPONENT SUBTRACTION IN STATE 51 FAILS.

FPU ROM FLOW - 51, 374, 331, 345, 214

4997 TEST 74 LDEXP*-M0*IMM*(EXP=-)

THE ONLY THING THAT SHOULD FAIL IS THE SIGNALS IN ROM STATES 361 OR 227.

FPU ROM FLOW - 15, 10, 260, 361, 227, 236

5030 TEST 75 LDEXP*-M0*(EXP=-200)

IF FRMA FIU(0)B H DOES NOT GO LOW EXECUTION WILL GO FROM STATE 226 TO STATE 205. THIS WILL CAUSE AN INTERRUPT TO OCCUR.

5034

THE ONLY OTHER POSSIBLE FAILURES ARE ROM STATES 226 OR 207.

THE FD BIT WILL BE SET TO ENSURE THE ADX AND A BRANCH ROMS ARE OK.

FPU ROM FLOW - 15, 10, 260, 361, 226, 207

5070 TEST 76 LDEXP*M0*(EXP>177)

THE ONLY POSSIBLE FAILURE IS ROM STATE 70 OR 163.

FPU ROM FLOW - 30, 15, 10, 260, 363, 70, 173

5101 TEST 77 NEG*-M0*INM*(EXP=0)

THE ONLY THING THAT SHOULD FAIL IS ROM STATES 176 OR 31 OF THE ADX ROM.

FPU ROM FLOW - 27, 217, 234, 176, 31, 211

5132 TEST 100 ADDP*M0*(SRC<<DST)

THIS TEST CHECKS THE ADD FLOWS WHEN THE SRC EXPONENT IS MUCH, MUCH LESS THAN THE DST EXPONENT.

NEXT, DIFFERENT VALUES OF EXPONENTS WILL BE CHECKED TO ENSURE THAT THE LOGIC THAT CAUSES FRML SWR(1) TO BE LOW FUNCTIONS PROPERLY.

2F3 BRANCH

IF FRMF ADD*SC<R DOES NOT GO HIGH OR DOES NOT GET TO PAD03 AS A HIGH, EXECUTION WILL GO TO STATE 265. THIS WILL CAUSE THE SRC FRACTION TO BE RIGHT SHIFTED BY 8 AND ADDED TO THE DST. THIS ERROR WOULD MOST LIKELY BE CAUSED BY FXPJ EALU SWR NOT GOING LOW WHICH WOULD MOST LIKELY BE CAUSED BY FXPP OUT

5146

OF RANGE NOT GOING HIGH.

IF FRMF SUB*SC<R DOES NOT GO HIGH OR DOES NOT GET TO PAD02 AS A HIGH, EXECUTION WILL GO TO STATE 271. THIS WILL CAUSE THE SRC FRACTION TO BE RIGHT SHIFTED BY 6 AND SUBTRACTED FROM THE DESTINATION.

IF FXPP SC09(1) DOES NOT GO LOW OR DOES NOT GET TO FRMA AS A LOW EXECUTION WILL GO TO STATE 277. THIS WILL CAUSE THE SRC TO BE PUT IN THE DST.

SF1 BRANCH

IF FXPP OUR OF RANGE DOES NOT GET TO FRMA AS A HIGH, EXECUTION WILL GO TO STATE 225. THIS WILL PROBABLY CAUSE

THE FPP TO HANG IN THE ALIGN LOOP.

IT SHOULD BE REMEMBERED THAT THE FR, IN THIS TEST, IS LOADED FROM THE ABS VAL ROM. ONLY A SMALL PORTION OF THAT ROM IS TESTED HERE. ALL OF IT WILL BE TESTED BY THE END OF THE PROGRAM.

FPU ROM FLOW - 30, 65, 240, 275, 100, 235

5274 TEST 101 ADDP*MO*(SRC>>DST)

THIS TEST CHECKS THE ADD FLOWS WHEN THE SRC EXPONENT IS MUCH, MUCH GREATER THAN THE DST EXPONENT.

2F3 BRANCH

IF FXPP SC09(1) DOES NOT GO HIGH OR DOES NOT GET TO FRMA AS A HIGH EXECUTION WILL GO TO STATE 275. THIS WILL CAUSE THE DST TO REMAIN UNCHANGED.

FPU ROM FLOW - 30, 65, 240, 277, 274, 306

5348 TEST 102 CMPF*-MO*TM*(SPC EXP=DST EXP)*-(SS=SD)

4F2 BRANCH

IF FRMF SD(1) H DOES NOT GET TO FRMA RAD01 AS A LOW EXECUTION WILL GO TO STATE 255. A MICRO-BREAK TRAP WILL SET HERE TO CATCH THIS FAILURE.

FPU ROM FLOW-11,131,67,320,302,257

5384 TEST 103 CMPF*MO*(SRC<DST)*(SS=SD)

4F2 BRANCH

IF FRMF SD(1) DOES NOT GET TO FRMA RAD01 AS A HIGH, EXECUTION WILL GO TO STATE 257. THIS FAILURE WILL BE DETECTED BY SETTING A MICRO-BREAK TRAP AT STATE 257.

5F2 BRANCH

IF FRHE AR59 DOES NOT GO LOW OR DOES NOT GET TO FRMA RAD01 AS A HIGH EXECUTION WILL GO TO STATE 323. THIS FAILURE WILL ALSO BE CAUGHT BY A MICRO-BREAK TRAP.

FPU ROM FLOW-30,67,320,302,255,321

5434 TEST 104 CMPF*MO*(SRC>DST)*(SS=SD)

5F2J BRANCH

IF FRHE AP59(1)L DOES NOT GO HIGH, EXECUTION WILL GO TO STATE 135. THIS WILL BE CAUGHT BY SETTING A MICRO-TRAP. THE ONLY WAY THIS SHOULD HAPPEN IS IF STATE 255 DID NOT SUBTRACT THE ONE PROPERLY (ALU FAILURE).

FPU ROM FLOW-30,67,320,302,255,323

5470 TEST 105 CMPF*-M0*(SPC EXP>DST EXP)
THE ONLY THING THAT SHOULD FAIL IS THE A BRANCH OR ADX RCV
FPU ROM FLOW-11,130,67,320,301

5494 TEST 106 LDCDF*-M0*-(EXP=0)
THE ONLY THING THAT SHOULD FAIL IS ROM STATE 47.
FPU ROM FLOW-17,220,47,46,137,231

5520 TEST 107 STCFI*-M0*(INT=0)
7F1 BRANCH
IF FXPB M0 DOES NOT GET TO PRMA AS A LOW, EXECUTION WILL GO TO STATE
204. THIS WILL CAUSE THE CPU TO DO A DATI INSTEAD OF A DATO.
FPU ROM FLOW-51,374,371,345,210,215

5558 TEST 110 STCFI*M0*(INT>2**15)
3F2 BRANCH
THIS BRANCH WILL ONLY FAIL IF THE EXPONENT SUBTRACTION IN STATE
51 OR 374 FAILED.
FPU ROM FLOW-51,374,333,373,345,214

5594 TEST 111 STCFD*M0*(EXP=0)
THE ONLY THING THAT SHOULD FAIL IS ROM STATE 127.
FPU ROM FLOW-44,35,167,122,123,157

5637 TEST 112 STCFD*-M0*IMM*-(EXP=0)
THE ONLY THING THAT SHOULD FAIL IS THE A BRANCH, OR ADX RCV, OR
STATES 123 OR 153.
FPU ROM FLOW-44,35,167,123,153,145

5662 TEST 113 STCDF*M0*(EXP=0)
THE ONLY THING THAT SHOULD FAIL IS STATE 120.
FPU ROM FLOW-44,35,166,120,123,157

5705 TEST 114 LDEXP*IMM*(EXP<-200)

5706

THE ONLY THING THAT SHOULD FAIL IS THE ADX ROM OR STATE LDE.95.

FPU ROM FLOW-15,10,260,261,227,237,207

5734 TEST 115 ABSF*-M0*(EXP=0)

THE ONLY THING THAT SHOULD FAIL IS THE FP REQUEST COUNTER GETTING THE WRONG CONSTANT IN STATE 176.

FPU ROM FLOW-27,217,224,176,31,211,216

5764 TEST 116 MSN

THIS TEST FIRST CHECKS THE MSN INSTRUCTION AND THEN CHECKS THE AR AND Q SHIFTER LOGIC.

THE SHIFTER LOGIC IS TESTED AS FOLLOWS: THE RIGHT SHIFT IS CHECKED WITH SHIFT COUNTS 8 THRU 1. THIS IS DONE WITH TWO "SOB" LOOPS. THE NESTED LOOP (LOOP2) CONTROLS TESTING OF THE PATTERN AND ITS COMPLIMENT, WHILE THE OUTSIDE LOOP (LOOP1) CONTROLS THE SHIFT COUNT. THE ACTUAL LOOP1 COUNT IS USED AS THE SHIFT COUNT.

THE TEST DATA IS LOADED FROM "COMMON TAGS" INTO A CHECK BUFFER AND INTO AC1 AND AC2. THE DATA CONSISTS OF 8 ONE'S (INCLUDING THE HIDDEN BIT) FOLLOWED BY 8 ZERO'S FOLLOWED BY 8 ONE'S, ETC. FOR THE RIGHT SHIFT BY 8. THE RIGHT SHIFT BY 7 DATA IS 7 ONE'S, FOLLOWED BY 7 ZERO'S, FOLLOWED BY 7 ONE'S, ETC. SHIFT COUNTS 6 THRU 1 FOLLOW THE SAME PATTERN OF BITS. THE DATA IN THE CHECK BUFFER IS THEN CONVERTED TO THE EXPECTED ANSWER. A MICRO-BREAK TRAP IS SET ON ROM STATE 275 AND THE DIFFERENCE IN THE SRC AND DST ACCUMULATORS IS MADE TO BE +300. THE "ADDD" IS THEN EXECUTED AND A TRAP OCCURS LEAVING THE DATA IN THE AR AND THE QR. THE SHIFT IS THEN PERFORMED, THE AR & QR STORED IN MEMORY, AND CHECKED.

ONLY ONE SHIFT COUNT (7) IS TESTED FOR THE LEFT SHIFT. BUT, TWO UNIQUE CONDITIONS ARE TESTED. THE FIRST CONDITION IS WHEN AR58 CLEARS AND AR59 SETS AND THE SECOND CONDITION IS WHEN AR58 AND AR59 BOTH CLEAR. SINCE THESE TWO BITS CANNOT BE EXPLICITLY STORED A RIGHT SHIFT BY 2 IS MADE, AFTER THE LEFT SHIFT, TO CHECK THESE TWO CONDITIONS. THE CHECK DATA FOR LEFT SHIFTS IS STORED IN "COMMON TAGS" RATHER THAN BEING CALCULATED.

NOTE: SYNC POINT ONLY SELECTABLE ON MSN AND STAO AND STQO

FPU ROM FLOW-30,37,163

605P TEST 117 ADDF*M0*(SRC=DST)*-(SS=SD)

2P3 BRANCH

IF PRMF SUB*SC<8 DOES NOT GO LOW OR DOES NOT GET TO PRMA PAD02 AS A HIGH, EXECUTION WILL GO TO STATE 275. A MICRO-BREAK TRAP WILL BE SET TO CATCH THIS FAILURE.

IF PRMF AD*SC<8 DOES NOT GO HIGH, EXECUTION WILL GO TO STATE 261. THIS WILL CAUSE THE NORMALIZED COMPLIMENT OF THE DST TO BE PUT IN THE DST.

THE OTHER BRANCHES IN THE FLOW SHOULD NOT FAIL UNLESS A SIGNAL FROM THE CONTROL STORE FAILS.

FPU ROM FLOW - 30, 65, 240, 271, 74, 263, 311

6141 TEST 120 SUBP*MO*(SRC=DST)*(SS=SD)

THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS THE ZP3 BRANCH. THIS WILL ONLY FAIL IF FRMF SS XOR SD XOR SUB DOES NOT GO LOW.

FPU ROM FLOW - 30, 65, 240, 271, 74, 263, 311

6171 TEST 121 ADD*-MO*-(SRC=0)*(DST=0)

THIS TEST USES THE ADDP AND ADDD INSTRUCTIONS TO TEST THE LDAC6*FD(1) FLOWS. THE DESTINATION WILL BE SET TO ZERO SO THAT THE RESULT WILL BE THE SOURCE.

FPU ROM FLOW-20,140,270,150,65,250,306

6208 TEST 122 CMPF*MO*(SRC=DST)

6210 THE ONLY THING THAT SHOULD FAIL IS THE SUBTRACTION IN ROM STATE 255.

FPU ROM FLOW - 30, 67, 320, 302, 255, 323, 135

6230 TEST 123 STCFI/STCFL*(EXP=2**16/2**32)*(SD=+)

THE ONLY POSSIBLE FAILURE WITH THIS TEST IS THE SUBTRACTION IN ROM STATE 51 OR 374 OR 370.

INTEGER MODE IS TESTED FIRST, AND THEN INTEGER LONG. WHEN INTEGER LONG IS TESTED, THE A BRANCH COULD FAIL.

FPU ROM FLOW - 51, 374/370, 333, 372, 335, 345, 214/204

6285 TEST 124 LDCIF*-MO*IMM*(INTG=0)

SINCE THIS IS THE FIRST TIME THIS INSTRUCTION HAS BEEN TESTED, THE ADX ROM IS CHECKED FIRST AND THEN THE ROM FLOW IS CHECKED. FINALLY, THE DATA IS CHECKED.

FPU ROM FLOW - 41, 174, 213, 314, 312, 253, 245

6356 TEST 125 STCDF*MO*-(EXP=0)

NONE OF THE MICRU-BRANCHES SHOULD FAIL.

IF FRHC FALU 59 DOES NOT GET TO FRHK NORM POS ENCODER AND FRHL A CONTROL 3 MUX AS A LOW, THE EXPONENT WILL INCREASE BY 1.

IF FRHC FALU59 DOES NOT GET TO FRHK NORM POS ENCODER AS A LOW,

THE EXPONENT WILL DECREASE BY 7 AND THE FRACTION WILL BE SHIFTED LEFT BY 7.

IF PRNC PALU59 DOES NOT GET TO PRHL ASHF CONT 3A(0) AS A LOW OR IF PRHM SHIFT CNT 3 DOES NOT GO LOW, THE EXPONENT WILL INCREASE BY 8.

IF A BIT IN PRHK WORN POS ENCODER FAILS TO GO LOW OR FAILS TO GET THROUGH TO PRHM SHIFT CNT OR IF PRHM SHIFT CNT FAILS TO GET TO THE BMX ON FXP, THE EXPONENT WILL BE DECREASED BY A NUMBER BETWEEN 1 AND 7.

IF PRNC PD(1) B L DOES NOT GO HIGH, DRIVEN BY FXPD STCF L, ROUNDING WILL NOT OCCUR

FPU ROM FLOW-44,35,166,121,300,377,157

6448 TEST 126 STCDF*-M0*IMM

IF PRHL ASHF CONT 3A(0) DOES NOT GO HIGH THE EXPONENT WILL BE SMALLER BY 7.

IF A BIT OF PRHK WORN POS ENCODER DOES NOT GO HIGH THE EXPONENT WILL BE LARGER BY A NUMBER BETWEEN 2 AND 8.

FPU ROM FLOW-44,35,166,121,300,377,153,145

6498 TEST 127 STCFI*M0*(EXP=2**15)

IF FXPJ EALU06 XOR EALU03 DOES NOT CAUSE PALU SWR TO GO LOW, THE RESULT WILL BE 52525 SINCE ONLY A RIGHT SHIFT BY 1 WILL GET EXECUTED.

FPU ROM FLOW-51,374,333,371,375,345,214

6537 TEST 130 STCFL*-M0*(EXP=2**31)

THE ONLY THING THAT SHOULD FAIL IS THE SUBTRACTION IN ROM STATE 370 OF THE 7F1 BRANCH.

IF PRMA IL+IMMEDIATE DOES NOT GO LOW, EXECUTION WILL GO TO ROM STATE 210 INSTEAD OF 200. THIS WOULD CAUSE THE CPU NOT TO STORE THE SECOND WORD.

FPU ROM FLOW-51,370,333,371,375,345,200,66,215

6566 TEST 131 ADD*-M0*IMM*(SPC=DST)

THIS TEST DOES AN ADD WITH THE EXPONENTS EQUAL AND THE FRACTIONS EQUAL EXCEPT FOR THE LEAST SIGNIFICANT BIT. THIS IS DONE TO TEST THE ROUND LOGIC AT BIT POSITION 2 OF THE FHX.

2F3 BRANCH

IF PRMF ADD*SC<R DOES NOT GO LOW, EXECUTION WILL GO TO STATE 271.

IF FXPP OUT OF RANGE DOES NOT GO LOW OR DOES NOT

GET TO PXPJ EALU SWP AS A HIGH, EXECUTION
WILL GO TO STATE 275.

THE 3F1 BRANCH, IN THE NORMALIZE FLOWS, SHOULD NOT FAIL SINCE IT
HAS ALREADY BEEN TESTED.
THE 5F2J BRANCH SHOULD NOT FAIL.

NOTE: SYNC POINT NOT SELECTABLE DURING FIRST PART OF
TEST. DEFAULT=265

FPU ROM FLOW-20,141,65,240,265,313,317

6671 TEST 132 SUBD*-M0*(SRC>DST)*-(SS=SD)

2F3 BRANCH

IF PRMF SS XOR SD XOR SUB DOES NOT GO HIGH EXECUTION WILL GO TO STATE
273. A MICRO-BREAK TRAP WILL BE SET TO CATCH THIS FAILURE.

IF THE ABS VAL ROM FAILS TO PUT 177 IN THE EALU, EXECUTION
WILL GO TO STATE 277.

THE FT BIT WILL BE SET TO TEST THAT PPLC PNX02 IS DISABLED.

FPU ROM FLOW-30,65,240,267,313,317

6739 TEST 133 ADDD*M0*(SRC<DST)*-(SS=SD)

THE 2F3 BRANCH WILL ONLY FAIL IF THE ABS VAL ROM FAILS. THIS
WOULD CAUSE EXECUTION TO GO TO STATE 275 INSTEAD OF 271.

LIKEWISE, THE 3F0 BRANCH WILL ONLY FAIL IF THE ABS VAL ROM FAILS
AND PUTS ZERO IN THE ER.

ANY OTHER FAILURE WOULD BE CAUSED BY CONTROL STORE SIGNALS IN STATES
271 OR 75.

A MODE 0 INSTRUCTION IS USED TO TEST THE A-BRANCH ROM.

FPU ROM FLOW-30,65,240,271,75,317

6796 TEST 134 ADDP*-M0*(SRC>DST)*-(SS=SD)

THE ONLY THING THAT SHOULD FAIL IS THE CONTROL STORE SIGNALS IN ROM STATES
273, 272, OR 261.

THE IL BIT IS SET TO TEST THE NO-MEM AND ADX ROMS.

FPU ROM FLOW-11,130,65,240,273,272,261,313,317

6847 TEST 135 NORMALIZATION SHIPT ENCODER

THIS TEST IS COMPOSED OF THREE SUBTESTS. THE FIRST SUBTEST RUNS A COUNT PATTERN THROUGH AR BITS <57:51> WITH AR BITS <59:58>00.

THE SECOND SUBTEST RUNS A COUNT PATTERN THROUGH AR BITS <57:51> WITH AR BITS <59:58>01.

THE THIRD SUBTEST TESTS THE UNIQUE DATA PATTERN WHERE AR BITS <59:51> ARE ALL ZERO. THIS IS THE ONLY PATTERN THAT CAUSES FRNK NORM POS SWR TO GO LOW.

IF AN ERROR IS DETECTED, LOOP ON ERPOP WILL CAUSE THE FAILING DATA PATTERN TO BE LOOPED ON. THE ERROR TYPEOUT INCLUDES THE AR DATA PATTERN (BITS <59:51>) THAT IS TRYING TO BE NORMALIZED.

REFER TO THE FLOW CHART AT THE END OF THIS LISTING FOR FURTHER INFORMATION.

FPU ROM FLOW-11,130,65,240,271,75,317

7013 TEST 136 ADDD*NO*(EXP DIFF=30)

2F3 BRANCH

IF PXPJ EALU06 XOR EALU04 DOES NOT CAUSE EALU SWR TO GO LOW, EXECUTION WILL GO TO STATE 265. THIS WILL BE CAUGHT BY SETTING A MICRO-BREAK TRAP ON STATE 265.

5F1 BRANCH

IF PXPJ OUT OF RANGE DOES NOT GET TO PRMA AS A HIGH OR DOES NOT GET TO PRMA RADO3 AS A LOW, EXECUTION WILL GO TO STATE 235. THIS WILL CAUSE THE DST TO BE UNCHANGED.

THE PD BIT HAS TO BE ON A ONE TO CAUSE THE ABS VAL ROM TO OUTPUT A HIGH ON THE MOST SIGNIFICANT BIT SO THAT "OUT OF RANGE" WILL GO LOW. IF THIS ROM SIGNAL FAILS, THE 5F1 BRANCH WILL ALSO FAIL.

THE IL BIT IS SET TO TEST THE A-BRANCH AND ADX ROMS ON ADDD.

FPU ROM FLOW-20,140,270,150,65,240,2(275,100,225),
265,313,317

7104 TEST 137 ADDD*NO*(FXP DIFF=50)

7106

THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS PXPJ EALU06 XOR EALU04. THIS WILL CAUSE THE 2F3 BRANCH TO GO TO STATE 265 INSTEAD OF 275.

THE ABS VAL ROM COULD ALSO FAIL AND NOT PUT 330 IN THE ER.

A MODE 0 INSTRUCTION IS USED WITH THE IL BIT ON TO FURTHER TEST THE

A-BRANCH ROM.

FPU ROM FLOW-30,65,740,5(775,100,725),265,313,317

7184 TEST 140 ADDP*MO*(EXP DIFF=-9)

THE 2F3 AND AND 5F1 BRANCHES WILL ONLY FAIL IF THE ABS VAL ROM FAILS. THE ONLY OTHER POSSIBLE FAILURE IS ROM STATE 264.

IF THE 2F3 BRANCH FAILS THE DST WILL ONLY GET SHIFTED ONE PLACE AND ADDED TO THE SRC.
IF THE 5F1 BRANCH FAILS THE RESULT WILL BE THE SAME AS THE SRC.

FPU ROM FLOW-30,65,240,277,264,267,313,317

7226 TEST 141 GUARD BIT TEST

THIS TEST CHECKS THE GUARD BITS IN THE AR AND QR DATA PATH

7301 TEST 142 LDCIF*MO*(INT=+)

THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS CONTROL STORE SIGNALS IN ROM STATES 253, 247, OR THE EXPONENT LOAD IN STATE 314.

FPU ROM FLOW-41,174,213,314,312,253,247,317

7326 TEST 143 LDCIF*MO*(INT=-)

THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS THE CONTROL STORE SIGNALS IN STATE 251 OR THE A OR NO-MEM ROMS.

7330 FPU ROM FLOW-30,40,174,213,314,312,251,313,317

7355 TEST 144 LDCIF*MO*(INT=+)

THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS ROM STATE 310 OR THE A-BRANCH ROM OR THE ADX ROM.

FPU ROM FLOW-41,164,174,213,310,312,253,247,317

7386 TEST 145 STCDF*MO

THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS THE NORMALIZE SHIFT IN STATE 121 OR STATE 144.

FPU ROM FLOW-44,35,166,121,300,377,153,144,154

7418 TEST 146 STCFD*MO

THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS THE "CLEAR AR LOW" SIGNAL IN STATE 167 OR STATES 155 OR 156.

FPU ROM FLOW-44,35,167,123,153,144,155,156

7454 TEST 147 STCDF*-M0*FV

5FO BRANCH

IF FXPJ EALU 08 DOES NOT GET TO FRMB BOU AS A HIGH OR
FRMB BOU DOES NOT CAUSE FRMA BOU+BZ TO GO LOW OR IF
FRMA BOU+BZ DOES NOT GET TO RAD00 AS A HIGH, EXECUTION
WILL GO TO STATE 377. THIS WILL CAUSE THE RESULT TO BE
STORED INSTEAD OF ZEROED.

THE IL BIT IS SET TO TEST THE A-BRANCH & ADX ROMS.

FPU ROM FLOW-44,35,166,121,300,376,127,125,153,144,154

7497 TEST 150 LDCDF*-M0*AR59

7498

THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS THE EXPONENT AND
SHIFT CONTROL SUBTRACTION IN STATE 231 OR STATE 161.

THE IL BIT IS SET TO TEST THE A-BRANCH AND ADX ROMS.

FPU ROM FLOW-17,220,47,46,137,231,161

7529 TEST 151 LDCDF*M0*BOU

THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS ROM STATES
162,243.

THE IL BIT IS SET TO TEST THE A-BRANCH ROM.

FPU ROM FLOW-30,46,137,231,161,162,243,70,173

7556 TEST 152 ADDF*-M0*BOU

THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS THE ROUND IN
STATE 313 OR THE EXPONENT MINUS SHIFT CONTROL IN STATE 317.

FPU ROM FLOW-11,130,65,240,265,313,317,161,162,243,70,173

7581 TEST 153 SUBP*-M0*(RESULT=0)

THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS STATE 160.

THE IL BIT IS SET TO TEST THE A-BRANCH AND ADX ROMS.

FPU ROM FLOW-11,130,65,240,271,74,263,313,317,160,162,243

7625 TEST 154 SUBD*-M0*IMM*-AR59*BOU

THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS THE 3F2 BRANCH
AFTER STATE 162.

FPU ROM FLOW-20,140,270,150,65,240,271,74,263,313,317,162,241,207

7665 TEST 155 STCFI*-M0*(INT=-1)

7666

THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS ROM STATES 347 OR 103 OR FXM BIT 35 OR FXM BIT 19.

FPU ROM FLOW-51,374,333,371,375,347,103,210,215

7696 TEST 156 STCFL*-M0*(INT=-1)

THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS FXM75 OR FXM19.

FPU ROM FLOW-51,370,333,371,375,347,103,345,200,66,215

7725 TEST 157 STCFI*M0*(EXP=2**16)*(SD=-)

THIS TEST FIRST CHECKS THE QR FILL LOGIC. THIS IS DONE BY FLOATING A "ONE" FROM BIT POSITION 43 TO BIT 57. IF QR FILL DOES NOT FILL PROPERLY THE RESULT WILL NOT BE ZERO.

A ONE IS THEN FLOATED FROM BIT 35 TO 42 TO CHECK THE LOGICAL AND FUNCTION OF THE ALU (SINGLE PRECISION).

FPU ROM FLOW-51,374,333,372,337,256,151,357,322,316,103,345,214

7784 TEST 160 STCDL*-M0*(EXP=7**32)*(SD=-)

THIS TEST FIRST ENSURES THAT THE ADX ROM IS OK. IT THEN FLOATS A "ONE" FROM BIT 27 TO 42 TO ENSURE THAT THE QR MASK WAS FORMED PROPERLY. IT THEN FLOATS A ONE FROM BIT 3 TO BIT 26 TO TEST THE LOGICAL AND OF THE FALU.

FPU ROM FLOW-51,370,333,372,337,252,353,256,151,357,322,316,103,345,200,66,215

7859 TEST 161 STCDL*(INT=-2)

THIS TEST INSURES THAT FRLC FXM02 GOES LOW WITH FXPR INTEGER INCREMENT OF A LOW, FD(1) HIGH, AND FT(1) LOW.

7861 TEST 162 ABS VAL ROM (EXP DIFF = +)

THIS TEST CHECKS THE CONTENTS OF THE ABS VAL ROM FOR ADDRESSES 001 THRU 077 AND 201 THRU 277.

IF A FAILURE IS DETECTED THE ROM ADDRESS BEING TESTED AND THE EXPECTED CONTENTS OF THE ROM ARE TYPED OUT.

8017 TEST 163 ABS VAL ROM (DIFF = -)

THIS TEST CHECKS THE CONTENTS OF THE ABS VAL ROM FOR ADDRESSES 100 THRU 177 AND 300 THRU 377

8150

END OF PASS ROUTINE

8153 INCREMENT THE PASS NUMBER (SPASS)
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

8199

SCOPE HANDLER ROUTINE

8202 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=INT

8277

ERROR HANDLER ROUTINE

8280 THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL

8282 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

8340

CONVERT FLOATING BINARY TO OCTAL ASCII

8341

THIS ROUTINE CONVERTS A 32 BIT FLOATING NUMBER TO AN OCTAL
ASCII 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 ASCII STRING ON THE STACK.

8413

CONVERT FLOATING DOUBLE BINARY TO OCTAL ASCII

8414

THIS ROUTINE CONVERTS A 64 BIT FLOATING NUMBER TO AN OCTAL
ASCII 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 ASCII STRING ON THE STACK.

8466 *****
 ROUTINE TO START THE LINE CLOCK

8467
 THIS ROUTINE SETS UP THE LINE CLOCK TO GUARANTEE THE
 MAXIMUM DELAY BEFORE IT WILL INTERRUPT.

8497 *****
 SAVE AND RESTORE R0-R5 ROUTINES

8500 SAVE R0-R5
 CALL:
 SSAVREG
 UPON RETURN FROM SSAVREG THE STACK WILL LOOK LIKE:

TOP---(+16)

8506 +2---(+18)
 +4---R5
 +6---R4
 +8---R3
 +10---R2
 +12---R1
 +14---R0

8527 RESTORE R0-R5
 CALL:
 RESREG

8543 *****
 TYPE ROUTINE

8546 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
 MESADP

8623 *****
 APT COMMUNICATIONS ROUTINE

8682 *****
 ERROR MESSAGE TIMEOUT ROUTINE

8684 THIS ROUTINE USES THE "ITEM CONTROL BYTE" (\$ITEMB) TO DETERMINE WHICH
 ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" (\$ERRTB),
 AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.

8688 THE DATA FORMATS FOR THIS PROGRAM ARE:
 0 16-BIT BINARY TO 6 DIGIT OCTAL
 1 32-BIT FLOATING TO 13-DIGIT OCTAL
 2 64-BIT FLOATING TO 75-DIGIT OCTAL
 3 BITS <15:7> TO 3-DIGIT OCTAL

8905 *****
 BINARY TO OCTAL (ASCII) AND TYPE

8908 THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
 OCTAL (ASCII) NUMBER AND TYPE IT.
 \$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
 CALL:

```

MOV     NUM, -(SP)      ;;NUMBER TO BE TYPED
TYPOS   TYPON          ;;CALL FOR TYPEOUT
.BYTE  N                ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
.BYTE  M                ;;M=1 OR 0
                        ;;1=TYPE LEADING ZEROS
                        ;;0=SUPPRESS LEADING ZEROS
    
```

\$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
 \$TYPOS OR \$TYPOC
 CALL:

```

MOV     NUM, -(SP)      ;;NUMBER TO BE TYPED
TYPON   TYPON          ;;CALL FOR TYPEOUT
    
```

\$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
 CALL:

```

MOV     NUM, -(SP)      ;;NUMBER TO BE TYPED
TYPOC   TYPON          ;;CALL FOR TYPEOUT
    
```

8883 *****
CONVERT BINARY TO DECIMAL AND TYPF ROUTINE

8886 THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
REPLACED WITH SPACES.
CALL:

```
MOV     NUM,-(SP)    ;;PUT THE BINARY NUMBER ON THE STACK.  
TYPDS                      ;;GO TO THE ROUTINE
```

8951 *****
DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE

8954 THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
UNSIGNED OCTAL ASCII NUMBER.
CALL

```
MOV     @PTR,-(SP)    ;;POINTER TO LOW WORD OF BINARY NUMBER  
JSR     PC,@$DR20    ;;CALL THE ROUTINE  
RETURN                      ;;THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK
```

8991 *****
UNEXPECTED TRAP TO 4 ROUTINE

9005 *****
UNEXPECTED TRAP TO 114 ROUTINE

9020 *****
UNEXPECTED TRAP TO 244 ROUTINE

9028 *****
TRAP DECODER

9031 THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
GO TO THAT ROUTINE.

9045 *****
TRAP TABLE

9047 THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
BY THE "TRAP" INSTRUCTION.

9063 *****
POWER DOWN AND UP ROUTINES

47	OPERATIONAL SWITCH SETTINGS
60	BASIC DEFINITIONS
187	CACHE REGISTER DEFINITIONS
198	CPU REGISTER DEFINITIONS
212	MEMORY MANAGEMENT DEFINITIONS
361	UNIBUS MAP REGISTER DEFINITIONS
448	FLOATING POINT REGISTER DEFINITIONS
462	FLOATING POINT STATUS BIT DEFINITIONS
478	FLOATING POINT EXCEPTION CODE DEFINITIONS
487	FLOATING POINT VECTOR DEFINITION
492	TRAP CATCHER
502	STARTING ADDRESS(PS)
505	ACT11 HOOKS
516	COMMON TAGS
577	APT MAILBOX-ETABLE
685	ERROR POINTER TABLE
1877	APT PARAMETER BLOCK
1962	T1 LDFPS*MO
2038	T2 INTERFACE DATA PATH
2152	T3 STFPS*MO
2228	T4 SET NODES
2299	T5 MICRO-BREAK TRAP
2344	T6 CPCC
2365	T7 LDF*-MO*IMM
2431	T10 STP*-MO*IMM
2497	T11 LDF*STP*MO
2609	T12 LDF*STP*-MO*-IMM
2686	T13 HIGH ORDER DATA PATH
2745	T14 HIGH ORDER SCRATCH PAD DUAL ADDRESSING
2798	T15 LDFPS/STFPS*-MO
2833	T16 CLRF*MO
2923	T17 TSTP*MO
2962	T20 CLRF*-MO*IMM
2998	T21 CLRF*-MO*-IMM
3026	T22 ABSF*MO*-(EXP=0)
3107	T23 NEGF*MO*-(EXP=0)
3149	T24 ADD*MO*(SRC=0)*-(DST=0)
3200	T25 ADD*MO*(SRC=0)*(DST=0)
3229	T26 SUB*MO*(SRC=0)*(DST=0)
3277	T27 CMP*MO*(SRC=0)*-(DST=0)
3326	T30 CMP*MO*-(SRC=0)*(DST=0)
3347	T31 CMP*MO*(SRC=0)*(DST=0)
3373	
3374	FOUR ROM STATES
3375	
3377	T32 LDEXP*-MO*(EXP=+)
3473	T33 ABSF*MO*(SRC=0)
3500	T34 NEGF*MO*(SRC=0)
3520	T35 ADD*MO*-(SRC=0)*(DST=0)
3565	T36 SUB*MO*-(SRC=0)*(DST=0)
3596	T37 CMP*MO*(SRC>DST)
3678	T40 CMP*MO*(SRC<DST)
3736	T41 LDD/STD*-MO*-IMM
3792	T42 LOW ORDER DATA PATH
3844	T43 LOW ORDER SCRATCH PAD DUAL ADDRESSING
3929	T44 STAO*STOQ

4004	T45	QR DATA PATH
4128	T46	DISABLE LOW FMX
4175	T47	DISABLE LOW QMX
4218	T50	ACDC3,0)COND_ACMX
4251	T51	LDD*-M0*IMM
4303	T52	LDCFD*M0*-(EXP=0)
4350	T53	LDCFD*M0*(EXP=0)
4395	T54	LDCDF*M0*(EXP=0)
4434	T55	DISABLE LOW & HI FMX ON ROUND
4495	T56	LDCDF*M0*-(EXP=0)*-AR59*-ROU
4565	T57	LDCFD*-M0*IMM*-(EXP=0)
4597	T60	LDCFD*-M0*-IMM*-(EXP=0)
4630		
4631		FIVE ROM STATES
4632		
4634	T61	LDEXP*M0*(EXP=+)
4655	T62	STEXP
4727	T63	CLRD*-M0*-IMM
4767	T64	ABSF*-M0*-(EXP=0)
4795	T65	NEGF*-M0*-(EXP=0)
4824	T66	ADDF*-M0*-(SRC=0)*(DST=0)
4850	T67	ADDF*-M0*IMM*-(SRC=0)*(DST=0)
4889	T70	SUBP*-M0*-(SRC=0)*(DST=0)
4924	T71	SUBP*-M0*IMM*-(SRC=0)*(DST=0)
4944	T72	STCFD*M0*-(EXP=0)
5007	T73	STCFI*M0*(INT=0)
5055		
5056		SIX ROM STATES
5057		
5059	T74	LDEXP*-M0*IMM*(EXP=-)
5093	T75	LDEXP*-M0*(EXP=-200)
5134	T76	LDEXP*M0*(EXP>177)
5166	T77	NEGF*-M0*IMM*(EXP=0)
5198	T100	ADDF*M0*(SRC<<DST)
5341	T101	ADDF*M0*(SRC>>DST)
5416	T102	CMPP*-M0*IMM*(SRC EXP=DST EXP)*-(SS=SD)
5453	T103	CMPP*M0*(SRC<DST)*(SS=SD)
5504	T104	CMPP*M0*(SRC>DST)*(SS=SD)
5541	T105	CMPP*-M0*(SRC EXP>DST EXP)
5566	T106	LDCDF*-M0*-(EXP=0)
5593	T107	STCFI*-M0*(INT=0)
5632	T110	STCFI*M0*(INT>2**15)
5669	T111	STCFD*M0*(EXP=0)
5713	T112	STCFD*-M0*IMM*-(EXP=0)
5739	T113	STCDF*M0*(EXP=0)
5787		
5783		SEVEN ROM STATES
5784		
5786	T114	LDEXP*IMM*(EXP<-200)
5916	T115	ABSF*-M0*(EXP=0)
5847	T116	MSN
6142	T117	ADDF*M0*(SRC=DST)*-(SS=SD)
6226	T120	SUBP*M0*(SRC=DST)*(SS=SD)
6257	T121	ADDD*-M0*-(SRC=0)*(DST=0)
6295	T122	CMPP*M0*(SRC=DST)
6318	T123	STCFI/STCFL*(EXP=2**16/2**32)*(SD=+)

6374	T124	LDCIF*-M0*IMM*(INTG=0)
6446	T125	STCDF*M0*-(EXP=0)
6539	T126	STCDF*-M0*IMM
6590	T127	STCFI*M0*(EXP=2**15)
6625	T130	STCPL*-M0*(FXP=2**31)
6660	T131	ADD*-M0*IMM*(SRC=DST)
6766	T132	SUBD*-M0*(SRC>DST*(-(SS=SD)
6834	T133	ADD*M0*(SRC<DST)*-(SS=SD)
6893	T134	ADDF*-M0*(SRC>DST)*-(SS=SD)
6945	T135	NORMALIZATION SHIFT ENCODER
7112	T136	ADD*-M0*(EXP DIFF=30)
7204	T137	ADD*M0*(EXP DIFF=50)
7285	T140	ADDF*M0*(EXP DIFF=-9)
7328	T141	GUARD BIT TEST
7404	T142	LDCIF*-M0*(INT=+)
7430	T143	LDCIF*M0*(INT=-)
7460	T144	LDCLF*-M0*(INT=+)
7492	T145	STCDF*-M0
7525	T146	STCFD*-M0
7567	T147	STCDF*-M0*FV
7606	T150	LDCDF*-M0*AR59
7639	T151	LDCDF*M0*BOU
7667	T152	ADDF*-M0*BOU
7693	T153	SUBF*-M0*(RESULT=0)
7738	T154	SUBD*-M0*IMM*-AR59*BOU
7779	T155	STCFI*-M0*(INT=-1)
7811	T156	STCPL*-M0*(INT=-1)
7841	T157	STCFI*M0*(EXP=2**16)*(SD=-)
7901	T160	STCDL*-M0*(EXP=2**32)*(SD=-)
7977	T161	STCDL*(INT=-2)
8000	T162	ABS VAL ROM (EXP DIFF = +)
8137	T163	ABS VAL ROM (EXP DIFF = -)
8272		END OF PASS ROUTINE
8321		SCOPE HANDLER ROUTINE
8400		ERROR HANDLER ROUTINE
8463		CONVERT FLOATING BINARY TO OCTAL ASCII
8536		CONVERT FLOATING DOUBLE BINARY TO OCTAL ASCII
8589		ROUTINE TO START THE LINE CLOCK
8620		SAVE AND RESTORE R0-R5 ROUTINES
8666		TYPE ROUTINE
8746		APT COMMUNICATIONS ROUTINE
8805		ERROR MESSAGE TYPEOUT ROUTINE
8928		BINARY TO OCTAL (ASCII) AND TYPE
9006		CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
9074		DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
9114		UNEXPECTED TRAP TO 4 ROUTINE
9128		UNEXPECTED TRAP TO 114 ROUTINE
9143		UNEXPECTED TRAP TO 244 ROUTINE
9151		TRAP DECODER
9168		TRAP TABLE
9186		POWER DOWN AND UP ROUTINES

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29

.TITLE PDP 11/45/55/70...FP11-C DIAGNOSTIC PART 1
;*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.
;*

30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57

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

```
58
59      .SBTTL  BASIC DEFINITIONS
60
61      ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
62      001100  STACK= 1100      ;;FIRST ADDRESS OF THE STACK
63      001100  KERSTK= STACK    ;;KERNEL STACK
64      000700  SUPSTK= STACK-200 ;;SUPERVISOR STACK
65      000600  USESTK= STACK-300 ;;USER STACK
66      .EQUIV  EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
67      .EQUIV  IOT,SCOPE      ;;BASIC DEFINITION OF SCOPE CALL
68      177776  PS= 177776     ;;PROCESSOR STATUS WORD
69      .EQUIV  PS,PSW
70      177774  STKLMT= 177774  ;;STACK LIMIT REGISTER
71      177772  PTRQ= 177772    ;;PROGRAM INTERRUPT REQUEST REGISTER
72      177570  DSNR= 177570    ;;HARDWARE SWITCH REGISTER
73      177570  DDISP= 177570  ;;HARDWARE DISPLAY REGISTER
74      177546  LKS= 177546    ;;LINE CLOCK (KW11-L) STATUS REGISTER
75
76      ;*MISCELLANEOUS DEFINITIONS
77      000011  HT= 11          ;;CODE FOR HORIZONTAL TAB
78      000012  LF= 12          ;;CODE LINE FEED
79      000015  CR= 15          ;;CODE CARRIAGE RETURN
80      000200  CPLF= 200      ;;CODE FOR CARRIAGE RETURN-LINE FEED
81
82      ;*GENERAL PURPOSE REGISTER DEFINITIONS
83      000000  R0= 0           ;;GENERAL REGISTER
84      000001  R1= 1           ;;GENERAL REGISTER
85      000002  R2= 2           ;;GENERAL REGISTER
86      000003  R3= 3           ;;GENERAL REGISTER
87      000004  R4= 4           ;;GENERAL REGISTER
88      000005  R5= 5           ;;GENERAL REGISTER
89      000006  R6= 6           ;;GENERAL REGISTER
90      000007  R7= 7           ;;GENERAL REGISTER
91      .EQUIV  R0,R10          ;;GENERAL REGISTER
92      .EQUIV  R1,R11          ;;GENERAL REGISTER
93      .EQUIV  R2,R12          ;;GENERAL REGISTER
94      .EQUIV  R3,R13          ;;GENERAL REGISTER
95      .EQUIV  R4,R14          ;;GENERAL REGISTER
96      .EQUIV  R5,R15          ;;GENERAL REGISTER
97      000006  SP= 6           ;;STACK POINTER
98      .EQUIV  SP,KSP          ;;KERNEL STACK POINTER
99      .EQUIV  SP,SSP          ;;SUPERVISOR STACK POINTER
100     .EQUIV  SP,USP          ;;USER STACK POINTER
101     000007  PC= 7           ;;PROGRAM COUNTER
102
103     ;*PRIORITY LEVEL DEFINITIONS
104     000000  PR0= 0           ;;PRICRITY LEVEL 0
105     000040  PR1= 40          ;;PRIORITY LEVEL 1
106     000100  PR2= 100         ;;PRIORITY LEVEL 2
107     000140  PR3= 140         ;;PRIORITY LEVEL 3
108     000200  PR4= 200         ;;PRIORITY LEVEL 4
109     000240  PR5= 240         ;;PRIORITY LEVEL 5
110     000300  PR6= 300         ;;PRIORITY LEVEL 6
111     000340  PR7= 340         ;;PRIORITY LEVEL 7
112
113     ;*"SWITCH REGISTER" SWITCH DEFINITIONS
```

114	100000	SW15=	100000
115	040000	SW14=	40000
116	020000	SW13=	20000
117	010000	SW17=	10000
118	004000	SW11=	4000
119	002000	SW10=	2000
120	001000	SW09=	1000
121	000400	SW08=	400
122	000200	SW07=	200
123	000100	SW06=	100
124	000040	SW05=	40
125	000020	SW04=	20
126	000010	SW03=	10
127	000004	SW02=	4
128	000002	SW01=	2
129	000001	SW00=	1
130		.EQUIV	SW09,SW9
131		.EQUIV	SW08,SW8
132		.EQUIV	SW07,SW7
133		.EQUIV	SW06,SW6
134		.EQUIV	SW05,SW5
135		.EQUIV	SW04,SW4
136		.EQUIV	SW03,SW3
137		.EQUIV	SW02,SW2
138		.EQUIV	SW01,SW1
139		.EQUIV	SW00,SW0

141		;*DATA BIT DEFINITIONS (BIT00 TO BIT15)	
142	100000	BIT15=	100000
143	040000	BIT14=	40000
144	020000	BIT13=	20000
145	010000	BIT12=	10000
146	004000	BIT11=	4000
147	002000	BIT10=	2000
148	001000	BIT09=	1000
149	000400	BIT08=	400
150	000200	BIT07=	200
151	000100	BIT06=	100
152	000040	BIT05=	40
153	000020	BIT04=	20
154	000010	BIT03=	10
155	000004	BIT02=	4
156	000002	BIT01=	2
157	000001	BIT00=	1
158		.EQUIV	BIT09,BIT9
159		.EQUIV	BIT08,BIT8
160		.EQUIV	BIT07,BIT7
161		.EQUIV	BIT06,BIT6
162		.EQUIV	BIT05,BIT5
163		.EQUIV	BIT04,BIT4
164		.EQUIV	BIT03,BIT3
165		.EQUIV	BIT02,BIT2
166		.EQUIV	BIT01,BIT1
167		.EQUIV	BIT00,BIT0

168
 169 ;*BASIC "CPU" TRAP VECTOR ADDRESSES

A5

```

170      000004      ERRVEC= 4      ;;TIME OUT AND OTHER ERRORS
171      000010      RESVEC= 10     ;;RESERVED AND ILLEGAL INSTRUCTIONS
172      000014      TBITVEC=14    ;;TM BIT
173      000014      TRTVEC= 14     ;;TRACE TRAP
174      000014      BPTVEC= 14     ;;BREAKPOINT TRAP (BPT)
175      000020      IOTVEC= 20     ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
176      000024      PWRVEC= 24     ;;POWER FAIL
177      000030      EMTVEC= 30     ;;EMULATOR TRAP (EMT) **ERROR**
178      000034      TRAPVEC=34     ;;"TRAP" TRAP
179      000060      TKVEC= 60      ;;TTY KEYBOARD VECTOR
180      000064      TPVEC= 64      ;;TTY PRINTER VECTOR
181      000100      LKVEC= 100     ;;LINE CLOCK (KW11-L) VECTOR
182      000114      CACHVEC=114   ;;CACHE ERROR INTERRUPT VECTOR
183      000240      PIQVEC=240    ;;PROGRAM INTERRUPT REQUEST VECTOR
184      000250      MMVEC= 250     ;;MEMORY MANAGEMENT VECTOR
  
```

.SBTTL CACHE REGISTER DEFINITIONS

```

189      177740      LOADRS = 177740 ;;LOWER 16 BITS OF ADDRESS THAT CAUSED ERROR
190      177742      HIADRS = 177742 ;;UPPER SIX BITS OF ADDRESS THAT CAUSED ERROR
191      177744      MEMERR = 177744 ;;CACHE ERROR REGISTER
192      177746      CONTRL = 177746 ;;MEMORY CONTROL REGISTER
193      177750      MAINT = 177750  ;;MEMORY MAINTENANCE REGISTER
194      177752      HITMIS = 177752 ;;HIT MISS REGISTER "1" IMPLIES HIT IN CACHE
  
```

.SBTTL CPU REGISTER DEFINITIONS

```

200      177760      SIZELO = 177760 ;;MEMORY SIZE REGISTER NUMBER TO PUT INTO A PAR
201      177762      SIZEHI = 177762 ;;TO GET TO THE LAST 32 WORDS OF MEMORY
202      177764      CPUERR = 177764 ;;HIGH SIZE REGISTER, RESERVED FOR FUTURE USE
203      177766      SYSTID = 177764 ;;CURRENTLY ALL ZERO
204      177768      CPUERR = 177764 ;;SYSTEM ID REGISTER
205      177770      CPUERR = 177766 ;;CPU ERROR REGISTER HOLDS CONDITION THAT CAUSED
206      177772      CPUERR = 177766 ;;THE TRAP TO ERRVEC (000004)
  
```

.SBTTL MEMORY MANAGEMENT DEFINITIONS

;*MEMORY MANAGEMENT STATUS REGISTER ADDRESSES

```

216      177572      MMR0= 177572
217      177574      MMR1= 177574
218      177576      MMR2= 177576
219      172516      MMR3= 172516
220      .EQUIV MMR0,SR0
221      .EQUIV MMR1,SR1
222      .EQUIV MMR2,SR2
223      .EQUIV MMR3,SR3
  
```

;*USER "I" PAGE DESCRIPTOR REGISTERS

226		
227	177600	UIPDR0= 177600
228	177602	UIPDR1= 177602
229	177604	UIPDR2= 177604
230	177606	UIPDR3= 177606
231	177610	UIPDR4= 177610
232	177612	UIPDR5= 177612
233	177614	UIPDR6= 177614
234	177616	UIPDR7= 177616
235		
236		;*USER "D" PAGE DESCRIPTOR REGISTERS
237		
238	177620	UDPDR0= 177620
239	177622	UDPDR1= 177622
240	177624	UDPDR2= 177624
241	177626	UDPDR3= 177626
242	177630	UDPDR4= 177630
243	177632	UDPDR5= 177632
244	177634	UDPDR6= 177634
245	177636	UDPDR7= 177636
246		
247		;*USER "I" PAGE ADDRESS REGISTERS
248		
249	177640	UIPAR0= 177640
250	177642	UIPAR1= 177642
251	177644	UIPAR2= 177644
252	177646	UIPAR3= 177646
253	177650	UIPAR4= 177650
254	177652	UIPAR5= 177652
255	177654	UIPAR6= 177654
256	177656	UIPAR7= 177656
257		
258		;*USER "D" PAGE ADDRESS REGISTERS
259		
260	177660	UDPAR0= 177660
261	177662	UDPAR1= 177662
262	177664	UDPAR2= 177664
263	177666	UDPAR3= 177666
264	177670	UDPAR4= 177670
265	177672	UDPAR5= 177672
266	177674	UDPAR6= 177674
267	177676	UDPAR7= 177676
268		
269		;*SUPERVISOR "I" PAGE DESCRIPTOR REGISTERS
270		
271	172200	SIPDR0= 172200
272	172202	SIPDR1= 172202
273	172204	SIPDR2= 172204
274	172206	SIPDR3= 172206
275	172210	SIPDR4= 172210
276	172212	SIPDR5= 172212
277	172214	SIPDR6= 172214
278	172216	SIPDR7= 172216
279		
280		;*SUPERVISOR "D" PAGE DESCRIPTOR REGISTERS
281		

282	172220	SDPDR0= 172220
283	172222	SDPDR1= 172222
284	172224	SDPDR2= 172224
285	172226	SDPDR3= 172226
286	172230	SDPDR4= 172230
287	172232	SDPDR5= 172232
288	172234	SDPDR6= 172234
289	172236	SDPDR7= 172236
290		
291		;*SUPERVISOR "I" PAGE ADDRESS REGISTERS
292		
293	172240	SIPAR0= 172240
294	172242	SIPAR1= 172242
295	172244	SIPAR2= 172244
296	172246	SIPAR3= 172246
297	172250	SIPAR4= 172250
298	172252	SIPAR5= 172252
299	172254	SIPAR6= 172254
300	172256	SIPAR7= 172256
301		
302		;*SUPERVISOR "D" PAGE ADDRESS REGISTERS
303		
304	172260	SDPAR0= 172260
305	172262	SDPAR1= 172262
306	172264	SDPAR2= 172264
307	172266	SDPAR3= 172266
308	172270	SDPAR4= 172270
309	172272	SDPAR5= 172272
310	172274	SDPAR6= 172274
311	172276	SDPAR7= 172276
312		
313		;*KERNEL "I" PAGE DESCRIPTOR REGISTERS
314		
315	172300	KIPDR0= 172300
316	172302	KIPDR1= 172302
317	172304	KIPDR2= 172304
318	172306	KIPDR3= 172306
319	172310	KIPDR4= 172310
320	172312	KIPDR5= 172312
321	172314	KIPDR6= 172314
322	172316	KIPDR7= 172316
323		
324		;*KERNEL "D" PAGE DESCRIPTOR REGISTERS
325		
326	172320	KDPDR0= 172320
327	172322	KDPDR1= 172322
328	172324	KDPDR2= 172324
329	172326	KDPDR3= 172326
330	172330	KDPDR4= 172330
331	172332	KDPDR5= 172332
332	172334	KDPDR6= 172334
333	172336	KDPDR7= 172336
334		
335		;*KERNEL "I" PAGE ADDRESS REGISTERS
336		
337	172340	KIPAR0= 172340

338	172342	KIPAR1= 172342
339	172344	KIPAR2= 172344
340	172346	KIPAR3= 172346
341	172350	KIPAR4= 172350
342	172352	KIPAR5= 172352
343	172354	KIPAR6= 172354
344	172356	KIPAR7= 172356

;*KERNEL "D" PAGE ADDRESS REGISTERS

348	172360	KDPAR0= 172360
349	172362	KDPAR1= 172362
350	172364	KDPAR2= 172364
351	172366	KDPAR3= 172366
352	172370	KDPAR4= 172370
353	172372	KDPAR5= 172372
354	172374	KDPAR6= 172374
355	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"

366		
367	170200	MAPL00 = 170200
368	170202	MAPH00 = 170202
369	170204	MAPL01 = 170204
370	170206	MAPH01 = 170206
371	170210	MAPL02 = 170210
372	170212	MAPH02 = 170212
373	170214	MAPL03 = 170214
374	170216	MAPH03 = 170216
375	170220	MAPL04 = 170220
376	170222	MAPH04 = 170222
377	170224	MAPL05 = 170224
378	170226	MAPH05 = 170226
379	170230	MAPL06 = 170230
380	170232	MAPH06 = 170232
381	170234	MAPL07 = 170234
382	170236	MAPH07 = 170236
383	170240	MAPL10 = 170240
384	170242	MAPH10 = 170242
385	170244	MAPL11 = 170244
386	170246	MAPH11 = 170246
387	170250	MAPL12 = 170250
388	170252	MAPH12 = 170252
389	170254	MAPL13 = 170254
390	170256	MAPH13 = 170256
391	170260	MAPL14 = 170260
392	170262	MAPH14 = 170262
393	170264	MAPL15 = 170264



394	170266	NAPH15 = 170266
395	170270	NAPL16 = 170270
396	170272	NAPH16 = 170272
397	170274	NAPL17 = 170274
398	170276	NAPH17 = 170276
399	170300	NAPL20 = 170300
400	170302	NAPH20 = 170302
401	170304	NAPL21 = 170304
402	170306	NAPH21 = 170306
403	170310	NAPL22 = 170310
404	170312	NAPH22 = 170312
405	170314	NAPL23 = 170314
406	170316	NAPH23 = 170316
407	170320	NAPL24 = 170320
408	170320	NAPH24 = 170320
409	170324	NAPL25 = 170324
410	170326	NAPH25 = 170326
411	170330	NAPL26 = 170330
412	170332	NAPH26 = 170332
413	170334	NAPL27 = 170334
414	170336	NAPH27 = 170336
415	170340	NAPL30 = 170340
416	170342	NAPH30 = 170342
417	170344	NAPL31 = 170344
418	170346	NAPH31 = 170346
419	170350	NAPL32 = 170350
420	170352	NAPH32 = 170352
421	170354	NAPL33 = 170354
422	170356	NAPH33 = 170356
423	170360	NAPL34 = 170360
424	170362	NAPH34 = 170362
425	170364	NAPL35 = 170364
426	170366	NAPH35 = 170366
427	170370	NAPL36 = 170370
428	170372	NAPH36 = 170372
429	170374	NAPL37 = 170374
430	170376	NAPH37 = 170376
431		.EQUIV NAPL00,MAPL0
432		.EQUIV NAPH00,MAPH0
433		.EQUIV NAPL01,MAPL1
434		.EQUIV NAPH01,MAPH1
435		.EQUIV NAPL02,MAPL2
436		.EQUIV NAPH02,MAPH2
437		.EQUIV NAPL03,MAPL3
438		.EQUIV NAPH03,MAPH3
439		.EQUIV NAPL04,MAPL4
440		.EQUIV NAPH04,MAPH4
441		.EQUIV NAPL05,MAPL5
442		.EQUIV NAPH05,MAPH5
443		.EQUIV NAPL06,MAPL6
444		.EQUIV NAPH06,MAPH6
445		.EQUIV NAPL07,MAPL7
446		.EQUIV NAPH07,MAPH7
447		.SBTTL FLOATING POINT REGISTER DEFINITIONS
448	000000	ACO =80
449	000001	AC1 =81

```

450      000002      AC2      =82
451      000003      AC3      =83
452      000004      AC4      =84
453      000005      AC5      =85
454      000006      AC6      =86
455      000007      AC7      =87
456      170004      MSN      =170004
457      001160      COUNT    =$REG0
458      001162      STEP     =$REG1
459      001164      OFFSET   =$REG2
460
461      .SBTTL      FLOATING POINT STATUS BIT DEFINITIONS
462      .EQUIV      BIT0,FC
463      .EQUIV      BIT1,FV
464      .EQUIV      BIT2,FZ
465      .EQUIV      BIT3,FM
466      .EQUIV      BIT4,FNM
467      .EQUIV      BIT5,FT
468      .EQUIV      BIT6,FL
469      .EQUIV      BIT7,FD
470      .EQUIV      BIT8,FIC
471      .EQUIV      BIT9,FIV
472      .EQUIV      BIT10,FIU
473      .EQUIV      BIT11,FIUV
474      .EQUIV      BIT14,FTD
475      .EQUIV      BIT15,FER
476
477      .SBTTL      FLOATING POINT EXCEPTION CODE DEFINITIONS
478      .EQUIV      BIT1,FOCE
479      .EQUIV      BIT2,FDZ
480      000006      FICE=6
481      000010      FO=10
482      000012      FU=12
483      000014      FUV=14
484      000016      FT=16
485
486      .SBTTL      FLOATING POINT VECTOR DEFINITION
487      000244      FPPVEC=244
488      000100      LKVEC   =100
489      177546      LKSTAT  =177546
490
491      .SBTTL      TRAP CATCHER
492
493      000000      .=0
494      ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
495      ;*SEQUENCE TO CATCH ILLFCAL TRAPS AND INTERRUPTS
496      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
497      .=174
498      000174      000000      DISPREG: .WORD 0          ;;SOFTWARE DISPLAY REGISTER
499      000176      000000      SWREG:   .WORD 0          ;;SOFTWARE SWITCH REGISTER
500
501      .SBTTL      STARTING ADDRESS(ES)
502      000200      000137      004634      JMP      @BSTART ;;JUMP TO STARTING ADDRESS OF PROGRAM
503
504      .SBTTL      ACT11 HOOKS
505

```

```
506 ;*****  
507 ;HOOKS REQUIRED BY ACT11  
508 000204 $SVPC= ;SAVE PC  
509 000046 .=46  
510 000046 044012 SENDAD ;1)SET LOC.46 TO ADDRESS OF SENDAD IN .SEGP  
511 000052 .=52  
512 000052 000000 .WORD 0 ;2)SET LOC.52 TO ZERO  
513 000204 .=SSVPC ; RESTORE PC
```

H5

```

514
515      .SBTTL  COMMON TAGS
516
517      ;;*****
518      ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
519      ;*USED IN THE PROGRAM.
520
521      001100      .=1100
522 001100      $CNTAG:      ;;START OF COMMON TAGS
523 001100      000000      .WOPD      0      ;;CONTAINS THE TEST NUMBER
524 001102      000      $STNM: .BYTE      0      ;;CONTAINS ERROR FLAG
525 001103      000      $ERFLG: .BYTE      0      ;;CONTAINS SUBTEST ITERATION COUNT
526 001104      000000      $ICNT: .WORD      0      ;;CONTAINS SCOPE LOOP ADDRESS
527 001106      000000      $LPADR: .WORD      0      ;;CONTAINS SCOPE RETURN FOR ERRORS
528 001110      000000      $LPERP: .WOPD      0      ;;CONTAINS TOTAL ERRORS DETECTED
529 001117      000000      $ERTTL: .WORD      0      ;;CONTAINS ITEM CONTROL BYTE
530 001114      000      $ITEMB: .BYTE      0      ;;CONTAINS MAX. ERRORS PER TEST
531 001115      001      $ERRMAX: .BYTE      1      ;;CONTAINS PC OF LAST ERROR INSTRUCTION
532 001116      000000      $ERRPC: .WORD      0      ;;CONTAINS ADDRESS OF "GOOD" DATA
533 001120      000000      $GADR: .WORD      0      ;;CONTAINS ADDRESS OF "BAD" DATA
534 001122      000000      $BDADR: .WOPD      0      ;;CONTAINS "GOOD" DATA
535 001124      000000      $GDDAT: .WOPD      0      ;;CONTAINS "BAD" DATA
536 001126      000000      $RDDAT: .WORD      0      ;;RESERVED--NOT TO BE USED
537 001130      000000      .WORD      0
538 001132      000000      .WORD      0
539 001134      000000      .WORD      0
540 001136      177570      $SWR: .WORD      DSWR      ;;ADDRESS OF SWITCH REGISTER
541 001140      177570      $DISPLAY: .WORD      DDISP      ;;ADDRESS OF DISPLAY REGISTER
542 001142      177560      $TKS: 177560      ;;TTY KBD STATUS
543 001144      177562      $TKB: 177562      ;;TTY KBD BUFFER
544 001146      177564      $TPS: 177564      ;;TTY PRINTER STATUS REG. ADDRESS
545 001150      177566      $TPB: 177566      ;;TTY PRINTER BUFFER REG. ADDRESS
546 001152      000      $NULL: .BYTE      0      ;;CONTAINS NULL CHARACTER FOR FILLS
547 001153      002      $FILLS: .BYTE      2      ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
548 001154      012      $PILLC: .BYTE      12      ;;INSERT FILL CHARS. AFTER A "LINE FEED"
549 001155      000      $TPFLG: .BYTE      0      ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
550 001156      000000      $REGAD: .WORD      0      ;;CONTAINS THE ADDRESS FROM
551      ;;WHICH ($REG0) WAS OBTAINED
552 001160      000000      $REG0: .WORD      0      ;;CONTAINS ((SREGAD)+0)
553 001162      000000      $REG1: .WORD      0      ;;CONTAINS ((SREGAD)+2)
554 001164      000000      $REG2: .WORD      0      ;;CONTAINS ((SREGAD)+4)
555 001166      000000      $REG3: .WORD      0      ;;CONTAINS ((SREGAD)+6)
556 001170      000000      $REG4: .WORD      0      ;;CONTAINS ((SREGAD)+10)
557 001172      000000      $REG5: .WORD      0      ;;CONTAINS ((SREGAD)+12)
558 001174      000000      $REG6: .WORD      0      ;;CONTAINS ((SREGAD)+14)
559 001176      000000      $REG7: .WOPD      0      ;;CONTAINS ((SREGAD)+16)
560 001200      000000      $TMP0: .WOPD      0      ;;USER DEFINED
561 001202      000000      $TMP1: .WORD      0      ;;USER DEFINED
562 001204      000000      $TMP2: .WORD      0      ;;USER DEFINED
563 001206      000000      $TMP3: .WORD      0      ;;USER DEFINED
564 001210      000000      $TMP4: .WORD      0      ;;USER DEFINED
565 001212      000000      $TMP5: .WORD      0      ;;USER DEFINED
566 001214      000000      $TMP6: .WOPD      0      ;;USER DEFINED
567 001216      000000      $TMP7: .WORD      0      ;;USER DEFINED
568 001220      000000      $TIMES: 0      ;;MAX. NUMBER OF ITERATIONS
569 001222      000000      $ESCAPE: 0      ;;ESCAPE ON ERROR ADDRESS
    
```

IS

570	001224	177607	000377	\$BELL:	.ASCII	<207><377><377>	;;CODE FOR BELL
571	001230	077		\$QUES:	.ASCII	/?	;;QUESTION MARK
572	001231	015		\$CRLF:	.ASCII	<15>	;;CARRIAGE RETURN
573	001232	000012		\$LF:	.ASCII	<12>	;;LINE FEED


```

574      ;*****
575
576      .SBTTL  APT MAILBOX-ETABLE
577
578      ;*****
579      .EVEN
580 001234  SMAIL:                ;;APT MAILBOX
581 001234 000000  SMSCTY: .WORD    AMSCTY  ;;MESSAGE TYPE CODE
582 001236 000000  SFATAL: .WORD    AFATAL  ;;FATAL ERROR NUMBER
583 001240 000000  STESTN: .WORD    ATESTN  ;;TEST NUMBER
584 001242 000000  SPASS:  .WORD    APASS   ;;PASS COUNT
585 001244 000000  SDEVCT: .WORD    ADEVCT  ;;DEVICE COUNT
586 001246 000000  SUNIT:  .WORD    AUNIT   ;;I/O UNIT NUMBER
587 001250 000000  SMSGAD: .WORD    AMSGAD  ;;MESSAGE ADDRESS
588 001252 000000  SMSGLG: .WORD    AMSGLG  ;;MESSAGE LENGTH
589 001254  SETABLE:                ;;APT ENVIRONMENT TABLE
590 001254      000  SFNV:  .BYTE    AENV    ;;ENVIRONMENT BYTE
591 001255      000  SENVM: .BYTE    AENVN   ;;ENVIRONMENT MODE BITS
592 001256 000000  SSWREG: .WORD    ASWRG   ;;APT SWITCH REGISTER
593 001260 000000  SUSWR:  .WORD    AUSWR   ;;USER SWITCHES
594 001262 000000  SPCUOP: .WORD    ACPUOP  ;;CPU TYPE,OPTIONS
595      ;*
596      ;*
597      ;*
598      ;*
599      ;*
600      ;*
601 001264  SETEND:
602      .MEXIT
603 001264 000000  SERPSW: .WORD                ;ERROR PSW
604 001266 000000  S$TSTNM: .WORD                ;TEST NUMBER FOR TYPEOUT
605 001270 000000  SFEC:  .WORD                ;FLOATING EXCEPTION CODE
606 001272 000000  SFEA:  .WORD                ;FLOATING EXCEPTICH ADDRESS
607 001274      000  SFT:  .BYTE                ;TRUNCATE/ROUND MODE
608 001275      000  SFL:  .BYTE                ;INTEGER/INTEGER LONG MODE
609 001776 000000  SFD:  .WORD                ;FLOATING/FLOATING DOUBLE MODE
610 001300 000004  SFLT:  .PLW      4          ;BUFFER FOR FLOATING CONVERSIONS
611 001310 000044  SFLBUF: .BLKB    44        ;BUFFER FOR ASCII FLOATING NUMBERS
612 001354 000000  S$BUFF: .WORD
613 001356 025325  $AC5:  .WORD    25325      ;ACCUMULATOR DATA
614 001360 052525  .WORD    52525
615 001362 121104  $AC4:  .WORD    121104
616 001364 042104  .WORD    42104
617 001366 014663  $AC3:  .WORD    14663
618 001370 031463  .WORD    31463
619 001372 110442  $AC2:  .WORD    110442
620 001374 021042  .WORD    21042
621 001376 004221  $AC1:  .WORD    4221
622 001400 010421  .WORD    10421
623 001402 000000  $AC0:  .WORD    0
624 001404 000000  .WORD    0
625 001406 000000  SPCUERR: .WORD                ;ADDRESS OF ERROR REG IF 11/70 OTHERWISE 0
626 001410 052525  S$AC5: .WORD    52525      ;ACCUMULATOR DATA
627 001412 052525  .WORD    52525
628 001414 042104  S$AC4: .WORD    42104      ;FOR DUAL ADDRESSING
629 001416 042104  .WORD    42104            ;TEST (LOW ORDER)

```

K5

630	001420	031463	SSAC3:	.WORD	31463
631	001422	031463		.WORD	31463
632	001424	021042	SSAC2:	.WORD	21042
633	001426	021042		.WORD	21042
634	001430	010421	SSAC1:	.WORD	10421
635	001432	010421		.WORD	10421
636	001434	000000	SSAC0:	.WORD	0
637	001436	000000		.WORD	0
638	001440	077777	RSH0:	.WORD	77777
639	001442	000377		.WORD	377
640	001444	000377		.WORD	377
641	001446	000377		.WORD	377
642	001450		LSH7:		
643	001450	077776	RSH7:	.WORD	77776
644	001452	001770		.WORD	1770
645	001454	007740		.WORD	7740
646	001456	037600		.WORD	37600
647	001460	077774	RSH6:	.WORD	77774
648	001462	007700		.WORD	7700
649	001464	176017		.WORD	176017
650	001466	140374		.WORD	140374
651	001470	077770	RSH5:	.WORD	77770
652	001472	037017		.WORD	37017
653	001474	101740		.WORD	101740
654	001476	174076		.WORD	174076
655	001500	077760	RSH4:	.WORD	77760
656	001502	170360		.WORD	170360
657	001504	170360		.WORD	170360
658	001506	170360		.WORD	170360
659	001510	077743	RSH3:	.WORD	77743
660	001512	107070		.WORD	107070
661	001514	161616		.WORD	161616
662	001516	070707		.WORD	70707
663	001520	077714	RSH2:	.WORD	77714
664	001522	146314		.WORD	146314
665	001524	146314		.WORD	146314
666	001526	146314		.WORD	146314
667	001530	077625	RSH1:	.WORD	77625
668	001532	052525		.WORD	52525
669	001534	052525		.WORD	52525
670	001536	052525		.WORD	52525
671	001540	000001	LSH7CK:	.WORD	00001
672	001542	176007		.WORD	176007
673	001544	170037		.WORD	170037
674	001546	140000		.WORD	140000
675	001550	000000	LPSHCK:	.WORD	00000
676	001552	077401		.WORD	77401
677	001554	176007		.WORD	176007
678	001556	170000		.WORD	170000
679	001560	000100	SLRSHCK:	.WORD	00100
680	001562	077401		.WORD	77401
681	001564	176007		.WORD	176007
682	001566	170000		.WORD	170000

;DATA FOR SHIFTER TEST

683
684
685

.SBTTL ERROR POINTER TABLE

LS

686 ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
687 ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
688 ;*LOCATION SITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
689 ;*NOTE1: IF SITEMB IS 0 THE ONLY PERTINENT DATA IS (\$ERRPC).
690 ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
691

692 ;* EM ;)POINTS TO THE ERROR MESSAGE
693 ;* DH ;)POINTS TO THE DATA HEADER
694 ;* DT ;)POINTS TO THE DATA
695 ;* DF ;)POINTS TO THE DATA FORMAT
696

697
698 001570 \$ERRTB:
699

700
701 ;ITEM 1
702 001570 050161 EM1 ;)TRCC F/CLASS DOES NOT GET TO E40(A0) OR E40 (A0) BAD
703 001572 050244 DH1 ;)ERRORPC TSTNO
704 001574 050264 DT1 ;)SERRPC,\$\$STSNM
705 001576 000000 0

706
707 ;ITEM 2
708 001600 050772 EM2 ;)TRCC F/CLASS DOES NOT GET TO 21(?) OR E21 IS BAD.
709 001602 050244 DH1
710 001604 050264 DT1
711 001606 000000 0

712
713 ;ITEM 3
714 001610 050345 EM3 ;)RACK E49 DOES NOT GET PP REG WT
715 001612 050244 DH1
716 001614 050264 DT1
717 001616 000000 0

718
719 ;ITEM 4
720 001620 050421 EM4 ;)RACK A2 RAB00 DID NOT GO LOW
721 001622 050244 DH1
722 001624 050264 DT1
723 001626 000000 0

724
725 ;ITEM 5
726 001630 050510 EM5 ;)RACK E50(A0) DID NOT GO LOW
727 001632 050244 DH1
728 001634 050264 DT1
729 001636 000000 0

730
731 ;ITEM 6
732 001640 050545 EM6 ;)TRCC F/CLASS DID NOT GO LOW
733 001642 050244 DH1
734 001644 050264 DT1
735 001646 000000 0

736
737 ;ITEM 7
738 001650 050600 EM7 ;)RACK E64(A0) DOES NOT GET (?)
739 001652 050244 DH1
740 001654 050264 DT1
741 001656 000000 0

M5

742				
743			;ITEM 10	
744	001660	050714	EM10	
745	001662	050244	DH1	
746	001664	050264	DT1	
747	001666	000000	0	
748				
749			;ITEM 11	
750	001670	051014	EM11	;BIT STUCK IN FPS DATA PATH
751	001672	051047	DH11	;ERRPC DATA TEST NO
752				; EXPECT ACTUAL
753	001674	051130	DT11	;PERRPC, SREG2, SPEG1, S\$TSTNM
754	001676	000000	0	
755				
756			;ITEM 12	
757	001700	051147	EM12	;MICRO-BREAK TRAP DID NOT TRAP
758	001702	050244	DH1	
759	001704	050264	DT1	
760	001706	000000	0	
761				
762			;ITEM 13	
763	001710	051200	EM13	;NO MEM BRANCH FAILED TO
764	001712	050244	DH1	;SET MODES POW STATE
765	001714	050264	DT1	
766	001716	000000	0	
767				
768			;ITEM 14	
769	001720	051320	EM14	;MICRO-PRAK TRAP TRAPPED TO WRONG
770	001722	050244	DH1	;VECTOR
771	001724	050264	DT1	
772	001726	000000	0	
773				
774			;ITEM 15	
775	001730	051450	EM15	;SETD FAILED
776	001732	051576	DH15	;FRPPC FPS TEST NO
777				; EXPECT ACTUAL
778	001734	051656	DT15	;SERKPC, STMP1, STMP0, S\$TSTNM
779	001736	000000	0	
780				
781			;ITEM 16	
782	001740	051670	EM16	;SET L FAILED
783	001742	051576	DH15	
784	001744	051656	DT15	
785	001746	000000	0	
786				
787			;ITEM 17	
788	001750	051730	EM17	;SETF FAILED
789	001752	051576	DH15	
790	001754	051656	DT15	
791	001756	000000	0	
792				
793			;ITEM 20	
794	001760	051764	EM20	;SETI FAILED
795	001762	051576	DH15	
796	001764	051656	DT15	
797	001766	000000	0	

798					
799				;ITEM 21	
800	001770	052020	EM21		;CFCC FAILED
801	001772	052150	DH21		;ERRPC PSM TEST NO
802					; EXPECT ACTUAL
803	001774	051656	DT15		
804	001776	000000	0		
805					
806				;ITEM 22	
807	002000	052730	EM27		;FRMF FPREQ DOES NOT GET TO RACK
808	002002	050244	DH1		;AS A LOW
809	002004	050264	DT1		
810	002006	000000	0		
811					
812				;ITEM 23	
813	002010	052301	EM23		;PACK BRCAB05 DOES NOT GO LOW ON BRQ*(T+CONF)
814	002012	050244	DH1		
815	002014	050264	DT1		
816	002016	000000	0		
817					
818				;ITEM 24	
819	002020	052354	EM24		;TRAPPED TO 4 BUT DON'T KNOW WHY
820	002022	052404	DH24		;ERRPC ERRREG TEST NO
821	002024	052434	DT24		;SERRPC,\$TMP0
822	002026	000000	0		
823					
824				;ITEM 25	
825	002030	052444	EM25		;STACKED PC IS WRONG
826	002032	052470	DH25		;ERRPC PC TEST NO
827					; EXPECT ACTUAL
828	002034	051656	DT15		
829	002036	000000	0		
830					
831				;ITEM 26	
832	002040	052546	EM26		;TRCD DSTCON<2 NOT GOING HIGH
833	002042	050244	DH1		
834	002044	050264	DT1		
835	002046	000000	0		
836					
837				;ITEM 27	
838	002050	052603	EM27		;A BRANCH FAILED TO STX*-M0
839	002052	050744	DH1		
840	002054	050264	DT1		
841	002056	000000	0		
842					
843				;ITEM 30	
844	002060	052652	EM30		;A BRANCH FAILED TO CLX*-M0
845	002062	050244	DH1		
846	002064	050264	DT1		
847	002066	000000	0		
848					
849				;ITEM 31	
850	002070	052712	EM31		;FXPB IMMEDIATE L DOES NOT GET
851	002072	050244	DH1		;TO FRMF AS A LOW
852	002074	050264	DT1		
853	002076	000000	0		

854				
855			;ITEM 32	
856	002100	052761	EM32	;FXPA AD2 DID NOT GO HIGH AT
857	002107	050244	DH1	;PUM ADDRESS 710
858	002104	050264	DT1	
859	002106	000000	0	
860				
861			;ITEM 33	
862	002110	053031	EM33	;FXPB FIRA10(0) H STUCK LOW
863	002112	050244	DH1	
864	002114	050264	DT1	
865	002116	000000	0	
866				
867			;ITEM 34	
868	002120	053071	EM34	;CONTROL STORE BRANCH FAILED
869	002122	053125	DH34	;ERRPC PUMADR TEST NO
870	002124	052434	DT24	;SERRPC,\$TMP0,\$STSTNM
871	002126	000000	0	
872				
873			;ITEM 35	
874	002130	053155	EM35	;DATA IS PAD
875	002132	053711	DH35	;ERRPC DATA TEST NO
876				; EXPECT ACTUAL
877	002134	053276	DT35	;SERRPC,\$TMP0,\$TMP2,\$STSTNM
878	002136	053310	DF35	;0,1,1,0
879				
880			;ITEM 36	
881	002140	053314	EM36	;IRCD DSTCON<4 DID NOT GO HIGH
882	002142	050244	DH1	
883	002144	050264	DT1	
884	002146	000000	0	
885				
886			;ITEM 37	
887	002150	053352	EM37	;IRCD DSTCON<10 DID NOT GO LOW
888	002152	050244	DH1	
889	002154	050264	DT1	
890	002156	000000	0	
891				
892			;ITEM 40	
893	002160	053410	EM40	;DEST REG DID NOT INCREMENT CORRECTLY
894	002162	051047	DH11	
895	002164	051656	DT15	
896	002166	000000	0	
897				
898			;ITEM 41	
899	002170	053454	EM41	;FRMJ PD(1)H NOT GETTING 10
900	002172	050244	DH1	;FXP AS A LOW
901	002174	050264	DT1	
902	002176	000000	0	
903				
904			;ITEM 42	
905	002200	053524	EM42	;LDF WORKED BUT STF FAILED AUTO-INCREMENT
906	002202	050244	DH1	
907	002204	050264	DT1	
908	002206	000000	0	
909				

910			;ITEM 43	
911	002210	053571	EM43	;FIFTH FIRST OR SECOND WORD DID
912	002212	053211	DH35	;NOT LOAD OR STORE
913	002214	053776	DT35	
914	002216	053310	DF35	
915				
916			;ITEM 44	
917	002220	053655	EM44	;BIT STUCK IN DATA PATH IN PPP
918	002222	053211	DH35	
919	002224	053276	DT35	
920	002226	053310	DF35	
921				
922			;ITEM 45	
923	002230	053713	EM45	;DUAL ADDRESSING IN QUAD C3, 23 S.P.'S
924	002232	053211	DH35	
925	002234	053276	DT35	
926	002236	053310	DF35	
927				
928			;ITEM 46	
929	002240	053756	EM46	;LDFPS*-NO FAILED
930	002242	051576	DH15	
931	002244	051656	DT15	
932	002246	000000	0	
933				
934			;ITEM 47	
935	002250	054014	EM47	;STFPS*-NO FAILED
936	002252	051576	DH15	
937	002254	051656	DT15	
938	002256	000000	0	
939				
940			;ITEM 50	
941	002260	054052	EM50	;CC'S BAD ON LOAD FLOAT
942	002262	051576	DH15	
943	002264	054102	DT50	;SERRPC,STMP4,STMP5,\$\$STSTNM
944	002266	000000	0	
945				
946			;ITEM 51	
947	002270	054114	EM51	;EXP CLEARED, FRACTION DIDN'T
948	002272	053211	DH35	
949	002274	053276	DT35	
950	002276	053310	DF35	
951				
952			;ITEM 52	
953	002300	054145	EM52	;FRACTION CLEARED, EXP DIDN'T
954	002302	053211	DH35	
955	002304	053276	DT35	
956	002306	053310	DF35	
957				
958			;ITEM 53	
959	002310	054176	EM53	;BAD CC'S ON CLEAR FLOAT
960	002312	051576	DH15	
961	002314	054102	DT50	
962	002316	000000	0	
963				
964			;ITEM 54	
965	002320	054243	EM54	;BAD CC'S ON STORE FLOAT

966	002322	051576	DH15	
967	002324	054107	DT50	
968	002326	000000	0	
969				
970			;ITEM 55	
971	002330	054310	EM55	;POM FLOW OK, BUT DATA BAD
972	002332	053211	DH35	
973	002334	053276	DT35	
974	002336	053310	DF35	
975				
976			;ITEM 56	
977	002340	054342	EM56	;CC'S BAD ON TSTP
978	002342	051576	DH15	
979	002344	051656	DT15	
980	002346	000000	0	
981				
982			;ITEM 57	
983	002350	054441	EM57	;BIT DID NOT SET ON TSTP
984	002352	051576	DH15	
985	002354	051656	DT15	
986	002356	000000	0	
987				
988			;ITEM 60	
989	002360	054525	EM60	;TRCB B0 PAR03 DID NOT GO LOW
990	002362	050244	DH1	
991	002364	050264	DT1	
992	002366	000000	0	
993				
994			;ITEM 61	
995	002370	054562	EM61	;TRCB B0 PAR01 DID NOT GO LOW
996	002372	050244	DH1	
997	002374	050264	DT1	
998	002376	000000	0	
999				
1000			;ITEM 62	
1001	002400	054617	EM62	;BIT STUCK IN CPU/FPU INTERFACE OR PDR
1002	002402	051047	DH11	
1003	002404	051130	DT11	
1004	002406	000000	0	
1005				
1006			;ITEM 63	
1007	002410	054665	EM63	;SIGN BIT DID NOT CLEAR
1008	002412	051047	DH11	
1009	002414	051656	DT15	
1010	002416	000000	0	
1011				
1012			;ITEM 64	
1013	002420	054714	EM64	;FXPONENT DID NOT CLEAR
1014	002422	051047	DH11	
1015	002424	051656	DT15	
1016	002426	000000	0	
1017				
1018			;ITEM 65	
1019	002430	054743	EM65	;FRACTION DID NOT CLEAR
1020	002432	051047	DH11	
1021	002434	051656	DT15	

1022	002436	000000	0	
1023				
1024				
1025	002440	054772	;ITEM 66	
1026	002442	050244	EM66	;THE ADX POW FAILED
1027	002444	050264	DH1	
1028	002446	000000	DT1	
1029			0	
1030				
1031	002450	055015	;ITEM 67	
1032	002452	050244	EM67	;ABSX BRANCH FAILED
1033	002454	050264	DH1	
1034	002456	000000	DT1	
1035			0	
1036				
1037	002460	055113	;ITEM 70	
1038	002462	050244	EM70	;PZ BRANCH FAILED
1039	002464	050264	DH1	
1040	002466	000000	DT1	
1041			0	
1042				
1043	002470	055213	;ITEM 71	
1044	002472	051576	EM71	;CC'S BAD
1045	002474	051656	DH15	
1046	002476	000000	DT15	
1047			0	
1048				
1049	002500	055237	;ITEM 72	
1050	002502	050244	EM72	;ABSX BRANCH FAILED
1051	002504	050264	DH1	
1052	002506	000000	DT1	
1053			0	
1054				
1055	002510	055336	;ITEM 73	
1056	002512	051047	EM73	;SIGN ON, BUT DATA CHANGED
1057	002514	051130	DH11	
1058	002516	000000	DT11	
1059			0	
1060				
1061	002520	055353	;ITEM 74	
1062	002522	050244	EM74	;6F1 BRANCH FAILED TO STATE 254
1063	002524	050264	DH1	
1064	002526	000000	DT1	
1065			0	
1066				
1067	002530	055454	;ITEM 75	
1068	002532	050244	EM75	;6F1 BRANCH FAILED TO STATE 240
1069	002534	050264	DH1	
1070	002536	000000	DT1	
1071			0	
1072				
1073	002540	055556	;ITEM 76	
1074	002542	050244	EM76	;6F1 BRANCH FAILED TO STATE 244.
1075	002544	050264	DH1	
1076	002546	000000	DT1	
1077			0	

1078			;ITEM 77	
1079	002550	054310	EM55	
1080	002552	051047	DH11	
1081	002554	051130	DT11	
1082	002556	000000	0	
1083				
1084			;ITEM 100	
1085	002560	055660	EM100	;RPM FLOW OK, BUT CC'S BAD
1086	002562	051576	DH15	
1087	002564	051656	DT15	
1088	002566	000000	0	
1089				
1090			;ITEM 101	
1091	002570	055712	EM101	;RPM STATE 334 DID NOT CLR DST.
1092	002572	051047	DH11	
1093	002574	051130	DT11	
1094	002576	000000	0	
1095				
1096			;ITEM 102	
1097	002600	055755	EM102	;BZ BRANCH FAILED
1098	002602	050244	DH1	
1099	002604	050764	DT1	
1100	002606	000000	0	
1101				
1102			;ITEM 103	
1103	002610	054743	EM65	
1104	002612	051047	DH11	
1105	002614	051130	DT11	
1106	002616	000000	0	
1107				
1108			;ITEM 104	
1109	002620	056062	EM104	;EITHER STATE 105 OR 362 DID NOT
1110	002622	051047	DH11	;LOAD THE FRACTION PROPERLY
1111	002624	051130	DT11	
1112	002626	000000	0	
1113				
1114			;ITEM 105	
1115	002630	056151	EM105	;FXPONENT DID NOT LOAD PROPERLY
1116	002632	051047	DH11	
1117	002634	051130	DT11	
1118	002636	000000	0	
1119				
1120			;ITEM 106	
1121	002640	056710	EM106	;RZ-BW BRANCH FAILED
1122	002642	050244	DH1	
1123	002644	050264	DT1	
1124	002646	000000	0	
1125				
1126			;ITEM 107	
1127	002650	056320	EM107	;RIT STUCK IN STEP COUNTER.
1128	002652	051047	DH11	
1129	002654	051130	DT11	
1130	002656	000000	0	
1131				
1132			;ITEM 110	
1133	002660	056353	EM110	;EITHER STATE 250 OR 377 FAILED TO LOAD

1134	002662	051047	DH11	;THE EXP OR FRACTION
1135	002664	051130	DT11	
1136	002666	000000	0	
1137				
1138			;ITEM 111	
1139	002670	056427	EM111	;FRMF SS(0)H XOR SUB DID NOT GO LOW
1140	002672	051047	DH11	;OR DID NOT GET TO SD FLIP-FLOP
1141	002674	051130	DT11	
1142	002676	000000	0	
1143				
1144			;ITEM 112	
1145	002700	056523	EM112	;FRMF SS(0)H XOR SUB DID NOT GO HIGH
1146	002702	051047	DH11	;OR DID NOT GET TO SD FLIP-FLOP
1147	002704	051130	DT11	
1148	002706	000000	0	
1149				
1150			;ITEM 113	
1151	002710	056620	EM113	;FXPL BCN NOT GOING HIGH WITH BAL009 LOW
1152	002712	050244	DH1	
1153	002714	050264	DT1	
1154	002716	000000	0	
1155				
1156			;ITEM 114	
1157	002720	056670	EM114	;FRMF SS XOR SD NOT GOING LOW
1158	002722	050244	DH1	
1159	002724	050264	DT1	
1160	002726	000000	0	
1161				
1162			;ITEM 115	
1163	002730	056757	EM115	;FRMF SS XOR SD NOT DOING WRIGHT THING
1164	002732	050244	DH1	;WITH EITHER WH OR HL INPUT.
1165	002734	050264	DT1	
1166	002736	000000	0	
1167				
1168			;ITEM 116	
1169	002740	057051	EM116	;BZ-BN BRANCH FAILED OR FRMF
1170	002742	050244	DH1	;SS(0)H FAILED
1171	002744	050264	DT1	
1172	002746	000000	0	
1173				
1174			;ITEM 117	
1175	002750	057206	EM117	;FRMF SS(0)H NOT GOING LOW
1176	002752	050244	DH1	
1177	002754	050264	DT1	
1178	002756	000000	0	
1179				
1180			;ITEM 120	
1181	002760	057263	EM120	;TRCD DSTCON<10 NOT GOING HIGH
1182	002762	051047	DH11	
1183	002764	051130	DT11	
1184	002766	000000	0	
1185				
1186			;ITEM 121	
1187	002770	057321	EM121	;ROM FLOW OK BUT DATA BAD
1188	002772	057347	DH121	;ERPPC DATA TEST NO
1189	002774	057414	DT121	;SERRPC,STMP0,STMP4,\$STSTNM

1190	002776	057422	DF121	30,2,2,0
1191				
1192			;ITEM 122	
1193	003000	053655	EM44	
1194	003002	057347	DH121	
1195	003004	057414	DT121	
1196	003006	057422	DF121	
1197				
1198			;ITEM 173	
1199	003010	057424	EM123	;DUAL ADDRESSING IN QUAD E1,03
1200	003012	057347	DH121	
1201	003014	057414	DT121	
1202	003016	057422	DF121	
1203				
1204			;ITEM 124	
1205	003020	057467	EM124	;FROM STATE 143 FAILED
1206	003022	057347	DH121	
1207	003024	057414	DT121	
1208	003026	057422	DF121	
1209				
1210			;ITEM 125	
1211	003030	057514	EM125	;A QUADRANT IS BAD
1212	003032	057347	DH121	
1213	003034	057414	DT121	
1214	003036	057422	DF121	
1215				
1216			;ITEM 176	
1217	003040	057525	EM126	;PRHC PMX 35 NOT GOING LOW
1218	003042	050244	DH1	
1219	003044	050264	DT1	
1220	003046	000000	0	
1221				
1222			;ITEM 127	
1223	003050	057557	EM127	;PRHE AR59(1)L NOT GOING HIGH
1224	003052	050244	DH1	
1225	003054	050264	DT1	
1226	003056	000000	0	
1227				
1228			;ITEM 130	
1229	003060	057660	EM130	;PRHC PMX 34 NOT GOING HIGH OR PALU IS BAD
1230	003062	057347	DH121	
1231	003064	057414	DT121	
1232	003066	057422	DF121	
1233				
1234			;ITEM 131	
1235	003070	057727	EM131	;STEPX FAILED
1236	003072	051047	DH11	
1237	003074	051130	DT11	
1238	003076	000000	0	
1239				
1240			;ITEM 132	
1241	003100	057744	EM132	;PRME EALU CIN DID NOT GO LOW
1242	003102	050244	DH1	
1243	003104	050264	DT1	
1244	003106	000000	0	
1245				

1246			;ITEM 133	
1247	003110	060001	EM133	;FXPC FCLD EN DID NOT GO LOW
1248	003112	050244	DH1	
1249	003114	050264	DT1	
1250	003116	000000	0	
1251				
1252			;ITEM 134	
1253	003120	060035	EM134	;FXPF ESXT DID NOT GO HIGH
1254	003122	050244	DH1	
1255	003124	050264	DT1	
1256	003126	000000	0	
1257				
1258			;ITEM 135	
1259	003130	060067	EM135	;CLRD FAILED
1260	003132	053211	DH35	
1261	003134	053276	DT35	
1262	003136	053310	DF35	
1263				
1264			;ITEM 136	
1265	003140	060103	EM136	;ABSF FAILED TO STORE CORRECT DATA
1266	003142	051047	DH11	
1267	003144	051130	DT11	
1268	003146	000000	0	
1269				
1270			;ITEM 137	
1271	003150	060145	EM137	;NEGF FAILED TO STORE CORRECT DATA
1272	003152	051047	DH11	
1273	003154	051130	DT11	
1274	003156	000000	0	
1275				
1276			;ITEM 140	
1277	003160	060707	EM140	;DATA BAD ON ADDF *-M0
1278	003162	057347	DH121	
1279	003164	057414	DT121	
1280	003166	057427	DF121	
1281				
1282			;ITEM 141	
1283	003170	060234	EM141	;DATA BAD
1284	003172	051047	DH11	
1285	003174	051130	DT11	
1286	003176	000000	0	
1287				
1288			;ITEM 142	
1289	003200	060245	EM142	;STATE 51 DID NOT SET BN
1290	003202	050244	DH1	
1291	003204	050264	DT1	
1292	003206	000000	0	
1293				
1294			;ITEM 143	
1295	003210	060275	EM143	;ROM STATE 361 OR 237 FAILED
1296	003212	051047	DH11	
1297	003214	051130	DT11	
1298	003216	000000	0	
1299				
1300			;ITEM 144	
1301	003220	060331	EM144	;LDFXP DID NOT CLEAR DST ON UNDERFLOW

1302	003222	053211	DH35	
1303	003224	053276	DT35	
1304	003226	053310	DF35	
1305				
1306			;ITEM 145	
1307	003230	060374	EM145	;STATE 363 FAILED TO LOAD DST
1308	003232	053211	DH35	
1309	003234	053276	DT35	
1310	003236	053310	DF35	
1311				
1312			;ITEM 146	
1313	003240	060431	EM146	;FXPJ QVF NOT GOING HIGH OR NOT
1314	003242	050244	DH1	;GETTING TO PRLN
1315	003244	050264	DT1	
1316	003246	000000	0	
1317				
1318			;ITEM 147	
1319	003250	060510	EM147	;QUAD 3 DID NOT CLEAR
1320	003252	051047	DH11	
1321	003254	051130	DT11	
1322	003256	000000	0	
1323				
1324			;ITEM 150	
1325	003260	060535	EM150	;FRWF ADD*SC<R DID NOT GO HIGH
1326	003262	050244	DH1	
1327	003264	050264	DT1	
1328	003266	000000	0	
1329				
1330			;ITEM 151	
1331	003270	060573	EM151	;FRWF SUB*SC<R DID NOT GO HIGH
1332	003272	050244	DH1	
1333	003274	050264	DT1	
1334	003276	000000	0	
1335				
1336			;ITEM 152	
1337	003300	060631	EM152	;FXPP OUT OF RANGE NOT GETTING TO PRNA
1338	003302	050244	DH1	;AS A H
1339	003304	050264	DT1	
1340	003306	000000	0	
1341				
1342			;ITEM 153	
1343	003310	060711	EM153	;FXPP SC09 XOR SC7 DID NOT GO LOW
1344	003312	050744	DH1	
1345	003314	050264	DT1	
1346	003316	000000	0	
1347				
1348			;ITEM 154	
1349	003320	060753	EM154	;FXPP SC09 XKR SC06 DID NOT GO LOW
1350	003322	050244	DH1	
1351	003324	050264	DT1	
1352	003326	000000	0	
1353				
1354			;ITEM 155	
1355	003330	061015	EM155	;FXPP SC09 DID NOT GO LOW
1356	003332	050244	DH1	
1357	003334	050264	DT1	

1358	003336	000000	0	
1359				
1360);ITEM 156	
1361	003340	061046	EM156);FXPP SC09 NOT GETTING TO PRMA
1362	003342	050244	DH1);AS A HIGH
1363	003344	050264	DT1	
1364	003346	000000	0	
1365				
1366);ITEM 157	
1367	003350	061116	EM157);STATE 274 FAILED TO LOAD DST
1368	003352	053211	DH35	
1369	003354	053276	DT35	
1370	003356	053310	DF35	
1371				
1372);ITEM 160	
1373	003360	061153	EM160);FXMF SD(1) NOT GETTING TO PRMA
1374	003362	050244	DH1	
1375	003364	050264	DT1	
1376	003366	000000	0	
1377				
1378);ITEM 161	
1379	003370	061232	EM161);4F2 BRANCH FAILED
1380	003372	050244	DH1	
1381	003374	050264	DT1	
1382	003376	000000	0	
1383				
1384);ITEM 162	
1385	003400	061310	EM162);5F2 BRANCH FAILED
1386	003402	050244	DH1	
1387	003404	050264	DT1	
1388	003406	000000	0	
1389				
1390);ITEM 163	
1391	003410	061457	EM163);PRMA FIU(0) NOT GOING LOW
1392	003412	050244	DH1	
1393	003414	050264	DT1	
1394	003416	000000	0	
1395				
1396);ITEM 164	
1397	003420	061511	EM164);SC09 NOT GETTING THRU SD MUX AS A LOW
1398	003422	050244	DH1	
1399	003424	050264	DT1	
1400	003426	000000	0	
1401				
1402);ITEM 165	
1403	003430	061564	EM165);SC09 NOT GETTING THRU SD MUX AS A HIGH
1404	003432	050244	DH1	
1405	003434	050264	DT1	
1406	003436	000000	0	
1407				
1408);ITEM 166	
1409	003440	061640	EM166);FRHE AR59(1)L DID NOT GO HIGH
1410	003442	050244	DH1	
1411	003444	050264	DT1	
1412	003446	000000	0	
1413				

1414			;ITEM 167	
1415	003450	061745	EM167	;DATA BAD ON STCFI*-MO
1416	003452	051047	DH11	
1417	003454	051130	DT11	
1418	003456	000000	0	
1419				
1420			;ITEM 170	
1421	003460	062014	EM170	;SUB IN STATE 374 DID NOT CLR BN 6 BZ
1422	003462	050244	DH1	
1423	003464	050264	DT1	
1424	003466	000000	0	
1425				
1426			;ITEM 171	
1427	003470	062065	EM171	;STATE 373 DID NOT CLEAR AC6
1428	003472	050244	DH1	
1429	003474	050264	DT1	
1430	003476	000000	0	
1431				
1432			;ITEM 172	
1433	003500	062121	EM172	;FRME FCC1(1) NOT GETTING TO
1434	003502	050244	DH1	;FRLN C(1) AS A HIGH
1435	003504	050264	DT1	
1436	003506	000000	0	
1437				
1438			;ITEM 173	
1439	003510	062176	EM173	;STATE 167 DID NOT CLEAR AR LOW
1440	003512	057347	DH121	
1441	003514	057414	DT121	
1442	003516	057422	DF121	
1443				
1444			;ITEM 174	
1445	003520	062235	EM174	;STATE 122 DID NOT CLEAR QUAD [3:2]
1446	003522	057347	DF121	
1447	003524	057414	DT121	
1448	003526	057422	DF121	
1449				
1450			;ITEM 175	
1451	003530	062300	EM175	;STATE 123 FAILED
1452	003532	051047	DH11	
1453	003534	051130	DT11	
1454	003536	000000	0	
1455				
1456			;ITEM 176	
1457	003540	062321	EM176	;FRLP FVINT NOT GOING HIGH
1458	003542	050244	DH1	
1459	003544	050264	DT1	
1460	003546	000000	0	
1461				
1462			;ITEM 177	
1463	003550	062415	EM177	;STAO DID NOT STORE ALL QUADS PROPERLY
1464	003552	057347	DH121	
1465	003554	062464	DT177	;SEPRPC,\$TMP0,\$REG0,\$STSTNM
1466	003556	057422	DF121	
1467				
1468			;ITEM 200	
1469	003560	062472	EM200	;STOO DID NOT STORE ALL QUAD'S PROPERLY

1470	003562	057347	DH121	
1471	003564	062540	DT200	;SEPRPC,\$TMP4,\$REGU,\$STSTNM
1472	003566	057422	DF121	
1473				
1474			;ITEM 201	
1475	003570	062546	EM201	;STATE 173 DID NOT CLEAR DST
1476	003572	051047	DH11	
1477	003574	051130	DT11	
1478	003576	000000	0	
1479				
1480			;ITEM 202	
1481	003600	062602	EM202	;AR DATA PAD ON SHIFT
1482	003602	062627	DH202	
1483	003604	062712	DT202	
1484	003606	062722	DF202	
1485				
1486			;ITEM 203	
1487	003610	062725	EM203	;OR DATA PAD ON SHIFT
1488	003612	062627	DH202	
1489	003614	062752	DT203	
1490	003616	062727	DF202	
1491				
1492			;ITEM 204	
1493	003620	062762	EM204	;FRMF SUB*SCOR NOT GOING LOW
1494	003622	050244	DH1	
1495	003624	050264	DT1	
1496	003626	000000	0	
1497				
1498			;ITEM 205	
1499	003630	063062	EM205	;FRMF SS XOR SD XOR SUB NOT GOING LOW
1500	003632	050244	DF1	
1501	003634	050264	DT1	
1502	003636	000000	0	
1503				
1504			;ITEM 206	
1505	003640	063127	EM206	;FRMJ IL(0)H NOT GOING LOW OR NOT
1506	003642	050244	DH1	;GETTING TO FRMA RAD02 AS A HIGH
1507	003644	050264	DT1	
1508	003646	000000	0	
1509				
1510			;ITEM 207	
1511	003650	063225	EM207	;FXPP ABS VAL ROM OUT NOT GETTING
1512	003652	050244	DH1	;TO OUT OF RANGE GATE AS A LOW
1513	003654	050264	DT1	
1514	003656	000000	0	
1515				
1516			;ITEM 210	
1517	003660	063324	EM210	;FRHC FD(1) B L NOT GOING HIGH
1518	003662	053211	DH35	
1519	003664	053276	DT35	
1520	003666	053310	DF35	
1521				
1522			;ITEM 211	
1523	003670	063361	EM211	;FRHC PALUS9 NOT GETTING TO FRHK AND
1524	003672	053211	DH35	;FRHL AS A LOW
1525	003674	053276	DT35	

1526	003676	053310	DF35	
1527				
1528			;ITEM 212	
1529	003700	063443	EM212	;FRHC FALU59 NOT GETTING TO FRHL
1530	003702	053211	DH35	;ASHF CONT 3A(0) AS A LOW
1531	003704	053276	DT35	
1532	003706	053310	DF35	
1533				
1534			;ITEM 213	
1535	003710	063534	EM213	;A BIT OF FRHK NORM POS ENCODER DID
1536	003712	053211	DH35	;NOT GO LOW OP DID NOT GET THRU THE
1537	003714	053276	DT35	;BMX ON FXP AS A LOW.
1538	003716	053310	DF35	
1539				
1540			;ITEM 214	
1541	003720	063667	EM214	;FRHC FALU59 DOES NOT GET TO FRHK NORM
1542	003722	053211	DH35	;POS ENCODER AS A LOW
1543	003724	053276	DT35	
1544	003726	053310	DF35	
1545				
1546			;ITEM 215	
1547	003730	063762	EM215	;FRMF FP REQ DID NOT LOAD IN STATE 176
1548	003732	050244	DH1	
1549	003734	050264	DT1	
1550	003736	000000	0	
1551				
1552			;ITEM 216	
1553	003740	064044	EM216	;FRHL ASHF CONT 3A(0) NOT GETTING
1554	003742	051047	DH11	;TO BMX AS A HIGH
1555	003744	051130	DT11	
1556	003746	000000	0	
1557				
1558			;ITEM 217	
1559	003750	064141	EM217	;A BIT OF FRHK NORM POS ENCODER
1560	003752	051047	DH11	;NOT GETTING TO THE BMX AS A HIGH
1561	003754	051130	DT11	
1562	003756	000000	0	
1563			;ITEM 220	
1564	003760	064241	EM220	;FXPJ EALU06 XOR EALU03 NOT GOING LOW
1565	003762	050244	DH1	
1566	003764	050264	DT1	
1567	003766	000000	0	
1568				
1569			;ITEM 221	
1570	003770	056670	EM114	
1571	003772	064306	DH221	
1572	003774	064400	DT221	
1573	003776	000000	0	
1574				
1575			;ITEM 222	
1576	004000	064416	EM222	;DOUBLE PRECISION ROUND FAILED
1577	004002	057347	DH121	
1578	004004	057414	DT121	
1579	004006	057422	DF121	
1580				
1581			;ITEM 223	

1582	004010	064576	EM223	;STATE 313 DID NOT SUB A MINUS ONE
1583	004012	057347	DH121	;FROM EXPONENT
1584	004014	057414	DT121	
1585	004016	057422	DF121	
1586				
1587			;ITEM 274	
1588	004020	064663	EM224	;FRWF SS XOR SD XOR SUB DID NOT GO HIGH
1589	004022	050244	DH1	;OR ADD *SC<8 DID NOT GET TO
1590	004024	050264	DT1	;FRWA RAD03 AS A HIGH.
1591	004026	000000	0	
1592				
1593			;ITEM 225	
1594	004030	065034	EM225	;FRLC FMX02 DID NOT GO LOW WITH FT(1) H
1595	004032	050244	DH1	
1596	004034	050264	DT1	
1597	004036	000000	0	
1598				
1599			;ITEM 226	
1600	004040	065106	EM226	;FRWF SS XOR SD XOR SUB DID NOT GO H
1601	004042	050244	DH1	
1602	004044	050264	DT1	
1603	004046	000000	0	
1604				
1605			;ITEM 227	
1606	004050	065155	EM227	;FXPP OUT OF RANGE NOT GOING HIGH
1607	004052	050244	DH1	;OR NOT GETTING TO FXPJ BALD SWR
1608	004054	050264	DT1	;AS A LOW
1609	004056	000000	0	
1610				
1611			;ITEM 230	
1612	004060	065270	EM230	;ABS VAL ROM FAILED
1613	004062	050244	DH1	
1614	004064	050264	DT1	
1615	004066	000000	0	
1616				
1617			;ITEM 231	
1618	004070	065313	EM231	;FRHC FMX34 NOT GOING LOW WITH FT(1)
1619	004072	050244	DH1	
1620	004074	050264	DT1	
1621	004076	000000	0	
1622				
1623			;ITEM 232	
1624	004100	065360	EM232	;FRHK NORM POS ENCODER FAILED
1625	004102	065415	DH232	
1626	004104	065522	DT232	
1627	004106	065536	DF232	
1628				
1629			;ITEM 233	
1630	004110	065543	EM233	;FRHK NORM POS SWR DID NOT GO LOW
1631	004112	065415	DH232	
1632	004114	065522	DT232	
1633	004116	065536	DF232	
1634				
1635			;ITEM 234	
1636	004120	065604	EM234	;FXPP OUT OF RANGE NOT GETTING TO FRMA
1637	004122	050244	DH1	;AS A LOW OR NOT GETTING TO RAD03

1638	004124	050264	DT1	;AS A HIGH
1639	004126	000000	0	
1640				
1641			;ITEM 235	
1642	004130	065706	EM235	;FXPJ EALU06 XOR EALU04 NOT GOING LOW
1643	004132	050244	DH1	
1644	004134	050264	DT1	
1645	004136	000000	0	
1646				
1647			;ITEM 236	
1648	004140	065754	EM236	;FXPJ EALU06 XOR EALU05 DID NOT GO LOW
1649	004142	050244	DH1	
1650	004144	050264	DT1	
1651	004146	000000	0	
1652				
1653			;ITEM 237	
1654	004150	066022	EM237	;ABS VAL POW MSP DID NOT GO HIGH
1655	004152	050244	DH1	
1656	004154	050264	DT1	
1657	004156	000000	0	
1658				
1659			;ITEM 240	
1660	004160	066062	EM240	;FRMC FMX34 NOT GOING LOW WHEN PD=1
1661	004162	050244	DH1	
1662	004164	050264	DT1	
1663	004166	000000	0	
1664				
1665			;ITEM 241	
1666	004170	066127	EM241	;STATE 247 DID NOT ROUND
1667	004172	053211	DH35	
1668	004174	053276	DT35	
1669	004176	053310	DF35	
1670				
1671			;ITEM 242	
1672	004200	066157	EM242	;STATE 121 DID NOT NORMALIZE PROPERLY
1673	004202	053211	DH35	
1674	004204	053276	DT35	
1675	004206	053310	DF35	
1676				
1677			;ITEM 243	
1678	004210	066224	EM243	;STATE 167 DID NOT CLEAR AR LOW
1679	004212	057347	DH121	
1680	004214	057414	DT121	
1681	004216	057422	DF121	
1682				
1683			;ITEM 244	
1684	004220	066763	EM244	;EITHER SFO BRANCH FAILED OR
1685	004222	051576	DH15	;CC'S WERE LOADED WRONG.
1686	004224	051656	DT15	
1687	004226	000000	0	
1688				
1689			;ITEM 245	
1690	004230	066433	EM245	;OVERFLOW DID NOT OCCUR
1691	004232	053211	DH35	
1692	004234	053276	DT35	
1693	004236	053310	DF35	

1694				
1695			;ITEM 246	
1696	004240	066462	EM246	;FRMA BOU+87 DID NOT GO LOW ON BZ
1697	004242	053211	DH35	
1698	004244	053276	DT35	
1699	004246	053310	DF35	
1700				
1701			;ITEM 247	
1702	004250	066523	EM247	;STATE 162 DID NOT ADD 1 TO EXP
1703	004252	051576	DH15	
1704	004254	051656	DT15	
1705	004256	000000	0	
1706				
1707			;ITEM 250	
1708	004260	066566	EM250	;FRHC FMX75 NOT GOING HIGH
1709	004262	051047	DH11	
1710	004264	051130	DT11	
1711	004266	000000	0	
1712				
1713			;ITEM 251	
1714	004270	066620	EM251	;FRLC FMX19 NOT GOING LOW
1715	004272	051047	DH11	
1716	004274	051130	DT11	
1717	004276	000000	0	
1718				
1719			;ITEM 252	
1720	004300	066651	EM252	;DATA BAD ON STCFI+INT=-1
1721	004302	051047	DH11	
1722	004304	051130	DT11	
1723	004306	000000	0	
1724				
1725			;ITEM 253	
1726	004310	066702	EM253	;FRLC FMX19 NOT GOING HIGH
1727	004312	064306	DH221	
1728	004314	064400	DT221	
1729	004316	000000	0	
1730				
1731			;ITEM 254	
1732	004320	066734	EM254	;FRHH Q FILL DID NOT FILL PROPERLY
1733	004322	051047	DH11	
1734	004324	051130	DT11	
1735	004326	000000	0	
1736				
1737			;ITEM 255	
1738	004330	066776	EM255	;FALU DID NOT TAKE LOGICAL AND PROPERLY
1739	004332	051047	DH11	
1740	004334	051130	DT11	
1741	004336	000000	0	
1742				
1743			;ITEM 256	
1744	004340	067045	EM256	;FRHH Q FILL FAILED ON INTEGER LCNG
1745	004342	064306	DH221	
1746	004344	064400	DT221	
1747	004346	000000	0	
1748				
1749			;ITEM 257	

1750	004350	067110	EM257	;PALU LOGICAL AND FAILED IN LOW 24 BITS
1751	004352	064306	DH221	
1752	004354	064400	DT221	
1753	004356	000000	0	
1754				
1755			;ITEM 260	
1756	004360	067157	EM260	;GUARD BITS FAILED IN AR
1757	004362	057347	DH121	
1758	004364	057414	DT121	
1759	004366	057422	DF121	
1760				
1761			;ITEM 261	
1762	004370	067207	EM261	;GUARD BITS FAILED IN QR
1763	004372	057347	DH121	
1764	004374	057414	DT121	
1765	004376	057422	DF121	
1766				
1767			;ITEM 262	
1768	004400	067237	EM262	;ABS VAL ROM FAILED
1769	004402	067262	DH262	;ERRPC DATA APS VAL ROM TEST NG
1770				; EXPECT ACTUAL ADDRESS ROMOUT
1771	004404	067370	DT262	;SERRPC,\$TMP0,\$TMP2,\$REG4,\$REG5,\$STSTW
1772	004406	067406	DF262	;0,1,1,0,0,0
1773				
1774			;ITEM 263	
1775	004410	067237	EM262	
1776	004412	067414	DH263	
1777	004414	067522	DT263	
1778	004416	067534	DF263	
1779				
1780			;ITEM 264	
1781	004420	067540	EM264	;BIT STUCK IN QR DATA PATH
1782	004422	057347	DH121	
1783	004424	057414	DT121	
1784	004426	057422	DF121	
1785				
1786			;ITEM 265	
1787	004430	067572	EM265	;IRCC DSTWO L NOT GOING LOW
1788	004432	050244	DH1	;CHECK FP11C JUMPER
1789	004434	050264	DT1	
1790	004436	000000	0	
1791				
1792			;ITEM 266	
1793	004440	067651	EM266	;FRMH WRITE ACD1 OR WRITE ACD0
1794	004442	057347	DH121	;NOT GOING HIGH WITH PD(1) LOW
1795	004444	057414	DT121	
1796	004446	057422	DF121	
1797				
1798			;ITEM 267	
1799	004450	067745	EM267	;FXPC FCLD FM STUCK LOW
1800	004452	050244	DH1	
1801	004454	050264	DT1	
1802	004456	000000	0	
1803				
1804			;ITEM 270	
1805	004460	067774	EM270	;FRHD DISABLE LOW FMX OR DISABLE

1806	004462	057347	DH121	;ROUND FMX NOT GOING HIGH
1807	004464	057414	DT121	
1808	004466	057422	DF121	
1809				
1810			;ITEM 271	
1811	004470	070065	EM271	;FRHD OR FRLA DISABLE LOW Q MUX
1812	004472	057347	DH121	;NOT GOING HIGH
1813	004474	057414	DT121	
1814	004476	057422	DF121	
1815				
1816			;ITEM 272	
1817	004500	070141	EM272	;EITHER FRHD OR FRLC DISABLE LOW FMX
1818	004502	057347	DH121	;NOT GOING HIGH OR FRLC FMX02 NOT
1819	004504	057414	DT121	;GOING LOW OR FALU BAD
1820	004506	057422	DF121	
1821				
1822			;ITEM 273	
1823	004510	070274	EM273	;EITHER FRHD DISABLE HI FMX OR
1824	004512	057347	DH121	;FMX34 NOT GOING HIGH OR FALU BAD
1825	004514	057414	DT121	
1826	004516	057422	DF121	
1827				
1828			;ITEM 274	
1829	004520	070373	EM274	;FRLC FMX02 NOT GOING LOW WITH
1830	004522	064306	DH221	;INTEGER INC ON A LOW
1831	004524	064400	DT221	
1832	004526	000000	0	
1833				
1834			;ITEM 275	
1835	004530	070456	EM275	;UNEXPECTED TRAP TO 4
1836	004532	070503	DH275	;ERRPC PCOFTPP ERRREG TEST NO
1837	004534	070540	DT275	;SERRPC,\$TMP0,\$TMP1,\$STSTNM
1838	004536	000000	0	
1839				
1840			;ITEM 276	
1841	004540	070456	EM275	;UNEXPECTED TRAP TO 4 (11/45)
1842	004542	050244	DH1	
1843	004544	050264	DT1	
1844	004546	000000	0	
1845				
1846			;ITEM 277	
1847	004550	070552	EM277	;UNEXPECTED TRAP TO 114
1848	004552	070601	DH277	;ERRPC PCOFTPP ERRREG LOADRS HIADRS TEST NO
1849	004554	070654	DT277	;SERRPC,\$TMP0,\$TMP1,\$TMP2,\$TMP3,\$STSTNM
1850	004556	000000	0	
1851				
1852			;ITEM 300	
1853	004560	070552	EM277	;UNEXPECTED TRAP TO 114 (11/45)
1854	004562	050244	DH1	
1855	004564	050264	DT1	
1856	004566	000000	0	
1857				
1858			;ITEM 301	
1859	004570	070677	EM301	;UNEXPECTED TRAP TO 244
1860	004572	070721	DH301	;ERRPC PCOFTPP PEC FFA TEST NO
1861	004574	070764	DT301	;SERRPC,\$TMP0,\$TMP1,\$TMP2,\$STSTNM

1862	004576	000000	0		
1863					
1864			;ITEM 302		
1865	004600	071000	EM302	;FXPL EXPA EQ 0 NOT GOING LOW	
1866	004602	071036	DW302	;ERRPC	EXPA DATA TEST NO
1867	004604	071100	DT302	;SERHPC,STMP0,\$STSTNM	
1868	004606	071110	DF302	;0,1,0	
1869					
1870			;ITEM 303		
1871	004610	071113	EM303	;FXPL EXPB EQ 0 NOT GOING LOW	
1872	004612	071150	DW303	;ERRPC	EXPB DATA TEST NO
1873	004614	071100	DT302		
1874	004616	071110	DF302		


```

1875
1876      .SBTTL  APT PARAMETER BLOCK
1877
1878      ;;*****
1879      ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
1880      ;;*****
1881      004620      .SX=.      ;;SAVE CURRENT LOCATION
1882      000024      .=24      ;;SET POWER FAIL TO POINT TO START OF PROGRAM
1883      000024      000200      200      ;;FOR APT START UP
1884      000044      .=44      ;;POINT TO APT INDIRECT ADDRESS PTR.
1885      000044      004620      $APTHDR ;;POINT TO APT HEADER BLOCK
1886      004620      .=.SX      ;;RESET LOCATION COUNTER
1887      ;;*****
1888      ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
1889      ;INTERFACE SPEC.
1890
1891      004620      $APTHD:
1892      004620      000000      $SHRFS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
1893      004622      001234      $MADR: .WORD $MAIL      ;;ADDRESS OF APT MAILBOX (BITS 0-15)
1894      004624      000007      $TSTM: .WORD 2      ;;RUN TIME OF LONGEST TEST
1895      004626      000005      $PSTM: .WORD 5      ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
1896      004630      000000      $UNITM: .WORD      ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
1897      004632      000014      .WORD      SETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
  
```

```

1898 004634
1899
1900 004634 012706 001100
1901 004640 005026
1902 004642 022706 001126
1903 004646 001374
1904 004650 012706 001100
1905
1906 004654 012737 044032 000020
1907 004662 012737 000340 000022
1908 004670 012737 044364 000030
1909 004676 012737 000340 000032
1910 004704 012737 047646 000034
1911 004712 012737 000340 000036
1912 004720 012737 047712 000024
1913 004726 012737 000340 000026
1914 004734 013737 043656 043650
1915 004742 005037 001220
1916 004746 005037 001222
1917 004752 112737 000001 001115
1918 004760 012737 004760 001106
1919 004766 012737 004766 001110
1920
1921
1922 004774 013746 000004
1923 005000 012737 005036 000004
1924 005006 012737 177570 001136
1925 005014 012737 177570 001140
1926 005022 022777 177777 174106
1927 005030 001013
1928
1929 005032 005737 000001
1930 005036 012737 000176 001136 64$:
1931 005044 012737 000174 001140
1932 005052 012716 005060
1933 005056 000006
1934 005060 012637 000004 65$:
1935 005064 005037 001247
1936 005070 132737 000200 001255
1937 005076 001403
1938 005100 012737 001756 001136
1939 005106
1940
1941 005106 005227 177777
1942 005112 001050
1943 005114 022737 044012 000042
1944 005122 001444
1945 005124 104400 005132
1946 005130 000441
1947
1948 005234
1949 005234 012737 005254 000004
1950 005242 005737 177766
1951 005246 012737 177766 001406
1952 005254 012737 047624 000244 LOOP:
1953 005262 012737 000340 000246

```

```

START:
;;CLEAR THE COMMON TAGS (SCMTAG) AREA
MOV #SCMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
CLR (R6)+ ;;CLEAR MEMORY LOCATION
CMP #SBDAT,R6 ;;DONE?
BNE -6 ;;LOOP BACK IF NO
MOV #STACK,SP ;;SETUP THE STACK POINTER
;;INITIALIZE A FEW VECTORS
MOV #SSCOPE,@PIOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
MOV #340,@PIOTVEC+2 ;;LEVEL 7
MOV #SERROR,@EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
MOV #340,@EMTVEC+2 ;;LEVEL 7
MOV #STRAP,@TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
MOV #340,@TRAPVEC+2;LEVEL 7
MOV #SPWRDN,@PWRVEC ;;POWER FAILURE VECTOR
MOV #340,@PWRVEC+2 ;;LEVEL 7
MOV #SENDCT,SEOPCT ;;SETUP END-OF-PROGRAM COUNTER
CLR $TIMES ;;INITIALIZE NUMBER OF ITERATIONS
CLR $ESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
MOV #1,$EMAX ;;ALLOW ONE ERROR PER TEST
MOV #,$SLPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
MOV #,$SLPEPR ;;SETUP THE ERROR LOOP ADDRESS
;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
MOV @ERRVEC,-(SP) ;;SAVE ERROR VECTOR
MOV #64,$@ERRVEC ;;SET UP ERROR VECTOR
MOV #DSWR,SWR ;;SETUP FOR A HARDWARE SWITCH REGISTER
MOV #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
CMP #-1,$SWR ;;TRY TO REFERENCE HARDWARE SWR
BNE 65$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
;;AND THE HARDWARE SWR IS NOT = -1
TST #1 ;;FORCE A TRAP THROUGH ERRVEC
MOV #SWREG,SWR ;;POINT TO SOFTWARE SWR
MOV #DISPREG,DISPLAY ;;POINT TO SOFTWARE DISPLAY REG
MOV #65,$(SP) ;;REPLACE OLD PC WITH NEW
RTT ;;RESTORE PC AND PSW
MOV (SP)+,@ERRVEC ;;RESTORE ERROR VECTOR
CLR $PASS ;;CLEAR PASS COUNT
BTTP #APTSIZE,$EMVM ;;TEST USER SIZE UNDER APT
BEQ 3$ ;;YES,USE NON-APT SWITCH
MOV #SSWREG,SWR ;;NO,USE APT SWITCH REGISTER
3$:
;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
INC #-1 ;;FIRST TIME?
BNE 66$ ;;BRANCH IF NO
CMP #SENDAD,@42 ;;ACT-11?
BEQ 66$ ;;BRANCH IF YES
TYPE ,67$ ;;TYPE ASCII STRING
BR 66$ ;;GET OVER THE ASCII
;;67$: .ASCII <CRLF>"MAINTDEC-11-DEFPAA-A...PDP 11-45/55/70...FP11-C DIAGNOSTIC PART 1"
66$:
MOV #LOOP,ERRVEC ;;SETUP ERRVEC FOR TIMEOUT
TST CPUERR ;;11/45 OR 11/70
MOV #CPUERR,$CPUERR ;;DO THIS IF 11/70
MOV #FPSPUR,@FPPVEC;SET VECTOR TO SPURIOUS HANDLER
MOV #PR7,@FPPVEC+2 ;SET PRIORITY 7 IN VECTOR PSW

```

```

1954 005270 012737 047504 000004
1955 005276 012737 000340 000006
1956 005304 012737 047550 000114
1957 005312 012737 000340 000116
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991 005320 000004
1992 005322 012737 000300 001220
1993 005330 012737 000001 001240
1994 005336 012737 005536 001222
1995 005344 012737 005462 000010
1996 005352 012737 005466 000024
1997 005360 012737 005506 000100
1998 005366 012737 005374 001110
1999 005374 000230
2000 005376 012700 125252
2001 005402 012701 040357
2002 005406 004737 045256
2003 005412 170101
2004 005414 000240
2005 005416 000237
2006 005420 004737 005516
2007 005424 020016
2008 005426 001004
2009 005430 012706 001100
  
```

```

MOV    #CPSPOR, @ERRVEC; SET TRAP TO 4 VECTOR
MOV    #PR7, @ERRVEC+2 ; SET TRAP TO 4 PSW
MOV    #CACHSPU, @CACHVEC; SET PARITY ERROR VECTOR
MOV    #PR7, @CACHVEC+2

;;*****
;*TEST 1      LDFPS*NO
;*
;*POSSIBLE CPU FAILURES
;*A FORK
;* IF RACK A7 RAB00 DOES NOT GO LOW A TRAP TO LOCATION 10
;* WILL OCCUR.
;*CFORK
;* IF IRCC F/CLASS DOES NOT GO LOW EXECUTION WILL GO TO STATE 200
;* WHICH WILL CAUSE A TRAP THRU LOCATION 24.
;* IF IRCC F/CLASS DOES NOT GET TO E40(A0) OR IF E(40) IS BAD EXECUTION
;* WILL GO TO STATE 201 (JSR.10). THIS WILL CAUSE R0 TO BE PUSHED
;* ONTO THE STACK.
;* IF IRCC F/CLASS DOES NOT GET TO IRCC F21 OR E21 IS BAD, EXECUTIN WILL GO TO
;* STATE 210 (NEG.90). THIS WILL CAUSE THE SR+1 TO BE PUT IN THE DR.
;* IF IRCC DSTMO DOES NOT GO LOW ON (SMO*IR(0:6)0*PCLASS) EXECUTION WILL GO TO
;* STATE MAT.00, CAUSING THE PROCESSOR TO HANG UNTIL THE LINE CLOCK INTERRUPTS.
;*BEN 17
;* IF RACK E64(A0) DOES NOT GO HIGH EXECUTION WILL GO TO STATE 153.
;* THIS WILL CAUSE A TRAP THRU LOCATIN 24. THE STACKED PC WILL BE
;* THE PC OF THE LDFPS INSTRUCTION.
;* IF RACK RIP+FP SYNC DOES NOT GO LOW EXECUTION WILL LOOP ON STATE 133.
;* THIS WILL HANG THE PROCESSOR UNTIL THE PRO (LINE CLOCK INTERRUPT) OCCURS.
;*BEN 7
;* IF RACK BE75 DOES NOT GET FP REG WT AS A LOW EXECUTION WILL GO TO
;* STATE 333. THIS WILL CAUSE THE DESTINATION REGISTER TO BE CLOBBED.
;*
;* CPU ROM FLOW - 101,314,211,133,173
;* FPP ROM FLOW -30,60
;;*****
TST1:  SCOPE
MOV    #300, $TIMES      ;;DO 300 ITERATTIONS
MOV    #STN-1, $TESTN    ;;SET TEST NUMBER IN MAIL BOX
MOV    #TST2, $ESCAPE    ;;ESCAPE TO TEST 2 ON EPROR
MOV    #2$, @RESVEC      ;SETUP RESERVED VEC
MOV    #3$, @PWRVEC      ;SETUP POWER VEC
MOV    #4$, @CLKVEC      ;SETUP CLOCK VEC
MOV    #6$, @SLPERR      ;SET ERROR LOOP
6$:    SPL 0              ;ENSURE PRIORITY IS 0
MOV    #125252, R0       ;SETUP R0
MOV    #40357, R1        ;SET DATA TO LOAD IN R1
JSR    PC, GETTIK        ;START LINE CLOCK
1$:    LDFPS R1           ;EXECUTE INSTRUCTION UNDER TEST
NOP
SPL 7                    ;LOCK OUT LINE CLOCK INTERRUPT
JSR    PC, WAIT          ;GO WAIT FOR LINE CLOCK TO FINISH
CMP    R0, (SP)          ;DID C FORK FAIL TO JSR.10?
BNE 7$                   ;BRANCH IF NC
MOV    #STACK, SP        ;RESTORE STACK
  
```

2010	005434	000005				KESFT);CLEAR PPU
2011	005436	104001				ERROR	1);C FORK FAILED TO JSR.10
2012	005440	022701	125253		7S:	CMP	#125253,R1);DID C FORK FAIL TO NEG.90?
2013	005444	001002				BNE	8S);BRANCH IF NO
2014	005446	000005				RESET);CLEAR PPU
2015	005450	104002				ERROR	2);C FORK FAILED TO NEG.90
2016	005452	022701	040357		8S:	CMP	#40357,R1);DID BEN 7 FAIL & CLOBBER #1?
2017	005456	001427				BEQ	TST2);)BRANCH IF NO
2018	005460	104003				ERROR	3);BEN 7 FAILED
2019	005462	022626			2S:	CMP	(SP)+,(SP)+);RESTORE STACK
2020	005464	104004				ERROR	4);A FORK FAILED
2021	005466	022626			3S:	CMP	(SP)+,(SP)+);RESTORE THE SP
2022	005470	022766	005412	177774		CMP	#1S,-4(SP));WAS STACKED PC SAME AS FP INSTR?
2023	005476	001001				BNE	9S);BRANCH IF NO
2024	005500	104005				ERROR	5);BEN 17 FAILED TO STATE POP.60
2025	005502	000005			9S:	RESET);CLEAR PPU
2026	005504	104006				ERROR	6);C FORK FAILED TO STATE ZAP.00
2027	005506	042737	000100	177546	4S:	BIC	#BIT6,LKSTAT);CLEAR IE BIT IN STATUS
2028	005514	104007				ERROR	7);BEN 17 FAILED TO STATE POP.30
2029);OR C FORK FAILED TO WAT.00.
2030	005516	042737	000100	177546	WAIT:	BIC	#BIT6,LKSTAT);CLEAR IE BIT
2031	005524	105737	177546		1S:	TSTB	LKSTAT);IS LYNE CLOCK DONE?
2032	005530	100375				BPL	1S);BRANCH IF NO
2033	005532	000230				SPL	0);RETURN TO LEVEL 0
2034	005534	000207				RTS	PC);RETURN

```

2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065

```

```

);*****
);*TEST 2      INTERFACE DATA PATH
);*
);*      THIS TEST TESTS THE INTERFACE DATA PATH BY FLOATING A ONE
);*      AND THEN A ZERO THROUGH THE DESTINATION REGISTER.
);*      A PIR TRAP IS USED TO YANK THE CPU OUT OF THE PFP
);*      SERVICE FLOW CAUSING THE DESTINATION REGISTER TO BE
);*      RESTORED FROM THE PDR REGISTER.
);*NOTE: A BIT FAILURE IN THIS TEST COULD BE CAUSED BY ONE OF
);*      THREE DATA PATHS: 1) CPU TO PDR; 2) PDR TO DOMUX; OR
);*      3) DOMUX TO CPU.
);*
);*CPU FAILURES
);*C FORK
);*      IF IRCC DSTND L DOES NOT GO HIGH EXECUTION WILL GO TO STATE
);*      POP.00. THIS WILL HANG THE PROCESSOR IN A LOOP BETWEEN
);*      ROM ADDRESSES 101 AND 314.
);*BEN 15
);*      IF IRCB FJ CLASS DOES NOT GO LOW EXECUTION WILL GO TO
);*      STATE D12.10. THIS WILL FINISH THE BUS OPERATION AND THEN
);*      THE B FORK WILL FAIL TO STATE S13.01. THIS WILL HANG THE
);*      PROCESSOR IN A LOOP AROUND ROM STATES 111,175,22, AND 27.
);*B FORK
);*      IF IRCB BO PA01 DOES NOT GO LOW EXECUTION WILL GO TO
);*      STATE JSR.00. THIS WILL CAUSE THE PC TO BE PUSHED ON
);*      THE STACK AND THE CONTENTS OF THE DESTINATION TO BE PUT
);*      IN THE PC.
);*      IF IRCB BO PA03 DOES NOT GO LOW EXECUTION WILL GO TO
);*      STATE S67.00. THIS WILL CAUSE A DESTINATION MODE 6
);*      TO START EXECUTING. THIS FAILURE WILL BE FOUND BY

```

```

2066 ;* MAKING THE INDEX ODD, SO AN ODD ADDRESS TRAP WILL OCCUR.
2067 ;*BEN 17
2068 ;*
2069 ;* IF PACK BRCA05 DOES NOT GO LOW (ON BRQ*(T+CONF)) EXECUTION
2070 ;* WILL GO TO STATE 327. THIS WILL CAUSE THE FPU TO COMPLETE
2071 ;* THE INSTRUCTION BEFORE THE INTERRUPT OCCURS.
2072 ;*
2073 ;* CPU ROM FLOW-101,314,111,135,36,347,150,245
2074 ;*****
2074 005536 000004 TST2: SCOPE
2075 005540 012737 000002 001240 MOV #STN-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX
2076 005546 012737 006150 001222 MOV #TST3,$ESCAPE ;;ESCAPE TO TEST 3 ON ERROR
2077 005554 012737 047712 000024 MOV #SPWRDN,PWRVEC ;RESTORE POWER FAIL VECTOR
2078 005562 012737 005636 000240 MOV #1,$PIRQVEC ;SET PIRQ VECTOR ADDRESS
2079 005570 012737 000340 000242 MOV #PR7,PIRQVEC+2 ;SET VECTOR PSM TO LEVEL 7
2080 005576 012737 005700 000004 MOV #2,$ERRVEC ;SET THE ERROR VECTOR
2081 005604 000237 SPL 7 ;ENSURE CPU IS AT LEVEL 7
2082 005606 012737 005614 001110 MOV #.+6,#$SLPERR ;SET ERROR LOOP
2083 005614 052737 100000 177772 BIS #BIT15,PIRQ ;SET PIR LEVEL 7
2084 005622 012737 177777 005634 MOV #-1,$S ;SET DATA WORD(MUST BE ODD)
2085 005630 000230 SPL 0 ;LOWER PRIORITY
2086 005632 170227 4S: STFPS (PC)+ ;AND EXECUTE INSTRUCTION
2087 005634 177777 3S: .WORD -1 ;DATA WORD
2088 005636 022716 005632 1S: CMP #4,$(SP) ;DID STACKED PC COME OUT OK?
2089 005642 001427 BEQ 7S ;BRANCH IF YES
2090 005644 012737 005632 001202 MOV #4,$$TMP1 ;SAVE EXPECTED VALUE
2091 005652 011637 001200 MOV (SP),$TMP0 ;SAVE RECEIVED VALUE
2092 005656 005037 177772 CLR PIRQ ;TURN OFF PIRQ
2093 005662 022626 CMP (SP)+,(SP)+ ;RESTORE THE STACK
2094 005664 022766 005636 177774 CMP #1,$-4(SP) ;DID FP INSTRUCTION COMPLETE?
2095 005672 001001 BNE 5S ;BRANCH IF NO
2096 005674 104023 ERROR 23 ;BEN 17 FAILED
2097 005676 104025 5S: ERROR 25 ;STACKED PC IS WRONG
2098 005700 005037 177772 2S: CLR PIRQ ;TURN OFF PIRQ
2099 005704 022626 CMP (SP)+,(SP)+ ;RESTORE THE STACK
2100 005706 022766 005636 177774 CMP #1,$-4(SP) ;DID A DM6 TRY TO EXECUTE?
2101 005714 001001 BNE 6S ;BRANCH IF NO
2102 005716 104060 ERROR 60 ;IRCB BO RAB03 DOES NOT GO LOW
2103 005720 104061 6S: ERROR 61 ;IRCB BO RAB01 DOES NOT GO LOW
2104 ;FLOAT A ONE THROUGH THE FDR
2105 005722 012706 001100 7S: MOV #STACK,SP ;RESTORE THE SP
2106 005726 012700 000001 MOV #1,P0 ;INITIALIZE DESTINATION REG
2107 005732 010037 001164 MOV R0,$REG2 ;SAVE EXPECTED VALUE IN $REG2
2108 005736 012737 005776 000240 MOV #R$,PIRQVEC ;SETUP PIRQ VECTOR
2109 005744 012737 047504 000004 MOV #CPSPUR,ERRVEC ;RESTORE ERROR VECTOR
2110 005752 012702 000020 MOV #20,R2 ;SET SOR LOOP
2111 005756 012737 005764 001110 MOV #.+6,#$SLPERR ;SET ERROR LOOP
2112 005764 052737 100000 177772 BIS #BIT15,PIRQ
2113 005772 000230 9S: SPL 0 ;LOWER PRIORITY
2114 005774 170220 STFPS (R0)+ ;AND EXECUTE FP INSTRUCTION
2115 005776 022626 8S: CMP (SP)+,(SP)+ ;RESTORE STACK
2116 006000 020037 001164 CMP R0,$REG2 ;IS DATA OK?
2117 006004 001010 BNE 10S ;BRANCH IF NO
2118 006006 105737 001103 TSTP $ERPLG ;ANY ERRORS?
2119 006012 001367 BNE 9S ;BRANCH IF YES
2120 006014 006300 ASL R0 ;ROTATE THE ONE
2121 006016 010037 001164 MOV R0,$REG2
  
```

```

2122 006022 077215          SOB      R2,95          ;CONTINUE
2123 006024 000405          BP       11$
2124 006026 010037 001162    10$:     MOV      R0,SREG1    ;SAVE RECEIVED VALUE
2125 006032 005037 177772    CLR      PTRQ         ;TURN OFF PTRQ
2126 006036 104062          ERROR    62          ;DATA PATTERN FAILED
2127                                ;FLOAT A ZERO
2128 006040 012700 177776    11$:     MOV      #-2,R0      ;INITIALIZE DATA PATTERN
2129 006044 010037 001164    MOV      R0,SREG2
2130 006050 012737 006102 000240  MOV      #13,PIRQVEC  ;SETUP PIRQ VECTOR
2131 006056 012707 000020    MOV      #70,R7      ;SET SOB COUNT
2132 006062 012737 006070 001110  MOV      #.+6,#SLPERP ;SET ERROR LOOP
2133 006070 052737 100000 177772  BITS    #R15,PIRQ
2134 006076 000230          12$:     SPL      0          ;LOWER PRIORITY
2135 006100 170220          STFPS   (R0)+        ;ANDEXECUTE FP INSTRUCTION
2136 006102 022626          13$:     CMP      (SP)+,(SP)+  ;RESTORE THE STACK
2137 006104 020037 001164    CMP      R0,SREG2    ;IS DATA OK?
2138 006110 001346          BNE     10$         ;BRANCH IF NO
2139 006117 105737 001103    TSTR    $RFLG       ;ANY ERRORS YET?
2140 006116 001367          BNE     12$         ;BRANCH IF YES
2141 006120 000261          SEC
2142 006122 006100          ROL     R0          ;ROTATE THE ZERO
2143 006124 010037 001164    MOV      R0,SREG7
2144 006130 077216          SOB     R2,12$      ;CONTINUE
2145 006132 005037 177772    CLR      PTRQ       ;TURN OFF THE PTRQ
2146 006136 012737 000242 000240  MOV      #PIRQVEC+2,PIRQVEC;RESTORE PIRQ VECTOR
2147 006144 005037 000242    CLR      PIRQVEC+2
2148
2149                                ;*****
2150                                ;*TEST 3          STFPS*MO
2151                                ;*
2152                                ;*   THIS TEST STARTS OUT JUST VERIFYING THAT
2153                                ;*   LOAD FPS AND STFPS GET DECODED PROPERLY IN THE FPU.
2154                                ;*
2155                                ;*CPU FAILURES
2156                                ;*BEN 7
2157                                ;*   IF PACK BE75 DOES NOT GET FP REG WT AS A HIGH
2158                                ;*   EXECUTION WILL GO TO STATE FET.06. THIS WILL
2159                                ;*   CAUSE THE DESTINATION REGISTER TO REMAIN UNCHANGED.
2160                                ;*
2161                                ;*   IT THEN FLOATS A ONE AND A ZERO THROUGH THE FPS
2162                                ;*   REGISTER.
2163                                ;*
2164                                ;*   CPU ROM FLOW 101,314,211,133,173,333,365
2165                                ;*   FPU ROM FLOW 62,23
2166                                ;*****
2167 006150 000004          TST3:   SCOPE
2168 006152 012737 000003 001240  MOV      #STN-1,STESTN ;;SET TEST NUMBER IN MAIL BOX
2169 006160 012737 006402 001222  MOV      #TST4,$ESCAPE ;;ESCAPE TO TEST 4 ON ERROR
2170 006166 012737 000017 000010  MOV      #12,@RESVEC
2171 006174 012737 050132 000024  MOV      #SPWER,@PWRVEC
2172 006202 005000          CLR     R0          ;PUT DATA TO LOAD IN R0
2173 006204 170100          LDFPS  R0          ;CLEAR THE FPS
2174 006206 005300          DFC    R0          ;MAKE R0 ALL ONE'S
2175 006210 170200          1$:     STFPS  R0      ;EXECUTE INSTRUCTION UNDER TEST
2176 006212 022700 177777    CMP     #-1,R0      ;DID R0 CHANGE?
2177 006216 001001          BNE     2$          ;BRANCH IF YES

```

```
2178 006220 104010          ERROR 10          ;EITHER LDFPS OR STFPS DID NOT
2179                                     ;TRANSFER THE DATA
2180                                     ;FLOAT A ONE
2181 006222 012703 000017    2S:  MOV    #17,R3          ;SET SOB COUNT
2182 006226 012700 000001    MOV    #1,R0           ;INITIALIZE R0 TO FLOAT A "1"
2183 006232 010002          MOV    R0,R2           ;INITIALIZE R2
2184 006234 005001          CLR    R1              ;INITIALIZE R1
2185 006236 012737 006244 001110  MOV    #.+6,##SLPERR  ;SET ERROR LOOP
2186 006244 170100          4S:  LDFPS  R0           ;LOAD DATA PATTERN
2187 006246 170201          STFPS  R1             ;BRING IT BACK
2188 006250 020102          CMP    R1,R2           ;IS IT OK?
2189 006252 001405          BEQ    3S             ;BRANCH IF YES
2190 006254 010137 001162    MOV    R1,$REG1        ;SAVE RECIEVED VALUE
2191 006260 010237 001164    MOV    R7,$REG2        ;SAVE EXPECTED VALUE
2192 006264 104011          ERROR 11             ;BIT FAILED WHILE FLOATING A "1"
2193 006266 105737 001103    3S:  TSTR   SERFLG        ;ANY ERRORS YET?
2194 006272 001364          BNE    4S             ;BRANCH IF YES
2195 006274 006300          ASL    R0              ;GET NEXT VALUE TO LOAD
2196 006276 010002          MOV    R0,R2           ;AND TO COMPARE
2197 006300 042702 030000    BIC    #30000,R2       ;CLEAR UNUSED BITS
2198 006304 077321          SOB    R3,4S          ;CONTINUE
2199                                     ;FLOAT A ZERO
2200 006306 012703 000017    MOV    #17,R3          ;SET SOB COUNT
2201 006312 012700 177776    MOV    #-2,R0          ;INITIALIZE R0 TO FLOAT A "0"
2202 006316 012702 147776    MOV    #147776,R2      ;INITIALIZE R2 FOR COMPARISON
2203 006322          5S:
2204 006322 012737 006330 001110  MOV    #.+6,##SLPERR  ;SET ERROR LOOP
2205 006330 005001          CLR    R1              ;INIT R1
2206 006332 170100          LDFPS  R0           ;LOAD DATA INTO FPP
2207 006334 170201          STFPS  R1             ;BRING IT BACK
2208 006336 020102          CMP    R1,R2           ;IS IT OK?
2209 006340 001405          BEQ    6S             ;BRANCH IF YES
2210 006342 010137 001162    MOV    R1,##$REG1      ;SAVE RECIEVED VALUE
2211 006346 010237 001164    MOV    R2,##$REG2      ;AND EXPECTED VALUE
2212 006352 104011          ERROR 11             ;BIT FAILED WHILE FLOATING A ZERO
2213 006354 105737 001103    6S:  TSTR   SERFLG        ;ANY ERRORS YET?
2214 006360 001360          BNE    5S             ;BRANCH IF YES
2215 006362 000261          SEC                    ;MOVE THE ZERO
2216 006364 006100          ROL    R0              ;TO THE NEXT BIT POSITION
2217 006366 010002          MOV    R0,R2           ;AND OF COURSE R2 ALSO GETS
2218 006370 042702 030000    BIC    #30000,R2       ;MOVED
2219 006374 077326          SOB    R3,5S          ;CONTINUE
2220 006376 005000          CLR    P0              ;
2221 006400 170100          LDFPS  R0              ;CLEAR FPS REGISTER
2222
2223
2224                                     ;*****
2225                                     ;*TEST 4          SET MODES
2226                                     ;*
2227                                     ;* THIS TEST ENSURES THAT THE FOUR INSTRUCTIONS "SETF", "SETD", "SETL", AND
2228                                     ;* "SETI" FUNCTION.
2229                                     ;*
2230                                     ;* FPU ROM FLOW - 30,33
2231                                     ;*****
2232 006402 000004          TST4:  SCOPE
2233 006404 012737 000004 001240  MOV    #STN-1,$TESTN  ;;SET TEST NUMBER IN MAIL BOX
```

2234	006412	012737	047624	000244	MOV	#FPPUR, @FPPVEC	;RESTORE FPP VECTOR
2235	006420	012737	006472	000004	MOV	#55, @ERRVEC	;SET ERROR VEC
2236	006426	012737	006434	001110	MOV	#.+6, @SLPERR	;SET ERROR LOOP
2237	006434	005000			CLR	R0	
2238	006436	012701	000001		MOV	#1, R1	;MAKE R1 ODD
2239	006442	170100			LDFPS	R0	;ENSURE FPS CLEAR
2240	006444	170011			SETD		;EXECUTE INSTRUCTION UNDER TEST
2241	006446	170201			STFPS	R1	;GET FPS
2242	006450	022701	000200		CMP	#FD, R1	;DID FD BIT SET?
2243	006454	001410			BEQ	25	;BRANCH IF YES
2244	006456	012737	000200	001202	MOV	#FD, \$TMP1	;SAVE FPS
2245	006464	010137	001200		MOV	R1, \$TMP0	
2246	006470	104015			ERROR	15	;SETD FAILED
2247	006472	022626			CMP	(SP)+, (SP)+	;RESTORE THE SP
2248	006474	104265			ERROR	265	;IRCC DSTWO L NOT GOING LOW
2249							;CHECK FP11-C JUMPER
2250							;NOW TEST THE SETL
2251	006476						25:
2252	006476	012737	006504	001110	MOV	#.+6, @SLPERR	;SET ERROR LOOP
2253	006504	005000			CLR	R0	
2254	006506	170100			LDFPS	R0	;ENSURE FPS CLEAR
2255	006510	170012			SETL		;EXECUTE INSTRUCTION UNDER TEST
2256	006512	170201			STFPS	R1	;GET FPS
2257	006514	022701	000100		CMP	#FL, R1	;DID FL BIT SET?
2258	006520	001406			BEQ	35	;BRANCH IF YES
2259	006522	012737	000100	001202	MOV	#FL, \$TMP1	
2260	006530	010137	001200		MOV	R1, \$TMP0	;SAVE FPS
2261	006534	104016			ERROR	16	;SETL FAILED
2262							;NOW TEST SETF
2263	006536						35:
2264	006536	012737	006544	001110	MOV	#.+6, @SLPERR	;SET ERROR LOOP
2265	006544	012700	000200		MOV	#FD, R0	
2266	006550	005037	001202		CLR	\$TMP1	;SAVE EXPECTED VALUE
2267	006554	170100			LDFPS	R0	;SET FD BIT IN FPS
2268	006556	170001			SETF		;EXECUTE INSTRUCTION UNDER TEST
2269	006560	170201			STFPS	R1	;GET FPS
2270	006562	005701			TST	R1	;DID THE FD BIT CLEAR?
2271	006564	001403			BEQ	45	;BRANCH IF YES
2272	006566	010137	001200		MOV	R1, \$TMP0	;SAVE FPS
2273	006572	104017			ERROR	17	;SETF FAILED
2274							;NOW TEST SETI
2275							45:
2276	006574						
2277	006574	012737	006602	001110	MOV	#.+6, @SLPERR	;SET ERROR LOOP
2278	006602	012700	000100		MOV	#FL, R0	
2279	006606	170100			LDFPS	R0	;SET FL IN FPS
2280	006610	000777			SCC		
2281	006612	170002			SETI		;EXECUTE INSTRUCTION UNDER TEST
2282	006614	013702	177776		MOV	PSW, R2	;GET PSW
2283	006620	170201			STFPS	R1	;GET FPS
2284	006622	005701			TST	R1	;DID FL BIT CLEAR?
2285	006624	001403			BEQ	65	;BRANCH IF YES
2286	006626	010137	001200		MOV	R1, \$TMP0	;SAVE FPS
2287	006632	104020			ERROR	20	;SETI FAILED.
2288	006634	042702	177760		BIC	#177760, R2	;MASK OFF CC'S
2289	006640	022702	000017		CMP	#17, R2	;DID PSW CC'S CHANGE?

2290 006644 001401
 2291 006646 104267

BEQ TST5 ;;BRANCH IF NO
 ERROR 267 ;;FXPC FCLD EN STUCK LOW

2292
 2293
 2294
 2295
 2296
 2297
 2298
 2299
 2300
 2301
 2302
 2303
 2304
 2305
 2306
 2307
 2308
 2309
 2310
 2311

```

;*****
;*TEST 5          MICRO-BREAK TRAP
;*
;*   THIS TEST ENSURES THAT THE MICRO-BREAK TRAP FUNCTIONS PROPERLY
;*   IT IS TESTED HERE SO THAT IT CAN BE USED TO DEBUG ANY INSTRUCTION
;*   FAILURES THAT MAY OCCUR LATER ON.
;*
;*   IF THE NO-MEM BRANCH FAILS EXECUTION WOULD GO TO ONE OF TWO STATES:
;*   SET MODES, OR ILLEGAL OP CODE. SET MODES WOULD BE CAUSED BY FXPB FIRA00 OR 01
;*   FAILING AND ILLEGAL OP CODE WOULD BE CAUSED BY FXPB FIRA 02 OR 03 FAILING.
;*
;*   IF THE TRAP FAILS, THE FAILURE COULD BE IN THE FPP ON THE BACKPLANE
;*   OR IN THE CPU ON TMC(0135).
;*   THE TRAP VECTOR COULD FAIL TO 4 INSTEAD OF 244. THIS WOULD BE
;*   CAUSED TMCB PPTRAP L OR DAPE TV05*07.
;*
;*   FPU ROM FLOW -30,33,2,6
;*****
  
```

2312 006650 000004
 2313 006652 012737 000005 001240
 2314 006660 012737 006760 001222
 2315 006666 005001
 2316 006670 012737 006746 000004
 2317 006676 012737 006754 000244
 2318 006704 012703 000033
 2319 006710 170003
 2320 006712 012700 000020
 2321 006716 012737 006724 001110
 2322 006724 170100
 2323 006726 170002
 2324 006730 000240
 2325 006732 170101
 2326 006734 032700 000300
 2327 006740 001401
 2328 006742 104013
 2329 006744 104012
 2330
 2331 006746 170101
 2332 006750 022626
 2333 006752 104014
 2334 006754 022626
 2335 006756 170101

```

TST5:  SCOPE
        MOV    #STW-1,$TESTM    ;;SET TEST NUMBER IN MAIL BOX
        MOV    #TST6,$ESCAPE    ;;ESCAPE TO TEST 6 ON ERROR
        CLR    R1                ;ENSURE W1 CLEAR
        MOV    #25,@ERRVEC      ;SETUP ERROR VECTOR
        MOV    #35,@FPPVEC      ;SETUP FLOATING POINT VECTOR
        MOV    #33,R3            ;SETUP MICRO-BREAK DATA
        LDR    @FMM,R0           ;SET MICRO-BREAK TO TRAP ON STATE 23.
        MOV    #.+6,@SLPERR     ;
        LDFPS  R0                ;SET ERROR LOOP
        SETI   R0                ;SET MAINTENANCE MODE
        NOP                               ;SHOULD TRAP ON THIS INSTRUCTION
1$:     SETI   R1                ;CLEAR MAINTENANCE MODE
        LDFPS  R1                ;DID PD OR PL BIT GET SET?
        BIT    #FD+FL,R0        ;BRANCH IF NO
        BEQ    4$                ;NO MEM BRANCH FAILED TO "SET MODES"
        ERROR  13                ;DID NOT TRAP
4$:     ERROR  12
;TRAPPED TO WRONG VECTOR
2$:     LDFPS  R1                ;CLEAR MAINTENANCE MODE
        CMP    (SP)+,(SP)+      ;RESTORE STACK POINTER
        ERROR  14                ;TRAPPED TO WRONG VECTOR
3$:     CMP    (SP)+,(SP)+      ;RESTORE STACK POINTER
        LDFPS  R1                ;CLEAR MAINTENANCE MODE
  
```

2336
 2337
 2338
 2339
 2340
 2341
 2342
 2343
 2344
 2345

```

;*****
;*TEST 6          CFCC
;*
;*   THIS TEST ENSURES THAT THE PSM CONDITION CODES GET LOADED FROM
;*   THE FPP CONDITION CODES
;*
;*   IF IT FAILS, EITHER FXPC FCLD EN IS NOT GOING LOW OR IT IS NOT
;*   GETTING TO IRCE OR ICE CC<BR IS NOT GOING HIGH.
;*****
  
```

```

2346 ;*****
2347 006760 000004 TST6: SCOPE
2348 006762 012737 000006 001240 MOV #STN-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX
2349 006770 005000 CLR R0 ;
2350 006772 170100 LDPPS R0 ;CLEAR FPP CONDITION CODES
2351 006774 000277 SCC ;SET ALL THE PSW CONDITION CODES
2352 006776 170000 1S: CFCC ;EXECUTE INSTRUCTION UNDER TEST
2353 007000 013737 177776 001200 MOV PSW,$TMP0 ;GET PSW
2354 007006 042737 177760 001200 BIC #177760,$TMP0 ;MASK OFF CC'S
2355 007014 001403 BEQ TST7 ;;BRANCH IF INSTRUCTION WORFED
2356 007016 005037 001202 CLR $TMP1 ;SAVE EXPECTED VALUE
2357 007022 104021 ERROR 21 ;CFCC FAILED
2358 ;*****
2359 ;*TEST 7 LDF*-M0*IMM
2360 ;*
2361 ;*CPU FAILURES
2362 ;*BEN 10
2363 ;* IF FP CLASS DOES NOT GET TO RACK EXECUTION WILL GO TO
2364 ;* STATE PSV.30. THIS WILL CAUSE THE PROCESSOR TO HANG
2365 ;* IN ROM STATE 265 WAITING FOR FP SYNC.
2366 ;*BEN 7
2367 ;* IF FRMP FPREQ DOES NOT GET TO RACK AS A LOW THE CPU WILL
2368 ;* CONTINUE TO DO BUS OPERATIONS UNTIL THE ADDRESS TIMES OUT.
2369 ;*
2370 ;* IF THE DESTINATION REGISTER (R7) FAILS TO AUTO-INCREMENT
2371 ;* THE WORD AT 3S WILL TRY TO BE EXECUTED. THIS WILL CAUSE
2372 ;* A RESERVED INSTRUCTION TRAP.
2373 ;*
2374 ;*FPU FAILURES
2375 ;* ONLY THOSE FAILURES WHICH CAN BE DETECTED IN THIS TEST
2376 ;* ARE DISCUSSED HERE. OTHER POSSIBLE FAILURES ARE DEFERRED
2377 ;* TO THE NEXT TEST.
2378 ;*A BRANCH
2379 ;* IF FXPB FIRA10(0) N IS NOT GOING LOW OR NOT GETTING THRU
2380 ;* THE ROM ADDRESS MUX, EXECUTION WILL GO TO STATE 25. THIS
2381 ;* WILL CAUSE THE DESTINATION TO BE CLEARED.
2382 ;* IF FXPB A BRANCH MUX(A3) DOES NOT GO HIGH, EXECUTION WILL
2383 ;* GO TO STATE 13. THIS WILL CAUSE A STX*-M0 TO BE EXECUTED.
2384 ;*
2385 ;* CPU ROM FLOW-101, 114, 111, 135, 36, 377, 367, 362, 307, 237
2386 ;* FPU ROM FLOW-12, 133, 16
2387 ;*****
2388 007024 000004 TST7: SCOPE
2389 007026 012737 000007 001240 MOV #STN-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX
2390 007034 012737 007200 001222 MOV #TST10,$ESCAPE ;;ESCAPE TO TEST 10 ON ERROR
2391 007042 012737 007140 000004 MOV #7,$@ERRVFC ;SETUP ERROR VECTOR
2392 007050 012737 007174 000010 MOV #7,$@RESVFC ;SETUP RESERVED VECTOR
2393 007056 012737 007064 001110 MOV #.+6,$SLPERR ;SET ERROR LOOP
2394 007064 172427 1S: LDF (PC)+,$ACO ;EXECUTE INSTRUCTION UNDER TEST
2395 007066 007001 3S: .WORD 7001 ;DATA TO LOAD ( MUST BE
2396 ; AN ILLEGAL OP-CODE)
2397 007070 000403 BR 45
2398 007072 104066 ERROR 66 ;IDX POW FAILED
2399 007074 104066 ERROR 66 ; *
2400 007076 104066 ERROR 66 ; *
2401 007100 170000 4S: CFCC ;ENSURE FPU FINISHES LOAD
  
```

```
2402 007102 022737 007001 007066      CMP      #7001,35      ;DID DATA WORD CHANGE?
2403 007110 001433                    BEQ      TST10        ;;BRANCH IF NO
2404 007112 005737 007066              TST      35           ;DID IT CLEAR?
2405 007116 001404                    BEQ      85           ;BRANCH IF YES
2406 007120 012737 007001 007066      MOV      #7001,35     ;RESTORE DATA WORD
2407 007126 104027                    ERROR    27           ;A BRANCH FAILED TO STX*-NO
2408 007130 012737 007001 007066 85:  MOV      #7001,35     ;RESTORE DATA WORD
2409 007136 104030                    ERROR    30           ;A BRANCH FAILED TO CLX*-NO
2410                                     ;TRAP TO 4 OCCURED
2411 007140 022626                    25:  CMP      (SP)+,(SP)+ ;RESTORE STACK POINTER
2412 007142 032777 000020 172736      BIT      #BIT4,@SCPUERR ;DID UNIBUS TIMEOUT?
2413 007150 001403                    BEQ      55           ;BRANCH IF NO
2414 007152 005077 172230              CLR      @SCPUERR     ;CLEAR ERROR REG
2415 007156 104027                    ERROR    27           ;BEN 7 FAILED
2416 007160 017737 172222 001200 55:  MOV      @SCPUERR,$TMP0 ;SAVE ERROR REG
2417 007166 005077 172214              CLR      @SCPUERR     ;CLEAN ERROR REG
2418 007172 104024                    ERROR    24           ;UNEXPECTED TRAP
2419                                     ;RESERVED INSTRUCTION TRAP
2420 007174 022626                    75:  CMP      (SP)+,(SP)+ ;RESTORE STACK POINTER
2421 007176 104026                    ERROR    26           ;DESTINATION CONSTANT FAILED
2422
2423                                     ;;*****
2424                                     ;*TEST 10      STF*-MO*IMM
2425                                     ;*
2426                                     ;*CPU FAILURES
2427                                     ;*      THE CPU SHOULD NOT FAIL
2428                                     ;*
2429                                     ;*FPU FAILURES
2430                                     ;*LDF A BRANCH
2431                                     ;*      IF FXPB FIRA08(0) H IS STUCK HIGH EXECUTION WILL GO TO THE
2432                                     ;*      LDAC6*FD(0) FLOWS.
2433                                     ;*      IF FXPB FD(1) H IS STUCK HIGH EXECUTION WILL GO TO LDD*-NO.
2434                                     ;*      THIS WILL BE TESTED LATER.
2435                                     ;*      IF FXPB NO IS STUCK HIGH EXECUTION WILL GO TO MO*-ST SPEC.
2436                                     ;*LDF IMM BRANCH
2437                                     ;*      IF FXPB IMMEDIATE DOES NOT GET TO FPMF AS A LOW, TWO WORDS
2438                                     ;*      WILL BE LOADED INSTEAD OF ONE. THIS FAILURE WILL BE DETECTED
2439                                     ;*      ON THE STF INSTRUCTION.
2440                                     ;*STF A BRANCH
2441                                     ;*      IF FXPB FIRA11(0) H IS STUCK HIGH EXECUTION WILL GO TO STATE
2442                                     ;*      ZFRO (0).
2443                                     ;*      IF FXPB FIRA10(0) H IS STUCK LOW A STCFI WILL OCCUR.
2444                                     ;*      IF FXPB FIRA08(0) H IS STUCK LOW EXECUTION WILL GO TO
2445                                     ;*      LDAC6*FD(0) FLOWS.
2446                                     ;*STF IMM BRANCH
2447                                     ;*      IF THIS BRANCH FAILS (SEE ABOVE) THE WORD FOLLOWING THE DATA
2448                                     ;*      WORD WILL BE OVER WRITTEN WITH DATA, THAT WILL EXECUTE AS
2449                                     ;*      A NOP.
2450                                     ;*
2451                                     ;*      IF AD2 DOES NOT ACERT THE DATA WORD FOLLOWING THE
2452                                     ;*      INSTRUCTION WILL BE EXECUTED.
2453                                     ;*
2454                                     ;*      FPU ROM FLOW-13,203
2455                                     ;;*****
2456 007200 000004                    TST10:  SCOPE
2457 007202 012737 000010 001240      MOV      #STN-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX
```

2458	007210	012737	007354	001222	MOV	#TST11,SESCAPE	;;ESCAPE TO TEST 11 ON ERROR
2459	007216	012737	000012	000010	MOV	#12,@PRESVEC	;RESTORE RESERVED VECTOR
2460	007224	012737	047504	000004	MOV	#CPSPUR,@SEPRVEC	;RESTORE ERROR VECTOR
2461	007232	012737	007240	001110	MOV	#.+6,@SLPERR	;SET ERROR LOOP
2462	007240	005037	007262		CLR	DATA1	
2463	007244	012702	000001		MOV	#1,R2	;SETUP R2 AND R0 TO
2464	007250	010200			MOV	R2,R0	;CATCH AD2 FAILURE
2465	007252	172427			LDF	(PC)+,ACO	;LOAD DATA
2466	007254	040277			.WORD	40277	;INTEGER 1.X IN FLOATING FORMAT
2467	007256	000240			NOP		;USED IN CASE IMMEDIATE FAILS
2468	007260	174027			15: STF	ACO,(PC)+	;EXECUTE INSTRUCTION UNDER TEST
2469	007262	000000			DATA1: .WORD	0	;DATA WILL BE STORED HERE
2470	007264	000404			35: BP	25	;WILL GET OVERWRITTEN WITH A
2471							;NOP IF IMMEDIATE FAILS.
2472	007266	012737	000404	007264	MOV	#404,35	;RESTORE BRANCH AT 35
2473	007274	104031			ERROR	31	;FXPB IMMEDIATE L DID NOT GO LOW
2474	007276	005700			25: TST	R0	;DID AD2 FAIL AND EXECUTE THE DATA?
2475	007300	001001			BWE	55	;BRANCH IF NO
2476	007302	104032			ERROR	32	;FXPA AD2 DID NOT GO HIGH
2477	007304	022737	040277	007262	55: CMP	#40277,DATA1	;DID DATA COME OUT OK?
2478	007312	001420			BEQ	TST11	;BRANCH IF YES
2479	007314	022737	000001	007262	CMP	#1,DATA1	;DID A STCFI GET EXECUTED?
2480	007327	001001			BWE	45	;BRANCH IF NO
2481	007324	104033			ERROR	33	;FXPB FYR10(0) IS STUCK LOW
2482	007326	012737	040277	001200	45: MOV	#40277,STMP0	;SAVE EXPECTED DATA
2483	007334	005037	001702		CLR	STMP1	;REMEMBER ITS FLOATING FORMAT
2484	007340	013737	007262	001204	MOV	@DATA1,STMP2	;SAVE THE RECEIVED DATA
2485	007346	005037	001206		CLR	STMP3	
2486	007352	104055			ERROR	55	;DATA BAD
2487							
2488							
2489							
2490							
2491							
2492							
2493							
2494							
2495							
2496							
2497							
2498							
2499							
2500							
2501							
2502							
2503							
2504							
2505							
2506							
2507							
2508	007354	000004			TST11: SCOPE		
2509	007356	012737	000011	001240	MOV	#STN-1,STESTN	;;SPT TEST NUMBER IN MAIL BOX
2510	007364	012737	007754	001227	MOV	#TST12,SESCAPE	;;ESCAPE TO TEST 12 ON ERROR
2511	007372	012737	007400	001110	MOV	#.+6,@SLPERR	;SET ERROR LOOP
2512	007400	005037	007420		CLR	DATA2	
2513	007404	172427			LDF	(PC)+,ACO	;PUT KNOWN DATA IN ACO

```

2514 007406 040200      .WORD 40200
2515 007410 172627      LDF (PC)+,AC2      ;PUT KNOWN DATA IN AC2
2516 007412 040277      .WORD 40277
2517 007414 172402      1S:  LDF AC2,ACO      ;EXECUTE INSTRUCTION UNDER TEST
2518 007416 174027      STF ACO,(PC)+      ;GET DATA IN ACO
2519 007420 000000      DATA2: .WORD 0
2520 007422 022737 040277 007420  CMP #40277,00DATA2 ;DID DATA IN ACO CHANGE?
2521 007430 001453      BEQ B1              ;BRANCH IF YES
2522
2523      ;*****
2524      ;CHECK ROM FLOW OF LDF*NO
2524 007432 032777 001000 171476  B0:  BIT #SW9,@SWR      ;SWITCH 9 ON?
2525 007440 001403      RFQ .+10           ;BRANCH IF NO
2526 007442 105737 001103      TSTB SERFLG        ;ANY ERRORS?
2527 007446 001027      BNE 4$             ;BRANCH IF YES
2528 007450 005005      CLR R5             ;USED TO CLEAR MAINTENANCE MODE
2529 007452 012737 007522 000244  MOV B1$,00FPPVEC    ;SET INTERRUPT VECTOR
2530 007460 012700 007746      MOV #ROM2,R0       ;GET ADDRESS OF TABLE
2531 007464 112003      3S:  MOVR (R0)+,R3     ;GET DATA FOR MICRO-BREAK REG
2532 007466 001420      BEQ 5$             ;BRANCH IF DONE
2533 007470 170127 000020      LDFPS #FHM         ;LOAD FPS REGISTER
2534 007474 170003      LDUR              ;LOAD MICRO-BREAK REGISTER
2535 007476 172401      LDF AC1,ACO        ;EXECUTE INSTRUCTION
2536 007500 005001      CLR R1
2537 007502 005201      2S:  INC R1           ;WAIT FOR FPP
2538 007504 001376      BNE 2$             ;TO FINISH IF NO INTERRUPT
2539 007506 170105      LDFPS R5           ;CLEAR MAINTENANCE MODE
2540 007510 114037 001200      MOVB -(R0),00STMP0 ;SAVE ADDRESS THAT FAILED
2541 007514 105037 001201      CLRB 00STMP0+1     ;GET RID OF SIGN EXT
2542 007520 104034      ERROR 34          ;FAILED TO TRAP ON THE ROM STATE
2543 007522 022626      1S:  CMP (SP)+,(SP)+ ;RESTORE STACK POINTER
2544 007524 000757      BR 3$             ;GO TO NEXT ROM STATE
2545 007526 104000      4S:  ERROR 0        ;ONLY EXECUTES WHILE LOOPING ON ERROR
2546 007530 170105      5S:  LDFPS R5       ;CLEAR MAINTENANCE MODE
2547 007532 012737 040277 001200  MOV #40277,STMP0    ;SAVE EXPECTED VALUE
2548 007540 005037 001202      CLR STMP1
2549 007544 013737 007420 001204  MOV DATA2,STMP2    ;SAVE RECEIVED VALUE
2550 007552 005037 001206      CLR STMP3
2551 007556 104035      ERROR 35          ;ROM FLOW OF, DATA BAD
2552
2553      ;*****
2553 007560      B1:
2554 007560 012737 007566 001110  MOV #.+6,00SLPERR  ;SET ERROR LOOP
2555 007566 005037 007606      CLR DATA3
2556 007572 172427      LDF (PC)+,ACO      ;PUT KNOWN DATA IN ACO
2557 007574 040200      .WORD 40200
2558 007576 172527      LDF (PC)+,AC1      ;PUT KNOWN DATA IN AC1
2559 007600 040277      .WORD 40277
2560 007602 174100      1S:  STF AC1,ACO      ;EXECUTE INSTRUCTION UNDER TEST
2561 007604 174027      STF ACO,(PC)+      ;GET ACC BACK
2562 007606 000000      DATA3: .WORD
2563 007610 022737 040277 007606  CMP #40277,DATA3    ;DID DATA IN ACO OK?
2564 007616 001456      BEQ TST12         ;BRANCH IF YES
2565
2566      ;*****
2566      ;CHECK ROM FLOW OF STP*NO
2567 007620 032777 001000 171310  B2:  BIT #SW9,@SWR      ;SWITCH 9 ON?
2568 007626 001403      BEQ .+10          ;BRANCH IF NO
2569 007630 105737 001103      TSTB SERFLG        ;ANY ERRORS?
  
```

2570	007634	001027				BNE	45		;BRANCH IF YES
2571	007636	005005				CLR	R4		;USED TO CLEAR MAINTENANCE MODE
2572	007640	012737	007710	000244		MOV	#15,#0PPPVEC		;SET INTERRUPT VECTOR
2573	007646	012700	007751			MOV	#ROM3,R0		;GET ADDRESS OF TABLE
2574	007652	112003			3S:	MOVR	(R0)+,R7		;GET DATA FOR MICRO-BREAK REG
2575	007654	001420				BEQ	55		;BRANCH IF DONE
2576	007656	170127	000020			LDFPS	#PMH		;LOAD FPS REGISTER
2577	007662	170003				LDUR			;LOAD MICRO-BREAK REGISTER
2578	007664	174100				STP	AC1,AC0		;EXECUTE INSTRUCTION
2579	007666	005001				CLR	R1		
2580	007670	005201			2S:	INC	R1		;WAIT FOR PPP
2581	007672	001376				BNE	25		;TO FINISH IF NO INTERRUPT
2582	007674	170105				LDFPS	R5		;CLEAR MAINTENANCE MODE
2583	007676	114037	001200			MOVR	-(R0),#STMP0		;SAVE ADDRESS THAT FAILED
2584	007702	105037	001201			CLRB	#STMP0+1		;GET RID OF SIGN EXT
2585	007706	104034				ERROR	34		;FAILED TO TRAP ON THE ROM STATE
2586	007710	022626			1S:	CMP	(SP)+,(SP)+		;RESTORE STACK POINTER
2587	007712	000757				BP	35		;GO TO NEXT ROM STATE
2588	007714	104000			4S:	ERROR	0		;ONLY EXECUTES WHILE LOOPING ON ERROR
2589	007716	170105			5S:	LDFPS	R5		;CLEAR MAINTENANCE MODE
2590	007720	012737	040777	001200		MOV	#40777,STMP0		;SAVE EXPECTED VALUE
2591	007726	005037	001202			CLR	STMP1		
2592	007732	013737	007606	001204		MOV	DATA3,STMP2		;SAVE RECEIVED VALUE
2593	007740	005037	001206			CLR	STMP3		
2594	007744	104035				ERROR	35		;ROM FLOW OF, DATA BAD
2595									
2596	007746	042	016	000	ROM2:	.BYTE	42,16,0		
2597	007751	043	262	000	ROM3:	.BYTE	43,262,0		
2598						.EVEN			
2599									
2600									
2601									
2602									
2603									
2604									
2605									
2606									
2607									
2608									
2609									
2610									
2611									
2612									
2613									
2614									
2615									
2616									
2617									
2618									
2619									
2620									
2621									
2622									
2623									
2624									
2625									

2626
2627
2628
2629
2630 007754 000004
2631 007756 012737 000012 001240
2632 007764 012737 010154 001222
2633 007772 005001
2634 007774 012700 001200
2635 010000 012720 040200
2636 010004 012720 177777
2637 010010 005020
2638 010012 005010
2639 010014 012737 010022 001110
2640 010022 012700 001200
2641 010026 172420
2642 010030 022700 001204
2643 010034 001423
2644 010036 022700 001200
2645 010042 001001
2646 010044 104036
2647 010046 022700 001210
2648 010052 001002
2649 010054 005201
2650 010056 000412
2651 010060 022700 001214
2652 010064 001001
2653 010066 104037
2654 010070 010037 001200
2655 010074 012737 001200 001202
2656 010102 104040
2657
2658 010104 012700 001204
2659 010110 174020
2660 010112 022700 001210
2661 010116 001407
2662 010120 022700 001220
2663 010124 001004
2664 010126 005701
2665 010130 001401
2666 010132 104041
2667 010134 104042
2668 010136 005737 001204
2669 010142 001403
2670 010144 005737 001206
2671 010150 001001
2672 010152 104043
2673
2674
2675
2676
2677
2678
2679
2680
2681 010154 000004

```
;*
;* FPU ROM FLOW-LDF:12,132,47,16
;* STF:13,203,202
**.....
TST12: SCOPE
MOV @STN-1,STESTN ;;SET TEST NUMBER IN MAIL BOX
MOV @TST13,SESCAPE ;;ESCAPE TO TEST 13 ON ERROR
CLR R1 ;CLEAR ERROR FLAG
MOV @STMP0,R0 ;PUT ADDRESS OF DATA IN R0
MOV @40200,(R0)+ ;PUT KNOWN DATA IN FIRST WORD
MOV @177777,(R0)+ ;PUT KNOWN DATA IN SECOND WORD
CLR (R0)+ ;CLEAR THE 2 WORDS WHERE
CLR (R0) ;THE DATA WILL BE STORED
MOV @-+6,@SLPERR ;SET ERROR LOOP
MOV @STMP0,R0
1S: LDF (R0)+,ACO ;EXECUTE INSTRUCTION UNDER TEST
CMP @STMP2,R0 ;DID R0 AUTO-INCREMENT PROPERLY?
BEQ 2S ;BRANCH IF YES
CMP @STMP0,R0 ;DID IT INCREMENT AT ALL?
BNE 3S ;BRANCH IF YES
ERROR 36 ;TRCD DSTCON<4 DID NOT GO HIGH
3S: CMP @STMP0+10,R0 ;DID IT INCREMENT BY 10?
BNE 4S ;BRANCH IF NO
INC R1 ;SET ERROR FLAG
BR 2S ;GO TRY STF
4S: CMP @STMP0+14,R0 ;DID IT INCREMENT BY 14?
BNE 5S ;BRANCH IF NO
ERROR 37
5S: MOV R0,STMP0 ;SAVE RECEIVED VALUE
MOV @STMP0,STMP1 ;SAVE EXPECTED VALUE
ERROR 40
;CHECK THE STF
2S: MOV @STMP2,R0
STF ACO,(R0)+ ;EXECUTE INSTRUCTION UNDER TEST
CMP @STMP4,R0 ;DID R0 AUTO-INCREMENT OK?
BEQ 6S ;BRANCH IF YES
CMP @STMP7+2,R0 ;DID LD & ST BOTH EXECUTE AS DOUBLE?
BNE 6S ;BRANCH IF NO
TST R1 ;DID LDF FAIL AUTO-INCREMENT?
BEQ 7S ;BRANCH IF NO
ERROR 41 ;PRMJ PD(1) NOT GETTING TO FXP.
7S: ERROR 42 ;LDF OK BUT STF FAILED AUTO-INC
6S: TST STMP2 ;DID FIRST WORD OF STORE COME OUT?
BEQ 8S ;BRANCH IF NO
TST STMP3 ;DID SECOND WORD COME OUT?
BNE TST13 ;;BRANCH IF YES
8S: ERROR 43 ;EITHER FIRST OR SECOND WORD
;DID NOT STORE
**.....
;*TEST 13 HIGH ORDER DATA PATH
;*
;* THIS TEST FLOATS A ONE AND A ZEPO THROUGH QUADRANTS 2 AND 3
;* OF ACO TO VERIFY THE HIGH ORDER DATA PATH.
**.....
TST13: SCOPE
```

2682	010156	012737	000013	001240	MOV	#STN-1,\$TESTN	;;SET TEST NUMBER IN MAIL BOX	
2683	010164	012737	000200	001200	MOV	#200,\$TMP0	;INITIALIZE DATA	
2684	010172	012737	000001	001202	MOV	#1,\$TMP1	;TO LOAD	
2685	010200	012701	000010		MOV	#10,R1	;SET SOB LOOP	
2686	010204	012700	000010		MOV	#10,R0	;SET SOB LOOP	
2687	010210	012737	010216	001110	MOV	#.+6,#\$SLPERR	;SET ERROR LOOP	
2688	010216	172437	001200		LDF	\$TMP0,AC0	;LOAD DATA PATTERN	
2689	010222	174037	001204		STF	AC0,\$TMP2	;GET IT BACK	
2690	010226	023737	001200	001204	CMP	\$TMP0,\$TMP2	;FIRST WORDS OK?	
2691	010234	001045			BNE	3\$;BRANCH IF NO	
2692	010236	023737	001202	001206	CMP	\$TMP1,\$TMP3	;SECOND WORDS OK?	
2693	010244	001041			BNE	3\$;BRANCH IF NO	
2694	010246	105737	001103		TSTB	\$ERFLG	;ANY ERRORS YET?	
2695	010252	001361			BNE	7\$;BRANCH IF YES	
2696	010254	006337	001200		ASL	\$TMP0	;SHIFT DATA	
2697	010260	006337	001202		ASL	\$TMP1	;PATTERN	
2698	010264	077024			SOB	R0,7\$;CONTINUE	
2699	010266	052737	000200	001200	BIS	#BIT7,\$TMP0	;DON'T LET EXPONENT GO TO ZERO	
2700	010274	077135			SOB	R1,1\$;CONTINUE	
2701	010276	006137	001200		ROL	\$TMP0		
2702	010302	052737	000200	001200	BIS	#BIT7,\$TMP0		
2703	010310	012700	000010		MOV	#10,R0	;SET SOB COUNT	
2704	010314	172437	001200		LDF	\$TMP0,AC0	;LOAD DATA PATTERN	
2705	010320	174037	001204		STF	AC0,\$TMP2	;GET IT BACK	
2706	010324	023737	001200	001204	CMP	\$TMP0,\$TMP2	;IS FIRST WORD OK?	
2707	010332	001006			BNE	3\$;BRANCH IF NO	
2708	010334	023737	001202	001206	CMP	\$TMP1,\$TMP3	;IS SECOND WORD OK?	
2709	010342	001002			BNE	3\$;BRANCH IF NO	
2710	010344	077015			SOB	R0,2\$;CONTINUE	
2711	010346	000401			BR	4\$;GO FLOAT A 0	
2712	010350	104044			BR	44	;BIT FAILED WHILE FLOATING A 1.	
2713					ERROR	44		
2714	010352	012737	177777	001200	4\$:	MOV	#-1,\$TMP0	;INITIALIZE
2715	010360	012737	177776	001202	MOV	#-2,\$TMP1	;DATA TO LOAD	
2716	010366	012700	000040		MOV	#40,R0	;SET SOB COUNT	
2717	010372	012737	010400	001110	MOV	#.+6,#\$SLPERR	;SET ERROR LOOP	
2718	010400	172437	001200		LDF	\$TMP0,AC0	;LOAD DATA	
2719	010404	174037	001204		STF	AC0,\$TMP2	;GET IT BACK	
2720	010410	023737	001200	001204	CMP	\$TMP0,\$TMP2	;IS FIRST WORD OK?	
2721	010416	001016			BNE	6\$;BRANCH IF NO	
2722	010420	023737	001202	001206	CMP	\$TMP1,\$TMP3	;IS SECOND WORD OK?	
2723	010426	001012			BNE	6\$;BRANCH IF NO	
2724	010430	105737	001103		TSTB	\$ERFLG	;ANY ERRORS YET?	
2725	010434	001361			BNE	5\$;BRANCH IF YES	
2726	010436	000261			SEC			
2727	010440	006137	001202		ROL	\$TMP1	;SHIFT THE ZERO	
2728	010444	006137	001200		ROL	\$TMP0	;THRU BOTH WORDS	
2729	010450	077025			SOB	R0,5\$;CONTINUE	
2730	010452	000401			BR	TST14	;GO TO NEXT TEST	
2731	010454	104044			BR	44	;BIT FAILED WHILE FLOATING A 0.	
2732					ERROR	44		

2733 ;*****
 2734 ;*TEST 14 HIGH ORDER SCRATCH PAD DUAL ADDRESSING
 2735 ;*
 2736 ;* THIS TEST ENSURES THAT THE HIGH ORDER SCRATCH PAD ACCUMULATOR ADDRESS LINES
 2737 ;* ARE FUNCTIONAL. THIS IS PERFORMED BY WRITTING THE ADDRESS OF


```

2738      ;*      THE ACCUMULATORS INTO THE ACCUMULATORS AND THEN READING THEM BACK
2739      ;*      AND CHECKING THEM. THE ADDRESS OF THE ACCUMULATOR IS WRITTEN INTO
2740      ;*      EACH CHIP OF THE ACCUMULATOR.
2741      ;*      *****
2742      010456 000004      TST14: SCOPE
2743      010460 012737 000014 001240      MOV      #STM-1, $TESTN      ;;SET TEST NUMBER IN MAIL BOX
2744      010466 172437 001356      LDF      $AC5, AC0      ;;GET AC5 DATA
2745      010472 174005      STP      AC0, AC5      ;;PUT IT IN AC5
2746      010474 172437 001362      LDF      $AC4, AC0      ;;GET AC4 DATA
2747      010500 174004      STP      AC0, AC4      ;;PUT IT IN AC4
2748      010502 172737 001366      LDF      $AC3, AC3      ;;LOAD THE
2749      010506 172637 001372      LDF      $AC2, AC2      ;;OTHER
2750      010512 172537 001376      LDF      $AC1, AC1      ;;FOUR
2751      010516 172437 001402      LDF      $AC0, AC0      ;;REGISTERS
2752      010522 174337 001700      STP      AC3, $TMP0      ;;GET THE DATA
2753      010526 174737 001204      STP      AC2, $TMP2      ;;FROM
2754      010532 174137 001210      STP      AC1, $TMP4      ;;THE FIRST
2755      010536 174037 001214      STP      AC0, $TMP6      ;;FOUR REGISTERS
2756      010542 012700 001200      MOV      #TMP0, R0      ;;GET ADDRESS OF RECEIVED DATA
2757      010546 012701 001366      MOV      #SAC3, R1      ;;GET ADDRESS OF EXPECTED DATA
2758      010552 012702 000010      MOV      #10, R2      ;;SET SOB COUNT
2759      010556 022021      15:      CMP      (R0)+, (R1)+      ;;CHECK THE DATA
2760      010560 001021      BNE      2$      ;;BRANCH IF NOT THE SAME
2761      010562 077203      SOB      R2, 1$      ;;CONTINUE
2762      010564 172405      LDF      AC5, AC0      ;;GET AC5
2763      010566 172504      LDF      AC4, AC1      ;;AND AC4 DATA
2764      010570 174037 001200      STP      AC0, $TMP0      ;;PUT IT IN
2765      010574 174137 001204      STP      AC1, $TMP2      ;;MEMORY
2766      010600 012701 001356      MOV      #SAC5, R1
2767      010604 012700 001200      MOV      #TMP0, R0
2768      010610 012702 000004      MOV      #4, R2      ;;SET SOB COUNT
2769      010614 022021      3$:      CMP      (R0)+, (R1)+      ;;ARE THEY THE SAME
2770      010616 001002      BNE      2$      ;;BRANCH IF NO
2771      010620 077203      SOB      R2, 3$      ;;CONTINUE CHECK
2772      010622 000416      BR      TST15      ;;GO TO NEXT TEST
2773      010624 010002      2$:      MOV      R0, R2      ;;SAVE R0
2774      010626 162700 001200      SOB      #TMP0, P0      ;;
2775      010632 032700 000002      BIT      #BIT1, R0      ;;FIRST WORD OR SECOND WORD?
2776      010636 001401      BEQ      4$      ;;BRANCH IF 2ND WORD
2777      010640 022221      CMP      (R2)+, (R1)+      ;;ADVANCE ADDRESSES BY ONE WORD
2778      010642 012700 001210      4$:      MOV      #TMP4, R0
2779      010646 014240      MOV      -(R2), -(R0)      ;;SAVE RECEIVED
2780      010650 014240      MOV      -(R2), -(R0)      ;;DATA
2781      010652 014140      MOV      -(R1), -(R0)      ;;SAVE EXPECTED
2782      010654 014140      MOV      -(R1), -(R0)      ;;DATA
2783      010656 104045      ERROR 45      ;;DUAL ADDRESSING PROBLEM
2784
2785      ;*      *****
2786      ;*TEST 15      LDFPS/STFPS*-NO
2787
2788      ;*      THE A BRANCH SHOULD NOT FAIL. IF THE INSTRUCTIONS DON'T WORK,
2789      ;*      THE THE CONTROL STORE IS PROBABLY BAD OR THE ADX ROM.
2790
2791      ;*      FPU ROM FLOW-LDFPS:22
2792      ;*      STFPS:62,23
2793      ;*      *****
  
```

2794 010660 000004
2795 010662 012737 000015 001240
2796 010670 012737 147757 001202
2797 010676 170137 001202
2798 010702 170200
2799 010704 022700 147757
2800 010710 001403
2801 010712 010037 001200
2802 010716 104046
2803 010720
2804 010720 012737 010726 001110
2805 010726 005037 001200
2806 010732 170237 001200
2807 010736 022737 147757 001200
2808 010744 001401
2809 010746 104047
2810
2811 010750
2812 010750 012737 010756 001110
2813 010756 170127
2814 010760 104066
2815 010762 170227
2816 010764 104066
2817 010766 170127 000000
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832 010772 000004
2833 010774 012737 000016 001240
2834 011002 005001
2835 011004 012737 040201 001200
2836 011012 012737 177777 001202
2837 011020 013737 001200 001204
2838 011026 013737 001202 001206
2839 011034 170127 000017
2840 011040 172437 001200
2841 011044 170237 001212
2842 011050 005737 001717
2843 011054 001401
2844 011056 104050
2845 011060
2846 011060 012737 011334 001227
2847 011066 012737 011074 001110
2848 011074 170127 000013
2849 011100 170400

```
TST15: SCOPE
MOV #STN-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX
MOV #147757,$TMP1 ;PUT DATA TO LOAD IN MEMORY
1S: LDFPS $TMP1 ;EXECUTE INSTRUCTION UNDER TEST
STFPS R0 ;GET FPS BACK
CMP #147757,R0 ;DID THE FPS GET LOADED?
BEQ C0 ;BRANCH IF YES
MOV R0,$TMP0 ;SAVE RECEIVED VALUE
ERROR 46 ;LDFPS*-NO FAILED

C0:
MOV #.+6,#$SLPERR ;SET ERROR LOOP
CLR $TMP0
1S: STFPS $TMP0 ;EXECUTE INSTRUCTION UNDER TEST
CMP #147757,$TMP0 ;DID STFPS WORK?
BEQ 2S ;BRANCH IF YES
ERROR 47 ;STFPS*-NO FAILED

;CHECK IMMEDIATE MODE
2S:
MOV #.+6,#$SLPERR ;SET ERROR LOOP
LDFPS (PC)+ ;EXECUTE INSTRUCTION
ERROR 66 ;ADX ROM FAILED
STFPS (PC)+
ERROR 66
LDFPS #0 ;CLEAR THE STATUS

;;*****
;*TEST 16 CLRF*M0
;*
;* IF FIRA 08(0)H DOES NOT GET TO THE A BRANCH ROM MUX AS A LOW,
;* EXECUTION WILL GO TO ROM STATE 30 AND THEN TO ROM STATE
;* 0 ON THE NO-MEM BRANCH. ALSO, IF FIP08(0)H DOES NOT FORCE
;* FXPR IR(11:6)0 L TO GO HIGH, EXECUTION WILL GO TO
;* ROM STATE 3 ON THE NO-MEM BRANCH.
;* BOTH THESE FAILURES WOULD CAUSE THE DESTINATION
;* NOT TO BE CLEARED.
;*
;* FPU ROM FLOW-30,55
;;*****
TST16: SCOPE
MOV #STN-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX
CLR R1 ;CLEAR ERROR FLAG
MOV #40201,$TMP0 ;GET DATA TO MAKE ACO
MOV #-1,$TMP1 ;NON ZERO
MOV $TMP0,$TMP2 ;ENSURE THAT PLACE WHERE
MOV $TMP1,$TMP3 ;DATA IS TO BE STORED IS NON-ZERO
LDFPS #17 ;LOAD COMPLIMENT CC'S
LDF $TMP0,AC0 ;LOAD DATA INTO ACO
STFPS $TMP5 ;GET CONDITION CODES
TST $TMP5 ;CONDITION CODES OK?
BEQ 2S ;BRANCH IF YES
ERROR 50 ;COND CODES BAD ON LOAD FLOAT

2S:
MOV #TST17,$ESCAPE ;;ESCAPE TO TEST 17 ON ERROR
MOV #.+6,#$SLPERR ;SET ERROR LOOP
LDFPS #13 ;PUT COMPLIMENT CC'S IN FPS
1S: CLRF ACO ;EXECUTE INSTRUCTION UNDER TEST
```

```

2850 011102 170237 001212      STFPS  STMP5      ;GET CONDITION CODES BACK
2851 011106 170127 000013      LDFPS  #13        ;LOAD COMPLIMENT CC'S
2852 011112 174037 001204      STF    ACO,STMP2 ;GET DATA BACK
2853 011116 170237 001214      STFPS  STMP6      ;GET CONDITION CODES
2854 011122 005737 001204      TST    STMP2      ;DID QUADRANT 3 CLEAR?
2855 011126 001406              BEQ    3S         ;BRANCH IF YES
2856 011130 013700 001204      MOV    STMP2,R0   ;GET QUAD 3 DATA
2857 011134 042700 000177      BIC    #177,R0    ;CLEAR FRACTION BITS.
2858 011140 001401              BEQ    3S         ;BRANCH IF EXPONET AND SIGN ZERO
2859 011142 005701              INC    R1         ;SET EXPONET ERROR FLAG
2860 011144 005737 001206      3S:   TST    STMP3      ;DID QUAD 2 CLEAR?
2861 011150 001404              BEQ    5S         ;BRANCH IF YES
2862 011152 005701              TST    R1         ;EXPCNET ERROR?
2863 011154 001401              BEQ    6S         ;BRANCH IF NO
2864 011156 000424              BR     DO         ;NEITHER EXP NOR FRACTION CLEARED
2865                                ;GO CHECK ROM FLOW
2866 011160 104051              6S:   ERROR  51      ;FRACTION DID NOT CLEAR, EXP DID
2867 011162 005701              5S:   TST    R1         ;EXPCNET ERROR?
2868 011164 001401              BEQ    7S         ;BRANCH IF NO
2869 011166 104052              ERROR  57      ;FRAC CLEARED, EXPONET DIDN'T
2870                                ;*****
2871                                ;DATA IS OK-NOW CHECK CC'S
2872 011170 022737 000004 001212 7S:   CMP    #4,STMP5    ;CC'S OK ON CLRFR?
2873 011176 001404              BEQ    8S         ;BRANCH IF YES
2874 011200 012737 000004 001210      MOV    #4,STMP4   ;SAVE EXPECTED VALUE
2875 011206 104053              ERROR  53        ;PAD CC'S ON CLRFR CHECK FXPL ACWX=0
2876 011210 022737 000013 001214 8S:   CMP    #13,STMP6 ;CC'S OK ON STF?
2877 011216 001446              BEQ    TST17     ;BRANCH IF YES
2878 011220 013737 001214 001212      MOV    STMP6,STMP5
2879 011226 104054              ERROR  54        ;BAD CC'S ON STF
2880                                ;CHECK CLRFR ROM FLOW
2881 011230 032777 001000 167700 00:   BIT    #SW9,#SWR  ;SWITCH 9 ON?
2882 011236 001403              BEQ    .+10      ;BRANCH IF NO
2883 011240 105737 001103              TSTB  SERFLC     ;ANY ERRORS?
2884 011244 001027              BNE    4S         ;BRANCH IF YES
2885 011246 005005              CLR    R5         ;USPD TO CLEAR MAINTENANCE MODE
2886 011250 012737 011320 000244      MOV    #15,@#FPPVEC ;SET INTERRUPT VECTOR
2887 011256 012700 011332              MOV    #ROMCLR,R0 ;GET ADDRESS OF TABLE
2888 011262 112003              3S:   MOVB  (R0)+,R3 ;GET DATA FOR MICRO-BREAK REG
2889 011264 001420              BEQ    5S         ;BRANCH IF DONE
2890 011266 170127 000020      LDFPS  #FMM      ;LOAD FPS REGISTER
2891 011272 170003      LDUP                    ;LOAD MICRO-BREAK REGISTER
2892 011274 170400      CLRFR  ACO        ;EXECUTE INSTRUCTION
2893 011276 005001      CLR    R1
2894 011300 005201              2S:   INC    R1         ;WAIT FOR FPP
2895 011302 001376              BNE    2S        ;TO FINISH IF NO INTERRUPT
2896 011304 170105      LDFPS  R5         ;CLEAR MAINTENANCE MODE
2897 011306 114037 001200      MOVB  -(R0),@#STMP0 ;SAVE ADDRESS THAT FAILED
2898 011312 105037 001201      CLRFR @#STMP0+1   ;GET RID OF SIGN EXT
2899 011316 104034              ERROR  34        ;FAILED TO TRAP ON THE ROM STATE
2900 011320 022626              1S:   CMP    (SP)+,(SP)+ ;RESTORE STACK POINTER
2901 011322 000757              BR     3S        ;GO TO NEXT ROM STATE
2902 011324 104000              4S:   ERROR  0         ;ONLY EXECUTES WHILE LOOPING ON ERROR
2903 011326 170105              5S:   LDFPS  R5         ;CLEAR MAINTENANCE MODE
2904 011330 104055              ERROR  55        ;ROM FLOW OK, EXP & FRAC DID NOT CLEAR
2905 011332 055 000      ROMCLR: .BYTE 55,0

```

```

2906          .EVEN
2907
2908          ;;*****
2909          ;*TEST 17      TSTF*MO
2910          ;*
2911          ;*      THE ONLY WAY THE A AND NO MEM BRANCHES SHOULD FAIL IS IF
2912          ;*      THE ROM(S) FAIL.
2913          ;*      A ONE (1) IS FLOATED THRU THE EXPONENT TO VERIFY THAT THE
2914          ;*      SIGNAL FXPL ACMX EQ 0 IS UNACERTED FOR ALL 8 INPUTS.
2915          ;*
2916          ;*      FPU ROM FLOW-30,56
2917          ;;*****
2918          011334 000004          TST17: SCOPE
2919          011336 012737 000017 001240          MOV      #STN-1,STESTN      ;;SET TEST NUMBER IN MAIL BOX
2920          011344 012737 011504 001222          MOV      #TST20,SESCAPE    ;;ESCAPE TO TEST 20 ON ERROR
2921          011357 012700 000010          MOV      #10,R0            ;SET SOB COUNT
2922          011356 012737 000200 001204          MOV      #BIT7,$TMP2      ;INITIALIZE DATA TO LOAD
2923          011364 012737 011372 001110          MOV      #.+6,#$LPERK     ;SET ERROR LOOP
2924          011372 172437 001204          2S:    LDF      $TMP2,ACO      ;LOAD DATA
2925          011376 170127 000017          LDFPS   #17                ;LOAD COMPLIMENT CC'S
2926          011402 170500          1S:    TSTF     ACO            ;EXECUTE INSTRUCTION UNDER TEST
2927          011404 170237 001200          STFPS   $TMP0              ;GET CC'S BACK
2928          011410 005737 001200          TST     $TMP0              ;CC'S OK?
2929          011414 001403          BEQ     3S                  ;BRANCH IF YES
2930          011416 005037 001202          CLR     $TMP1              ;SAVE EXPECTED VALUE
2931          011422 104056          ERROR   56                  ;CC'S BAD
2932          011424 105737 001103          3S:    TSTF     $ERFLG       ;ANY ERRORS YET?
2933          011430 001360          BNE     2S                  ;BRANCH IF YES
2934          011432 006337 001204          ASL     $TMP2              ;SHIFT DATA PATTERN
2935          011436 077023          SOB     R0,2S              ;CONTINUE
2936          011440 012737 011446 001110          MOV      #.+6,#$LPERK     ;SET ERROR LOOP
2937          011446 172427 140000          LDF     #-0140000,ACO      ;LOAD A NEGATIVE NUMBER
2938          011452 170127 000007          LDFPS   #7                  ;SET CC'S TO COMPLIMENT OF EXPECTED VALUE
2939          011456 170500          TSTF     ACO                ;EXECUTE INSTRUCTION UNDER TEST
2940          011460 170237 001200          STFPS   $TMP0              ;GET CC'S
2941          011464 022737 000010 001200          CMP     #FN,$TMP0          ;CC'S OK?
2942          011472 001404          BEQ     TST20              ;;BRANCH IF YES
2943          011474 012737 000010 001202          MOV      #FN,$TMP1        ;SAVE EXPECTED VALUE
2944          011502 104057          ERROR   57                  ;CC'S BAD ON NEGATIVE NUMBER
2945
2946          ;;*****
2947          ;*TEST 20      CLRFB*-MO*IMM
2948          ;*
2949          ;*      THE ONLY WAY THIS SHOULD FAIL, IS IF THE CONTROL STORE FAILS.
2950          ;*
2951          ;*      FPU ROM FLOW-25,211
2952          ;;*****
2953          011504 000004          TST20: SCOPE
2954          011506 012737 000020 001240          MOV      #STN-1,STESTN      ;;SET TEST NUMBER IN MAIL BOX
2955          011514 012737 011636 001222          MOV      #TST21,SESCAPE    ;;ESCAPE TO TEST 21 ON ERROR
2956          011522 012737 140203 011536          MOV      #140203,2S        ;MAKE SURE DATA WORD IS NON-ZERO
2957          011530 170127 000013          LDFPS   #13                ;LOAD COMPLIMENT OF EXPECTED CC'S
2958          011534 170427          1S:    CLRFB  (PC)+          ;EXECUTE INSTRUCTION UNDER TEST
2959          011536 140203          2S:    .WOPD  140203        ;SHOULD CLEAR THIS LOCATION
2960          ;*      WILL HALT HERE IF UPDATE CONSTANTS FAIL
2961          011540 170237 001212          STFPS   $TMP5              ;GET CC'S
  
```

```

2962 011544 022737 000004 001217      CMP      #FZ,$TMP5      ;ARE THEY OK?
2963 011552 001404                      BFQ      3S            ;BRANCH IF YES
2964 011554 012737 000004 001210      MOV      #FZ,$TMP4      ;SAVE EXPECTED VALUE
2965 011562 104053                      ERROR    53            ;CC'S BAD (ROM STATE 25)
2966 011564 005737 011536              3S:     TST      2S            ;DID THE INSTRUCTION WORK?
2967 011570 001422                      BEQ      TST21         ;;BRANCH IF YES
2968 011572 012737 011536 001200      MOV      #2S,$TMP0      ;SAVE RECEIVED DATA
2969 011600 005037 001202                      CLR      $TMP1         ;SAVE EXPECTED DATA
2970 011604 013700 011536                      MOV      2S,R0         ;SAVE 2S
2971 011610 010001                      MOV      R0,R1
2972 011612 010102                      MOV      R1,R2
2973 011614 042700 077777      BIC      #77777,R0      ;DID SIGN BIT FAIL?
2974 011620 001401                      BEQ      4S            ;BRANCH IF NO
2975 011622 104063                      ERROR    63            ;SIGN BIT DID NOT CLEAR
2976 011624 042701 100177      4S:     BIC      #100177,R1   ;DID EXPONENT FAIL?
2977 011630 001401                      BEQ      5S            ;BRANCH IF NO
2978 011632 104064                      ERROR    64            ;EXPONENT DID NOT CLEAR
2979 011634 104065      5S:     ERROR    65            ;FRACTION DID NOT CLEAR

```

```

2980
2981
2982      ;;*****
2983      ;*TEST 21      CLRF*-MO*-IMM
2984      ;*
2985      ;*      THE ONLY WAY THIS TEST SHOULD FAIL, IS IF THE CONTROL STORE,
2986      ;*      (ROM STATE 216) FAILS.
2987      ;*
2988      ;*      FPU ROM FLOW-25,211,216
2989      ;;*****

```

```

2989 011636 000004      TST21:  SCOPE
2990 011640 012737 000021 001240      MOV      #STN-1,$TESTN  ;;SET TEST NUMBER IN MAIL BOX
2991 011646 012737 011736 001227      MOV      #TST22,$ESCAPE ;;ESCAPE TO TEST 22 ON ERROR
2992 011654 012737 177777 001204      MOV      #-1,$TMP2      ;WAKE DESTINATION
2993 011662 012737 177777 001206      MOV      #-1,$TMP3      ;NON-ZERO
2994 011670 005037 001200                      CLR      $TMP0         ;SAVE EXPECTED
2995 011674 005037 001202                      CLR      $TMP1         ;VALUES
2996 011700 012700 001204                      MOV      #STMP2,R0     ;PUT ADDRESS OF DESTINATION IN R0
2997 011704 170420      1S:     CLRF      (R0)+        ;EXECUTE INSTRUCTION UNDER TEST
2998 011706 022700 001210      CMP      #STMP4,R0      ;DID ADX ROM WORK OK?
2999 011712 001401                      BEQ      2S            ;BRANCH IF YES
3000 011714 104066                      ERROR    66            ;ADX ROM FAILED
3001 011716 005737 001204      2S:     TST      $TMP2      ;DID QUAD 3 CLEAR?
3002 011722 001401                      BEQ      3S            ;BRANCH IF YES
3003 011724 104052                      ERROR    52            ;EXPONENT DID NOT CLEAR
3004 011726 005737 001206      3S:     TST      $TMP3      ;DID QUAD 2 CLEAR?
3005 011732 001401                      BEQ      TST22         ;;BRANCH IF YES
3006 011734 104051                      ERROR    51            ;FRACTION DID NOT CLEAR

```

```

3007
3008      ;;*****
3009      ;*TEST 22      ABSF*MO*-(EXP=0)
3010      ;*
3011      ;*      THE A BRANCH SHOULD NOT FAIL.
3012      ;*      THE NO-MEM BRANCH SHOULD NOT FAIL.
3013      ;*ABSX BRANCH
3014      ;*      IF FIRB06(1)H DOES NOT GO LOW OR DOES NOT GET TO FRPA RADR03 AS
3015      ;*      A HIGH, EXECUTION WILL GO TO PCM STATE 134. THIS WILL CAUSE THE
3016      ;*      SIGN BIT TO BE COMPLIMENTED.
3017      ;*BZ BRANCH

```

```

3018 ;* IF PRMB BZ(0)H DOES NOT GO HIGH OR DOES NOT GET TO PRMA RADOO
3019 ;* AS A LOW, EXECUTION WILL GO TO STATE 106. THIS WILL CAUSE THE
3020 ;* EXPONENT AND FRACTION TO BE CLEARED.
3021 ;*
3022 ;* FPU ROM FLOW-30,57,124
3023 ;*****
3024 011736 000004 TST22: SCOPE
3025 011740 012737 000022 001240 MOV #STW-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX
3026 011746 012737 012212 001222 MOV #TST23,$ESCAPE ;;ESCAPE TO TEST 23 ON ERROR
3027 011754 172427 LDF (PC)+,$ACO ;LOAD DATA TO TEST
3028 011756 040200 2S: .WORD 40200
3029 011760 170127 LDFPS (PC)+ ;LOAD COMPLIMENT OF EXPECTED CC'S
3030 011762 000017 3S: .WORD 17
3031 011764 170600 1S: ARSF ACO ;EXECUTE INSTRUCTION UNDER TEST
3032 011766 170237 001200 STFPS $TMP0 ;GET CC'S
3033 011772 174037 001162 STF ACO,$REG1 ;GET DATA BACK
3034 011776 022737 040200 001162 CMP #40200,$REG1 ;IS DATA OK?
3035 012004 001407 BEQ 4S ;BRANCH IF YES
3036 012006 005737 001162 TST $REG1 ;DID SIGN COMPLIMENT OR EXPONENT CLEAR?
3037 012012 100402 BMI 5S ;BRANCH IF SIGN COMPLIMENTED
3038 012014 001402 BEQ 6S ;BRANCH IF EXPONENT CLEARED.
3039 012016 000410 BR 60 ;GO CHECK ROM FLOW
3040 012020 104067 5S: ERROR 67 ;ABSX BRANCH FAILED
3041 012022 104070 6S: EPROR 70 ;BZ BRANCH FAILED
3042 ;CHECK CC'S
3043 012024 005737 001200 4S: TST $TMP0 ;ARE CC'S OK?
3044 012030 001447 BEQ E1 ;BRANCH IF YES
3045 012032 005037 001202 CLR $TMP1 ;SAVE EXPECTED VALUE
3046 012036 104071 ERROR 71 ;CC'S BAD
3047 ;*****
3048 ;CHECK ROM FLOW OF ARSF
3049 012040 032777 001000 167070 20: BIT #SW9,$SWK ;SWITCH 9 ON?
3050 012046 001403 BEQ .+10 ;BRANCH IF NO
3051 012050 105737 001103 TSTP $ERFLG ;ANY ERRORS?
3052 012054 001027 BNE 4S ;BRANCH IF YES
3053 012056 005005 CLR R5 ;USED TO CLEAR MAINTENANCE MODE
3054 012060 012737 012130 000744 MOV #1S,@#FPPVEC ;SET INTERRUPT VECTOR
3055 012066 012700 012206 MOV #POMARS,$RO ;GET ADDRESS OF TABLE
3056 012072 112003 3S: MOV# (RO)+,$R3 ;GET DATA FOR MICRO-BREAK REG
3057 012074 001420 BEQ 5S ;BRANCH IF DONE
3058 012076 170127 000020 LDFPS #FPM ;LOAD FPS REGISTER
3059 012102 170003 LDUP ;LOAD MICRO-BREAK REGISTER
3060 012104 170600 ARSF ACO ;EXECUTE INSTRUCTION
3061 012106 005001 CLR R1
3062 012110 005201 2S: INC R1 ;WAIT FOR FPP
3063 012112 001376 BNE 2S ;TO FINISH IF NO INTERRUPT
3064 012114 170105 LDFPS R5 ;CLEAR MAINTENANCE MODE
3065 012116 114037 001200 MOV# -(RO),@#$TMP0 ;SAVE ADDRESS THAT FAILED
3066 012122 105037 001201 CLR# @#$TMP0+1 ;GET RID OF SIGN EXT
3067 012126 104034 ERROR 34 ;FAILED TO TRAP ON THE ROM STATE
3068 012130 022626 1S: CMP (SP)+,(SP)+ ;RESTORE STACK POINTER
3069 012132 000757 BR 3S ;GO TO NEXT ROM STATE
3070 012134 104000 4S: ERROR 0 ;ONLY EXECUTES WHILE LOOPING ON ERROR
3071 012136 170105 5S: LDFPS R5 ;CLEAR MAINTENANCE MODE
3072 012140 012737 040200 001164 MOV #40200,$REG2 ;SAVE EXPECTED DATA
3073 012146 104077 ERROR 77 ;FLOW'S OK, BUT DATA CHANGED

```

```

3074      ;*****
3075      ;CHECK IT WITH A NEGATIVE NUMBER
3076 012150      E1:
3077 012150 012737 012156 001110      MOV      #.+6,#SLPERR      ;SET ERROR LOOP
3078 012156 172427 140200      LDF      #0140200,ACO      ;LOAD A NEGATIVE NUMBER
3079 012162 170600      1S:      ARSF      ACO      ;EXECUTE INSTRUCTION UNDER TEST
3080 012164 174037 001162      STF      ACO,$REG1      ;GET DATA BACK
3081 012170 005737 001162      TST      $REG1      ;DID SIGN GET CHANGED?
3082 012174 100006      BPL      TST23      ;BRANCH IF YES
3083 012176 012737 040200 001164      MOV      #40200,$REG2      ;SAVE EXPECTED VALUE
3084 012204 104063      ERROR    63      ;ABSF DID NOT CLEAR SIGN BIT
3085 012206      057      124      000      ROMABS: .BYTE 57,124,0
3086      012217      .EVEN
3087
3088      ;*****
3089      ;*TEST 23      NEGF*MO*-(EXP=0)
3090      ;*
3091      ;*      THE ONLY POSSIBLE FAILURE SHOULD BE THE ABSX BRANCH. IF FIRB06(1)
3092      ;*      DOES NOT GO HIGH OR DOES NOT GET TO FRMA RAO3 AS A LOW, EXECUTION
3093      ;*      WILL GO TO STATE 124. THIS WILL CAUSE AN ARSX TO BE EXECUTED.
3094      ;*
3095      ;*      FPU ROM FLOW-30,57,134
3096      ;*****
3097 012212 000004      TST23: SCOPE
3098 012214 012737 000023 001240      MOV      #STN-1,$TESTN      ;SET TEST NUMBER IN MAIL BOX
3099 012222 012737 012356 001222      MOV      #TST24,$ESCAPE      ;ESCAPE TO TEST 24 ON ERROR
3100 012230 172427      LDF      (PC)+,ACO      ;LOAD ACO WITH POSITIVE NUMBER
3101 012232 040277      2S:      .WORD    40277
3102 012234 170127 000007      LDFPS   #7      ;LOAD CC'S TO COMPLIMENT OF EXPECTED
3103 012240 170700      1S:      NEGF      ACO      ;EXECUTE INSTRUCTION UNDER TEST
3104 012242 170237 001200      STFPS   $TMP0      ;GET CC'S BACK
3105 012246 174037 001162      STF      ACO,$REG1      ;GET DATA BACK
3106 012252 022737 140277 001162      CMP      #140277,$REG1      ;DID DATA COME BACK OK?
3107 012260 001406      BEQ      3S      ;BRANCH IF YES
3108 012262 100401      BMI      4S      ;BRANCH IF SIGN BIT WORKED,
3109      ;BUT FRACTION OR EXPONENT FAILED
3110 012264 104072      ERROR    72      ;ABSX BRANCH FAILED
3111 012266 012737 140277 001164      4S:      MOV      #140277,$REG2      ;SAVE EXPECTED VALUE
3112 012274 104073      ERROR    73      ;SIGN OK, BUT DATA CHANGED
3113      ;CHECK CONDITION CODES
3114 012276 022737 000010 001200      3S:      CMP      #10,$TMP0      ;CC'S OK?
3115 012304 001404      BEQ      5S      ;BRANCH IF YES
3116 012306 012737 000010 001202      MOV      #10,$TMP1      ;SAVE EXPECTED VALUE
3117 012314 104071      ERROR    71      ;CC'S BAD
3118      ;CHECK NEG OF A NEGATIVE NUMBER
3119 012316      5S:
3120 012316 012737 012324 001110      MOV      #.+6,#SLPERR      ;SET ERROR LOOP
3121 012324 172427 140277      LDF      #0140277,ACO      ;LOAD A NEGATIVE NUMBER
3122 012330 170700      NEGF      ACO      ;EXECUTE INSTRUCTION UNDER TEST
3123 012332 174037 001162      STF      ACO,$REG1      ;GET DATA BACK
3124 012336 022737 040277 001162      CMP      #40277,$REG1      ;IS DATA OK?
3125 012344 001404      BEQ      TST24      ;BRANCH IF YES
3126 012346 012737 040277 001164      MOV      #40277,$REG2      ;SAVE EXPECTED VALUE
3127 012354 104073      ERROR    73      ;DATA BAD
3128
3129      ;*****

```

3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148 012356 000004
3149 012360 012737 000024 001240
3150 012366 012737 012524 001222
3151 012374 012703 000240
3152 012400 170003
3153 012402 172427 040277
3154 012406 172527 000177
3155 012412 170127 000037
3156 012416 012737 012514 000244
3157 012424 172001
3158 012426 170237 001200
3159 012432 042737 000020 001200
3160 012440 170127 000000
3161 012444 174037 001162
3162 012450 022737 040277 001162
3163 012456 001410
3164 012460 005737 001162
3165 012464 001001
3166 012466 104074
3167 012470 012737 040277 001164
3168 012476 104073
3169
3170 012500 005737 001200
3171 012504 001407
3172 012506 005037 001202
3173 012512 104071
3174
3175 012514 022626
3176 012516 170127 000000
3177 012522 104075
3178
3179
3180
3181
3182
3183
3184
3185

```
;*TEST 24      ADDP*MO*(SRC=0)*-(DST=0)
;*
;*           NEITHER THE A NOR THE NO-MEM BRANCHES SHOULD FAIL.
;*6F1 BRNACH
;*           IF FRMJ EXPB0(1)H DOES NOT GO HIGH OR DOES NOT GET TO FRMA
;*           RADO2 AS A LOW EXECUTION WILL GO TO STATE 240. SINCE THIS
;*           FAILURE WOULD NOT BE DETECTED BY THE ANSWFR, THE MICRO-BREAK
;*           REGISTER IS SETUP TO TRAP ON STATE 240.
;*           IF FRMJ EXPA0(1)H DOES NOT GO LOW OR DOES NOT GET TO FRMA RADO3
;*           AS A HIGH, EXECUTION WILL GO TO STATE 254. THIS WOULD CAUSE THE
;*           DESTINATION TO BE CLEARED.
;*
;*           NOTE: SYNC POINT NOT SELECTIBLE--DEFAULT=240
;*
;*           NOTE: SYNC POINT NOT SELECTABLE. DEFAULT=240
;*
;*           FPU ROM FLOW-30,65,244
;*****
TST24: SCOPE
NOV      #STW-1,STESTN    ;;SET TEST NUMBER IN MAIL BOX
NOV      #TST25,SESCAPE  ;;ESCAPE TO TEST 25 ON ERROR
NOV      #240,R3         ;
LDUR     ;LOAD THE MICRO-BREAK REGISTER
LDF      #040277,ACO     ;MAKE THE DESTINATION NON-ZERO
LDF      #0177,AC1      ;MAKE THE SOURCE=ZERO
LDFPS    #FNM+17        ;LOAD COMPLIMENT OF EXPECTED CC'S
NOV      #25,FPPVEC     ;LOAD INTERRUPT VECTOR
15:      ADDP            AC1,ACO ;EXECUTE INSTRUCTION UNDER TEST
STFPS    $TMP0          ;GET FPS BACK
BIC      #FNM,$TMP0     ;GET RID OF FNM BIT
LDFPS    #0             ;CLEAR FNM BIT IN FPS
STF      ACO,$REG1     ;GET DESTINATION BACK
CMP      #40277,$REG1  ;IS IT OK?
BFQ      3$             ;BRANCH IF YES
TST      $REG1         ;DID IT CLEAR?
BNE      4$             ;BRANCH IF IT DIDN'T CLEAR
ERROR    74            ;6F1 BRANCH FAILED TO STATE 254
45:      MOV      #40277,$RFG2 ;SAVE EXPECTED VALUE
ERROR    73            ;DATA CHANGED
;CHECK CONDITION CODES
35:      TST      $TMP0   ;CC'S OK?
BEQ      TST25         ;;BRANCH IF YES
CLR      $TMP1         ;SAVE EXPECTED VALUE
ERROR    71            ;CC'S BAD
;MICRO-BREAK TRAP OCCURED
25:      CMP      (SP)+,(SP)+ ;RESTORE THE SP
LDFPS    #0           ;CLEAR MM MODE.
ERROR    75           ;6F1 BRANCH FAILED TO STATE 240.
;*****
;*TEST 25      ADDP*MO*(SRC=0)*-(DST=0)
;*
;*6F1 BRANCH
;*           IF FRMJ EXPA0(1)H DOES NOT GO HIGH OR DOES NOT GET TO FRMA RADO3,
;*           EXECUTION WILL GO TO STATE 244. THIS WILL CAUSE THE DESTINATION PRACTICM
;*           TO REMAIN UNCHANGED.
```



```

3186
3187
3188
3189 012524 000004
3190 012526 012737 000025 001240
3191 012534 012737 047624 000244
3192 012542 012737 012626 001222
3193 012550 172427 000177
3194 012554 172527 000000
3195 012560 170127 000013
3196 012564 172001
3197 012566 170237 001200
3198 012572 174037 001162
3199 012576 005737 001162
3200 012602 001401
3201 012604 104076
3202 012606 022737 000004 001200
3203 012614 001404
3204 012616 012737 000004 001202
3205 012624 104071
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215 012626 000004
3216 012630 012737 000026 001240
3217 012636 012737 013002 001222
3218 012644 172427 000177
3219 012650 172527 000000
3220 012654 173001
3221 012656 174037 001162
3222 012662 005737 001162
3223 012666 001445
3224
3225
3226 012670 032777 001000 166240
3227 012676 001403
3228 012700 105737 001103
3229 012704 001027
3230 012706 005005
3231 012710 012737 012760 000244
3232 012716 012700 012776
3233 012722 112003
3234 012724 001420
3235 012726 170127 000020
3236 012732 170003
3237 012734 173001
3238 012736 005001
3239 012740 005201
3240 012742 001376
3241 012744 170105

; *
; * FPU ROM FLOW-30,65,254
; *****
TST75: SCOPE
MOV #STN-1,STESTN ;;SET TEST NUMBER IN MAIL BOX
MOV #PPSPUR,FPPVEC ;;RESTORE FP VECTOR
MOV #TST26,SESCAPE ;;ESCAPE TO TEST 26 ON ERROR
LDF #0177,ACO ;;MAKE DST EXP=0 BUT FRAC NON-ZERO
LDF #0,AC1 ;;MAKE SRC=0
LDFPS #13 ;;LOAD COMPLIMENT COND. CODES
1S: ADDP AC1,ACO ;;EXECUTE INSTRUCTION UNDER TEST
STFPS $TMP0 ;;SAVE COND CODES
STP ACO,$REG1 ;;GET DATA BACK
TST $REG1 ;;DID DEST FRAC GET CLEARED?
BEQ 2S ;;BRANCH IF YES
EPROR 76 ;;6FI BRANCH FAILED TO STATE 244
2S: CMP #4,$TMP0 ;;CC'S OK?
BEQ TST26 ;;BRANCH IF YES
MOV #4,$TMP1 ;;SAVE EXPECTED VALUE
ERROR 71 ;;CC'S BAD

; *****
; *TEST 26 SUBP*MO*(SRC=0)*(DST=0)
; *
; * THE ONLY POSSIBLE FAILURE WOULD BE A BAD CELL IN EITHER THE A BRANCH
; * ROM OR THE NO MEM ROM.
; *
; * FPU ROM FLOW-30,65,254
; *****
TST26: SCOPE
MOV #STN-1,STESTN ;;SET TEST NUMBER IN MAIL BOX
MOV #TST27,SESCAPE ;;ESCAPE TO TEST 27 ON ERROR
LDF #0177,ACO ;;MAKE DST EXP=0, FRAC NON-ZERO
LDF #0,AC1 ;;MAKE SRC=0
1S: SUBP AC1,ACO ;;EXECUTE INSTRUCTION UNDER TEST
STP ACO,$REG1 ;;GET DATA BACK
TST $REG1 ;;DID THE DST GET CLEARED?
BEQ TST27 ;;BRANCH IF YES

; *****
;CHECK THE ROM FLOW
FO: BIT #SW9,$SWR ;;SWITCH 9 ON?
BEQ .+10 ;;BRANCH IF NO
TSTB $ERFLG ;;ANY ERRORS?
BNE 4S ;;BRANCH IF YES
CLR R5 ;;USED TO CLEAR MAINTENANCE MODE
MOV #1S,$#FPPVEC ;;SET INTERRUPT VECTOR
MOV #POMSUB,R0 ;;GET ADDRESS OF TABLE
3S: MOVR (R0)+,R3 ;;GET DATA FOR MICRO-BREAK REG
BEQ 5S ;;BRANCH IF DONE
LDFPS #PMM ;;LOAD FPS REGISTER
LDUR ;;LOAD MICRO-BREAK REGISTER
SUBP AC1,ACO ;;EXECUTE INSTRUCTION
CLR R1
2S: INC R1 ;;WAIT FOR FPP
BNE 2S ;;TO FINISH IF NO INTERRUPT
LDFPS R5 ;;CLEAR MAINTENANCE MODE

```

```

3242 012746 114037 001200      MOVF      -(R0),@R$TMP0      ;SAVE ADDRESS THAT FAILED
3243 012752 105037 001201      CLRR      @R$TMP0+1        ;GET RID OF SIGN EXT
3244 012756 104034              ERROR      34              ;FAILED TO TRAP ON THE ROM STATE
3245 012760 022626              1S:      CMP      (SP)+,(SP)+  ;RESTORE STACK POINTER
3246 012762 000757              BR        3S              ;GO TO NEXT ROM STATE
3247 012764 104000              4S:      ERROR      0              ;ONLY EXECUTES WHILE LOOPING ON EPROR
3248 012766 170105              5S:      LDFPS     R5              ;CLEAR MAINTENANCE MODE
3249 012770 005037 001164      CLR      $REG2            ;SAVE EXPECTED VALUE
3250 012774 104077              ERROR      77              ;FLOW OF, DATA BAD
3251 012776      065      254      000      ROMSUB: .BYTE 65,254,0
3252                                .EVEN
3253
3254                                ;;*****
3255                                ;*TEST 27      CNPF*MO*(SRC=0)*--(DST=0)
3256                                ;*
3257                                ;*      THE ONLY THING THAT SHOULD FAIL IS THE CONTROL STORE
3258                                ;*      FOR ROM STATE 324, OR THE A OR NO-MEM ROMS.
3259                                ;*
3260                                ;*      FPU ROM FLOW-30,67,324
3261                                ;;*****
3262 013002 000004      TST27:  SCOPE
3263 013004 012737 000027 001240      MOV      @STN-1,$TESTN    ;;SFT TEST NUMBER IN MAIL BOX
3264 013012 012737 013162 001222      MOV      @TST30,$ESCAPE  ;;ESCAPE TO TEST 30 ON ERROR
3265 013020 172427 140777              LDF      @-0140277,AC0    ;MAKE DST NON-ZERO
3266 013024 172527 000000              LDF      @0,AC1          ;MAKE SRC=ZERO
3267 013030 170127 000017              LDFPS    @17            ;LOAD COMPLIMENT OF EXPECTED CC'S
3268 013034 173401              1S:      CNPF      AC1,AC0    ;EXECUTE INSTRUCTION UNDER TEST
3269 013036 170237 001200              STFPS    $TMP0          ;GET FPS
3270 013042 005737 001200              TST      $TMP0          ;ARE CC'S OK?
3271 013046 001445              BEQ      TST30          ;;BRANCH IF YES
3272
3273                                ;;*****
3274 013050 032777 001000 166060      GO:      BIT      @SW9,@SWR  ;SWITCH 9 ON?
3275 013056 001403              BEQ      .+10          ;BRANCH IF NO
3276 013060 105737 001103              TSTB     $ERFLG        ;ANY ERRORS?
3277 013064 001027              BNE      4S            ;BRANCH IF YES
3278 013066 005005              CLR      R5            ;USED TO CLEAR MAINTENANCE MODE
3279 013070 012737 013140 000744      MOV      @1S,@FPPVEC    ;SET INTERRUPT VECTOR
3280 013076 012700 013156              MOV      @ROMCMP,R0     ;GET ADDRESS OF TABLE
3281 013102 112003              3S:      MOVB     (R0)+,R3   ;GET DATA FOR MICRO-BREAK REG
3282 013104 001420              BEQ      5S            ;BRANCH IF DONE
3283 013106 170127 000020              LDFPS    @FMW          ;LOAD FPS REGISTER
3284 013112 170003              LDUR     ;LOAD MICRO-BREAK REGISTER
3285 013114 173401              CNPF     AC1,AC0        ;EXECUTE INSTRUCTION
3286 013116 005001              CLR      R1
3287 013120 005201              2S:      INC      R1          ;WAIT FOR FPP
3288 013122 001376              BNE      2S            ;TO FINISH IF NO INTERRUPT
3289 013124 170105              LDFPS    R5            ;CLEAR MAINTENANCE MODE
3290 013126 114037 001200      MOVF     -(R0),@R$TMP0  ;SAVE ADDRESS THAT FAILED
3291 013137 105037 001701      CLRR     @R$TMP0+1      ;GET RID OF SIGN EXT
3292 013136 104034              ERROR    34            ;FAILED TO TRAP ON THE ROM STATE
3293 013140 022626              1S:      CMP      (SP)+,(SP)+  ;RESTORE STACK POINTER
3294 013142 000757              BR       3S            ;GO TO NEXT ROM STATE
3295 013144 104000              4S:      ERROR    0              ;ONLY EXECUTES WHILE LOOPING ON EPROR
3296 013146 170105              5S:      LDFPS    R5            ;CLEAR MAINTENANCE MODE
3297 013150 005037 001202      CLR      $TMP1          ;SAVE EXPECTED CC'S

```

3298 013154 104100
3299 013156 067 324 000 ROMCMP:
3300 013162
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310 013162 000004
3311 013164 012737 000030 001240
3312 013172 172427 000000
3313 013176 172527 140200
3314 013202 170127 000013
3315 013206 173401
3316 013210 170237 001200
3317 013214 022737 000010 001200
3318 013222 001404
3319 013224 012737 000010 001202
3320 013232 104071
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330 013234 000004
3331 013236 012737 000031 001240
3332 013244 012737 013334 001222
3333 013252 172427 000177
3334 013256 172527 000000
3335 013262 170127 000013
3336 013266 173401
3337 013270 170237 001200
3338 013274 174037 001162
3339 013300 005737 001162
3340 013304 001403
3341 013306 005037 001164
3342 013312 104101
3343 013314 022737 000004 001200
3344 013322 001404
3345 013324 012737 000004 001202
3346 013332 104071
3347
3348
3349
3350
3351
3352
3353

ERROR 100 ;FLOW OK, CC'S BAD
.PYTE 67,324,0
.EVEN

;*TEST 30 CMPF*W0*(SRC=0)*(DST=0)
;*
;* THE ONLY POSSIBLE FAILURE IS THE CONTROL STORE SIGNALS IN ROM
;* STATE 330 OR THE SD MUX.
;*
;* FPU ROM FLOW-30,67,330

TST30: SCOPE
MOV #STN-1,STESTH ;SET TEST NUMBER IN MAIL BOX
LDF #0,AC0 ;MAKE THE DST=0
LDF #0140200,AC1 ;MAKE THE SRC NON-ZERO
LDFPS #13 ;LOAD COMP OF EXPECTED CC'S
1\$: CMPF AC1,AC0 ;EXECUTE INSTRUCTION UNDER TEST
STFPS \$TMP0 ;GET CC'S
CMP #10,\$TMP0 ;CC'S OK?
BEQ TST31 ;BRANCH IF YES
MOV #10,\$TMP1 ;SAVE EXPECTED CC'S
ERROR 71 ;CC'S BAD

;*TEST 31 CMPF*W0*(SRC=0)*(DST=0)
;*
;* THE ONLY POSSIBLE FAILURE IS THE CONTROL STORE SIGNALS
;* IN ROM STATE 334.
;*
;* FPU ROM FLOW-30,67,334

TST31: SCOPE
MOV #STN-1,STESTH ;SET TEST NUMBER IN MAIL BOX
MOV #TST37,\$ESCAPE ;ESCAPE TO TEST 32 ON ERROR
LDF #0177,AC0 ;MAKE DST EXP=0, FRAC NON-ZERO
LDF #0,AC1 ;MAKE SPC=0
LDFPS #13 ;LOAD COMPLIMENT OF EXPECTED CC'S
1\$: CMPF AC1,AC0 ;EXECUTE INSTRUCTION UNDER TEST
STFPS \$TMP0 ;GET CC'S BACK
STF AC0,\$REG1 ;GET DATA BACK
TST \$REG1 ;DID DST FRACTION CLEAR?
BEQ 2\$;BRANCH IF YES
CLR \$REG2 ;STORE EXPECTED VALUE
ERROR 101 ;DST FRAC DID NOT CLEAR
2\$: CMP #4,\$TMP0 ;CC'S OK?
BEQ TST32 ;BRANCH IF YES
MOV #4,\$TMP1 ;SAVE EXPECTED VALUE
EPROR 71 ;CC'S BAD

.SBTTL
.SBTTL FOUR ROM STATES
.SBTTL

;*TEST 32 LDEXP*-M0*(EXP=+)
;*

```

3354 ;* THE A AND NO-MEM BRANCHES SHOULD NOT FAIL.
3355 ;*BZ-BN BRANCH
3356 ;* IF PRMB BN(0)W DOES NOT GO HIGH OR DOES NOT GET TO PAD01 AS A LOW,
3357 ;* EXECUTION WILL GO TO STATE 360. A MICRO-BREAK TRAP WILL BE SET
3358 ;* TO CATCH THIS FAILURE.
3359 ;* THE BZ SIGNAL SHOULD NOT FAIL.
3360 ;*
3361 ;*
3362 ;* A ZERO AND ONE ARE THEN FLOATED THROUGH BITS <6:0> TO CHECK THE
3363 ;* STEP COUNTER.
3364 ;*
3365 ;* FPU ROM FLOW-15,10,260,36?
3366 ;*.....
3366 013334 000004 TST32: SCOPE
3367 013336 012737 000032 001240 MOV #STN-1,STESTN ;;SET TEST NUMBER IN MAIL BOX
3368 013344 012737 013744 001222 MOV #TST33,SESCAPE ;;ESCAPE TO TEST 33 ON ERROR
3369 013352 012703 000360 MOV #360,R3 ;SET THE MICROBREAK
3370 013356 170003 LDUB ;REGISTER TO TRAP IF BRANCH FAILS
3371 013360 012737 013504 000244 MOV #25,#PPPVEC ;SET THE TRAP VECTOR
3372 013366 012737 013374 001110 MOV #.+6,#SLPERR ;SET ERROR LOOP
3373 013374 172427 000177 LDF #0177,ACO ;MAKE ACO EXP=0 FRAC NON-ZERO
3374 013400 105737 001103 TSTR SERFLG ;ANY ERRORS YET?
3375 013404 001002 BNE 100$ ;BRANCH IF YES
3376 013406 170127 000037 LDFPS #FMM+17 ;SET FMM & CC'S TO COMPLIMENT OF EXPECTED
3377 013417 005037 001204 100$: CLR $TMP2 ;PUT EXPONENT TO LOAD IN $TMP0
3378 013416 012700 001204 MOV #TMP2,R0 ;PUT ADDRESS OF DATA IN R0
3379 013422 176420 1$: LDEXP (R0)+,ACO ;EXECUTE INSTRUCTION UNDER TEST
3380 013424 170237 001200 STFPS $TMP0 ;GET CC'S BACK
3381 013430 170137 001204 LDFPS $TMP2 ;CLEAR FMM BIT
3382 013434 022700 001206 CMP #TMP3,R0 ;ADX POW OK?
3383 013440 001401 BEQ 9$ ;BRANCH IF YES
3384 013442 104066 ERROR 66 ;ADX POW FAILED
3385 013444 174037 001162 9$: STF ACO,$REG1 ;GET DATA BACK
3386 013450 022737 040177 001162 CMP #40177,$REG1 ;IS DATA OK?
3387 013456 001416 BEQ 3$ ;BRANCH IF YES
3388 013460 012737 040177 001164 MOV #40177,$REG2 ;SAVE EXPECTED DATA
3389 013466 013701 001162 MOV $REG1,R1 ;GET DATA
3390 013472 042701 177600 BIC #177600,R1 ;MASK FRACTION BITS
3391 013476 001001 BNE 4$ ;BRANCH IF THEY ARE OK.
3392 013500 104104 ERROR 104 ;FRAC BITS GOT CHANGED. (CONT STORE)
3393 013502 104105 4$: ERROR 105 ;EXP. DID NOT LOAD PROPERLY (CONT STORE)
3394 ;BZ-BN BRANCH FAILED
3395 013504 022626 2$: CMP (SP)+,(SP)+ ;RESTORE THE STACK POINTER
3396 013506 170137 001204 LDFPS $TMP2 ;CLEAR FMM BIT
3397 013512 104106 ERROR 106 ;BN FAILED
3398 ;CHECK CC'S
3399 013514 042737 000020 001200 3$: BIC #FMM,$TMP0
3400 013522 005737 001200 TST $TMP0 ;CC'S OK?
3401 013526 001403 BEQ 5$ ;BRANCH IF YES
3402 013530 005037 001202 CLR $TMP1 ;SAVE EXPECTED VALUE
3403 013534 104071 ERROR 71 ;CC'S BAD (CONT STORE FAILURE)
3404 ;*.....
3405 ;CHECK THE STEP COUNTER - FLOAT A ONE
3406 013536 012737 000001 001204 5$: MOV #1,$TMP2 ;SET FIRST DATA PATTERN
3407 013544 012702 040377 MOV #40377,P2 ;INITIALIZE CHECK DATA
3408 013550 012700 000007 MOV #7,R0 ;SET SUB COUNT
3409 013554 012701 000200 MOV #BIT7,R1 ;INITIALIZE R1

```

```

3410 013560 012737 013566 001110      MOV      #.+6,#BSLPERR      ;SET ERROR LOOP
3411 013566 176437 001204      LDEXP   $TMP2,AC0          ;EXECUTE INSTRUCTION
3412 013572 174037 001162      STF     AC0,$REG1         ;GET DATA BACK
3413 013576 020237 001162      CMP     R2,$REG1         ;DATA OK?
3414 013602 001055              BNE     B5                ;BRANCH IF NO
3415 013604 105737 001103      TSTB   $ERFLG           ;ANY ERRORS YET?
3416 013610 001366              BNE     65                ;BRANCH IF YES
3417 013612 006337 001204      ASL     $TMP2            ;ROTATE DATA TO LOAD
3418 013616 012702 040177      MOV     #40177,R2        ;INITIALIZE CHECK DATA
3419 013622 006301              ASL     R1                ;
3420 013624 050102              BIS     R1,R2            ;SET NEW CHECK DATA
3421 013626 077021              SOB     R0,65            ;CONTINUE
3422
3423      ;*****
3424 013630 012737 000176 001204      MOV     #176,$TMP2       ;INITIALIZE DATA TO LOAD
3425 013636 012702 077577              MOV     #77577,R2       ;INITIALIZE CHECK DATA
3426 013647 012700 000007              MOV     #7,R0          ;SET SOB COUNT
3427 013646 012737 013654 001110      MOV     #.+6,#BSLPERR   ;SET ERROR LOOP
3428 013654 176437 001204      LDEXP   $TMP2,AC0          ;EXECUTE INSTRUCTION
3429 013660 174037 001162      STF     AC0,$REG1         ;GET DATA BACK
3430 013664 020237 001162      CMP     R2,$REG1         ;DATA OK?
3431 013670 001022              BNE     B5                ;BRANCH IF NO
3432 013672 105737 001103      TSTB   $ERFLG           ;ANY ERRORS YET?
3433 013676 001366              BNE     75                ;BRANCH IF YES
3434 013700 000261              SEC
3435 013702 106137 001704      ROLR    $TMP2            ;ROTATE THE ZERO
3436 013706 042737 000200 001204      BIC     #BIT7,$TMP2      ;DON'T LET BIT 7 COME ON
3437 013714 000261              SEC
3438 013716 006102              ROL     R2                ;ROTATE CHECK DATA
3439 013720 042702 100000      BIC     #BIT15,R2        ;DON'T LET SIGN CHANGE
3440 013724 077025              SOB     R0,75            ;CONTINUE
3441 013726 012737 047624 000244      MOV     #FPPSPUR,#FPPVEC ;RESTORE FPP VECTOR
3442 013734 000403              BP     TST33             ;GO TO NEXT TEST
3443 013736 010237 001164      85:    MOV     R2,$REG2     ;SAVE EXPECTED DATA
3444 013742 104107      ERROR   107              ;BIT IN SC FAILED
3445
3446      ;*****
3447      ;*TEST 33      ABSF*NO*(SRC=0)
3448      ;*
3449      ;*BZ BRANCH
3450      ;*      IF PRMB BZ(0)H DOES NOT GO LOW OR DOES NOT GET TO FRNA RADROO
3451      ;*      AS A HIGH, EXECUTION WILL GO TO ROM STATE 3. THIS WILL CAUSE THE
3452      ;*      FRACTION NOT TO BE CLEARED.
3453      ;*
3454      ;*      FPU ROM FLOW-30,57,124,106
3455      ;*****
3456 013744 000004      TST33: SCUPE
3457 013746 012737 000033 001240      MOV     #STN-1,$TESTN    ;SET TEST NUMBER IN MAIL BOX
3458 013754 012737 014034 001222      MOV     #TST34,$ESCAPE  ;ESCAPE TO TEST 34 ON ERROR
3459 013762 172427 000177              LDF     #*0177,AC0      ;LOAD DATA; EXP=0, PRAC NON-ZERO
3460 013766 170127 000013              LDFPS  #13              ;LOAD COMPLIMENT OF EXPECTED CC'S
3461 013772 170600      15:    ABSF   AC0                ;EXECUTE INSTRUCTION UNDER TEST
3462 013774 170237 001200      STFPS  $TMP0            ;GET CC'S
3463 014000 174037 001162      STF     AC0,$REG1         ;GET DATA BACK
3464 014004 005737 001162      TST    $REG1            ;DID FRACTION CLEAR?
3465 014010 001401      BEQ    25                ;BRANCH IF YES

```

```

3466 014012 104102
3467 014014 022737 000004 001200 2S:
3468 014022 001404
3469 014024 012737 000004 001202
3470 014032 104071
3471
3472
3473
3474
3475
3476
3477
3478
3479
3480 014034 000004
3481 014036 012737 000034 001240
3482 014044 012737 014100 001222
3483 014052 172427 000177
3484 014056 170700
3485 014060 174037 001162
3486 014064 005737 001162
3487 014070 001403
3488 014072 005037 001164
3489 014076 104103
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501
3502 014100 000004
3503 014102 012737 000035 001240
3504 014110 012737 014250 001222
3505 014116 172427 000177
3506 014122 172527 040200
3507 014126 170127 000017
3508 014132 172001
3509 014134 170237 001200
3510 014140 174037 001162
3511 014144 022737 040200 001162
3512 014152 001406
3513 014154 100404
3514 014156 012737 040200 001164
3515 014164 104110
3516 014166 104111
3517
3518 014170 005737 001200
3519 014174 001403
3520 014176 005037 001202
3521 014202 104071
  
```

```

ERROR 102 ;R2 BRANCH FAILED
CMP #4,$TMP0 ;CC'S OK?
BEQ TST34 ;;BRANCH IF YES
MOV #4,$TMP1 ;SAVE EXPECTED VALUE
ERROR 71 ;CC'S BAD IN ROM STATE 106.

;*****
;*TEST 34 NEGF*MO*(SRC=0)
;*
;* THE ONLY POSSIBLE FAILURE WOULD BE THE CONTROL STORE IN ROM
;* STATES 134 OR 106.
;*
;* FPU ROM FLOW-30,57,134,106
;*****
TST34: SCOPE
MOV #STN-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX
MOV #TST35,$ESCAPE ;;ESCAPE TO TEST 35 ON ERROR
LDF #0177,AC0 ;LOAD DATA EXP=0, FRAC NON-ZERO
1S: NEGF AC0 ;EXECUTE INSTRUCTION UNDER TEST
STF AC0,$REG1 ;GET DATA BACK
TST $REG1 ;DID SIGN & FRACT. CLEAR?
BEQ TST35 ;;BRANCH IF YES
CLR $REG2 ;SAVE EXPECTED VALUE
ERROR 103 ;CONTROL STORE FAILURE.

;*****
;*TEST 35 ADDP*MO*--(SRC=0)*(DST=0)
;*
;* THE A AND NO-WEM BRANCHES SHOULD NOT FAIL.
;*6F1 BRANCH
;* IF FRMJ EXPB0(1)H DOES NOT GO LOW OR DOES NOT GET TO FRMA RAO2 AS A
;* HIGH, EXECUTION WILL GO TO STATE 754. THIS WILL CAUSE THE DESTINATION FRAC. TO
;* BE CLEAPED.
;*
;* FPU ROM FLOW-30,65,250,306
;*****
TST35: SCOPE
MOV #STN-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX
MOV #TST36,$ESCAPE ;;ESCAPE TO TEST 36 ON ERROR
LDF #0177,AC0 ;MAKE DST EXP=0, FRAC NON-ZERO
LDF #040200,AC1 ;MAKE SRC NON-ZERO
LDFPS #17
1S: ADDP AC1,AC0 ;LOAD COMPLIMENT OF EXPECTED CC'S
STFPS $TMP0 ;EXECUTE INSTRUCTION UNDER TEST
STF AC0,$REG1 ;GET CC'S BACK
CMP #40200,$REG1 ;GET DST BACK
BEQ 35 ;DID INSTRUCTION WORK?
BMI 25 ;BRANCH IF YES
MOV #40200,$REG2 ;BRANCH IF SIGN WRONG
ERROR 110 ;SAVE EXPECTED VALUE
2S: ERROR 111 ;SIGN OK, BUT EXP OR FRAC WRONG
;CHECK THE CC'S ;SIGN BAD
3S: TST $TMP0 ;CC'S OK?
BEQ 45 ;BRANCH IF YES
CLR $TMP1 ;SAVE EXPECTED VALUE
ERROR 71 ;CC'S BAD
  
```

```

3522 ;
3523 ;CHECK THAT SIGN OF DST GETS CHANGED ON NEGATIVE SRC
3524 014204 4S:
3525 014204 012737 014212 001110 MOV #.+6,0#SLPERR ;SET ERROR LOOP
3526 014212 172427 000177 LDF #~0177,ACO ;LOAD DST
3527 014216 172527 140200 LDF #~0140200,AC1 ;LOAD NEGATIVE SRC
3528 014222 172001 ADDP AC1,ACO ;EXECUTE INSTRUCTION
3529 014224 174037 001162 STF ACO,$REG1 ;GET DST BACK
3530 014230 022737 140200 001162 CMP #140200,$REG1 ;SIGN OK?
3531 014236 001404 BEQ TST36 ;BRANCH IF YES
3532 014240 012737 140200 001164 MOV #140200,$REG2 ;SAVE EXPECTED VALUE
3533 014246 104112 ERROR 112 ;PRMF SS XOR SUB FAILED
3534
3535 ;*****
3536 ;*TEST 36 SUBP*MO*-(SRC=0)*(DST=0)
3537 ;*
3538 ;* THE ONLY POSSIBLE FAILURE IN THIS TEST IS THE SIGNAL PRMF SS(0)W XOR SUB.
3539 ;*
3540 ;* FPU ROM FLOW-30,65,250,306
3541 ;*****
3542 014250 000004 TST36: SCOPE
3543 014252 012737 000036 001240 MOV #STM-1,$TESTN ;SET TEST NUMBER IN MAIL BOX
3544 014260 012737 014370 001222 MOV #TST37,$ESCAPE ;ESCAPE TO TEST 37 ON ERROR
3545 014266 172427 000177 LDF #~0177,ACO ;LOAD DESTINATION
3546 014272 172527 140200 LDF #~0140200,AC1 ;LOAD SRC
3547 014276 173001 1S: SUBP AC1,ACO ;EXECUTE INSTRUCTION
3548 014300 174037 001162 STF ACO,$REG1 ;GET DATA BACK
3549 014304 022737 040200 001162 CMP #40200,$REG1 ;DATA OK?
3550 014312 001404 BEQ 2S
3551 014314 012737 040200 001164 MOV #40200,$REG2 ;SAVE EXPECTED DATA
3552 014322 104111 ERROR 111 ;SD DID NOT CLEAR
3553 ;NOW TEST THE SUB WITH POSITIVE SRC
3554 014324 2S:
3555 014324 012737 014332 001110 MOV #.+6,0#SLPERR ;SET ERROR LOOP
3556 014332 172427 000177 LDF #~0177,ACO ;LOAD DST
3557 014336 172527 040200 LDF #~040200,AC1 ;LOAD SRC
3558 014342 173001 SUBP AC1,ACO ;EXECUTE INSTRUCTION
3559 014344 174037 001162 STF ACO,$REG1 ;GET DATA
3560 014350 022737 140200 001162 CMP #140200,$REG1 ;DATA OK?
3561 014356 001404 BEQ TST37 ;BRANCH IF YES
3562 014360 012737 140200 001164 MOV #140200,$REG2 ;SAVE EXPECTED VALUE
3563 014366 104112 ERROR 112 ;SD DID NOT SET
3564
3565 ;*****
3566 ;*TEST 37 CMPF*MO*(SRC>DST)
3567 ;*
3568 ;* THE A, NO-MEM, AND 6F1 BRANCHES SHOULD NOT FAIL.
3569 ;*3F2 BRANCH
3570 ;*
3571 ;* THE BZ BIT SHOULD NOT FAIL.
3572 ;* IF FXPL BCN L DOES NOT GO HIGH WITH EALU09 ON A LOW, EXECUTION WILL
3573 ;* GO TO STATE 301. THIS WILL CAUSE THE N BIT TO BE WRONG.
3574 ;*
3575 ;* A ONE(1) IS THEN FLOATED THROUGH EXPONENT A TO VERIFY
3576 ;* THAT FXPL EXPA EQ 0 GOES LOW FOR ALL 8 BITS.
3577 ;*
3578 ;* FPU ROM FLOW-30,67,320,303

```

```

3578 ;*****
3579 014370 000004 TST37: SCOPE
3580 014372 012737 000037 001240 MOV #STN-1,STFSTN ;SET TEST NUMBER IN MAIL BOX
3581 014400 012737 014660 001222 MOV #TST40,SESCAPE ;ESCAPE TO TEST 40 ON ERROR
3582 014406 172427 040000 LDF #040000,AC0 ;LOAD THE DST WITH EXP=200
3583 014412 172527 040200 LDF #040200,AC1 ;LOAD THE SRC WITH EXP=201
3584 014416 170127 000017 LDFPS #17 ;LOAD COMPLIMENT OF EXPECTED CC'S
3585 014422 173401 1S: CMPF AC1,AC0 ;EXECUTE INSTRUCTION UNDER TEST
3586 014424 170237 001200 STFPS $TMP0 ;GET CC'S BACK
3587 014430 005737 001200 TST $TMP0 ;CC'S OK?
3588 014434 001410 BEQ 3S ;BRANCH IF YES
3589 014436 005037 001200 CLK $TMP1 ;SAVE EXPECTED VALUE
3590 014447 022737 000010 001200 CMP #10,$TMP0 ;IS N BIT WRONG?
3591 014450 001001 BNE 2S ;BRANCH IF NO OR ALSO OTHER BITS
3592 014452 104113 ERROR 113 ;RZ-BN BRANCH FAILED
3593 014454 104071 2S: ERROR 71 ;CC'S BAD
3594 ;*****
3595 ;CHECK SS(1) 0 AND SD(1) 1
3596 014456 170700 3S: NEGF AC0 ;MAKE DST NEGATIVE
3597 014460 012737 014466 001110 MOV #.+6,#SLPERR ;SET ERROR LOOP
3598 014466 170127 000017 LDFPS #17 ;LOAD COMPLIMENT CC'S
3599 014472 173401 CMPF AC1,AC0 ;EXECUTE INSTRUCTION
3600 014474 170000 CFCC ;GET CC'S
3601 014476 100001 BPL 4S ;BRANCH IF N OK
3602 014500 104114 ERROR 114 ;PRMF SS XOR SD DID NOT GO LOW
3603 ;*****
3604 ;CHECK SS(1) 1 AND SD(1) 0
3605 014502 170701 4S: NEGF AC1 ;MAKE SRC NEGATIVE
3606 014504 170700 NEGF AC0 ;MAKE DST POSITIVE
3607 014506 012737 014514 001110 MOV #.+6,#SLPERR ;SET ERROR LOOP
3608 014514 170127 000007 LDFPS #7 ;LOAD COMPLIMENT CC'S
3609 014520 173401 CMPF AC1,AC0 ;EXECUTE INSTRUCTION
3610 014522 170000 CFCC ;GET CC'S
3611 014524 100401 BMI 5S ;BRANCH IF N BIT OK
3612 014526 104115 ERROR 115 ;PRMF SS XOR SD FAILED
3613 ;*****
3614 ;CHECK SS(1) 1 AND SD(1) 1
3615 014530 170700 5S: NEGF AC0 ;MAKE DST NEG
3616 014532 012737 014540 001110 MOV #.+6,#SLPERR ;SET ERROR LOOP
3617 014540 170127 000007 LDFPS #7 ;SET COMPLIMENT CC'S
3618 014544 173401 CMPF AC1,AC0 ;EXECUTE INSTRUCTION
3619 014546 170000 CFCC ;GET CC'S
3620 014550 100401 BMI 6S ;BRANCH IF N OK
3621 014552 104115 ERROR 115
3622 ;*****
3623 ;CHECK FXPL EXPA EQ 0
3624 014554 172527 077600 014616 6S: LDF #077600,AC1 ;LOAD THE SRC EXP=377
3625 014560 012737 000200 MOV #200,7S ;INIT THE DST DATA
3626 014566 012703 000320 MOV #320,R3 ;LOAD MICRO-BREAK
3627 014572 170003 LDUR
3628 014574 012737 014650 000244 MOV #8S,FPVVEC
3629 014602 170127 000020 LDFPS #FMM
3630 014606 012737 014614 001110 MOV #.+6,#SLPEPR ;SET ERROR LOOP
3631 014614 172427 9S: LDF (PC)+,AC0 ;LOAD THE DST
3632 014616 000200 7S: .WORD 200
3633 014620 173401 CMPF AC1,AC0 ;EXECUTE TEST INSTRUCTION

```


3634 014622 170000
3635
3636 014624 170200
3637 014626 032700 000020
3638 014632 001770
3639 014634 013737 014616 001200
3640 014642 005037 001202
3641 014646 104302
3642 014650 022626
3643 014652 006137 014616
3644 014656 100356
3645
3646
3647
3648
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658 014660 000004
3659 014662 012737 000040 001240
3660 014670 012737 015056 001222
3661 014676 172427 040200
3662 014702 172527 040000
3663 014706 170127 000007
3664 014712 173401
3665 014714 170237 001200
3666 014720 022737 000010 001200
3667 014726 001401
3668 014730 104116
3669
3670
3671
3672
3673 014732 170700
3674 014734 012737 014742 001110
3675 014742 170127 000017
3676 014746 173401
3677 014750 170000
3678 014752 100001
3679 014754 104117
3680
3681
3682 014756 012737 000200 015014
3683 014764 012737 015046 000244
3684 014772 012703 000320
3685 014776 170003
3686 015000 170127 000020
3687 015004 012737 015012 001110
3688 015012 172427
3689 015014 000200

```
CFCC ;WAIT FOR TRAP
;NO TRAP OCCUPRED-LOOPING ON ERROR?
STFPS R0 ;GET FPS
BIT #FMM,R0 ;MAINT MODE ON?
BEQ 9S ;BRANCH IF NO
MOV 7S,STMP0 ;SAVE PATTERN
CLR STMP1 ;THAT FAILED
ERROR 302 ;FXPL EXPA EQ 0 DID NOT GO LOW
8S: CMP (SP)+,(SP)+ ;RESTORE THE SP
ROL 7S ;SELECT NEXT PATTERN
BPL 9S ;CONTINUE UNTIL DONE

;;*****
;*TEST 40 CMPF*MO*(SRC<DST)
;*
;*3F2 BRANCH
;* IF FXPL BCN L DOES NOT GO LOW WITH EVALU09 ON A HIGH, EXECUTION WILL
;* GO TO STATE 303. THIS WILL CAUSE THE N BIT TO BE WRONG.
;*
;* A ONE(1) IS THEN FLOATED THROUGH EXPONENT 8 TO VERIFY THAT
;* FXPL EXPB EQ 0 GOES LOW FOR ALL 8 BITS.
;*
;* FPU ROM FLOW-30,67,320,301
;;*****
TST40: SCOPE
MOV #STW-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX
MOV #TST41,$ESCAPE ;;ESCAPE TO TEST 41 ON ERROR
LDF #040200,ACO ;LOAD THE DST
LDF #040000,AC1 ;LOAD THE SRC
LDFPS #7 ;LOAD COMPLIMENT CC'S
1S: CMPF AC1,ACO ;EXECUTE INSTRUCTION UNDER TEST
STFPS STMP0 ;GET CC'S BACK
CMP #10,$STMP0 ;CC'S OK?
BEQ 2S ;BRANCH IF YES
EPROR 116 ;BZ-BN BRANCH FAILED OR
;FRMF SS(0)H FAILED TO GET
;THRU SD WUX

;;*****
;CHECK FRMF SS(0) GOING LOW
2S: NEGF ACO ;MAKE THE DST NEGATIVE
MOV #.+6,@#SLPERR ;SET ERROR LOOP
LDFPS #17 ;LOAD COMPLIMENT CC'S
CMPF AC1,ACO ;EXECUTE INSTRUCTION
CFCC ;GET CC'S
BPL 6S ;BRANCH IF N BIT OK
ERROR 117 ;FRMF SS(0)H NOT GOING HIGH

;;*****
;CHECK FXPL EXPR EQ 0
6S: MOV #200,3S ;INIT THE SPC DATA
MOV #4S,FPVVEC
MOV #320,R3
LDUR
LDFPS #FMM
MOV #.+6,@#SLPERR ;SET ERROR LOOP
5S: LDF (PC)+,ACO ;LOAD THE SRC
3S: .WORD 200
```

```

3690 015016 173500
3691 015020 170000
3697
3693 015022 170200
3694 015024 032700 000020
3695 015030 001770
3696 015032 013737 015014 001200
3697 015040 005037 001202
3698 015044 104303
3699 015046 022626
3700 015050 006137 015014
3701 015054 100356
3702
3703
3704
3705
3706
3707
3708
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3720 015056 000004
3721 015060 012737 000041 001240
3722 015066 012737 015262 001222
3723 015074 012737 040377 001200
3724 015102 012737 177777 001202
3725 015110 012737 177777 001204
3726 015116 012737 177777 001206
3727 015124 012700 001200
3728 015130 005037 001210
3729 015134 005037 001212
3730 015140 005037 001214
3731 015144 005037 001216
3732 015150 170127 000200
3733 015154 172420
3734 015156 022700 001210
3735 015162 001406
3736 015164 010037 001162
3737 015170 012737 001210 001164
3738 015176 104120
3739
3740 015200 174020
3741 015202 022700 001220
3742 015206 001406
3743 015210 010037 001162
3744 015214 012737 001220 001164
3745 015222 104120
  
```

```

CMPF ACO,AC1 ;EXECUTE THE TEST INSTRUCTION
CFCC ;WAIT FOR TRAP
;NO TRAP OCCURRED-LOOPING ON ERROR?
STFPS R0
BIT #FMM,P0 ;MAINT MODE?
BEQ 55 ;BRANCH IF NO
MOV 35,$TMP0 ;SAVE PATTERN
CLR $TMP1 ;THAT FAILED
ERROR 303 ;FXPL EXPN EQ 0 DID NOT GO LOW
4$: CMP (SP)+,(SP)+ ;RESTORE THE SP
ROL 35 ;SELECT NEXT PATTERN
BPL 55 ;CONTINUE UNTIL DONE

;*****
;*TEST 41 LDD/STD*-MO*-IMM
;*
;* IF FRMJ FD(1) DOES NOT GO HIGH, ONLY 2 WORDS WILL BE LOADED
;* AND STOPED.
;* IF THE ADX POM FAILS, THE DESTINATION REGISTER WILL AUTO-INCREMENT
;* BY 4 INSTEAD OF 10. THIS WILL ALSO OCCUR IF AD1 & AD2 DO NOT GET
;* TO TRCD AS A LOW.
;*
;* IF FRMF FPREQ DOES NOT STAY ASCERTED FOR 4 CYCLES THE CPU & FP
;* WILL GET OUT OF SYNC.
;*
;*LDD
;* FPU ROM FLOW-21,142,165,201,42,16
;*STD
;* FPU ROM FLOW-13,203,202,206,212
;*****
TST41: SCOPE
MOV #STN-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX
MOV #TST42,$ESCAPE ;;ESCAPE TO TEST 42 ON ERROR
MOV #40377,$TMP0 ;LOAD TPE
MOV #-1,$TMP1 ;DATA TO LOAD
MOV #-1,$TMP2 ;INTO MEMORY
MOV #-1,$TMP3 ;LOCATIONS
MOV $TMP0,P0 ;SETUP SRC REGISTER
CLR $TMP4 ;SETUP LOCATIONS
CLR $TMP5 ;WHERE THE
CLR $TMP6 ;DATA WILL
CLR $TMP7 ;BE STORED
LDFPS #FD ;SET THE FD BIT IN FPS
1$: LDD (R0)+,ACO ;EXECUTE INSTRUCTION UNDER TEST
CMP #TMP4,P0 ;DID R0 AUTO-INC OK?
BEQ 2$ ;BRANCH IF YES
MOV R0,$REG1 ;SAVE RECEIVED ADDRESS
MOV #TMP4,$REG2 ;SAVE EXPECTED ADDRESS
ERROR 120 ;TRCD DSTCON<10 NOT GOING HIGH
;NOW DO THE STD
2$: STD ACO,(R0)+ ;EXECUTE INSTRUCTION UNDER TEST
CMP #TMP7+2,R0 ;DID R0 AUTO-INC OK?
BEQ 3$ ;BRANCH IF YES
MOV R0,$REG1 ;SAVE RECEIVED ADDRESS
MOV #TMP7+2,$REG2 ;SAVE EXPECTED ADDRESS
ERROR 120 ;TRCD DSTCON<10 NOT GOING HIGH
  
```

```

3746 ;CHECK THE DATA THAT CAME BACK
3747 015224 022737 040377 001210 3S:  CMP #40377,STMP4 ;FIRST WORD OK?
3748 015232 001012                BNE 4S ;BRANCH IF NO
3749 015234 022737 177777 001212    CMP #-1,STMP5 ;SECOND WORD OK?
3750 015242 001006                BNE 4S ;BRANCH IF NO
3751 015244 005737 001214            TST STMP6 ;DID THIRD WORD CHANGE?
3752 015250 001403                BEQ 4S ;BRANCH IF NO
3753 015252 005737 001216            TST STMP7 ;DID FOURTH WORD CHANGE?
3754 015256 001001                BNE TST42 ;BRANCH IF YES
3755 015260 104121                4S:  ERROR 121 ;ROW FLOW OK, BUT DATA BAD
3756
3757
3758 ;*****
3759 ;*TEST 42 LOW ORDER DATA PATH
3760 ;*
3761 ;* THIS TEST FLOATS A ONE AND A ZERO THROUGH QUADRANTS 0 AND 1
3762 ;* OF ACO TO VERIFY THE LOW ORDER DATA PATH.
3763 ;*****
3764 015262 000004                TST42: SCOPE
3765 015264 012737 000042 001240    MOV #STW-1,STFSTN ;SET TEST NUMBER IN MAIL BOX
3766 015272 012737 000200 001200    MOV #200,STMP0 ;INITIALIZE
3767 015300 005037 001202                CLR STMP1 ;DATA
3768 015304 005037 001204                CLR STMP2 ;TO LOAD
3769 015310 012737 000001 001206    MOV #1,STMP3 ;INTG ACO
3770 015316 012701 000040                MOV #40,R1 ;SET SOB COUNT
3771 015322 012737 015330 001110    MOV #-+6,@$LPERR ;SET ERROR LOOP
3772 015330 170127 000200                6S:  LDFPS #FD ;SET THE PD BIT
3773 015334 172437 001200                1S:  LDD STMP0,ACO ;LOAD DATA PATTERN
3774 015340 174037 001210                STD ACO,STMP4 ;GET IT BACK
3775 015344 023737 001204 001714    CMP STMP2,STMP6 ;QUAD 1 OK?
3776 015352 001015                BNE 2S ;BRANCH IF NO
3777 015354 023737 001206 001216    CMP STMP3,STMP7 ;QUAD 0 OK?
3778 015362 001011                BNE 2S ;BRANCH IF NO
3779 015364 105737 001103            TSTR $ERFLG ;ANY ERRORS YET?
3780 015370 001357                BNE 6S ;BRANCH IF YES
3781 015372 006337 001206            ASL STMP3 ;SHIFT THE
3782 015376 006137 001204            ROL STMP2 ;DATA PATTERN
3783 015402 077124                SOB R1,1$ ;CONTINUE
3784 015404 000401                BR 3S
3785 015406 104122                2S:  ERROR 122 ;BIT FAILED WHILE FLOATING A ONE
3786 ;*****
3787 ;FLOAT A ZERO
3788 015410 012737 177777 001204    3S:  MOV #-1,STMP2 ;INITIALIZE
3789 015416 012737 177776 001206    MOV #-2,STMP3 ;DATA TO LOAD
3790 015424 012701 000040                MOV #40,R1 ;SET SOB COUNT
3791 015430 012737 015436 001110    MOV #-+6,@$LPERR ;SET ERROR LOOP
3792 015436 170127 000200                7S:  LDFPS #FD
3793 015442 172437 001200                4S:  LDD STMP0,ACO ;LOAD DATA PATTERN
3794 015446 174037 001210                STD ACO,STMP4 ;GET IT BACK
3795 015452 023737 001204 001214    CMP STMP2,STMP6 ;QUAD 1 OK?
3796 015460 001016                BNE 5S ;BRANCH IF NO
3797 015462 023737 001206 001216    CMP STMP3,STMP7 ;QUAD 2 OK?
3798 015470 001012                BNE 5S ;BRANCH IF NO
3799 015472 105737 001103            TSTR $ERFLG ;ANY ERRORS YET?
3800 015476 001357                BNE 7S ;BRANCH IF YES
3801 015500 000261                SEC

```

3802 015502 006137 001206
 3803 015506 006137 001204
 3804 015512 077125
 3805 015514 000401
 3806 015516 104122
 3807
 3808
 3809
 3810
 3811
 3812
 3813
 3814
 3815
 3816
 3817
 3818 015520 000004
 3819 015522 012737 000043 001240
 3820 015530 012700 001410
 3821 015534 012701 001210
 3822 015540 012721 040000
 3823 015544 005021
 3824 015546 012021
 3825 015550 012021
 3826 015552 170127 000200
 3827 015556 172437 001210
 3828 015562 174005
 3829 015564 012041
 3830 015566 012041
 3831 015570 172437 001210
 3832 015574 174004
 3833 015576 012021
 3834 015600 012021
 3835 015602 172737 001210
 3836 015606 012041
 3837 015610 012041
 3838 015612 172637 001210
 3839 015616 012021
 3840 015620 012021
 3841 015622 172537 001210
 3842 015626 005041
 3843 015630 005041
 3844 015632 172437 001210
 3845 015636 012737 015644 001110
 3846 015644 012701 001214
 3847 015650 170127 000200
 3848 015654 012700 001420
 3849 015660 174337 001210
 3850 015664 022021
 3851 015666 001044
 3852 015670 022021
 3853 015672 001042
 3854 015674 174737 001210
 3855 015700 022041
 3856 015702 001036
 3857 015704 022041

ROL STMP3 ;ROTATE THE
 ROL STMP2 ;ZERO
 SOB R1,4\$;CONTINUE
 BR TST43 ;GO TO NEXT TEST
 55: ERROR 122 ;BIT FAILED WHILE FLOATING A ZERO

;;*****
 ;*TEST 43 LOW ORDER SCRATCH PAD DUAL ADDRESSING
 ;*
 ;* THIS TEST ENSURES THAT THE LOW ORDER SCRATCH PAD ACCUMULATOR ADDRESS
 ;* LINES ARE FUNCTIONAL. THIS IS PERFORMED BY WRITING THE ADDRESS
 ;* OF THE ACCUMULATORS INTO THE ACCUMULATORS AND THEN READING
 ;* THEM BACK AND CHECKING THEM. THE ADDRESS OF THE ACCUMULATOR
 ;* IS WRITTEN INTO EACH CHIP OF THE ACCUMULATOR.
 ;;*****

TST43: SCOPE
 MOV #STN-1,STPSTN ;SET TEST NUMBER IN MAIL BOX
 MOV #\$\$AC5,R0 ;GET ADDRESS OF ACS DATA
 MOV #STMP4,R1 ;GET ADDRESS OF BUFFER AREA
 MOV #40000,(R1)+ ;SET UP QUAD 3
 CLR (R1)+ ;SET UP QUAD 2
 MOV (R0)+,(R1)+ ;LOAD THE APPROPRIATE
 MOV (R0)+,(R1)+ ;DATA INTO ALL THE
 LDFPS #FD ;ACCUMULATORS
 LDD STMP4,AC0 ; *
 STD AC0,AC5 ; *
 MOV (R0)+,-(R1) ; *
 MOV (R0)+,-(R1) ; *
 LDD STMP4,AC0 ; *
 STD AC0,AC4 ; *
 MOV (R0)+,(R1)+ ; *
 MOV (R0)+,(R1)+ ; *
 LDD STMP4,AC3 ; *
 MOV (R0)+,-(R1) ; *
 MOV (R0)+,-(R1) ; *
 LDD STMP4,AC2 ; *
 MOV (R0)+,(R1)+ ; *
 MOV (R0)+,(R1)+ ; *
 LDD STMP4,AC1 ; *
 CLR -(R1) ; *
 CLR -(R1) ; *
 LDD STMP4,AC0 ; *
 MOV #.+6,#SLPERM ;SET ERROR LOOP
 MOV #STMP6,R1
 LDFPS #FD
 MOV #\$\$AC3,R0 ;GET ADDRESS OF AC3 DATA
 STD AC3,STMP4 ;GET AC3 DATA
 CMP (R0)+,(R1)+ ;QUAD 1 OK?
 BNE IS ;BRANCH IF NO
 CMP (R0)+,(R1)+ ;QUAD 0 OK?
 BNE IS ;BRANCH IF NO
 STD AC2,STMP4 ;GET AC2 DATA
 CMP (R0)+,-(R1) ;QUAD 0 OK?
 BNE IS ;BRANCH IF NO
 CMP (R0)+,-(R1) ;QUAD 1 OK?

3858	015706	001034				BNE	1\$;BRANCH IF NO
3859	015710	174137	001210			STD	AC1,STMP4		;GET AC1 DATA
3860	015714	022021				CMP	(R0)+,(R1)+		;QUAD 1 OK?
3861	015716	001030				BNE	1\$;BRANCH IF NO
3862	015720	022021				CMP	(R0)+,(R1)+		;QUAD 0 OK?
3863	015722	001026				BNE	1\$;BRANCH IF NO
3864	015724	174037	001210			STD	AC0,STMP4		;GET AC0 DATA
3865	015730	022041				CMP	(R0)+,-(R1)		;QUAD 0 OK?
3866	015732	001022				BNE	1\$;BRANCH IF NO
3867	015734	022041				CMP	(R0)+,-(R1)		;QUAD 1 OK?
3868	015736	001020				BNE	1\$;BRANCH IF NO
3869	015740	012700	001410			MOV	\$\$\$AC5,R0		;GET ADDRESS OF AC5 DATA
3870	015744	172405				LDD	AC5,AC0		
3871	015746	174037	001210			STD	AC0,STMP4		;GET AC5 DATA
3872	015752	022021				CMP	(R0)+,(R1)+		;QUAD 1 OK?
3873	015754	001011				BNE	1\$;BRANCH IF NO
3874	015756	022021				CMP	(R0)+,(R1)+		;QUAD 0 OK?
3875	015760	001007				BNE	1\$;BRANCH IF NO
3876	015762	172404				LDD	AC4,AC0		;GET AC4
3877	015764	174037	001210			STD	AC0,STMP4		;DATA
3878	015770	022041				CMP	(R0)+,-(R1)		;QUAD 0 OK?
3879	015772	001002				BNE	1\$;BRANCH IF NO
3880	015774	022041				CMP	(R0)+,-(R1)		;QUAD 1 OK?
3881	015776	001421				BEQ	TST44		;BRANCH IF YES
3882	016000	012737	040000	001200	1\$:	MOV	#40000,STMP0		;SAVE EXPECTED
3883	016006	005037	001202			CLR	STMP1		;DATA
3884	016012	010002				MOV	R0,R2		;SAVE R0
3885	016014	162700	001410			SUB	\$\$\$AC5,R0		;FIND OUT WHICH DATA PATTERN
3886	016020	032700	000002			BIT	#BIT1,R0		;ODD OR EVEN?
3887	016024	001001				BNE	2\$;BRANCH IF ODD
3888	016026	005742				TST	-(R2)		;BACK UP THE ADDRESS
3889	016030	011237	001204		2\$:	MOV	(R2),STMP2		;SAVE EXPECTED
3890	016034	011237	001206			MOV	(R2),STMP3		;DATA
3891	016040	104123				ERROR	123		

```

;*****
;*TEST #4      STAO*STQO
;*
;* THE STAO AND STQO INSTRUCTIONS ARE TESTED BY DOING A CMPD
;* INSTRUCTION & TRAPPING OUT AT ROM STATE CMP-25. THIS LEAVES THE
;* SRC & DST DATA IN THE AR AND OR RESPECTIVELY. THE ER WILL CONTAIN
;* THE -ABS VAL OF THE DIFFERENCE IN EXPONENTS.
;*
;* THIS TEST ALSO VERIFIES THAT THE LDAC6*PD(1) FLOW BINGS IN ALL FOUR
;* WORDS.
;*
;* NOTE:  SYNC POINT ONLY SELECTABLE ON STAO AND STQO
;*
;* CMPD FPU ROM FLOW-20,140,270,150,67,370
;* STAO FPU ROM FLOW-30,36
;* STQO FPU ROM FLOW-30,53
;*****

```

3910	016042	000004				TST44:	SCOPE		
3911	016044	012737	000044	001240		MOV	#STN-1,STESTN		;SET TEST NUMBER IN MAIL BOX
3912	016052	012737	016360	001222		MOV	#TST45,SESCAPE		;ESCAPE TO TEST 45 ON ERROR
3913	016060	012737	016166	000244		MOV	#2\$,FPPVEC		

3914	016066	012703	000320		MOV	#320,R3		;PUT POW STATE 320 IN
3915	016072	170003			LDUP			
3916	016074	012700	001700		MOV	#STMP0,R0		;MICRO-BREAK, TO TRAP OUT
3917	016100	012720	077652		MOV	#77652,(R0)+		;PUT DATA TO LOAD INTO QR INTO
3918	016104	012720	125252		MOV	#125252,(R0)+		;MEMORY EXP=377,FRAC=525.....
3919	016110	012720	125252		MOV	#125252,(R0)+		; *
3920	016114	012720	125252		MOV	#125252,(R0)+		; *
3921	016120	012720	017725		MOV	#17725,(R0)+		;PUT DATA TO LOAD INTO AR INTO
3922	016124	012720	052525		MOV	#52525,(R0)+		;MEMORY EXP=77,FRAC=125252.....
3923	016130	012720	052525		MOV	#52525,(R0)+		; *
3924	016134	012720	052525		MOV	#52525,(R0)+		; *
3925	016140	012700	001210		MOV	#STMP4,R0		;SETUP R0 TO CHECK ADX ROM
3926	016144	170127	000700		LDFPS	#FD		
3927	016150	170400			CLRD	ACO		;ENSURE ACO STARTS OUT AS ZERO
3928	016152	172537	001200		LDD	STMP0,AC1		;LOAD THE DST
3929	016156	170127	000220		LDFPS	#FD+FMH		
3930	016162	173520			CMPD	(R0)+,AC1		;DO THE CMPD TO LOAD AR & QR
3931	016164	170000			CPC			;WAIT FOR TRAP
3932	016166	022626		2S:	CMP	(SP)+,(SP)+		;RESTORE THE SP
3933	016170	022700	001220		CMP	#STMP7+2,R0		;AUTO INC OK?
3934	016174	001401			BEQ	15		;BRANCH IF YES
3935	016176	104066			ERROR	66		;ADX ROM FAILED
3936	016200	117703	162732		MOVB	#SMP,R3		
3937	016204	170003		1S:	LDUP			
3938	016206	170007			STQO			;EXECUTE INSTRUCTION UNDER TEST
3939	016210	174037	001160		STD	ACO,\$REG0		;GET DATA BACK
3940	016214	042737	177600	001160	BIC	#177600,\$REG0		;GET RID OF STGN
3941	016222	022737	000052	001160	CMP	#00052,\$REG0		;QUAD 3 OK?
3942	016230	001014			BNE	35		;BRANCH IF NO
3943	016232	022737	125252	001162	CMP	#125252,\$REG1		;QUAD 2 OK?
3944	016240	001010			BNE	35		;BRANCH IF NO
3945	016242	022737	125252	001164	CMP	#125252,\$REG2		;QUAD 1 OK?
3946	016250	001004			BNE	35		;BRANCH IF NO
3947	016252	022737	125252	001166	CMP	#125252,\$REG3		;QUAD 0 OK?
3948	016260	001404			BEQ	45		;BRANCH IF YES
3949	016262	012737	000052	001200	3S: MOV	#00052,STMP0		;SAVE EXPECTED QUAD 3 DATA
3950	016270	104200			ERROR	200		;STQO DID NOT STORE ALL QUAD PROPERLY
3951								
3952	016272	170400			4S: ;NOW CHECK STAO			
3953	016274	170005			CLRD	ACO		;INITIALIZE ACO
3954	016276	174037	001160		STAO			;EXECUTE INSTRUCTION UNDER TEST
3955	016302	042737	177600	001160	STD	ACO,\$REG0		;GET DATA BACK
3956	016310	022737	000125	001160	BIC	#177600,\$REG0		;GET RID OF SIGN
3957	016316	001014			CMP	#00125,\$REG0		;QUAD 3 OK?
3958	016320	022737	052525	001162	BNE	55		;BRANCH IF NO
3959	016326	001010			CMP	#52525,\$REG1		;QUAD 2 OK?
3960	016330	022737	052525	001164	BNE	55		;BRANCH IF NO
3961	016336	001004			CMP	#52525,\$REG2		;QUAD 1 OK?
3962	016340	022737	052525	001166	BNE	55		;BRANCH IF NO
3963	016346	001404			CMP	#52525,\$REG3		;QUAD 0 OK?
3964	016350	012737	000125	001210	5S: BEQ	TST45		;BRANCH IF YES
3965	016356	104177			MOV	#00125,STMP4		;SAVE EXPECTED QUAD 3 DATA
3966					ERROR	177		;STAO DID NOT STORE ALL QUADS PROPERLY
3967								
3968					*****			
3969					;*TEST 45	QR DATA PATH		
					;*			

```

3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3980
3981
3982
3983 016360 000004
3984 016362 012737 000045 001240
3985 016370 012737 017156 001222
3986 016376 012737 040000 001160
3987 016404 005037 001162
3988 016410 005037 001164
3989 016414 012737 000001 001166
3990 016422 005037 001200
3991 016426 005037 001202
3992 016432 005037 001204
3993 016436 012737 000001 001206
3994 016444 012737 016510 000244
3995 016452 012700 000067
3996 016456 012737 016464 001110
3997 016464 012703 000067
3998 016470 170003
3999 016472 170127 000220
4000 016476 172437 001160
4001 016502 173427 050000
4002 016506 170000
4003 016510 022626
4004 016512 170127 000200
4005 016516 117703 162414
4006 016522 170003
4007 016524 170007
4008 016526 174037 001210
4009 016532 042737 177600 001210
4010 016540 012701 001200
4011 016544 012702 001210
4012 016550 022122
4013 016552 001037
4014 016554 022122
4015 016556 001035
4016 016560 022122
4017 016562 001033
4018 016564 021112
4019 016566 001031
4020 016570 032777 001000 162340
4021 016576 001403
4022 016600 105737 001103
4023 016604 001327
4024 016606 006137 001166
4025 016612 006137 001164
  
```

```

;* THIS TEST ENSURES THAT THERE ARE NO BITS STUCK
;* IN THE QM AND QR. THIS IS DONE BY EXECUTING A
;* CMPD INSTRUCTION AND TRAPPING ON STATE 67 WHICH
;* LEAVES THE DST ACC IN THE QR. A STQO IS THEN
;* EXECUTED TO GET THE QR DATA BACK. THE DATA IS
;* COMPOSED OF A FLOATING "ONE" AND THEN A FLOATING "ZERO".
;*
;* THE ONLY SYNC POINT AVAILABLE DURING THE CMPD IS
;* STATE 67. THE QR GETS LOADED IN THIS STATE
;*
;* NOTE: SYNC POINT ONLY SELECTABLE ON STQO INSTRUCTION.
;*)*****
;*)*****
TST45: SCOPE
MOV #STN-1,STESTM ;;SET TEST NUMBER IN MAIL BOX
MOV #TST46,SESCAPE ;;ESCAPE TO TEST 46 ON ERROR
MOV #40000,SREG0 ;INITIALIZE
CLR SREG1 ;DATA TO LOAD
CLR SREG2 ;INTO THE QR
MOV #1,SREG3 ;
CLR STMP0 ;INITIALIZE
CLR STMP1 ;CHECK
CLR STMP2 ;DATA
MOV #1,STMP3 ;
MOV #25,FPPVEC
MOV #67,R0 ;SET LOOP COUNT
MOV #.+6,#SLPERR ;SET ERROR LOOP
5S: MOV #67,R3
LDUB
LDFPS #FD+FMH
LDD SREG0,ACO ;LOAD THE DST
1S: CMPD #050000,ACO ;LOAD THE QR
CFCC ;WAIT FOR TRAP
2S: CMP (SP)+,(SP)+ ;RESTORE THE SP
LDFPS #FD ;TURN OFF FMH
MOV #SWR,R3
LDUB
STQO
STD ACO,STMP4 ;GET THE QR
BIC #177600,STMP4 ;GET RID OF SIGN AND EXPONENT
MOV #STMP0,R1
MOV #STMP4,R2
CMP (R1)+,(R2)+ ;QUAD 3 OK?
BNE 3S ;BRANCH IF NO
CMP (R1)+,(R2)+ ;QUAD 2 OK?
BNE 3S ;BRANCH IF NO
CMP (R1)+,(R2)+ ;QUAD 1 OK?
BNE 3S ;BRANCH IF NO
CMP (R1),(R2) ;QUAD 0 OK?
BNE 3S ;BRANCH IF NO
BIT #SW9,#SWR ;LOOP ON ERROR?
BEG 4S ;BRANCH IF NO
TSTR SEFLG ;ANY ERRORS YET
BNE 5S ;BRANCH IF YES
4S: ROL SREG3 ;SELECT NEXT
ROL SREG2 ;DATA PATTERN
  
```

4026	016616	006137	001162		ROL	\$REG1	
4027	016622	106137	001160		ROLP	\$REG0	; * ;SELECT NEXT
4028	016626	006137	001206		ROL	\$TMP3	;CHECK PATTERN
4029	016632	006137	001204		ROL	\$TMP2	
4030	016636	006137	001202		ROL	\$TMP1	
4031	016642	006137	001200		ROL	\$TMP0	
4032	016646	077072			SOB	#0,\$S	;CONTINUE
4033	016650	000401			BR	65	
4034	016652	104264			35: ERROR	264	;BIT STUCK IN QR DATA PATH
4035					;*****		
4036					;NOW FLOAT A ZERO		
4037	016654	012737	040177	001160	65: MOV	#40177,\$REG0	;INITIALIZE
4038	016662	012737	177777	001162	MOV	#-1,\$REG1	;DATA
4039	016670	012737	177777	001164	MOV	#-1,\$REG2	;TO LOAD
4040	016676	012737	177776	001166	MOV	#-2,\$REG3	;INTO QR
4041	016704	012737	000177	001200	MOV	#177,\$TMP0	;INITIALIZE
4042	016712	012737	177777	001202	MOV	#-1,\$TMP1	;CHECK
4043	016720	012737	177777	001204	MOV	#-1,\$TMP2	;DATA
4044	016726	012737	177776	001206	MOV	#-2,\$TMP3	; *
4045	016734	012737	017000	000244	MOV	#75,\$PPVEC	
4046	016742	012700	000067		MOV	#67,R0	;SET LOOP COUNT
4047	016746	012737	016754	001110	MOV	#,+6,\$SLPERR	;SET ERROR LOOP
4048	016754	012703	000320		85: MOV	#320,R3	
4049	016760	170003			LDUR		
4050	016762	170127	000220		LDFPS	#FD+\$PM	
4051	016766	172437	001160		LDD	\$REG0,AC0	;LOAD THE DST
4052	016772	173427	050000		CMPD	#*050000,AC0	;LOAD THE QR
4053	016776	170000			CFCC		;WAIT FOR TRAP
4054	017000	022626			75: CMP	(SP)+,(SP)+	;RESTORE THE SP
4055	017002	117703	162130		MOV	#SWR,R3	
4056	017006	170003			LDUB		
4057	017010	170127	000200		LDFPS	#FD	
4058	017014	170007			STQ0		
4059	017016	174037	001210		STD	AC0,\$TMP4	;GET QR DATA
4060	017022	042737	177600	001210	BYC	#177600,\$TMP4	;GET PID OF SIGN AND EXP
4061	017030	012701	001200		MOV	#TMP0,R1	
4062	017034	012702	001210		MOV	#TMP4,R2	
4063	017040	022122			CMP	(R1)+,(R2)+	;QUAD 3 OK?
4064	017042	001044			BNE	95	;BRANCH IF NO
4065	017044	022122			CMP	(R1)+,(R2)+	;QUAD 2 OK?
4066	017046	001042			BNE	95	;BRANCH IF NO
4067	017050	022122			CMP	(R1)+,(R2)+	;QUAD 1 OK?
4068	017052	001040			BNE	95	;BRANCH IF NO
4069	017054	021112			CMP	(R1),(R2)	;QUAD 0 OK?
4070	017056	001036			BNE	95	;BRANCH IF NO
4071	017060	032777	001000	162050	BIT	#SW9,\$SWR	;LOOP ON ERROR?
4072	017066	001403			BEQ	105	;BRANCH IF NO
4073	017070	105737	001103		TSTR	\$ENFLG	;ANY ERRORS YET?
4074	017074	001327			BNE	85	;BRANCH IF YES
4075	017076	000261			105: SEC		
4076	017100	006137	001166		ROL	\$REG3	;SELECT NEXT
4077	017104	006137	001164		ROL	\$REG2	;DATA PATTERN
4078	017110	006137	001162		ROL	\$REG1	; *
4079	017114	106137	001160		ROLP	\$REG0	; *
4080	017120	000261			SEC		
4081	017122	006137	001206		ROL	\$TMP3	;SELECT NEXT

4082 017126 006137 001204
 4083 017132 006137 001202
 4084 017136 106137 001200
 4085 017147 042737 177600 001200
 4086 017150 077077
 4087 017152 000401
 4088 017154 104264

ROL STMP2 ;CHECK PATTERN
 ROL STMP1 ; *
 ROLR STMP0 ; *
 BIC #177600,STMP0 ; *
 SOB R0,R5 ;CONTINUE
 BP TST46 ;;GO TO NEXT TEST
 95: ERROR 264 ;BIT STUCK IN QR DATA PATH

4089
 4090
 4091
 4092
 4093
 4094
 4095
 4096
 4097
 4098
 4099
 4100
 4101

;;*****
 ;*TEST 46 DISABLE LOW FMX
 ;*
 ;* THIS TEST INSURES THAT PRHD AND FRLC DISAPLE LOW FMX
 ;* AND PRHD DISABLE POUND FMX AND FRLC DISAPLE FMX19
 ;* ARE NOT STUCK LOW. THIS IS DONE BY MOVING ALL ONE'S
 ;* FROM AC1 TO AC0 AND TRAPPING ON STATE 65 AND EXAMINING
 ;* THE AR.
 ;*
 ;* NOTE: SYNC POINT NOT SELECTABLE. DEFAULT=65
 ;*
 ;;*****

4102 017156 000004
 4103 017160 012737 000046 001240
 4104 017166 012737 017350 001222
 4105 017174 012737 017312 000244
 4106 017202 012737 040377 001160
 4107 017210 012737 177777 001162
 4108 017216 012737 177777 001164
 4109 017224 012737 177777 001166
 4110 017232 012737 000177 001200
 4111 017240 012737 177777 001202
 4112 017246 005037 001204
 4113 017257 005037 001206
 4114 017256 012737 017264 001110
 4115 017264 170011
 4116 017266 172537 001160
 4117 017272 012703 000065
 4118 017276 170003
 4119 017300 170400
 4120 017302 170127 000020
 4121 017306 172100
 4122 017310 170000
 4123 017312 022626
 4124 017314 170011
 4125 017316 170005
 4126 017320 174037 001210
 4127 017324 042737 177600 001210
 4128 017332 005737 001714
 4129 017336 001003
 4130 017340 005737 001216
 4131 017344 001401
 4132 017346 104270

TST46: SCOPE
 MOV #STN-1,STESTN ;;SET TEST NUMBER IN MAIL BOX
 MOV #TST47,SESCAPE ;;ESCAPE TO TEST 47 ON ERROR
 MOV #15,PPPVEC
 MOV #40377,SREG0 ;PUT DATA IN
 MOV #-1,SREG1 ;MEMORY TO
 MOV #-1,SREG2 ;INIT AC1
 MOV #-1,SREG3 ; *
 MOV #177,STMP0 ;SAVE EXPECTED
 MOV #-1,STMP1 ;VALUE OF RESULT
 CLR STMP2 ; *
 CLR STMP3 ; *
 MOV #.+6,#SLPERR ;SET ERROR LOOP
 SETD
 LDD SREG0,AC1 ;PUT DATA IN AC1
 MOV #65,R3
 LDUB
 CLRD AC0 ;INIT AC0
 LDPPS #PMW
 ADDF AC0,AC1 ;MOVE THE ONE'S AND TRAP
 CFCC ;WAIT FOR TRAP
 15: CMP (SP)+,(SP)+ ;RESTORE THE SP
 SETD
 STAO
 STD AC0,STMP4 ;GET THE AR DATA
 BIC #177600,STMP4 ;GET RID OF EXPONENT
 TST STMP6 ;QUAD 1 OK?
 BNE 25 ;BRANCH IF NO
 TST STMP7 ;QUAD 0 OK?
 BEQ TST47 ;;BRANCH IF YES
 25: EPROR 270 ;FRHD DISABLE LOW FMX OR DISAPLE
 ;POUND FMX NOT GOING HIGH

4133
 4134
 4135
 4136
 4137

;;*****
 ;*TEST 47 DISABLE LOW QMX


```

4194 017626 012737 177777 001166
4195 017634 012737 017642 001110
4196 017642 170011
4197 017644 172437 001160
4198 017650 170401
4199 017652 170001
4200 017654 172500
4201 017656 170011
4202 017660 174137 001210
4203 017664 005737 001214
4204 017670 001003
4205 017672 005737 001216
4206 017676 001401
4207 017700 104266
4208
4209
4210
4211
4212
4213
4214
4215
4216
4217
4218
4219 017702 000004
4220 017704 012737 000051 001240
4221 017712 012737 020070 001222
4222 017720 012700 001200
4223 017724 012701 020100
4224 017730 010120
4225 017732 010120
4226 017734 010120
4227 017736 010120
4228 017740 012737 017746 001110
4229 017746 170127 000200
4230 017752 172437 001200
4231 017756 170127 000717
4232 017762 172427
4233 017764 104066
4234 017766 000403
4235 017770 104066
4236 017772 104066
4237 017774 104066
4238 017776 170737 001200
4239 020002 174037 001210
4240 020006 012701 001210
4241 020012 022721 104066
4242 020016 001006
4243 020020 005721
4244 020022 001004
4245 020024 005721
4246 020026 001002
4247 020030 005721
4248 020032 001406
4249 020034 005040
  
```

```

MOV      #-1,$REG3      ;TEST
MOV      #-+6,@$SLPERR ;SET ERROR LOOP
SFTD
LDD      $PEGO,ACO      ;PUT DATA IN ACO
CLR      AC1            ;INSURE AC1 CLEAR
SETF
LDF      ACO,AC1        ;EXEC INSTR TO TEST FUNCTION
SETD
STD      AC1,$TMP4      ;GET RESULT
TST      $TMP6          ;QUAD 1 OK?
BNE      1$            ;BRANCH IF NO
TST      $TMP7          ;QUAD 0 OK?
BEQ      TST51         ;;BRANCH IF YES
1$:      ERROR          266 ;FRMH WRITE ACD1 OR WRITE ACD0
                          ;NOT GOING HIGH WITH PD(1) LOW

;;*****
;*TEST 51      LDD*-M0*IMM
;*
;*      THE A AND IMMEDIATE BRANCHES SHOULD NOT FAIL.
;*      IF THE ADX ROM FAILS, THE PC WILL AUTO INCREMENT
;*      BY SOMETHING OTHER THAN 2.
;*
;*      FPU FOM FLOW - 21, 143, 16
;;*****
TST51:  SCOPE
MOV      #$TN-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX
MOV      #TST52,$ESCAPE ;;ESCAPE TO TEST 52 ON ERROR
MOV      #STMPO,P0      ;GET ADDRESS OF BUFFER
MOV      #20100,R1      ;WAKE BUFFER
MOV      R1,(R0)+      ;NON-ZERO
MOV      R1,(R0)+      ;
MOV      R1,(R0)+      ;
MOV      R1,(R0)+      ;
MOV      #-+6,@$SLPERR ;SET ERROR LOOP
LDFPS   #PD             ;ENSURE PD BIT SET
LDD      $TMP0,ACO      ;ENSURE ACO NON-ZERO
LDFPS   #PD+17         ;SET CC'S TO COMPLEMENT OF EXPECTED
1$:      LDD            (PC)+,ACO ;EXECUTE INSTRUCTION UNDER TEST
          ERROR          66      ;ADX ROM FAILED
          BR             2$
          ERROR          66      ;ADX ROM FAILED
          ERROR          66      ;ADX ROM FAILED
          ERROR          66      ;ADX ROM FAILED
2$:      STFPS          $TMP0    ;GET CC'S BACK
          STD            ACO,$TMP4 ;GET DATA BACK
          MOV           #STMPO,P1 ;GET ADDRESS OF DATA
          CMP           #104066,(R1)+ ;QUAD 3 OK?
          BNE          3$      ;BRANCH IF NO
          TST          (R1)+   ;QUAD 2 OK?
          BNE          3$      ;BRANCH IF NO
          TST          (R1)+   ;QUAD 1 OK?
          BNE          3$      ;BRANCH IF NO
          TST          (R1)+   ;QUAD 0 OK?
          BEQ          4$      ;BRANCH IF YES
3$:      CLR            -(R0)    ;SAVE
  
```

```

4250 020036 005040          CLR      -(R0)          ;EXPECTED
4251 020040 005040          CLR      -(R0)          ;VALUE OF DATA
4252 020042 012740 104066  MOV      #104066, -(R0)
4253 020046 104124          ERROR    124          ;ROM STATE 143 FAILED
4254                                ;CHECK THE CC'S
4255 020050 022737 000210 001200 45:  CMP      #FD+FN, STMP0    ;CC'S OK?
4256 020056 001404          BEQ      TST52          ;;BRANCH IF YES
4257 020060 012737 000210 001202  MOV      #FD+FN, STMP1    ;SAVE EXPECTED VALUE
4258 020066 104071          ERROR    71          ;CC'S BAD ON LDD*-NO*IMM
4259
4260
4261                                ;;*****
4262                                ;*TEST 52          LDCPD*NO*-(EXP=0)
4263                                ;*
4264                                ;*          NONE OF THE MICRO BRANCHES SHOULD FAIL.
4265                                ;*          IF FRMH CLR AR<34:00> DOES NOT GO LOW, OR GET TO ALL THE
4266                                ;*          CHIPS OF THE LOWER AR, THE AR WON'T CLEAN.
4267                                ;*
4268                                ;*          FPU ROM FLOW - 30, 46, 136
4269                                ;;*****
4270 020070 000004          TST52:  SCOPE
4271 020072 012737 000052 001240  MOV      #STN-1, STSTN    ;;SET TEST NUMBER IN MAIL BOX
4272 020100 012737 020250 001222  MOV      #TST53, SESCPE   ;;ESCAPE TO TEST 53 ON ERROR
4273 020106 012700 001200          MOV      #STMP0, R0       ;GET BUFFER ADDRESS
4274 020112 012720 040200          MOV      #40200, (R0)+    ;LOAD BUFFER
4275 020116 012720 177777          MOV      #-1, (R0)+      ;WITH A NON ZERO
4276 020122 012720 177777          MOV      #-1, (R0)+      ;FLOATING POINT
4277 020126 012720 177777          MOV      #-1, (R0)+      ;NUMBER
4278 020132 012737 020140 001110  MOV      #-6, #SLPERR    ;SET ERROR LOOP
4279 020140 170127 000200          LDFPS   #FD             ;ENSURE FD BIT SET
4280 020144 172437 001200          LDD     STMP0, ACO       ;LOAD THE SRC
4281 020150 172537 001200          LDD     STMP0, AC1       ;& DST ACC'S WITH THE WORD
4282 020154 170127 000217          LDFPS   #FD+17         ;LOAD COMPLEMENT CC'S
4283 020160 177401          15:  LDCPD   AC1, ACO         ;EXECUTE INSTRUCTION UNDER TEST
4284 020162 170203          STFPS   R3             ;GET CC'S BACK
4285 020164 174037 001210          STD     ACO, STMP4       ;GET DATA BACK
4286 020170 012701 001216          MOV     #STMP7, P1       ;GET ADDRESS OF DATA
4287 020174 005040          CLR     -(R0)           ;SAVE EXPECTED DATA
4288 020176 021011          CMP     (R0), (R1)       ;QUAD 0 OK?
4289 020200 001007          BNE     25              ;BRANCH IF NO
4290 020202 005040          CLR     -(R0)           ;SAVE EXPECTED VALUE
4291 020204 021041          CMP     (R0), -(R1)      ;QUAD 1 OK?
4292 020206 001004          BNE     25              ;BRANCH IF NO
4293 020210 024041          CMP     -(R0), -(R1)     ;QUAD 2 OK?
4294 020212 001002          BNE     25              ;BRANCH IF NO
4295 020214 024041          CMP     -(R0), -(R1)     ;QUAD 3 OK?
4296 020216 001403          BEQ     35              ;BRANCH IF YES
4297 020220 005037 001204          25:  CLR     STMP2         ;ENSURE QUAD 1 DATA IS SAVED
4298 020224 104125          ERROR   125           ;ONE OF THE QUADRANTS FAILED
4299                                ;CHECK THE CC'S
4300 020226 022703 000200          35:  CMP     #FD, R3        ;CC'S OK?
4301 020232 001406          BEQ     TST53          ;;BRANCH IF YES
4302 020234 010337 001200          MOV     R3, STMP0       ;SAVE RECEIVED DATA
4303 020240 012737 000200 001202  MOV     #FD, STMP1       ;SAVE EXPECTED DATA
4304 020246 104071          ERROR   71          ;CC'S BAD
4305

```

4306
 4307
 4308
 4309
 4310
 4311
 4312
 4313
 4314 020250 000004
 4315 020252 012737 000053 001240
 4316 020260 012737 020422 001222
 4317 020266 012700 001200
 4318 020272 012720 000177
 4319 020276 012720 177777
 4320 020302 012720 177777
 4321 020306 012720 177777
 4322 020312 170127 000200
 4323 020316 172437 001200
 4324 020322 172537 001200
 4325 020326 170127 000213
 4326 020332 177401
 4327 020334 170703
 4328 020336 174037 001210
 4329 020342 005040
 4330 020344 005040
 4331 020346 005040
 4332 020350 005040
 4333 020352 012701 001210
 4334 020356 022021
 4335 020360 001006
 4336 020362 022021
 4337 020364 001004
 4338 020366 022021
 4339 020370 001002
 4340 020372 021011
 4341 020374 001401
 4342 020376 104125
 4343
 4344 020400 022703 000204
 4345 020404 001406
 4346 020406 010337 001200
 4347 020412 012737 000204 001202
 4348 020420 104071
 4349
 4350
 4351
 4352
 4353
 4354
 4355
 4356
 4357
 4358 020422 000004
 4359 020424 012737 000054 001740
 4360 020432 012737 020562 001222
 4361 020440 012700 001200

```

;*****
;*TEST 53      LDCFD*MO*(EXP=0)
;*
;*      THE ONLY THING THAT SHOULD FAIL IS ROM STATE 276.
;*
;*      FPU ROM FLOW - 30, 46, 136, 276
;*****
TST53:  SCOPE
        MOV      #STN-1,STESTN      ;;SET TEST NUMBER IN MAIL BOX
        MOV      #TST54,SESCAPE    ;;ESCAPE TO TEST 54 ON ERROR
        MOV      #STMP0,R0         ;GET ADDRESS OF BUFFER
        MOV      #177,(R0)+        ;LOAD BUFFER
        MOV      #-1,(R0)+         ;WITH ZERO EXP
        MOV      #-1,(R0)+         ;AND NON-ZERO
        MOV      #-1,(R0)+         ;FRACTION
        LDFPS    #FD                ;ENSURE FD BIT SET
        LDD      STMP0,AC0          ;LOAD SRC &
        LDD      STMP0,AC1          ;DST WITH EXP=0, FRAC=NON-ZERO
        LDFPS    #FD+13            ;LOAD COMPLEMENT CC'S
15:     LDCFD    AC1,AC0            ;EXECUTE INSTRUCTION UNDER TEST
        STFPS    R?                ;SAVE CC'S
        STD      AC0,STMP4          ;GET DATA BACK
        CLR      -(R0)              ;SAVE EXPECTED VALUES
        CLR      -(R0)
        CLR      -(R0)
        CLR      -(R0)
        MOV      #STMP4,R1         ;GET ADDRESS OF RECEIVED DATA
        CMP      (R0)+,(R1)+       ;QUAD 3 OK?
        BNE     3$                 ;BRANCH IF NO
        CMP      (R0)+,(R1)+       ;QUAD 2 OK?
        BNE     3$                 ;BRANCH IF NO
        CMP      (R0)+,(R1)+       ;QUAD 1 OK?
        BNE     3$                 ;BRANCH IF NO
        CMP      (R0),(R1)         ;QUAD 0 OK?
        BEQ     2$                 ;BRANCH IF YES
3$:     ERROR    175                ;A QUAD IS BAD
;CHECK CC'S
2$:     CMP      #FD+4,R3          ;CC'S OK?
        BEQ     TST54              ;;BRANCH IF YES
        MOV      R3,STMP0          ;SAVE RECEIVED VALUE
        MOV      #FD+4,STMP1       ;SAVE EXPECTED VALUE
        EPRDR   71                 ;CC'S BAD

```

```

;*****
;*TEST 54      LDCFD*MO*(EXP=0)
;*
;*      THE ONLY THING THAT SHOULD FAIL IS ROM STATE 230.
;*
;*      FPU ROM FLOW - 30, 46, 137, 270
;*****
TST54:  SCOPE
        MOV      #STN-1,STFSTN    ;;SET TEST NUMBER IN MAIL BOX
        MOV      #TST55,SESCAPE    ;;ESCAPE TO TEST 55 ON ERROR
        MOV      #STMP0,R0         ;GET ADDRESS OF BUFFER

```

```

4362 020444 012720 000177
4363 020450 012720 177777
4364 020454 012720 177777
4365 020460 012720 177777
4366 020464 170127 000200
4367 020470 172437 001200
4368 020474 172537 001200
4369 020500 170127 000013
4370 020504 177401
4371 020506 170203
4372 020510 174037 001204
4373 020514 024040
4374 020516 005040
4375 020520 005040
4376 020522 022037 001204
4377 020526 001003
4378 020530 021037 001206
4379 020534 001401
4380 020536 104035
4381
4382 020540 022703 000004
4383 020544 001406
4384 020546 010337 001200
4385 020552 012737 000004 001202
4386 020560 104071
4387
4388
4389
4390
4391
4392
4393
4394
4395
4396
4397
4398
4399
4400
4401 020562 000004
4402 020564 012737 000055 001740
4403 020572 012737 021030 001222
4404 020600 012737 040377 001160
4405 020606 012737 177777 001162
4406 020614 012737 177777 001164
4407 020622 012737 177777 001166
4408 020630 005037 001700
4409 020634 012737 000001 001202
4410 020642 005037 001204
4411 020646 005037 001206
4412 020652 012737 020706 000244
4413 020660 012703 000067
4414 020664 170003
4415 020666 170011
4416 020670 172537 001160
4417 020674 170127 000220
  
```

```

NOV #177,(R0)+ ;WAKE BUFFER
NOV #-1,(R0)+ ;EXP=0
NOV #-1,(R0)+ ;AND
NOV #-1,(R0)+ ;FRAC=NON-ZERO
LDFPS #FD ;ENSURE FD BIT SET
LDD $TMP0,AC0 ;PUT DATA INTO
LDD $TMP0,AC1 ;SRC AND DST ACC'S
LDFPS #13 ;LOAD COMPLEMENT CC'S
15: LDCDF AC1,AC0 ;EXECUTE INSTRUCTION UNDER TEST
STFPS R3 ;SAVE CC'S
STF AC0,$TMP2 ;GET DATA BACK
CMP -(R0),-(R0) ;READJUST R0
CLR -(R0) ;SAVE EXPECTED
CLR -(R0) ;VALUE OF DATA
CMP (R0)+,$TMP2 ;QUAD 3 OK?
BNE 2$ ;BRANCH IF NO
CMP (R0),$TMP3 ;QUAD 2 OK?
BEQ 3$ ;BRANCH IF YES
25: ERROR 35
;CHECK CC'S
35: CMP #4,P3 ;CC'S OK?
BEQ TST55 ;BRANCH IF YES
NOV R3,$TMP0 ;SAVE RECEIVED VALUE
NOV #4,$TMP1 ;SAVE EXPECTED VALUE
ERROR 71 ;CC'S BAD
  
```

```

;;*****
;*TEST 55 DISABLE LOW & HI FMX ON ROUND
;*
;* THIS TEST INSURES THAT FRHD DISABLE LOW FMX AND DISABLE
;* HI FMX AND FRLC DISABLE LOW FMX GO HIGH WHEN ROUND IS
;* ENABLED. THIS IS DONE BY LOADING THE QR WITH ALL ONE'S
;* AND THEN EXECUTING A LDCDF, TRAPPING ON STATE 137, AND
;* EXAMINING THE AR.
;*
;* NOTE: SYNC POINT NOT SELECTABLE. DEFAULT=67
;*
;;*****
  
```

```

TST55: SCOPE
NOV #STN-1,$TSTN ;SET TEST NUMBER IN MAIL BOX
NOV #TST56,$ESCAPE ;ESCAPE TO TEST 56 ON ERROR
NOV #40377,$REG0 ;PUT DATA TO LOAD
NOV #-1,$REG1 ;INTC QR INTO MEMCRV
NOV #-1,$REG2 ;
NOV #-1,$REG3 ;
CLR $TMP0 ;SAVE EXPECTED
NOV #1,$TMP1 ;RESULT
CLR $TMP2 ;
CLR $TMP3 ;
NOV #1$,FPPVEC
NOV #67,R3
LDUP
SETD
LDD $PEGO,AC1
LDFPS #FD+FMH
  
```

```

4418 020700 173527 040200
4419 020704 170000
4420 020706 022626
4421 020710 012737 020744 000244
4422 020716 012737 020724 001110
4423 020724 012703 000137
4424 020730 170003
4425 020732 170127 000020
4426 020736 177527 040200
4427 020742 170000
4428 020744 022626
4429 020746 012704 000001
4430 020752 170004
4431 020754 170005
4432 020756 170011
4433 020760 174037 001210
4434 020764 042737 177600 001210
4435 020772 005737 001214
4436 020776 001003
4437 021000 005737 001216
4438 021004 001401
4439 021006 104272
4440
4441
4442 021010 005737 001210
4443 021014 001004
4444 021016 022737 000001 001212
4445 021024 001401
4446 021026 104273
4447
4448
4449
4450
4451
4452
4453
4454
4455
4456
4457
4458
4459
4460
4461
4462
4463 021030 000004
4464 021032 012737 000056 001240
4465 021040 012737 021320 001222
4466 021046 012737 021224 000244
4467 021054 170127 000200
4468 021060 172527 040200
4469 021064 170127 000000
4470 021070 012703 000161
4471 021074 170003
4472 021076 170127 000037
4473 021102 012737 021110 001110
  
```

```

CMPD #1,AC1 ;EXEC INSTR TO LOAD QR
CFCC ;WAIT FOR TRAP
1S: CMP (SP)+,(SP)+ ;RESTORE THE SP
MOV #25,PPPVEC
MOV #.+6,#SLPERR ;SET ERROR LOOP
MOV #137,R3
LDUR
LDFPS #FMW
LDCDF #1,AC1 ;EXFC INSTR TO CAUSE ROUND
CFCC ;WAIT FOR TRAP
2S: CMP (SP)+,(SP)+ ;RESTORE THE SP
MOV #1,R4 ;PUT THE SHIFT COUNT IN R4
MSN ;GET AR BIT 2
STAO
SETD
STD ACO,$TMP4 ;GET AR DATA
BIC #177600,$TMP4 ;GET RID OF EXPONENT
TST $TMP6 ;QUAD 1 OK?
BNE 3S ;BRANCH IF NO
TST $TMP7 ;QUAD 0 OK?
BEQ 4S ;BRANCH IF YES
3S: ERROR 272 ;EITHER FRHD OR FRLC DISABLE LOW
;FMX NOT GOING HIGH OR FRLC FMX02
;NOT GOING HIGH.
4S: TST $TMP4 ;QUAD 3 OK?
BNE 5S ;BRANCH IF NO
CMP #1,$TMP5 ;QUAD 2 OK?
BEQ TST56 ;BRANCH IF YES
5S: ERROR 273 ;EITHER FRHD DISABLE HI FMX OR
;FMX34 NOT GOING HIGH
  
```

```

;*****
;*TEST 56 LDCDF*NO*-(EXP=0)*-AR59*-BOU
;*
;* THE ONLY THING THAT SHOULD FAIL IS THE 5F7J BRANCH.
;* IF FRHE AR59(1)L DOES NOT GO HIGH OR DOFS NOT GET TO FRMA RAD01 AS A LCM,
;* EXECUTION WILL GO TO STATE 161.
;* A MICRO-TRAP WILL BE SET TO CATCH THIS FAILURE.
;* IF FRMA BOU+BZ DOES NOT GO HIGH, EXECUTION WILL GO TO STATE 162.
;* THIS WILL CAUSE THE DESTINATION TO BE CLEARED.
;*
;* IF FRHC FMX 35 DOES NOT GO LOW, THE DESTINATION WILL HAVE A "1" IN THE LSB.
;*
;* FPU ROM FLOW - 30, 46, 137, 231
;*****
  
```

```

TST56: SCOPE
MOV #STN-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX
MOV #TST57,$ESCAPE ;;ESCAPE TO TEST 57 ON ERROR
MOV #25,PPPVEC ;SETUP TRAP VECTOR
LDFPS #FD ;ENSURE FD BIT SET
LDD #*040200,AC1 ;LOAD THE SRC ACC
LDFPS #0 ;GO BACK TO F FORMAT
MOV #161,P3 ;LOAD MICPC-BREAK
LDUR ;WITH ROM ADR 161.
LDFPS #FMW+17 ;LOAD COMPLIMENT CC'S
MOV #.+6,#SLPERR ;SET ERROR LOOP
  
```

```

4474 021110 012737 150377 001200      MOV      #150377,STMP0      ;PUT DATA IN MEMORY FOR DST
4475 021116 012737 177777 001202      MOV      #-1,STMP1
4476 021124 172437 001200              LDF      STMP0,AC0        ;MAKE SURE DST IS NON ZERO
4477 021130 177401              LDCDF    AC1,AC0        ;EXECUTE INSTRUCTION UNDER TEST
4478 021132 170204              STFPS    R4              ;GET CC'S BACK
4479 021134 174037 001204              STF      AC0,STMP2       ;GET DATA BACK
4480 021140 022737 040200 001204      CMP      #40200,STMP2    ;QUAD 3 OK?
4481 021146 001010              BNE      35              ;BRANCH IF NO
4482 021150 005737 001206              TST      STMP3           ;QUAD 2 OK?
4483 021154 001413              BEQ      45              ;BRANCH IF YES
4484 021156 022737 000001 001206      CMP      #1,STMP3       ;DID PMX BIT 35 COME THRU?
4485 021164 001001              BNE      35              ;BRANCH IF NO
4486 021166 104126              ERROR    126            ;PRHC PMX 35 STUCK HIGH
4487 021170 012737 040200 001200 35:    MOV      #40200,STMP0    ;SAVE
4488 021176 005037 001202              CLR      STMP1          ;EXPECTED DATA
4489 021202 104035              ERROR    35            ;DATA WRONG
4490                                ;CHECK THE CC'S
4491 021204 042704 000020 45:    BIC      #PMX,R4        ;GET RID OF PMX BIT IN SAVED STATUS
4492 021210 001411              BEQ      55              ;BRANCH IF CC'S OK
4493 021212 010437 001200              MOV      R4,STMP0       ;SAVE RECEIVED CC'S
4494 021216 005037 001202              CLR      STMP1          ;SAVE EXPECTED CC'S
4495 021222 104071              ERROR    71            ;CC'S BAD
4496                                ;AR59*BOU BRANCH FAILED
4497 021224 022626 25:    CMP      (SP)+,(SP)+    ;RESTORE THE SP
4498 021226 170127 000000              LDFPS    #0              ;CLEAR THE PMX BIT
4499 021232 104127              ERROR    127          ;PRHC AR59(1)L NOT GOING HIGH
4500                                ;*****
4501                                ;CHECK PRHC PMX 34
4502                                55:
4503 021234 012737 021247 001110      MOV      #.+6,#$SLPERR  ;SET PRPOR LOOP
4504 021242 170127 000200              LDFPS    #FD            ;CLEAR PMX & SET D FORMAT
4505 021246 005037 001202              CLR      STMP1          ;DATA
4506 021252 012737 100000 001204      MOV      #100000,STMP2  ;IN
4507 021260 005037 001206              CLR      STMP3          ;MEMORY
4508 021264 172537 001200              LDD      STMP0,AC1      ;LOAD DATA WITH ACC BIT 34 ON.
4509 021270 170127 000000              LDFPS    #0              ;CLEAR FD BIT
4510 021274 177401              LDCDF    AC1,AC0        ;EXECUTE INSTRUCTION
4511 021276 174037 001210              STF      AC0,STMP4       ;GET DATA
4512 021302 022737 000001 001212      CMP      #1,STMP5       ;QUAD 2 OK?
4513 021310 001403              BEQ      TST57          ;BRANCH IF YES
4514 021312 005037 001204              CLR      STMP2          ;PRHC PMX 34 NOT GOING HIGH OR PALU BAD
4515 021316 104130              ERROR    136
4516
4517
4518                                ;*****
4519                                ;*TEST 57      LDCDF*-NO*INH*-(EXP=0)
4520                                ;*
4521                                ;*      THE ONLY POSSIBLE FAILURES ARE ROM STATES 17 OR 221 OR THE ADX ROM.
4522                                ;*
4523                                ;*      FPU ROM FLOW - 17, 221, 136
4524                                ;*****
4525 021320 000004      TST57:  SCOPE
4526 021322 012737 000057 001240      MOV      #STN-1,STESTN  ;SET TEST NUMBER IN MAIL BOX
4527 021330 012737 021442 001222      MOV      #TST60,SESCAPE ;ESCAPE TO TEST 60 ON ERROR
4528 021336 012737 047624 000244      MOV      #PPSPUR,FPPVEC ;PESTORE FP VECTCH
4529 021344 012737 177777 001200      MOV      #-1,STMP0      ;PUT DATA TO

```


4530 021352 012737 177777 001202
 4531 021360 172437 001200
 4532 021364 170127 000200
 4533 021370 177427
 4534 021372 104066
 4535 021374 000403
 4536 021376 104066
 4537 021400 104066
 4538 021402 104066
 4539 021404 174037 001204
 4540 021410 022737 104066 001204
 4541 021416 001003
 4542 021420 005737 001206
 4543 021424 001406
 4544 021426 012737 104066 001200
 4545 021434 005037 001202
 4546 021440 104035
 4547
 4548
 4549
 4550
 4551
 4552
 4553
 4554
 4555
 4556 021442 000004
 4557 021444 012737 000060 001240
 4558 021452 012737 021602 001222
 4559 021460 012737 177777 001200
 4560 021466 012737 177777 001202
 4561 021474 172437 001200
 4562 021500 012737 040200 001200
 4563 021506 012737 125252 001202
 4564 021514 012700 001200
 4565 021520 170127 000200
 4566 021524 177420
 4567 021526 022700 001204
 4568 021532 001401
 4569 021534 104066
 4570 021536 174037 001210
 4571 021542 022737 040200 001210
 4572 021550 001007
 4573 021552 022737 125252 001212
 4574 021560 001003
 4575 021562 005737 001214
 4576 021566 001405
 4577 021570 005037 001204
 4578 021574 005037 001206
 4579 021600 104125

```

MOV #1,STMP1 ;INITIALIZE ACO IN MEMORY
LDF STMP0,ACO ;INITIALIZE ACO
LDPPS #FD ;SET THE FD BIT
LDCFD (PC)+,ACO ;EXECUTE INSTRUCTION UNDER TEST
ERROR 66 ;ADX ROM FAILED
BR 25 ;ADX ROM OK
ERROR 66 ;ADX ROM FAILED
ERROR 66 ;
ERROR 66 ;
STD ACO,STMP2 ;GET DATA BACK
CMP #104066,STMP2 ;QUAD 3 OK?
BNE 35 ;BRANCH IF NO
TST STMP3 ;QUAD 2 OK?
BEQ TST60 ;;BRANCH IF YES
MOV #104066,STMP0
CLR STMP1
ERROR 35 ;DATA BAD IN ACO

```

```

;*****
;*TEST 60 LDCFD*-M0*IMM*-(EXP=0)
;*
;* THE ONLY POSSIBLE FAILURE IN THIS TEST SHOULD BE ROM STATE 720.
;*
;* FPU ROM FLOW - 17, 220, 46, 136
;*****

```

```

TST60: SCOPE
MOV #STN-1,STESTN ;;SET TEST NUMBER IN MAIL BOX
MOV #TST61,SESCAPE ;;ESCAPE TO TEST 61 ON ERROR
MOV #1,STMP0
MOV #1,STMP1
LDF STMP0,ACO ;INITIALIZE ACO
MOV #40200,STMP0 ;INITIALIZE DATA
MOV #125252,STMP1 ;TO LOAD
MOV #STMP0,R0 ;GET ADDRESS OF DATA
LDPPS #FD ;SET FD BIT
LDCFD (R0)+,ACO ;EXECUTE INSTRUCTION UNDER TEST
CMP #STMP2,R0 ;ADX ROM OK?
BEQ 35 ;BRANCH IF YES
ERROR 66 ;ADX ROM FAILED
STD ACO,STMP4 ;GET DATA BACK
CMP #40200,STMP4 ;QUAD 3 OK?
BNE 25 ;BRANCH IF NO
CMP #125252,STMP5 ;QUAD 2 OK?
BNE 25 ;BRANCH IF NO
TST STMP6 ;QUAD 1 OK?
BEQ TST61 ;;BRANCH IF YES
CLR STMP2 ;SAVE EXPECTED
CLR STMP3 ;VALUE OF QUAD 1 & 0
ERROR 125 ;QUAD IS BAD

```

```

.SBTTL
.SBTTL FIVE ROM STATES
.SBTTL

```

```

;*****

```

T61

```

4586
4587
4588
4589
4590
4591
4592
4593 021602 000004
4594 021604 012737 000061 001740
4595 021612 170400
4596 021614 005000
4597 021616 176400
4598 021620 174037 001162
4599 021624 022737 040000 001162
4600 021632 001404
4601 021634 012737 040000 001164
4602 021642 104105
4603
4604
4605
4606
4607
4608
4609
4610
4611
4612
4613
4614
4615
4616 021644 000004
4617 021646 012737 000062 001240
4618 021654 012737 022140 001227
4619
4620
4621 021662 172427 040000
4622 021666 012700 125252
4623 021672 170127 000013
4624 021676 012737 000013 177776
4625 021704 175000
4626 021706 013737 177776 001264
4627 021714 170237 001200
4628 021720 005700
4629 021722 001411
4630 021724 022700 177777
4631 021730 001405
4632 021732 010037 001162
4633 021736 005037 001164
4634 021742 104131
4635 021744 104132
4636
4637 021746 022737 000004 001200
4638 021754 001411
4639 021756 022737 000014 001202
4640 021764 001001
4641 021766 104164
  
```

```

;*TEST 61      LDEXP*MO*(EXP=+)
;*
;*      THE ONLY THING THAT SHOULD FAIL IS THE A BRANCH ROM,
;*      THE NO-MEM POM, OR POM STATE 45.
;*
;*      FPU ROM FLOW - 30, 45, 10, 260, 362
;*****
TST61:  SCOPE
        MOV      #STN-1,STESTN    ;;SET TEST NUMBER IN MAIL BOX
        CLRFP   ACO                ;START WITH EXP=0
        CLR     RO                  ;SETUP RO
1S:     LDEXP   RO,ACO              ;EXECUTE INSTRUCTION UNDER TEST
        STF     ACO,$REG1          ;GET DATA BACK
        CMP     #40000,$REG1       ;EXPONENT OK?
        BEQ     TST62              ;;BRANCH IF YES
        MOV     #40000,$REG7       ;SAVE EXPECTED DATA
        ERROR   105                ;EXPONENT DID NOT LOAD PROPERLY
  
```

```

;*TEST 62      STEXP
;*
;*      THE A BRANCH SHOULD ONLY FAIL, IF THE ROM FAILS. THE EXPONENT BEING WRONG
;*      COULD BE A FUNCTION OF THE EALU CARRY LOGIC OR ANY OF THE CONTROL
;*      STORE SIGNALS.
;*      IF THE SIGN IS WRONG, THEN FXPP SC09 IS NOT GETTING TO THE SD MUX OR
;*      THE SD MUX IS BAD.
;*
;*      FPU ROM FLOW - 50, 64, 147, 63, 152
;*****
  
```

```

TST62:  SCOPE
        MOV      #STN-1,STESTN    ;;SET TEST NUMBER IN MAIL BOX
        MOV      #TST63,SESCAPE    ;;ESCAPE TO TEST 63 ON ERROR
;*****
;SECTION 1 - EXP=200 AND MO
        LDF     #040000,ACO        ;INITIALIZE EXPONENT
        MOV     #125252,RO         ;MAKE SURE DST IS NON ZERO
        LDFPS  #13                 ;LOAD COMPLEMENT CC'S
        MOV     #13,PSW            ;SET CPU CC'S TO COMPLEMENT OF EXPECTED
1S:     STEXP   ACO,RO              ;EXECUTE INSTRUCTION UNDER TEST
        MOV     PSW,$RPSW          ;SAVE CC'S OF PSW
        STFPS  $TMP0               ;SAVE CC'S OF FPU
        TST    RO                  ;IS DATA OK?
        BEQ    2S                  ;BRANCH IF YES
        CMP    #-1,RO              ;DID FRMP EALU CIN GO LOW?
        BEQ    3S                  ;BRANCH IF NO
        MOV    PO,$REG1            ;SAVE RECEIVED VALUE
        CLR    $REG2               ;SAVE EXPECTED VALUE
        ERROR  131                 ;DATA BAD ON STEXP*MO
3S:     ERROR   132                 ;FRMP EALU CIN DID NOT GO LOW
;CHECK CONDITION CODES
2S:     CMP    #4,$TMP0            ;FPU'S CC'S OK?
        BEQ    4S                  ;BRANCH IF YES
        CMP    #14,$TMP1           ;DID SC09 FAIL?
        BNE    10S                 ;BRANCH IF NO
        ERROR  164                 ;FXPP SC09 NOT GETTING TO
  
```

```

4647
4643 021770 012737 000004 001207 10$: MOV #4,$TMP1 ;SD MUX AS A LOW
4644 021776 104071 ERROR 71 ;SAVE EXPECTED VALUE
4645 022000 042737 177760 001264 4$: BIC #177760,$ERPSW ;CC'S BAD IN FPU
4646 022006 022737 000004 001264 CMP #4,$ERPSW ;MASK CC'S OF CPU
4647 022014 001401 BEQ 55 ;CC'S OK?
4648 022016 104133 ERROR 133 ;BRANCH IF YES
4649 ;*****
4650 ;SECTION 2 - EXP=0 AND -NO
4651 022020 5$:
4652 022020 012737 022026 001110 MOV #.+6,$SLPERR ;SET ERROR LOOP
4653 022026 170400 CLRF ACO ;MAKE ACO=ZERO
4654 022030 012700 001162 MOV #SREG1,R0
4655 022034 170127 000007 LDFPS #7 ;LOAD COMPLEMENT OF EXPECTED CC'S
4656 022040 175020 STEXP ACO,(R0)+ ;EXECUTE INSTRUCTION UNDER TEST
4657 022042 022700 001164 CMP #SREG2,R0 ;ADJ ROM OK?
4658 022046 001401 BEQ 75 ;BRANCH IF YES
4659 022050 104066 ERROR 66 ;ADJ ROM FAILED
4660 022052 170237 001200 7$: STFPS $TMP0 ;GET CC'S
4661 022056 022737 177600 001162 CMP #177600,$SREG1 ;IS DATA OK?
4662 022064 001411 BEQ 85 ;BRANCH IF YES
4663 022066 022737 000200 001162 CMP #200,$SREG1 ;DID DIRM SIGN EXTEND?
4664 022074 001404 BEQ 65 ;BRANCH IF NO
4665 022076 012737 177600 001164 MOV #177600,$SREG2 ;SAVE EXPECTED DATA
4666 022104 104131 ERROR 171 ;DATA IS BAD
4667 022106 104134 ERROR 134 ;FXPF ESXT DID NOT GO HIGH
4668 022110 022737 000010 001200 8$: CMP #10,$TMP0 ;CC OK?
4669 022116 001410 BEQ TST63 ;BRANCH IF YES
4670 022120 005737 001200 TST $TMP0 ;IS N CLEAR & NO OTHER BITS ON?
4671 022124 001001 BNE 95 ;BRANCH IF NO
4672 022126 104165 ERROR 165 ;SC09 NOT GETTING TO SD MUX AS A H
4673 022130 012737 000010 001202 9$: MOV #10,$TMP1 ;SAVE EXPECTED VALUE
4674 022136 104071 ERROR 71
4675
4676 ;*****
4677 ;*TEST 63 CLR0*-NO*-IMM
4678 ;*
4679 ;* THE ONLY THING THAT SHOULD FAIL IS ROM STATE 222 OR THE ADX ROM.
4680 ;*
4681 ;* FPU ROM FLOW - 25, 211, 216, 227, 212
4682 ;*****
4683 022140 000004 TST63: SCOPE
4684 022142 012737 000063 001240 MOV #STN-1,$TESTN ;SET TEST NUMBER IN MAIL BOX
4685 022150 012737 022252 001222 MOV #TST64,$ESCAPE ;ESCAPE TO TEST 64 ON ERPOP
4686 022156 012700 001210 MOV #TMP4,R0 ;GET ADDRESS OF BUFFER
4687 022162 012701 177777 MOV #-1,R1 ;PUT DATA TO LOAD IN R1
4688 022166 010120 MOV R1,(R0)+ ;ENSURE BUFFER
4689 022170 010120 MOV R1,(R0)+ ;IS NOT
4690 022172 010120 MOV R1,(R0)+ ;ALREADY
4691 022174 010110 MOV R1,(R0) ;CLEAR.
4692 022176 170127 000200 LDFPS #FD ;SET THE FD BIT
4693 022202 012700 001210 MOV #TMP4,R0 ;PUT ADDRESS OF BUFFER IN R0
4694 022206 170420 1$: CLRD (R0)+ ;EXECUTE INSTRUCTION UNDER TEST
4695 022210 022700 001220 CMP #TMP7+7,R0 ;DID ADX ROM WORK?
4696 022214 001401 BEQ 25 ;BRANCH IF YES
4697 022216 104066 ERROR 66 ;ADJ ROM FAILED

```

```

4698 022220 005737 001214 2S: TST STMP6 ;DID QUAD 1 CLEAR?
4699 022224 001063 BNE 3S ;BRANCH IF NO
4700 022226 005737 001216 TST STMP7 ;DID QUAD 0 CLEAR?
4701 022732 001407 BEQ TST64 ;;BRANCH IF YES
4702 022234 012700 001200 3S: MOV #STMP0,R0 ;SAVE
4703 022240 005020 CLR (R0)+ ;EXPECTED
4704 022242 005020 CLR (R0)+ ;DATA
4705 022244 005020 CLR (R0)+ ;
4706 022246 005010 CLR (R0) ;
4707 022250 104135 EPROR 135 ;CLRD FAILED
  
```

4708
4709
4710
4711
4712
4713
4714
4715
4716
4717
4718
4719
4720

```

;;*****
;*TEST 64 ABSF*-NO*-(EXP=0)
;*
;* IF FRMB FP CLASS DOES NOT GO LOW OR DOES NOT GET TO BACK AS
;* A LOW, THE CPU AND FP WILL GET OUT OF SYNC.
;*
;* THE BRANCHES IN THE FPU SHOULD NOT FAIL.
;*
;* CPU ROM FLOW - 101, 314, 111, 135, 36, 327, 367, 342, 265, 225, 362, 307, 237
;* FPU ROM FLOW - 77, 717, 274, 177, 212
;;*****
  
```

```

4721 022252 000004
4722 022254 012737 000064 001240
4723 022262 012737 022360 001227
4724 022270 012737 140377 001162
4725 022276 012700 001162
4726 022302 170127 000017
4727 022306 170620
4728 022310 022700 001166
4729 022314 001401
4730 022316 104066
4731 022320 022737 040377 001162
4732 022326 001404
4733 022330 012737 040377 001164
4734 022336 104136
4735 022340 170237 001200
4736 022344 005737 001200
4737 022350 001403
4738 022352 005037 001202
4739 022356 104071
  
```

```

TST64: SCOPE
MOV #STN-1,STESTN ;;SET TEST NUMBER IN MAIL BOX
MOV #TST65,SESCAPE ;;ESCAPE TO TEST 65 ON ERROR
MOV #140377,$REG1 ;PUT TEST DATA IN MEMORY
MOV #SREG1,R0 ;PUT ADDRESS OF DATA IN R0
LDPPS #17 ;LOAD COMPLEMENT CC'S
1S: ABSF (R0)+ ;EXECUTE INSTRUCTION UNDER TEST
CMP #SREG3,R0 ;DID AUTO-INC WORK OK?
BEQ 2S ;BRANCH IF YES.
ERROR 66 ;NOX ROM PATLFD
2S: CMP #40377,$REG1 ;DID SIGN CLEAR?
BEQ 3S ;BRANCH IF YES
MOV #40377,$REG2 ;SAVE EXPECTED DATA
ERROR 136 ;ABSF FAILED
3S: STFPS STMP0 ;GET CC'S
TST STMP0 ;CC'S OK?
BEQ TST65 ;;BRANCH IF YES
CLR STMP1 ;SAVE EXPECTED CC'S
ERROR 71 ;CC'S BAD
  
```

4740
4741
4742
4743
4744
4745
4746
4747
4748

```

;;*****
;*TEST 65 NEGFP*-NO*-(EXP=0)
;*
;* THE ONLY THING THAT SHOULD FAIL IS PCW STATE 234.
;*
;* FPU ROM FLOW - 77, 717, 234, 177, 212
;;*****
  
```

```

4749 022360 000004
4750 022362 012737 000065 001240
4751 022370 012737 022472 001227
4752 022376 012737 040377 001162
4753 022404 012700 001162
  
```

```

TST65: SCOPE
MOV #STN-1,STFSTN ;;SET TEST NUMBER IN MAIL BOX
MOV #TST66,SESCAPE ;;ESCAPE TO TEST 66 ON ERROR
MOV #40377,$REG1 ;PUT DATA IN MEMORY
MOV #SREG1,R0 ;PUT ADDRESS OF DATA IN R0
  
```

4754 022410 170127 000007
4755 022414 170720
4756 022416 022700 001166
4757 022427 001401
4758 022424 104066
4759 022426 022737 140377 001162
4760 022434 001404
4761 022436 012737 140377 001164
4762 022444 104137
4763 022446 170237 001200
4764 022452 022737 000010 001200
4765 022460 001404
4766 022462 012737 000010 001202
4767 022470 104071
4768
4769
4770
4771
4772
4773
4774
4775
4776
4777 022472 000004
4778 022474 012737 000066 001240
4779 022502 012737 022632 001227
4780 022510 170127 000200
4781 022514 170400
4782 022516 012737 140377 001700
4783 022524 012737 177777 001202
4784 022532 012737 177777 001204
4785 022540 012700 001200
4786 022544 170127 000000
4787 022550 172020
4788 022552 022700 001204
4789 022556 001401
4790 022560 104066
4791 022562 170127 000200
4792 022566 174037 001210
4793 022572 022737 140377 001210
4794 022600 001007
4795 022602 022737 177777 001212
4796 022610 001003
4797 022612 005737 001214
4798 022616 001405
4799 022620 005037 001204
4800 022624 005037 001206
4801 022630 104140
4802
4803
4804
4805
4806
4807
4808
4809

LDPPS #7 ;LOAD COMPLEMENT CC'S
15: NEGF (R0)+ ;EXECUTE INSTRUCTION UNDER TEST
CMP #SREG3,R0 ;ADX ROM OK?
BEQ 25 ;BRANCH IF YES
ERROR 66 ;ADX ROM FAILED
25: CMP #140377,\$REG1 ;IS DATA OK?
BEQ 35 ;BRANCH IF YES
MOV #140377,\$REG2 ;SAVE EXPECTED DATA
ERROR 137 ;NEGF FAILED
35: STPFS \$TMP0 ;GET CC'S
CMP #10,\$TMP0 ;CC'S OK?
BEQ TST66 ;BRANCH IF YES
MOV #10,\$TMP1 ;SAVE EXPECTED CC'S
ERROR 71 ;CC'S WRONG

;;*****
;*TEST 66 ADDP*-M0*-(SRC=0)*(DST=0)
;*
;* THE ONLY THING THAT SHOULD FAIL IS THE LDAC6 FLOWS.
;*
;* FPU ROM FLOW - 11, 130, 65, 250, 306
;;*****
TST66: SCOPE
MOV #STN-1,\$TESTN ;SET TEST NUMBER IN MAIL BOX
MOV #TST67,\$ESCAPE ;ESCAPE TO TEST 67 ON ERPCP
LDPPS #PD
CLRD ACO ;ENSURE DST=0
MOV #140377,\$TMP0 ;PUT SRC DATA
MOV #-1,\$TMP1 ;IN MEMORY
MOV #-1,\$TMP2 ;WAKE QUAD 1 NON-ZERO INCASE
MOV #STMP0,P0 ;PUT ADDRESS OF SRC DATA IN R0
LDPPS #0 ;A BRANCH FAILS
15: ADDP (R0)+,ACO ;EXECUTE INSTRUCTION UNDER TEST
CMP #STMP2,R0 ;ADX ROM OK?
BEQ 25 ;BRANCH IF YES
ERROR 66 ;ADX ROM FAILED
25: LDPPS #PD
STD ACO,\$TMP4 ;GET DST BACK
CMP #140377,\$TMP4 ;QUAD 3 OK?
BNE 35 ;BRANCH IF NC
CMP #-1,\$TMP5 ;QUAD 2 OK?
BNE 35 ;BRANCH IF NN
TST \$TMP6 ;QUAD 1 OK?
BEQ TST67 ;BRANCH IF YES
35: CLR \$TMP2 ;SAVE EXPECTED VALUE
CLR \$TMP3 ;OF QUAD 1 & 0
ERROR 140 ;DATA BAD

;;*****
;*TEST 67 ADDP*-M0*TMW*-(SRC=0)*(DST=0)
;*
;* THE ONLY THING THAT SHOULD FAIL IS ROM STATE 131 OR THE ADX ROM
;*
;* FPU ROM FLOW - 11, 131, 65, 250, 306

4810
4811 022632 000004
4812 022634 012737 000067 001240
4813 022642 012737 022734 001222
4814 022650 170127 000700
4815 022654 170400
4816 022656 170127 000000
4817 022662 172027
4818 022664 104066
4819 022666 000403
4820 022670 104066
4821 022672 104066
4822 022674 104066
4823 022676 174037 001204
4824 022702 022737 104066 001204
4825 022710 001003
4826 022712 005737 001206
4827 022716 001406
4828 022720 012737 104066 001200
4829 022726 005037 001202
4830 022732 104035
4831
4832
4833
4834
4835
4836
4837
4838
4839
4840 022734 000004
4841 022736 012737 000070 001240
4842 022744 012737 023074 001222
4843 022752 170127 000200
4844 022756 170400
4845 022760 170127 000000
4846 022764 012737 140377 001200
4847 022772 012737 177777 001202
4848 023000 012737 177777 001204
4849 023006 012700 001200
4850 023012 173020
4851 023014 022700 001204
4852 023020 001401
4853 023022 104066
4854 023024 170127 000200
4855 023030 174037 001210
4856 023034 022737 040377 001210
4857 023042 001007
4858 023044 022737 177777 001212
4859 023052 001003
4860 023054 005737 001214
4861 023060 001405
4862 023062 005037 001204
4863 023066 005037 001206
4864 023072 104121
4865

```
;;*****  
TST67: SCOPE  
MOV #STN-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX  
MOV #TST70,$ESCAPE ;;ESCAPE TO TEST 70 ON ERROR  
LDFPS #FD  
CLRD AC0 ;ENSURE DST=0  
LDFPS #0  
1S: ADDF (PC)+,AC0 ;EXECUTE INSTRUCTION UNDER TEST  
ERROR 66 ;ADX ROM FAILED  
BR 25 ;ADX ROM OK  
ERROR 66 ;ADX ROM FAILED  
ERROR 66 ;ADX ROM FAILED  
2S: STF AC0,$TMP2 ;GET DATA BACK  
CMP #104066,$TMP2 ;QUAD 3 OK?  
BNE 35 ;BRANCH IF NO  
TST $TMP3 ;DID QUAD 2 REMAIN CLEAR?  
BEQ TST70 ;;BRANCH IF YES  
3S: MOV #104066,$TMP0 ;SAVE EXPECTED  
CLR $TMP1 ;VALUE OF DATA  
ERROR 35 ;ADDF*-NO*IMM MESSED UP DATA
```

```
;;*****  
;*TEST 70 SUBF*-NO*-(SRC=0)*(DST=0)  
;*  
;* THE ONLY THING THAT SHOULD FAIL IS THE A BRANCH OR ADX ROM.  
;*  
;* FPU ROM FLOW - 11, 130, 65, 250, 306  
;;*****
```

```
TST70: SCOPE  
MOV #STN-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX  
MOV #TST71,$ESCAPE ;;ESCAPE TO TEST 71 ON ERROR  
LDFPS #FD  
CLRD AC0 ;ENSURE DST=0  
LDFPS #0  
MOV #140377,$TMP0 ;PUT SRC  
MOV #-1,$TMP1 ;DATA IN  
MOV #-1,$TMP2 ;MEMORY  
MOV #TMP0,R0 ;PUT ADDRESS OF DATA IN R0  
1S: SUBF (R0)+,AC0 ;EXECUTE INSTRUCTION UNDER TEST  
CMP #TMP2,R0 ;ADX ROM OK?  
BEQ 25 ;BRANCH IF YES  
ERROR 66 ;ADX ROM FAILED  
2S: LDFPS #FD  
STD AC0,$TMP4 ;GET DST BACK  
CMP #40377,$TMP4 ;QUAD 3 OK?  
BNE 35 ;BRANCH IF NO  
CMP #-1,$TMP5 ;QUAD 2 OK?  
BNE 35 ;BRANCH IF NO  
TST $TMP6 ;QUAD 1 REMAIN ZERO?  
BEQ TST71 ;;BRANCH IF YES  
3S: CLR $TMP2 ;SAVE EXPECTED VALUE OF  
CLR $TMP3 ;QUAD 1 & 0  
EPROX 171 ;DATA BAD.
```

4866
4867
4868
4869
4870
4871
4872 073074 000004
4873 023076 012737 000071 001240
4874 023104 012737 023140 001222
4875 023112 170127 000200
4876 023116 170400
4877 023120 170127 000000
4878 023124 173027
4879 023126 104066
4880 023130 000403
4881 023132 104066
4882 023134 104066
4883 023136 104066
4884
4885
4886
4887
4888
4889
4890
4891
4892
4893
4894
4895
4896
4897
4898
4899
4900
4901
4902 023140 000004
4903 023142 012737 000072 001240
4904 023150 012737 023372 001222
4905 023156 012737 023366 000244
4906 023164 012703 000171
4907 023170 170003
4908 023172 012700 001200
4909 023176 012720 177777
4910 023202 012720 177777
4911 023206 012720 177777
4912 023212 012720 177777
4913 023216 012720 140200
4914 023222 005020
4915 023224 012710 177777
4916 023230 170127 000200
4917 023234 172437 001200
4918 023240 172537 001210
4919 023244 170127 000027
4920 023250 176100
4921 023252 170237 001200

```
;;*****  
;*TEST 71 SUBP*-M0*IMM*-(SRC=0)*(DST=0)  
;*  
;* THE ONLY THING THAT SHOULD FAIL IS THE ADX ROM  
;;*****  
TST71: SCOPE  
MOV #STN-1,STESTN ;;SET TEST NUMBER IN MAIL BOX  
MOV #TST72,SESCAPE ;;ESCAPE TO TEST 72 ON ERROR  
LDFPS #FD  
CLRD ACO ;ENSURE ACO CLEAR  
LDFPS #0  
1S: SUBF (PC)+,ACO ;EXECUTE INSTRUCTION UNDER TEST  
ERROR 66 ;ADY ROM FAILED  
BP TST72 ;;GO TO NEXT TEST  
ERROR 66 ;ADX ROM FAILED  
ERROR 66 ;  
ERROR 66 ;  
;;*****  
;*TEST 72 STCFD*M0*-(EXP=0)  
;*  
;*6F2J BRANCH  
;* IF PRLP FVINT DOES NOT GO HIGH OR DOES NOT GET TO FRMA PAD01 AS  
;* A LOW, EXECUTION WILL GO TO STATE 171. THIS WILL CAUSE AN  
;* INTERRUPT.  
;*  
;* THE ONLY OTHER POSSIBLE FAILURES SHOULD BE FROM SIGNALS IN THE  
;* CONTROL STORE.  
;*  
;* NOTE: SYNC POINT NOT SELECTABLE. DEFAULT=171  
;*  
;* FPU ROM FLOW - 44, 35, 167, 123, 157  
;;*****  
TST72: SCOPE  
MOV #STN-1,STESTN ;;SET TEST NUMBER IN MAIL BOX  
MOV #TST73,SESCAPE ;;ESCAPE TO TEST 73 ON ERROR  
MOV #55,FPPVEC  
MOV #171,R3  
LDUP  
MOV #STMP0,PC  
MOV #-1,(R0)+ ;PUT DUMMY DATA  
MOV #-1,(R0)+ ;FOR DST IN MEMORY  
MOV #-1,(R0)+ ;  
MOV #-1,(R0)+ ;  
MOV #140200,(R0)+ ;PUT DATA TO  
CLR (R0)+ ;CONVERT INTO  
MOV #-1,(R0) ;MEMORY  
LDFPS #FD  
LDD STMP0,ACO ;LOAD THE DESTINATION  
LDD STMP4,AC1 ;LOAD THE SOURCE  
LDFPS #FMM+7 ;LOAD COMPLEMENT CC'S  
1S: STCFD AC1,ACO ;EXECUTE INSTRUCTION UNDER TEST  
STFPS STMP0 ;GET CC'S
```

4922	023256	170127	000200			LDFPS	#FD		
4923	023262	174037	001210			STD	ACO,\$TMP4		;GET THE DATA BACK
4924	023266	022737	140200	001710		CMP	#140200,\$TMP4		;QUAD 3 OK?
4925	023274	001012				BNE	2\$;BRANCH IF NO
4926	023276	005737	001212			TST	\$TMP5		;QUAD 2 OK?
4927	023302	001007				BNE	2\$;BRANCH IF NO
4928	023304	005737	001214			TST	\$TMP6		;QUAD 1 OK?
4929	023310	001003				BNE	3\$;BRANCH IF NO
4930	023312	005737	001216			TST	\$TMP7		;QUAD 0 OK?
4931	023316	001413				BEQ	4\$;BRANCH IF YES
4932	023320	104173			3\$:	ERKOR	173		;STATE 167 DID NOT CLEAR AN LOW
4933	023322	012737	140200	001200	2\$:	MOV	#140200,\$TMP0		;SAVE EXPECTED
4934	023320	005037	001202			CLR	\$TMP1		;VALUE
4935	023334	005037	001204			CLR	\$TMP2		;*
4936	023340	005037	001206			CLR	\$TMP3		;*
4937	023344	104125				ERROR	125		;QUAD IS BAD
4938									;CHECK THE CC'S
4939	023346	022737	000030	001200	4\$:	CMP	#PMM+10,\$TMP0		;CC'S OK?
4940	023354	001406				BEQ	TST73		;BRANCH IF YES
4941	023356	012737	000030	001202		MOV	#PMM+10,\$TMP1		;SAVE EXPECTED
4942	023364	104071				ERROR	71		;CC'S BAD
4943									
4944	023366	022626			5\$:	CMP	(SP)+,(SP)+		;RESTORE THE SP
4945	023370	104176				ERROR	176		;PRLP FVINT DID NOT GO HIGH
4946									
4947									
4948									
4949									
4950									
4951									
4952									
4953									
4954									
4955									
4956									
4957									
4958									
4959									
4960	023372	000004							
4961	023374	012737	000073	001240		MOV	#STN-1,\$TESTN		;SET TEST NUMBER IN MAIL BOX
4962	023402	012737	023562	001222		MOV	#TST74,\$ESCAPE		;ESCAPE TO TEST 74 ON ERROR
4963	023410	172427	140000			LDF	#^0140000,ACO		;LOAD SRC ACC WITH INT=0
4964	023414	012703	000333			MOV	#333,R3		;SET
4965	023420	170003				LDUB			;MICRO-BREAK TRAP
4966	023422	012737	023552	000244		MOV	#2\$,FPPVEC		;SET FPP VECTOR
4967	023430	012737	023436	001110		MOV	#.+6,@\$SLPERR		;SET ERROR LOOP
4968	023436	012700	177777			MOV	#-1,R0		;INITIALIZE R0 TO NON-ZERO
4969	023442	105737	001103			TSTB	\$ERFLG		;ANY ERRORS YET?
4970	023446	001002				BNE	5\$;BRANCH IF YES
4971	023450	170127	000033			LDFPS	#PMM+13		;LOAD COMPLEMENT CC'S IN FPP
4972	023454	000277			5\$:	SCC			;AND IN
4973	023456	000244				CLZ			;CPU
4974	023460	175400			1\$:	STCFI	ACO,R0		;EXECUTE INSTRUCTION UNDER TEST
4975	023462	013737	177776	001264		MOV	PSW,\$ERPSW		;SAVE CPU CC'S
4976	023470	170237	001200			STFPS	\$TMP0		;AND FPU CC'S
4977	023474	005700				TST	R0		;IS DATA OK?

```

;*****
;*TEST 73      STCFI*MO*(INT=0)
;*
;*SF1 BRANCH
;*      THIS BRANCH WILL BE TESTED LATER
;*
;*BN BRANCH
;*      A MICRO-BREAK TRAP WILL BE SET ON STATE 333 IN-CASE THE EXPONENT
;*      SUBTRACTION IN STATE 51 FAILS.
;*
;*      FPU ROM FLOW - 51, 374, 331, 345, 214
;*****

```


4978 023476 001405
 4979 023500 010037 001162
 4980 023504 005037 001164
 4981 023510 104141
 4982 023512 022737 000024 001200 3S:
 4983 023520 001404
 4984 023522 012737 000024 001202
 4985 023530 104071
 4986 023532 042737 177760 001264 4S:
 4987 023540 022737 000004 001264
 4988 023546 001405
 4989 023550 104133
 4990
 4991 023552 022626
 4992 023554 170127 000000
 4993 023560 104142
 4994
 4995
 4996
 4997
 4998
 4999
 5000
 5001
 5002
 5003 023562 000004
 5004 023564 012737 000074 001240
 5005 023572 012737 023700 001222
 5006 023600 012737 047624 000244
 5007 023606 175427
 5008 023610 104066
 5009 023612 170400
 5010 023614 170127 000017
 5011 023620 176427
 5012 023622 177776
 5013 023624 000403
 5014 023626 104066
 5015 023630 104066
 5016 023632 104066
 5017 023634 170237 001200 2S:
 5018 023640 175037 001162
 5019 023644 022737 177776 001162
 5020 023652 001404
 5021 023654 012737 177776 001164
 5022 023662 104143
 5023 023664 005737 001200 4S:
 5024 023670 001403
 5025 023672 005037 001202
 5026 023676 104071
 5027
 5028
 5029
 5030
 5031
 5032
 5033

```

BEQ 35 ;BRANCH IF YES
MOV R0,SREG1 ;SAVE RECEIVED DATA
CLR $REG2 ;SAVE EXPECTED DATA
ERROR 141 ;DID NOT STORE ZERO IN R0
CMP #FMM+FZ,$TMP0 ;FPU CC'S OK?
BEQ 4S ;BRANCH IF YES
MOV #FMM+FZ,$TMP1 ;SAVE EXPECTED CC'S
ERROR 71 ;CC'S BAD IN FPU
BIC #177760,$ERPSW ;MASK CC'S IN PSW
CMP #4,$ERPSW ;CC'S OK?
BEQ TST74 ;;BRANCH IF YES
ERROR 133 ;NO MEM ROM DTD NOT ENABLE PCLD
;RN BRANCH FAILED
2S: CMP (SP)+,(SP)+ ;RESTORE SP
LDFPS #0
ERROR 142 ;STATE 51 DTD NOT SET BN

```

;;*****
 ;*TEST 74 LDEXP*-NO*IMM*(EXP=-)
 ;*
 ;* THE ONLY THING THAT SHOULD FAIL IS THE SIGNALS IN ROM STATES 361 OR 227.
 ;*
 ;* FPU ROM FLOW - 15, 10, 260, 361, 227, 236
 ;******
 TST74: SCOPE
 MOV #STN-1,\$TESTN ;;SET TEST NUMBER IN MAIL BOX
 MOV #TST75,\$ESCAPE ;;ESCAPE TO TEST 75 ON ERROR
 MOV #FPSPUR,\$PPVEC ;RESTORE FP VECTOR
 1S: LDEXP (PC)+,ACO ;EXECUTE INSTRUCTION UNDER TEST
 ERROR 66 ;CHECK ADX ROM FIRST
 CLRF ACO ;INITIALIZE ACO
 LDFPS #17 ;LOAD COMPLEMENT CC'S
 LDEXP (PC)+,ACO ;EXECUTE INSTRUCTION UNDER TEST
 .WORD -2 ;EXPONENT TO LOAD
 BR 2S
 ERROR 66 ;ADX ROM FAILED
 EPROR 66 ;
 ERROR 66 ;
 2S: STFPS \$TMP0 ;GET CC'S
 STEXP ACO,\$REG1 ;GET EXPONENT BACK
 CMP #-2,\$REG1 ;DID EXP LOAD OK?
 BEQ 4S ;BRANCH IF YES
 MOV #-2,\$REG2 ;SAVE EXPECTED VALUE
 ERROR 143 ;EXPONENT WRONG
 4S: TST \$TMP0 ;CC'S OK?
 BEQ TST75 ;;BRANCH IF YES
 CLR \$TMP1 ;SAVE EXPECTED CC'S
 ERROR 71 ;CC'S BAD

```

;;*****
;*TEST 75 LDEXP*-NO*(EXP=-200)
;*
;* IF FRMA FIU(0)B H DOES NOT GO LOW EXECUTION WILL GO FROM STATE 226
;* TO STATE 205. THIS WILL CAUSE AN INTERRUPT TO OCCUR.

```

```

5034 ;*
5035 ;* THE ONLY OTHER POSSIBLE FAILURES ARE ROM STATES 226 OR 207.
5036 ;*
5037 ;* THE FD BIT WILL BE SET TO ENSURE THE ADX AND A BRANCH ROMS ARE OK.
5038 ;*
5039 ;* FPU ROM FLOW - 15, 10, 260, 361, 226, 207
5040 ;*****

```

```

TST75: SCOPE
5041 023700 000004 MOV #STN-1,STESTN ;;SET TEST NUMBER IN MAIL BOX
5042 023702 012737 000075 001240 MOV #TST76,SESCAPE ;;ESCAPE TO TEST 76 ON ERROR
5043 023710 012737 024036 001222 MOV #45,FPPVEC ;SETUP FPP VECTOR
5044 023716 012737 024032 000244 LDF #0140377,ACO ;INITIALIZE ACO
5045 023724 172427 140377 MOV #SREG2,R0 ;PUT ADDRESS OF EXPONENT IN R0
5046 023730 012700 001154 MOV #-200,(R0) ;PUT EXPONENT TO LOAD IN MEMORY
5047 023734 012710 177600 LDFPS #FD+13 ;LOAD COMPLEMENT CC'S
5048 023740 170127 000213 1S: LDEXP (R0)+,ACO ;EXECUTE INSTRUCTION UNDER TEST
5049 023744 176420 CMP #SREG3,R0 ;ADX ROM OK?
5050 023746 022700 001166 BEQ 25 ;BRANCH IF YES
5051 023752 001401 ERROR 66 ;ADX ROM FAILED
5052 023754 104066 2S: STFPS $TMP0 ;GET CC'S BACK
5053 023756 170237 001200 LDFPS #0 ;CLEAR FD BIT
5054 023762 170127 000000 STF ACO,$TMP2 ;GET ACO BACK
5055 023766 174037 001204 TST $TMP2 ;DID ACO CLEAR?
5056 023772 005737 001204 BEQ 35 ;BRANCH IF YES
5057 023776 001405 CLR $TMP0 ;SAVE EXPECTED
5058 024000 005037 001200 CLR $TMP1 ;VALUE OF ACO
5059 024004 005037 001202 ERROR 144 ;LDEXP DID NOT UNDERFLOW
5060 024010 104144 3S: CMP #FD+4,$TMP0 ;CC'S OK?
5061 024012 022737 000204 001200 BEQ TST76 ;;BRANCH IF YES
5062 024020 001406 MOV #FD+4,$TMP1 ;SAVE EXPECTED VALUE
5063 024022 012737 000204 001202 ERROR 71 ;CC'S BAD
5064 024030 104071 4S: CMP (SP)+,(SP)+ ;RESTORE THE SP
5065 024032 022626 EPROR 163 ;PRMA FTU(0) NOT GOING LOW
5066 024034 104163

```

```

5067 ;*****
5068 ;*TEST 76 LDEXP*M0*(EXP>177)
5069 ;*
5070 ;* THE ONLY POSSIBLE FAILURE IS ROM STATE 70 OR 163.
5071 ;*
5072 ;* FPU ROM FLOW - 30, 15, 10, 260, 363, 70, 173
5073 ;*****

```

```

TST76: SCOPE
5076 024036 000004 MOV #STN-1,STESTN ;;SET TEST NUMBER IN MAIL BOX
5077 024040 012737 000076 001240 MOV #TST77,SESCAPE ;;ESCAPE TO TEST 77 ON ERROR
5078 024046 012737 024154 001222 MOV #201,R0 ;SETUP EXPONENT TO LOAD
5079 024054 012700 000201 LDF #0140377,ACO ;INITIALIZE ACO
5080 024060 172427 140377 LDFPS #5 ;LOAD COMPLEMENT CC'S
5081 024064 170127 000005 1S: LDEXP R0,ACO ;EXECUTE INSTRUCTION UNDER TEST
5082 024070 176400 STFPS $TMP0 ;SAVE CC'S
5083 024072 170237 001200 STF ACO,$TMP2 ;GET ACO BACK
5084 024076 174037 001204 TST $TMP2 ;IS ACO EXP 6 FRAC OK?
5085 024102 005737 001204 BEQ 25 ;BRANCH IF YES
5086 024106 001405 CLR $TMP0 ;SAVE EXPECTED
5087 024110 005037 001200 CLR $TMP1 ;STATE 173 FAILED TO LOAD DST
5088 024114 005037 001202 EPROR 145
5089 024120 104145

```

5090 024122 022737 000006 001200 2S:
5091 024130 001411
5092 024132 022737 000004 001200
5093 024140 001404
5094 024142 012737 000006 001202
5095 024150 104071
5096 024152 104146 3S:

CMP #6,\$TMP0 ;CC'S OK?
BEQ TST77 ;;BRANCH IF YES
CMP #4,\$TMP0 ;DID V PIT FAIL?
BEQ 3S ;BRANCH IF YES
MOV #6,\$TMP1 ;SAVE EXPECTED VALUE
ERROR 71 ;CC'S BAD
ERROR 146 ;FXPJ OVF NOT GOING HIGH OR NOT
;GETTING TO FRLN AS A HIGH.

5097
5098
5099

5100

;;*****

5101

;*TEST 77 NEG*MO*IMM*(EXP=0)

5102

;*
;*
;*
;*
;*
;*

5103

THE ONLY THING THAT SHOULD FAIL IS PCM STATES 176 OR 31 OR THE
ADX ROM.

5104

5105

5106

FPU ROM FLOW - 27, 217, 234, 176, 31, 211

5107

5108

024154 000004

TST77: SCOPE

5109

024156 012737 000077 001240

MOV #STN-1,\$TESTN ;;SET TEST NUMBER IN MAIL BOX

5110

024164 012737 024266 001222

MOV #TST100,\$ESCAPE ;;ESCAPE TO TEST 100 ON ERROR

5111

024172 012737 000177 024206

MOV #177,\$S ;INITIALIZE DATA LOCATION

5112

024200 170127 000013

LDFPS #13 ;LOAD COMPLIMENT CC'S

5113

024204 170727

1S: NEG* (PC)+

;EXECUTE INSTRUCTION UNDER TEST

5114

024206 000177

2S: .WORD 177

;WILL HALT HERE IF ADX ROM FAILS

5115

024210 000403

BR 3S

5116

024212 104066

ERROR 66 ;ADX ROM FAILED

5117

024214 104066

ERROR 66 ;

5118

024216 104066

ERROR 66 ;

5119

024220 170737 001200

3S: STFPS \$TMP0

;GET CC'S

5120

024224 005737 024206

TST 2S

;DID DATA WORD CLEAR?

5121

024230 001406

BEQ 4S

;BRANCH IF YES

5122

024232 013737 024206 001167

MOV 2S,\$REG1 ;SAVE RECEIVED VALUE

5123

024240 005037 001164

CLR \$REG2 ;SAVE EXPECTED VALUE

5124

024244 104147

ERROR 147 ;QUAD 3 DID NOT CLEAR

5125

024246 022737 000004 001200

4S: CMP #FZ,\$TMP0

;CC'S OK?

5126

024254 001404

BEQ TST100 ;;BRANCH IF YES

5127

024256 012737 000004 001202

MOV #FZ,\$TMP1 ;SAVE EXPECTED VALUE

5128

024264 104071

ERROR 71 ;CC'S BAD

5129

5130

5131

;;*****

5132

;*TEST 100 ADD*MO*(SRC<<DST)

5133

;*
;*
;*
;*
;*
;*

5134

THIS TEST CHECKS THE ADD FLOWS WHEN THE SRC EXPONENT IS
MUCH, MUCH LESS THAN THE DST EXPONENT.

5135

5136

5137

NEXT, DIFFERENT VALUES OF EXPONENTS WILL BE CHECKED TO ENSURE
THAT THE LOGIC THAT CAUSES FRLN SWR(1) TO BE LOW FUNCTIONS PROPERLY.

5138

5139

5140

;*2F3 BRANCH

5141

IF FRLN ADD*SC<8 DOES NOT GO HIGH OR DCFS NOT GET TO RAD03

5142

AS A HIGH, EXECUTION WILL GO TO STATE 265. THIS WILL CAUSE THE

5143

SRC FRACTION TO BE RIGHT SHIFTED BY 8 AND ADDED TO THE DST.

5144

THIS ERROR WOULD MOST LIKELY BE CAUSED BY FXPJ EALU SWR

5145

NOT GOING LOW WHICH WOULD MOST LIKELY BE CAUSED BY FXPP OUT

```

5146 ;* OF RANGE NOT GOING HIGH.
5147 ;*
5148 ;* IF FRMP SUB*SC<8 DOES NOT GO HIGH OR DOES NOT GET TO RAO2
5149 ;* AS A HIGH, EXECUTION WILL GO TO STATE 271. THIS WILL CAUSE THE
5150 ;* SRC FRACTION TO BE RIGHT SHIFTED BY 8 AND SUBTRACTED FROM THE
5151 ;* DESTINATION.
5152 ;*
5153 ;* IF FXPP SC09(1) DOES NOT GO LOW OR DOES NOT GET TO
5154 ;* FRMA AS A LOW EXECUTION WILL GO TO STATE 277. THIS
5155 ;* WILL CAUSE THE SRC TO BE PUT IN THE DST.
5156 ;*
5157 ;*5F1 BRANCH
5158 ;* IF FXPP OUR OF RANGE DOES NOT GET TO FRMA AS A HIGH,
5159 ;* EXECUTION WILL GO TO STATE 225. THIS WILL PROBABLY CAUSE
5160 ;* THE FPP TO HANG IN THE ALIGN LOOP.
5161 ;*
5162 ;* IT SHOULD BE REMEMBERED THAT THE ER, IN THIS TEST, IS LOADED FROM
5163 ;* THE ABS VAL ROM. ONLY A SMALL PORTION OF THAT ROM IS TESTED
5164 ;* HERE. ALL OF IT WILL BE TESTED BY THE END OF THE PROGRAM.
5165 ;*
5166 ;* FPU ROM FLOW - 30, 65, 240, 275, 100, 235
5167 ;*****
5168 024266 000004 TST100: SCOPE
5169 024270 012737 000100 001240 MOV #STN-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX
5170 024276 012737 024734 001222 MOV #TST101,$ESCAPE ;;ESCAPE TO TEST 101 ON ERROR
5171 024304 012737 024602 000244 MOV #ADD2,PPPVEC ;SETUP FP VECTOR
5172 024312 012703 000225 MOV #225,R3 ;SET MICRO-BREAK
5173 024316 170003 LDUB ;TO CATCH 5F1 BRANCH FAILURE
5174 024320 012737 024326 001110 MOV #.+6,@$LPERR ;SET ERROR LOOP
5175 024326 172427 077600 LDF #*077600,ACO ;LOAD DESTINATION
5176 024332 172527 017600 LDF #*017600,AC1 ;LOAD SOURCE
5177 024336 105737 001103 TSTP $ERFLG ;ANY ERRORS YET?
5178 024342 001002 BNE 15 ;BRANCH IF YES
5179 024344 170127 000037 LDFFS #FMM+17 ;LOAD COMPLIMENT CC'S
5180 024350 172001 15: ADDP AC1,ACO ;EXECUTE INSTRUCTION UNDER TEST
5181 024352 170237 001200 STPPS $TMP0 ;GET CC'S
5182 024356 170127 000000 LDFFS #0 ;CLEAR FMM BIT
5183 024362 174037 001204 STF ACO,$TMP2 ;GET DEST BACK
5184 024366 022737 077600 001204 CMP #77600,$TMP2 ;QUAD 3 OK?
5185 024374 001005 BNE 35 ;BRANCH IF NO
5186 024376 005737 001206 TST $TMP3 ;QUAD 2 OK?
5187 024402 001466 BEQ ADD1 ;BRANCH IF YES
5188 024404 100006 BPL 55 ;BRANCH IF 5F1 BRANCH DID NOT FAIL
5189 024406 104150 ERROR 150 ;FRMP ADD*SC<8 DID NOT GO HIGH
5190 024410 022737 077577 001204 35: CMP #77577,$TMP2 ;DID 2F3 BRANCH FAIL TO THE SUB FLOW?
5191 024416 001001 BNE 55 ;BRANCH IF NO
5192 024420 104151 EPROR 151 ;FRMP SUB*SC<8 DID NOT GO HIGH
5193 ;*****
5194 ;CHECK THE ROM FLOW
5195 024422 022737 017600 001204 55: CMP #17600,$TMP2 ;DID 2F3 BRANCH FAIL TO STATE 277?
5196 024430 001001 BNE 65 ;BRANCH IF NO
5197 024432 104155 ERROR 155 ;FXPP SC09(1) NOT GOING LOW
5198 024434 172427 077600 65: LDF #*077600,ACO ;RESTORE DST
5199 024440 172527 017600 LDF #*017600,AC1 ;AND SRC
5200 024444 032777 001000 154464 ADD: BIT #SW9,@SWK ;SWITCH 9 ON?
5201 024452 001403 BEQ .+10 ;BRANCH IF NO
  
```

```

5202 024454 105737 001103      TSTR    $ERFLG      ;ANY ERRORS?
5203 024460 001027      BNE     4$          ;BRANCH IF YES
5204 024462 005005      CLR     R5          ;USED TO CLEAR MAINTENANCE MODE
5205 024464 012737 024534 000244  MOV     #1$,@#FPPVEC ;SET INTERRUPT VECTOR
5206 024472 012700 024726      MOV     #ROMADD,R0  ;GET ADDRESS OF TABLE
5207 024476 112003      3$:    MOVB   (R0)+,R3  ;GET DATA FOR MICRO-BREAK REG
5208 024500 001420      BEQ     5$          ;BRANCH IF DONE
5209 024502 170127 000020      LDFPS  #FMM        ;LOAD FPS REGISTER
5210 024506 170003      LDUB   ;LOAD MICRO-BREAK REGISTER
5211 024510 172001      ADDF   AC1,AC0     ;EXECUTE INSTRUCTION
5212 024512 005001      CLR     R1
5213 024514 005201      2$:    INC     R1          ;WAIT FOR FPP
5214 024516 001376      BNE     2$          ;TO FINISH IF NO INTERRUPT
5215 024520 170105      LDFPS  R5          ;CLEAR MAINTENANCE MODE
5216 024522 114037 001200      MOVB   -(R0),@#STMP0 ;SAVE ADDRESS THAT FAILED
5217 024526 105037 001201      CLRB   @#STMP0+1  ;GET RID OF SIGN EXT
5218 024532 104034      ERROR  34          ;FAILED TO TRAP ON THE ROM STATE
5219 024534 022626      1$:    CMP     (SP)+,(SP)+ ;RESTORE STACK POINTER
5220 024536 000757      BR     3$          ;GO TO NEXT ROM STATE
5221 024540 104000      4$:    ERROR  0          ;ONLY EXECUTES WHILE LOOPING ON ERROR
5222 024542 170105      5$:    LDFPS  R5          ;CLEAR MAINTENANCE MODE
5223 024544 012737 077600 001700  MOV     #77600,STMP0 ;SAVE EXPECTED
5224 024552 005037 001202      CLR     $TMP1      ;VALUE OF DST
5225 024556 104055      ERROR  55          ;ROM FLOW OK, DATA BAD
5226
5227      ;*****
5228 024560 042737 000020 001200  ADD1:   BIC     #FMM,STMP0 ;DATA OK, CHECK CC'S
5229 024566 005737 001700      TST     STMP0      ;CC'S OK?
5230 024572 001405      BEQ     ADD3       ;BRANCH IF YES
5231 024574 005037 001202      CLR     $TMP1      ;SAVE EXPECTED VALUE
5232 024600 104071      ERROR  71          ;CC'S BAD
5233
5234      ;*****
5235 024602 022626      ADD2:   CMP     (SP)+,(SP)+ ;RESTORE THE SP
5236 024604 104152      ERROR  152        ;FXPP OUT OF RANGE NOT GETTING TO PRMA AS A B
5237
5238      ;*****
5239      ;NOW CHECK SOME MORE COMBINATIONS OF SWR
5240      ;     EXP DIFF=200---SC=200 AND EALU <6:0>=000
5241
5242      ;     NOTE: SYNC POINT NOT SELECTABLE. DEFAULT=265
5243 024606 012737 024650 000244  ADD3:   MOV     #2$,FPPVEC ;SETUP VECTOR
5244 024614 012737 024622 001110  MOV     #.+6,@#SLPERR ;SET ERROR LCCP
5245 024622 012703 000265      MOV     #265,R3    ;SETUP MICRO BREAK
5246 024626 170003      LDUB   ;TO CATCH FAILURE
5247 024630 172427 060177      LDF    #^060177,AC0 ;LOAD DST
5248 024634 172527 020000      LDF    #^020000,AC1 ;LOAD SRC
5249 024640 170127 000020      LDFPS  #FMM        ;LOAD MAINTENANCE MODE
5250 024644 172001      1$:    ADDF   AC1,AC0     ;EXECUTE INSTRUCTION UNDER TEST
5251 024646 000402      BR     3$          ;
5252 024650 022626      2$:    CMP     (SP)+,(SP)+ ;RESTORE STACK POINTER
5253 024652 104153      ERROR  153        ;FXPP SC9 XOR SC7 NOT GOING LOW
5254
5255      ;*****
5256 024654 012737 024722 000244  3$:    MOV     #4$,FPPVEC ;SETUP VECTOR
5257 024662 012737 024670 001110  MOV     #.+6,@#SLPERR ;SET ERROR LOOP

```

```

5258 024670 012703 000265
5259 024674 170003
5260 024676 170127 000020
5261 024702 172427 060177
5262 024706 172527 040000
5263 024712 172001
5264 024714 170127 000000
5265 024720 000405
5266 024722 022626
5267 024724 104154
5268 024726 065 240 275
5269 024731 100 235 000
5270
5271
5272
5273
5274
5275
5276
5277
5278
5279
5280
5281
5282
5283
5284
5285 024734 000004
5286 024736 012737 000101 001240
5287 024744 012737 025210 001222
5288 024752 172427 020077
5289 024756 172527 046000
5290 024762 172001
5291 024764 174037 001204
5292 024770 022737 046000 001204
5293 024776 001413
5294 025000 022737 020077 001204
5295 025006 001001
5296 025010 104156
5297
5298 025012 012737 046000 001200 3S:
5299 025020 005037 001202
5300 025024 104157
5301
5302
5303
5304
5305
5306
5307 025026 012737 025070 000244 2S:
5308 025034 012737 025042 001110
5309 025042 012703 000265
5310 025046 170003
5311 025050 170127 000020
5312 025054 172427 036000
5313 025060 172527 060177
  
```

```

MOV #265,R3
LDUR
LDFPS #FHM
LDF #060177,AC0 ;LOAD DST
LDF #040000,AC1 ;LOAD SRC
ADDF AC1,AC0 ;EXECUTE INSTRUCTION UNDER TEST
LDFPS #0 ;CLEAR FHM BIT
BR TST101 ;;GO TO NEXT TEST
4S: CMP (SP)+,(SP)+ ;RESTORE THE SP
ERROR 154 ;FXPP SC9 XOR SC6 NOT GOING LOW
ROMADD: .BYTE 65,240,275,100,235,0
.EVEN

;;*****
; *TEST 101 ADDF*MO*(SRC>>DST)
; *
; * THIS TEST CHECKS THE ADD FLOWS WHEN THE SRC EXPONENT IS
; * MUCH, MUCH GREATER THAN THE DST EXPONENT.
; *2F3 BRANCH
; * IF FXPP SC09(1) DOES NOT GO HIGH OR DOES NOT GET TO PRMA AS
; * A HIGH EXECUTION WILL GO TO STATE 275. THIS WILL CAUSE THE DST
; * TO REMAIN UNCHANGED.
; *
; * FPU ROM FLOW - 30, 65, 240, 277, 274, 306
; * *****
TST101: SCOPE
MOV #STN-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX
MOV #TST102,$ESCAPE ;;ESCAPE TO TEST 102 ON ERROR
LDF #020077,AC0 ;LOAD THE DST
LDF #046000,AC1 ;LOAD THE SRC
1S: ADDF AC1,AC0 ;EXECUTE INSTRUCTION UNDER TEST
STF AC0,$TMP2 ;GET DST BACK
CMP #46000,$TMP2 ;IS DATA CORRECT?
BEQ 25 ;BRANCH IF YES
CMP #20077,$TMP2 ;DID FXPP SC09 FAIL TO GET TO PRMA?
BNE 35 ;BRANCH IF NO
ERROR 156 ;FXPP SC09(1) NOT GETTING TO PRMA
;AS A HIGH
3S: MOV #46000,$TMP0 ;SAVE EXPECTED
CLR $TMP1 ;VALUE OF DATA
ERROR 157 ;STATE 274 FAILED TO LOAD DATA
;;*****
; NOW CHECK SOME MORE COMBINATIONS OF SWR
; EXP DIFF=110---SC=1670 AND EALU<6:0>=170
;
; NOTE: SYNC POINT NOT SELECTABLE. DEFAULT=265
;
2S: MOV #4S,$PPVEC ;SETUP VECTOR
MOV #.+6,$SLPEPR ;SET ERROR LOOP
MOV #265,R3 ;SET MICRO-BREAK TO TRAP
LDUR ;IF SWR FAILS
LDFPS #FHM
LDF #036000,AC0 ;LOAD THE DST
LDF #060177,AC1 ;LOAD THE SRC
  
```

```

5314 025064 172001      ADDF  AC1,AC0      ;EXECUTE INSTRUCTION
5315 025066 000402      BR    55
5316 025070 022626      4S:  CMP    (SP)+,(SP)+  ;RESTORE THE SP
5317 025072 104154      ERROR  154      ;FXPP SC09 XOR SC06 DID NOT GO LOW
5318
5319      ;*****
5320 025074 012737 025136 000244 5S:  MOV    #6$,FPPVEC  ;EXP DIFF=210---SC=1570 AND FALU<6:0>=170
5321 025102 012737 025110 001110  MOV    #.+6,#$SLPERR ;SET FRPCR LOOP
5322 025110 012703 000265  MOV    #26$,R3
5323 025114 170003      LDUP
5324 025116 170127 000020  LDFPS #FMM
5325 025122 172427 020000  LDF    #*020000,AC0  ;LOAD THE DST
5326 025126 172527 062177  LDF    #*062177,AC1  ;LOAD THE SRC
5327 025132 172001      ADDF  AC1,AC0      ;EXECUTE INSTRUCTION
5328 025134 000402      BR    75
5329 025136 022626      6S:  CMP    (SP)+,(SP)+  ;RESTORE THE SP
5330 025140 104153      ERROR  153      ;FXPP SC09 XOR SC07 DID NOT GO LOW
5331
5332      ;*****
5333 025142 012737 025204 000244 7S:  MOV    #RS$,FPPVEC  ;EXP DIFF=100---SC=1700 AND EALU<6:0>=000
5334 025150 012737 025156 001110  MOV    #.+6,#$SLPERR ;SET ERROR LOOP
5335 025156 012703 000265  MOV    #26$,R3
5336 025162 170003      LDUP
5337 025164 170127 000020  LDFPS #FMM
5338 025170 172427 040000  LDF    #*040000,AC0  ;LOAD THE DST
5339 025174 172527 060000  LDF    #*060000,AC1  ;LOAD THE SRC
5340 025200 172001      ADFDF AC1,AC0      ;EXECUTE INSTRUCTION
5341 025202 000402      BR    TST107
5342 025204 022626      8S:  CMP    (SP)+,(SP)+  ;RESTORE THE SP
5343 025206 104207      ERROR  207      ;FXPP ARS VAL ROM OUT NOT GETTING
5344
5345
5346
5347      ;*****
5348      ;*TEST 102      CMPF*-MO*TM*(SRC EXP=DST EXP)*--(SS=SD)
5349
5350      ;*4P2 BRANCH
5351      ;*      IF FRMP SD(1) H DOES NOT GET TO FRMA PAD01 AS A LOW EXECUTION
5352      ;*      WILL GO TO STATE 255. A MICRO-PREAF TRAP WILL SET HERE TO CATCH
5353      ;*      THIS FAILURE.
5354
5355      ;*      FPU ROM FLOW-11,131,67,320,302,757
5356      ;*****
5357 025210 000004      TST102: SCOPE
5358 025212 012737 000102 001240  MOV    #STN-1,$TESTN  ;SET TEST NUMBER IN MAIL BOX
5359 025220 012737 025332 001222  MOV    #TST103,$ESCAPE ;ESCAPE TO TEST 103 ON ERROR
5360 025226 012703 000255  MOV    #255,R3      ;LOAD THE MICRO BREAK
5361 025232 170003      LDUP      ;REGISTER IN CASE BRANCH FAILS
5362 025234 012737 025326 000244  MOV    #25$,FPPVEC
5363 025242 012737 025250 001110  MOV    #.+6,#$SLPERK ;SET ERROR LOOP
5364 025250 172427 004000  LDF    #*04000,AC0  ;LOAD DESTINATION
5365 025254 105737 001103  TSTR  $ERPLG      ;ANY ERRORS YET?
5366 025260 001002      BNE  15      ;BRANCH IF YES
5367 025262 170127 000027  LDFPS #FMM+7      ;LOAD COMPLIMENT CC'S
5368 025266 173427      1S:  CMPF  (PC)+,AC0  ;EXECUTE INSTRUCTION UNDER TEST
5369 025270 104066      ERROR  66      ;ADX ROM FAILED
  
```

5370	025272	000403				BR	35	
5371	025274	104066				ERROR	66	;ADX ROM FAILED
5372	025276	104066				ERROR	66	; *
5373	025300	104066				ERROR	66	; *
5374	025307	170737	001200		35:	STFPS	STMP0	;GET CC'S
5375	025306	022737	000030	001200		CMP	#FNM+FN,STMP0	;CC'S OK?
5376	025314	001406				BEQ	TST103	;BRANCH IF YES
5377	025316	012737	000030	001202		MOV	#FNM+FN,STMP1	;SAVE EXPECTED VALUE
5378	025324	104071				ERROR	71	;CC'S BAD
5379								;4F2 BRANCH FAILED
5380	025326	022626			25:	CMP	(SP)+,(SP)+	;RESTORE THE SP
5381	025330	104160				ERROR	160	;FRMF SD(1) NOT GETTING TO FRMA AS A R
5382								
5383								;*****
5384								;*TEST 103 CMPF*MO*(SRC<DST)*(SS=SD)
5385								;*
5386								;*4F2 BRANCH
5387								;* IF FRMF SD(1) DOES NOT GET TO FRMA PAD01 AS A HIGH, EXECUTION WILL
5388								;* GO TO STATE 257. THIS FAILURE WILL BE DETECTED BY SETTING A MICRO-BREAK
5389								;* TRAP AT STATE 257.
5390								;*
5391								;*5F2 BRANCH
5392								;* IF FRHE AR59 DOES NOT GO LOW OR DOES NOT GET TO FRMA RAD01 AS A
5393								;* HIGH EXECUTION WILL GO TO STATE 323. THIS FAILURE WILL ALSO BE CAUGHT BY
5394								;* A MICRO-BREAK TRAP.
5395								;*
5396								;* FPU ROM FLOW-30,67,320,307,255,321
5397								;*****
5398	025332	000004				TST103: SCOPE		
5399	025334	012737	000103	001240		MOV	#STN-1,STESTN	;SET TEST NUMBER IN MAIL BOX
5400	025342	012737	025512	001222		MOV	#TST104,SESCAPE	;ESCAPE TO TEST 104 ON ERROR
5401	025350	012703	000257			MOV	#257,R3	;LOAD MICRO-BREAK TO CATCH
5402	025354	170003				LDUR		;4F2 BRANCH FAILURE
5403	025356	012737	025364	001110		MOV	#.+6,#SLPERR	;SET ERROR LOOP
5404	025364	172427	040100			LDF	#040100,ACO	;LOAD THE DST
5405	025370	172527	040000			LDF	#040000,AC1	;LOAD THE SPC
5406	025374	012737	025420	000744		MOV	#75,PPPVEC	
5407	025402	105737	001103			TSTB	SEKFLG	;ANY PRGRS YET?
5408	025406	001002				BNE	15	;BRANCH IF YES
5409	025410	170127	000020			LDFPS	#FNM	
5410	025414	173401			15:	CMPF	AC1,ACO	;CHECK THE 4F2 BRANCH FIRST
5411	025416	000402				BR	35	
5412	025420	022626			25:	CMP	(SP)+,(SP)+	
5413	025422	104161				ERROR	161	;FRMF SD(1) DID NOT GET TO FRMA
5414								;PAD01 AS A HIGH.
5415	025424	012737	025506	000244	35:	MOV	#45,PPPVEC	
5416	025432	012703	000323			MOV	#323,R3	;SET MICRO-BREAK TO CATCH
5417	025436	170003				LDUR		;A FAILURE IN 5F2 BRANCH
5418	025440	170127	000027			LDFPS	#FNM+7	;LOAD COMPLIMENT CC'S
5419	025444	012737	025452	001110		MOV	#.+6,#SLPERR	;SET ERROR LOOP
5420	025452	173401				CMPF	AC1,ACO	;EXECUTE INSTRUCTION UNDER TEST
5421	025454	170237	001200			STFPS	STMP0	;GET THE CC'S
5422	025460	042737	006020	001200		BIC	#FNM,STMP0	
5423	025466	022737	000010	001200		CMP	#FN,STMP0	;CC'S OK?
5424	025474	001406				BEQ	TST104	;BRANCH IF YES
5425	025476	012737	000010	001202		MOV	#FN,STMP1	

5426 025504 104071
 5427 025506 022626
 5428 025510 104162

```

45:  ERROR 71
      CMP (SP)+,(SP)+ ;RESTORE THE SP
      ERROR 162 ;FRHE AR59(1) NOT GOING LOW OR NOT
                ;GETTING TO FRMA OR FXPW ACS0
                ;NOT GOING HIGH IN STATE 302
  
```

5429
 5430
 5431
 5432
 5433
 5434
 5435
 5436
 5437
 5438
 5439
 5440
 5441
 5442
 5443

```

;*****
;*TEST 104      CMPF*MO*(SRC>DST)*(SS=SD)
;*
;*5P2J BRANCH
;* IF FRHE AR59(1)L DOES NOT GO HIGH, EXECUTION WILL GO TO
;* STATE 175. THIS WILL BE CAUGHT BY SETTING A MICRO-TRAP. THE ONLY
;* WAY THIS SHOULD HAPPEN IS IF STATE 255 DID NOT SUBTRACT THE ONE
;* PROPERLY (ALU FAILURE).
;*
;* FPU ROM FLOW-30,67,320,302,255,323
;*****
  
```

5444 075517 000004
 5445 025514 012737 000104 001240
 5446 025522 012737 025644 001222
 5447 025530 012737 025640 000244
 5448 025536 012703 000135
 5449 025542 170003
 5450 025544 172727 040000
 5451 025550 012737 040000 001704
 5452 025556 012737 000001 001206
 5453 025564 172437 001204
 5454 025570 172537 001704
 5455 025574 170127 000037
 5456 025600 012737 025606 001110
 5457 025606 173403
 5458 025610 170737 001200
 5459 025614 042737 000020 001200
 5460 025622 022737 000010 001200
 5461 025630 001405
 5462 025632 005037 001202
 5463 025636 104071
 5464 025640 022626
 5465 025642 104166

```

TST104: SCOPE
        MOV #STN-1,STESTN ;;SET TEST NUMBER IN MAIL BOX
        MOV #TST105,$ESCAPE ;;ESCAPE TO TEST 105 ON ERROR
        MOV #2,$FPVEC ;;SETUP FP VECTOR
        MOV #135,R3 ;;LOAD THE MICRO-BREAK TO
        LDUR ;;CATCH A FAILURE OF 5P2J BRANCH
        LDF #040000,AC3 ;;LOAD THE SRC
        MOV #40000,$TMP2 ;;SETUP DATA IN MEMORY
        MOV #BIT0,$TMP3 ;;FOR THE DESTINATION
        LDF $TMP2,AC0 ;;LOAD THE DESTINATION
        LDF $TMP2,AC1
        LDFPS #FNM+17 ;;LOAD COMPLIMENT CC'S
        MOV #.+6,$SLPERR ;;SET ERROR LOOP
15:     CMPF AC3,AC0 ;;EXECUTE INSTRUCTION UNDER TEST
        STFPS $TMP0 ;;SAVE CC'S
        BIC #FNM,$TMP0
        CMP #FNM,$TMP0 ;;CC'S OK?
        BEQ TST105 ;;BRANCH IF YES
        CLR $TMP1 ;;SAVE EXPECTED VALUE
        ERROR 71 ;;CC'S BAD
25:     CMP (SP)+,(SP)+ ;;RESTORE THE SP
        ERROR 166 ;;FRHE AR59(1) DID NOT GO HIGH
                ;OR FXPW ACS1 DID NOT GO HIGH
                ;IN STATE 302
  
```

5466
 5467
 5468
 5469
 5470
 5471
 5472
 5473
 5474
 5475

```

;*****
;*TEST 105      CMPF*-MO*(SRC EXP>DST EXP)
;*
;* THE ONLY THING THAT SHOULD FAIL IS THE A BRANCH OR ADX ROM
;*
;* FPU ROM FLOW-11,130,67,320,301
;*****
  
```

5476 025644 000004
 5477 025646 012737 000105 001240
 5478 025654 012737 025736 001222
 5479 025662 172427 040000
 5480 025666 012737 040200 001200
 5481 025674 170127 000017

```

TST105: SCOPE
        MOV #STN-1,STESTN ;;SET TEST NUMBER IN MAIL BOX
        MOV #TST106,$ESCAPE ;;ESCAPE TO TEST 106 ON ERROR
        LDF #040000,AC0 ;;LOAD DST
        MOV #40200,$TMP0 ;;LOAD SRC INTO MEMORY
        LDFPS #17 ;;LOAD COMPLIMENT CC'S
  
```

5482 025700 012700 001700
 5483 025704 173420
 5484 025706 022700 001204
 5485 025712 001401
 5486 025714 104066
 5487 025716 170237 001200
 5488 025722 005737 001200
 5489 025726 001403
 5490 025730 005037 001202
 5491 025734 104071

```

MOV      #STMP0,R0      ;PUT ADDRESS OF DATA IN R0
1S:     CMPF      (R0)+,AC0 ;EXECUTE INSTRUCTION UNDER TEST
        CMP      #STMP2,R0 ;ADX ROM OK?
        BEQ      2$      ;BRANCH IF YES
        ERROR    66      ;ADX ROM FAILED
2S:     STFPS     $TMP0   ;GET CC'S
        TST      $TMP0   ;CC'S OK?
        BEQ      TST106  ;;BRANCH IF YES
        CLR      $TMP1   ;SAVE EXPECTED VALUE
        EPROR    71
  
```

```

;;*****
;*TEST 106      LDCDF*-M0*-(EXP=0)
;*
;*      THE ONLY THING THAT SHOULD FAIL IS ROM STATE 47.
;*
;*      FPU ROM FLOW-17,220,47,46,137,231
;;*****
  
```

5500 025736 000004
 5501 025740 012737 000106 001240
 5502 025746 012737 026042 001222
 5503 025754 012737 040000 001200
 5504 025762 012737 125252 001202
 5505 025770 012737 100000 001204
 5506 025776 012700 001700
 5507 026002 17040C
 5508 026004 177420
 5509 026006 022700 001210
 5510 026012 001401
 5511 026014 104066
 5512 026016 174037 001204
 5513 026022 022737 125253 001206
 5514 026030 001404
 5515 026032 012737 125253 001202
 5516 026040 104035

```

TST106: SCOPE
MOV      #STN-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX
MOV      #TST107,$ESCAPE ;;ESCAPE TO TEST 107 ON EPROR
MOV      #40000,$TMP0  ;LOAD DATA FOR
MOV      #125252,$TMP1 ;QUAD 3 AND 2
MOV      #BIT15,$TMP2  ;SET BIT 15 IN 3RD WORD. THIS
MOV      #STMP0,R0     ;WILL ROUND INTO QUAD 2 IF 3RD
CLRPF    AC0           ;WORD IS LOADED PROPERLY
1S:     LDCDF      (R0)+,AC0 ;EXECUTE INSTRUCTION UNDER TEST
        CMP      #STMP4,R0 ;ADX ROM OK?
        BEQ      2$      ;BRANCH IF YES
        EPROR    66      ;ADX ROM FAILED
2S:     STF       AC0,$TMP2 ;GET DATA BACK
        CMP      #125253,$TMP3 ;DID QUAD 2 GET ROUND BIT?
        BEQ      TST107  ;;BRANCH IF YES
        MOV      #125253,$TMP1 ;SAVE EXPECTED VALUE
        ERROR    35      ;QUAD 2 BAD
  
```

```

;;*****
;*TEST 107      STCFI*-M0*(INT=0)
;*
;*      *7F1 BRANCH
;*      IF FXPB MO DOES NOT GET TO PRMA AS A LOW, EXECUTION WILL GO TO STATE
;*      204. THIS WILL CAUSE THE CPU TO DO A DATI INSTEAD OF A DATO.
;*
;*      FPU ROM FLOW-51,374,331,345,210,215
;;*****
  
```

5528 026042 000004
 5529 026044 012737 000107 001240
 5530 026057 012737 026704 001227
 5531 026060 172427 040000
 5532 026064 012700 001162
 5533 026070 012710 177777
 5534 026074 170127 000013
 5535 026100 000277
 5536 026102 000744
 5537 026104 175420

```

TST107: SCOPE
MOV      #STN-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX
MOV      #TST110,$ESCAPE ;;ESCAPE TO TEST 110 ON EPROR
LDF      #040000,AC0   ;LOAD SRC WITH INTG=0
MOV      #SREG1,R0    ;PUT ADDRESS OF DST IN R0
MOV      #-1,(R0)     ;MAKE SURE DST NOW ZERO
LDFPS    #13          ;LOAD COMPLIMENT CC'S
SCC
CLZ
1S:     STCFI     AC0,(R0)+ ;AND PSW TOO
        ;EXECUTE INSTRUCTION UNDER TEST
  
```

```
5538 026106 013737 177776 001264      MOV      PSW,$ERPSW      ;SAVE CC'S OF PSW
5539 026114 170237 001200              STFPS      $TMP0        ;AND FPS TOO
5540 026120 005737 001162              TST       $REG1        ;DATA OK?
5541 026124 001403                      BEQ       2$           ;BRANCH IF YES
5542 026126 005037 001164              CLR       $REG2        ;SAVE EXPECTED VALUE
5543 026132 104167                      ERROR    167          ;DATA DID NOT STORE
5544 026134 022700 001164      2$:    CMP      #SREG2,R0    ;ADX ROM OK?
5545 026140 001401                      BEQ      3$           ;BRANCH IF YES
5546 026142 104066                      ERROR    66           ;ADX ROM FAILED
5547 026144 022737 000004 001700      3$:    CMP      #FZ,$TMP0    ;FP CC'S OK?
5548 026152 001404                      BEQ      4$           ;BRANCH IF YES
5549 026154 012737 000004 001202      MOV      #FZ,$TMP1    ;SAVE EXPECTED VALUE
5550 026162 104071                      ERROR    71           ;CC'S BAD
5551 026164 042737 177760 001764      4$:    BIC      #177760,$ERPSW ;PSW CC'S OK?
5552 026172 022737 000004 001264      CMP      #4,$ERPSW
5553 026200 001401                      BEQ      TST110       ;;BRANCH IF YES
5554 026202 104133                      ERROR    133         ;FXPC FCLD EN DID NOT GO LOW
5555
5556
5557
5558
5559
5560
5561
5562
5563
5564
5565
5566 026204 000004
5567 026206 012737 000110 001240      ;*****
;*TEST 110      STCFI*MO*(INT>2**15)
;*
;*3F2 BRANCH
;* THIS BRANCH WILL ONLY FAIL IF THE EXPONENT SUBTRACTION IN STATE
;* 51 OR 374 FAILED.
;*
;* FPU ROM FLOW-51,374,333,373,345,214
;*****
TST110: SCOPE
5568 026214 012737 026340 001222      MOV      #STN-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX
5569 026222 012737 144277 001200      MOV      #TST111,$ESCAPE ;;ESCAPE TO TEST 111 ON ERROR
5570 026230 012737 177777 001202      MOV      #144277,$TMP0 ;LOAD THE SRC WITH EXP=2**17
5571 026236 172437 001200      MOV      #-1,$TMP1     ;AND ENSURE QUAD 2
5572 026242 012700 125257      LDF      $TMP0,AC0     ;IS NON-ZERO
5573 026246 170127 000012      MOV      #125252,R0    ;ENSURE DST NON-ZERO
5574 026252 175400      1$:    LDFPS     #12         ;LOAD COMPLIMENT CC'S
5575 026254 170237 001200      STCFI    AC0,R0        ;EXECUTE INSTRUCTION UNDER TEST
5576 026260 005700      STFPS    $TMP0        ;GET CC'S
5577 026262 001411      TST     R0            ;PO CLEAR?
5578 026264 022700 177777      BEQ     3$           ;BRANCH IF YES
5579 026270 001001      CMP     #-1,R0       ;DID R0 GET A MINUS 1?
5580 026272 104171      BNE     4$           ;BRANCH IF NO
5581 026274 010037 001162      4$:    ERROR 171        ;STATE 373 DID NOT CLEAR AC6
5582 026300 005037 001164      MOV     R0,$REG1     ;SAVE RECEIVED VALUE
5583 026304 104141      CLR     $REG2        ;SAVE EXPECTED VALUE
5584 026306 022737 000005 001200      3$:    ERROR 141        ;DATA BAD
5585 026314 001411      CMP     #FZ+FC,$TMP0 ;CC'S OK?
5586 026316 022737 000004 001200      BEQ     TST111       ;;BRANCH IF YES
5587 026324 001001      CMP     #FZ,$TMP0    ;DID C BIT FAIL?
5588 026326 104172      BNE     5$           ;BRANCH IF NO
5589 026330 012737 000005 001202      5$:    ERROR 172        ;FRAME FCC1(1) NOT GETTING TO C BIT AS A H
5590 026336 104071      MOV     #FZ+FC,$TMP1 ;SAVE EXPECTED VALUE
5591
5592
5593      ERROR 71          ;CC'S BAD
;*****
```

```
5594 ;*TEST 111 STCFD*MO*(EXP=0)
5595 ;*
5596 ;* THE ONLY THING THAT SHOULD FAIL IS ROM STATE 122.
5597 ;*
5598 ;* FPU ROM FLOW-44,35,167,122,123,157
5599 ;*.....
5600 026340 000004 TST111: SCOPE
5601 026342 012737 000111 001240 MOV #STN-1,STESTN ;;SET TEST NUMBER IN MAIL BOX
5602 026350 012737 026550 001222 MOV #TST112,$ESCAPE ;;ESCAPE TO TEST 112 ON ERROR
5603 026356 012737 177777 001200 MOV #-1,$TMP0 ;PUT DST DATA
5604 026364 012737 177777 001202 MOV #-1,$TMP1 ;IN MEMORY
5605 026372 012737 177777 001204 MOV #-1,$TMP2
5606 026400 012737 177777 001206 MOV #-1,$TMP3 ;PUT SRC DATA
5607 026406 005037 001210 CLR $TMP4 ;IN MEMORY
5608 026412 012737 177777 001212 MOV #-1,$TMP5
5609 026420 170127 000200 LDFPS #FD
5610 026424 172437 001200 LDD $TMP0,AC0 ;LOAD DST ACC
5611 026430 172537 001210 LDD $TMP4,AC1 ;LOAD SRC ACC
5612 026434 170127 000013 LDFPS #13 ;LOAD COMPLIMENT CC'S
5613 026440 176100 1S: STCFD AC1,AC0 ;EXECUTE INSTRUCTION UNDER TEST
5614 026442 170237 001200 STFPS $TMP0 ;GET CC'S
5615 026446 170127 000200 LDFPS #FD
5616 026452 174037 001210 STD AC0,$TMP4 ;GET DATA
5617 026456 005737 001210 TST $TMP4 ;QUAD 3 OK?
5618 026462 001011 BNE 2S ;BRANCH IF NO
5619 026464 005737 001212 TST $TMP5 ;QUAD 2 OK?
5620 026470 001006 BNE 2S ;BRANCH IF NO
5621 026472 005737 001214 TST $TMP6 ;QUAD 1 OK?
5622 026476 001003 BNE 2S ;BRANCH IF NO
5623 026500 005737 001216 TST $TMP7 ;QUAD 0 OK?
5624 026504 001411 BEQ 3S ;BRANCH IF YES
5625 026506 005037 001200 2S: CLR $TMP0 ;SAVE EXPECTED
5626 026512 005037 001202 CLR $TMP1 ;VALUE OF DATA
5627 026516 005037 001204 CLR $TMP2
5628 026522 005037 001206 CLR $TMP3
5629 026526 104174 ERROR 174 ;DST DID NOT CLEAR
5630 ;CHECK THE CC'S
5631 026530 022737 000004 001200 3S: CMP #FZ,$TMP0 ;CC'S OK?
5632 026536 001404 BEQ TST112 ;;BRANCH IF YES
5633 026540 012737 000004 001202 MOV #FZ,$TMP1 ;SAVE EXPECTED VALUE
5634 026546 104071 ERROR 71 ;CC'S BAD
5635 ;*.....
5636 ;*TEST 112 STCFD*-MO*IMM*-(EXP=0)
5637 ;*
5638 ;* THE ONLY THING THAT SHOULD FAIL IS THE A BRANCH, OR ADX ROM, OR
5639 ;* STATES 123 OR 153.
5640 ;*
5641 ;* FPU ROM FLOW-44,35,167,123,153,145
5642 ;*.....
5643 ;*.....
5644 026550 000004 TST112: SCOPE
5645 026552 012737 000112 001240 MOV #STN-1,STESTN ;;SET TEST NUMBER IN MAIL BOX
5646 026560 012737 026642 001222 MOV #TST113,$ESCAPE ;;ESCAPE TO TEST 113 ON ERROR
5647 026566 012737 104066 026607 MOV #104066,3S
5648 026574 172527 104000 LDF #0104000,AC1 ;LOAD THE SRC
5649 026600 176127 1S: STCFD AC1,(PC)+ ;EXECUTE INSTRUCTION UNDER TEST
```

```

5650 026602 104066 3S: ERROR 66 ;ADX ROM FAILED
5651 026604 000403 BR 2S
5652 026606 104066 ERROR 66 ;ADX ROM FAILED
5653 026610 104066 ERROR 66 ;
5654 026612 104066 ERROR 66 ;
5655 026614 022737 104000 026602 2S: CMP #104000,3S ;DATA OK?
5656 026622 001407 BEQ TST113 ;;BRANCH IF YES
5657 026624 013737 026602 001162 MOV 3S,$REG1 ;SAVE RECEIVED DATA
5658 026632 012737 104000 001164 MOV #104000,$REG2 ;SAVE EXPECTED DATA
5659 026640 104175 ERROR 175 ;STATE 175 FAILED TO OUTPUT DATA

```

```

;;*****
;*TEST 113 STCDF*MO*(EXP=0)
;*
;* THE ONLY THING THAT SHOULD FAIL IS STATE 120.
;*
;* FPU ROM FLOW-44,35,166,120,123,157
;;*****

```

```

5668 026642 000004 TST113: SCOPE
5669 026644 012737 000113 001240 MOV #STN-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX
5670 026652 012737 027024 001222 MOV #TST114,$ESCAPE ;;ESCAPE TO TEST 114 ON ERROR
5671 026660 012700 001200 MOV #STMP0,R0
5672 026664 012701 177777 MOV #-1,R1
5673 026670 010120 MOV R1,(R0)+ ;PUT DATA TO
5674 026672 010120 MOV R1,(R0)+ ;INITIALIZE THE DST
5675 026674 010120 MOV R1,(R0)+ ;INTO MEMORY
5676 026676 010110 MOV R1,(R0) ;
5677 026700 172527 000000 LDF #0,AC1 ;LOAD THE SOURCE
5678 026704 170127 000200 LDFPS #FD
5679 026710 172437 001200 LDD $STMP0,ACO ;LOAD THE DST
5680 026714 170127 000213 LDFPS #FD+13 ;LOAD COMPLIMENT CC'S
5681 026720 176100 1S: STCDF AC1,ACO ;EXECUTE INSTRUCTION UNDER TEST
5682 026722 170237 001200 STFPS $STMP0 ;GET THE CC'S
5683 026726 174037 001210 STD ACO,$STMP4 ;GET THE DATA BACK
5684 026732 005737 001210 TST $STMP4 ;QUAD 3 OK?
5685 026736 001011 BNE 2S ;BRANCH IF NO
5686 026740 005737 001212 TST $STMP5 ;QUAD 2 OK?
5687 026744 001006 BNE 2S ;BRANCH IF NO
5688 026746 005737 001214 TST $STMP6 ;QUAD 1 OK?
5689 026752 001003 BNE 2S ;BRANCH IF NO
5690 026754 005737 001216 TST $STMP7 ;QUAD 0 OK?
5691 026760 001411 BEQ 3S ;BRANCH IF YES
5692 026762 005037 001200 2S: CLR $STMP0 ;SAVE EXPECTED DATA
5693 026766 005037 001202 CLR $STMP1 ;
5694 026772 005037 001204 CLR $STMP2 ;
5695 026776 005037 001206 CLR $STMP3 ;
5696 027002 104125 ERROR 175 ;QUAD BAD
5697 ;CHECK THE CC'S
5698 027004 022737 000204 001200 3S: CMP #FD+F7,$STMP0 ;CC'S OK?
5699 027012 001404 BEQ TST114 ;;BRANCH IF YES
5700 027014 012737 000204 001202 MOV #FD+FZ,$STMP1 ;SAVE EXPECTED VALUE
5701 027022 104071 ERROR 71 ;CC'S BAD

```

```

;;*****
;*TEST 114 LDEXP*IMM*(EXP<-200)

```

```

5706
5707
5708
5709
5710
5711 027024 000004
5712 027026 012737 000114 001240
5713 027034 012737 027124 001222
5714 027042 172427 040200
5715 027046 012737 027120 000244
5716 027054 170127 000200
5717 027060 176427
5718 027062 177507
5719 027064 000403
5720 027066 104066
5721 027070 104066
5722 027072 104066
5723 027074 170127 000000
5724 027100 174037 001162
5725 027104 005737 001162
5726 027110 001405
5727 027112 005037 001164
5728 027116 104201
5729
5730 027120 022626
5731 027122 104066
5732
5733
5734
5735
5736
5737
5738
5739
5740
5741 027124 000004
5742 027126 012737 000115 001240
5743 027134 012737 027242 001222
5744 027142 012700 001210
5745 027146 012737 000177 001210
5746 027154 012737 177777 001212
5747 027162 012737 177777 001214
5748 027170 170620
5749 027172 022700 001214
5750 027176 001401
5751 027200 104066
5752 027202 005737 001210
5753 027206 001010
5754 027210 005737 001212
5755 027214 001005
5756 027216 022737 177777 001214
5757 027224 061406
5758 027226 104125
5759 027230 005037 001200
5760 027234 005037 001202
5761 027240 104035

;
;* THE ONLY THING THAT SHOULD FAIL IS THE ADX ROM OR STATE LDE.95.
;*
;* FPU ROM FLOW-15,10,260,361,227,237,207
;*****
TST114: SCOPE
MOV #STN-1,STESTN ;;SET TEST NUMBER IN MAIL BOX
MOV #TST115,$ESCAPE ;;ESCAPE TO TEST 115 ON ERROR
LDF #040200,ACO ;INITIALIZE ACO
MOV #2,$PPPVEC ;SETUP FPP VECTOR
LDFPS #FD ;SET FD BIT TO CHECK AN ADDRESS OF ADX ROM
1$: LDEXP (PC)+,ACO ;EXECUTE INSTRUCTION UNDER TEST
.WORD 177507 ;IF ADX ROM FAILS, THIS WILL EXECUTE AND TRAP
BR 3$
ERROR 66 ;ADX ROM FAILED
ERROR 66 ;
ERROR 66 ;
3$: LDFPS #0 ;CLEAR FD BIT
STF ACO,$REG1 ;GET DATA BACK
TST $REG1 ;DATA OK?
BEQ TST115 ;;BRANCH IF YES
CLR $REG2 ;SAVE EXPECTED VALUE
ERROR 201 ;STATE 173 DID NOT CLEAR ACO
;ADX ROM FAILED
2$: CMP (SP)+,(SP)+ ;RESTORE THE SP
ERROR 66 ;ADX ROM FAILED
;*****
;*TEST 115 ABSF*-M0*(EXP=0)
;*
;* THE ONLY THING THAT SHOULD FAIL IS THE FP REQUEST COUNTER GETTING
;* THE WRONG CONSTANT IN STATE 176.
;*
;* FPU ROM FLOW-27,217,224,176,31,211,216
;*****
TST115: SCOPE
MOV #STN-1,STESTN ;;SET TEST NUMBER IN MAIL BOX
MOV #TST116,$ESCAPE ;;ESCAPE TO TEST 116 ON ERROR
MOV #STMP4,R0
MOV #177,STMP4 ;PUT DATA IN MEMORY
MOV #-1,STMP5 ;EXP=0 AND FRAC=NON-ZERO
MOV #-1,STMP6 ;MAKE *3RD WORD NON ZERO INCASE FP REQ FAILS
1$: ABSF (R0)+ ;EXECUTE INSTRUCTION UNDER TEST
CMP #STMP6,R0 ;ADX ROM OK?
BEQ 3$ ;BRANCH IF YES
ERROR 66 ;ADX ROM FAILED
3$: TST STMP4 ;QUAD 3 OK?
BNE 2$ ;BRANCH IF NO
TST STMP5 ;QUAD 2 OK?
BNE 2$ ;BRANCH IF NO
CMP #-1,STMP6 ;DID 3RD WORD CHANGE?
BEQ TST116 ;;BRANCH IF NO
ERROR 125 ;PRMF FP REQ DID NOT GO H AFTER 2 WORDS
2$: CLR STMP0 ;SAVE EXPECTED VALUE
CLR STMP1 ;
ERROR 35 ;DATA BAD

```

5762
5763
5764
5765
5766
5767
5768
5769
5770
5771
5772
5773
5774
5775
5776
5777
5778
5779
5780
5781
5782
5783
5784
5785
5786
5787
5788
5789
5790
5791
5792
5793
5794
5795
5796
5797
5798
5799
5800
5801
5802
5803
5804
5805
5806
5807
5808
5809
5810
5811
5812
5813
5814
5815
5816
5817

027242 000004
027244 012737 000116 001240
027252 012737 030646 001222
027260 032777 001000 151650
027266 001403
027270 105737 001103
027274 001031
027276 005005
027300 012737 027354 000244
027306 012700 030642
027317 112003
027314 001422
027316 170127 000020
027322 170003
027324 012704 000000
027330 170004
027332 005001
027334 005201
027336 001376
027340 170105
027342 114037 001200

```
;;*****  
;*TST 116 MSN  
;* THIS TEST FIRST CHECKS THE MSN INSTRUCTION AND THEN CHECKS THE AR AND  
;* Q SHIFTER LOGIC.  
;* THE SHIFTER LOGIC IS TESTED AS FOLLOWS: THE RIGHT SHIFT IS CHECKED WITH  
;* SHIFT COUNTS 8 THRU 1. THIS IS DONE WITH TWO "SOB" LOOPS. THE NESTED  
;* LOOP (LOOP2) CONTROLS TESTING OF THE PATTERN AND ITS COMPLIMENT, WHILE  
;* THE OUTSIDE LOOP (LOOP1) CONTROLS THE SHIFT COUNT. THE ACTUAL LOOP1 COUNT  
;* IS USED AS THE SHIFT COUNT.  
;* THE TEST DATA IS LOADED FROM "COMMON TAGS" INTO A CHECK BUFFER AND  
;* INTO AC1 AND AC2. THE DATA CONSISTS OF 8 ONE'S (INCLUDING THE HIDDEN BIT ) FOLLO  
;* 8 ZERO'S FOLLOWED BY 8 ONE'S, ETC. FOR THE RIGHT SHIFT BY 8. THE RIGHT  
;* SHIFT BY 7 DATA IS 7 ONE'S, FOLLOWED BY 7 ZERO'S, FOLLOWED BY 7 ONE'S, ETC.  
;* SHIFT COUNTS 6 THRU 1 FOLLOW THE SAME PATTERN OF BITS.  
;* THE DATA IN THE CHECK BUFFER IS THEN CONVERTED TO  
;* THE EXPECTED ANSWER. A MICROC-BREAK TRAP IS SET ON ROM STATE 275 AND  
;* THE DIFFERENCE IN THE SRC AND DST ACCUMULATORS IS MADE TO BE +300. THE  
;* "ADDD" IS THEN EXECUTED AND A TRAP OCCURS LEAVING THE DATA IN THE AR  
;* AND THE QP. THE SHIFT IS THEN PERFORMED, THE AR & QR STORED IN MEMORY,  
;* AND CHECKED.  
;* ONLY ONE SHIFT COUNT (7) IS TESTED FOR THE LEFT SHIFT. BUT, TWO UNIQUE  
;* CONDITIONS ARE TESTED. THE FIRST CONDITION IS WHEN AR58 CLEARS AND  
;* AR59 SETS AND THE SECOND CONDITION IS WHEN AR58 AND AR59 BOTH CLEAR.  
;* SINCE THESE TWO BITS CANNOT BE EXPLICITLY STORED A RIGHT SHIFT BY 2  
;* IS MADE, AFTER THE LEFT SHIFT, TO CHECK THESE TWO CONDITIONS. THE CHECK DATA  
;* FOR LEFT SHIFTS IS STORED IN "COMON TAGS" RATHER THAN BEING CALCULATED.  
;* NOTE: SYNC POINT ONLY SELECTABLE ON MSN AND STAO AND STCO  
;* FPU ROM FLOW-30,37,163  
;;*****  
TST116: SCOPE  
MOV #STN-1,STESTN ;;SET TEST NUMBER IN MAIL BOX  
MOV #TST117,$ESCAPE ;;ESCAPE TO TEST 117 ON ERROR  
;CHECK THE ROM FLOW OF MSN  
MSN0: BIT #SW9,@SWR ;SWITCH 9 ON?  
REQ .+10 ;BRANCH IF NO  
TSTB $ERFLG ;ANY ERRORS?  
BNE 4$ ;BRANCH IF YES  
CLR R5 ;USED TO CLEAR MAINTENANCE MODE  
MOV #1$,@#FPPVEC ;SET INTERRUPT VECTOR  
MOV #ROMMSN,R0 ;GET ADDRESS OF TABLE  
3$: MOVR (P0)+,R3 ;GET DATA FOR MICRO-BREAK REG  
BEQ 5$ ;BRANCH IF DONE  
LDFPS #FPM ;LOAD FPS REGISTER  
LDUP ;LOAD MICRO-BREAK REGISTER  
MOV #0,R4 ;PUT THE SHIFT COUNT IN R4  
;  
CLR R1  
2$: INC R1 ;WAIT FOR FPP  
BNE 2$ ;TO FINISH IF NO INTERRUPT  
LDFPS R5 ;CLEAR MAINTENANCE MODE  
MOVR -(R0),@#STMPO ;SAVE ADDRESS THAT FAILED
```

```

5818 027346 105037 001201      CLR  #STMP0+1      ;GET RID OF SIGN EXT
5819 027352 104034      ERROR 34          ;FAILED TO TRAP ON THE ROM STATE
5820 027354 022626      1$:  CMP  (SP)+,(SP)+ ;RESTORE STACK POINTER
5821 027356 000755      BP 3$           ;GO TO NEXT ROM STATE
5822 027360 104000      4$:  ERROR 0       ;ONLY EXECUTES WHILE LOOPING ON ERROR
5823 027362 170105      5$:  LDFPS R5       ;CLEAR MAINTENANCE MODE
5824                                ;*****
5825                                ;ROM FLOW OF, NOW CHECK THE SHIFTERS
5826 027364 052737 000002 001450  BTS  #BIT1,RSH7    ;ENSURE RSH7 DATA RESTORED
5827 027372 012737 027720 000244  MOV  #6$,PPPVEC    ;SET THE PPP VECTOR
5828 027400 012700 000010      MOV  #10,R0        ;SETUP LOOP1 COUNT
5829 027404 012701 001440      MOV  #RSH8,R1      ;GET ADDRESS OF DATA TO LOAD
5830 027410 012703 000002      7$:  MOV  #2,R3         ;SET LOOP2 COUNT
5831 027414 012702 001200      MOV  #STMP0,R2     ;GET ADDRESS OF CHECK LOCATION
5832 027420 012122      MOV  (R1)+,(R2)+  ;PUT DATA INTO CHECK LOCATIONS
5833 027422 012122      MOV  (R1)+,(R2)+  ;
5834 027424 012122      MOV  (R1)+,(R2)+  ;
5835 027426 012112      MOV  (R1)+,(R2)+  ;
5836 027430 170127 000200      8$:  LDFPS #FD       ;LOAD DST AND
5837 027434 172537 001200      LDD  STMP0,AC1     ;SRC ACCUMULATORS
5838 027440 172637 001200      LDD  STMP0,AC2     ;SAVE IN AC3
5839 027444 172701      LDD  AC1,AC3       ;CHANGE SRC EXP SO DIFF. IN
5840 027446 176627 177677      LDEXP #177677,AC2 ;EXPONENTS IS +300
5841
5842                                ;GENERATE CHECK DATA
5843 027452 042737 177600 001200  BIC  #177600,STMP0 ;GET RID OF EXPONENT
5844 027460 052737 000200 001200  BIS  #BIT7,STMP0   ;INSERT HIDDEN BIT
5845 027466 013704 001204      MOV  STMP2,R4      ;GET LEAST SIGNIFICANT
5846 027472 013705 001206      MOV  STMP3,R5      ;32 BITS TO RIGHT SHIFT
5847 027476 010002      MOV  R0,R2         ;GET LOOP COUNT
5848 027500 005402      NEG  R2            ;MAKE IT A RIGHT SHIFT COUNT
5849 027502 073402      ASHC R2,R4        ;SHIFT LS32 BITS TO THE RIGHT,
5850                                ;BY THE SHIFT COUNT
5851 027504 005037 001160      CLR  $REG0         ;INITIALIZE MASK
5852 027510 010237 001162      MOV  R2,$REG1     ;SAVE SHIFT COUNT
5853 027514 000261      11$: SEC          ;GENERATE A BIT MASK
5854 027516 006037 001160      ROR  $REG0         ;FOR THE HIGH ORDER PART
5855 027522 105237 001162      INCB $REG1        ;OF R4 SO THAT BITS FROM
5856 027526 001372      BNE  11$          ;STMP1 CAN BE INSERTED
5857 027530 043704 001160      BIC  $REG0,R4     ;CLEAR THE APPROPRIATE BITS
5858 027534 005137 001160      COM  $REG0         ;GET COMPLIMENT BIT MASK FOR HIGH WORD
5859 027540 010037 001162      MOV  R0,$REG1     ;GET LOOP COUNT
5860 027544 010346      MOV  R3,-(SP)     ;SAVE R3
5861 027546 013703 001202      MOV  STMP1,R3     ;GET HIGH WORD
5862 027552 012737 000020 001164  MOV  #20,$REG2    ;GENERATE LEFT SHIFT COUNT
5863 027560 163737 001162 001164  SUB  $REG1,$REG2  ;TO GET SIG. BITS OF HIGH WORD
5864 027566 072337 001164      ASH  $REG2,R3     ;SHIFT THE HIGH WORD
5865 027572 043703 001160      BIC  $REG0,R3     ;CLEAR OFF LOWER BITS
5866 027576 010337 001162      MOV  R3,$REG1     ;SAVE IT
5867 027602 012603      MOV  (SP)+,R3     ;RESTORE R3
5868 027604 053704 001162      BTS  $REG1,R4     ;SET THOSE BITS INTO R4
5869 027610 010437 001204      MOV  R4,STMP2     ;SAVE R4 & R5 AS
5870 027614 010537 001206      MOV  R5,STMP3     ;THE EXPECTED ANSWER
5871 027620 013704 001200      MOV  STMP0,R4     ;GET MOST SIGNIFICANT
5872 027624 013705 001202      MOV  STMP1,R5     ;32 BITS
5873 027630 073402      ASHC R2,R4        ;RIGHT SHIFT THEM

```


5874	027632	010437	001200		MOV	R4,STMP0	;SAVE AS THE EXPECTED
5875	027636	010537	001202		MOV	R5,STMP1	;ANSWER
5876							
5877							
5878	027642	012737	027650	001110			;EXECUTE THE TEST
5879	027650	010346			MOV	#.+6,#SLPERR	;SET ERROR LOOP
5880	027652	012703	000275		26\$:	MOV R3,-(SP)	;SAVE R3
5881	027656	170003			MOV	#275,R3	;LOAD THE MICRO-BREAK
5882	027660	170127	000220		LDUB		;TO TRAP IN THE ADDD FLOW
5883	027664	012603			LDFPS	#FD+FMH	
5884	027666	032777	001000	151242	MOV	(SP)+,R3	;RESTORE R3
5885	027674	001407			BIT	#SW9,#SWR	;LOOP ON ERROR?
5886	027676	105737	001103		BEQ	27\$;BRANCH IF NO
5887	027702	001404			TSTB	\$ERFLG	;ANY ERRORS YET?
5888	027704	172503			BEQ	27\$;BRANCH IF NO
5889	027706	172603			LDD	AC3,AC1	;RELOAD THE DST
5890	027710	176627	177677		LDD	AC3,AC2	;RELOAD THE SRC
5891	027714	172102			LDEXP	#177677,AC2	;MAKE EXP DIFF +300
5892	027716	170000			27\$:	ADDD AC2,AC1	;EXECUTED PARTIAL ADD
5893	027720	022626			CPCC		;WAIT FOR TRAP
5894	027722	032777	000400	151206	6\$:	CMP (SP)+,(SP)+	;RESTORE THE SP
5895	027730	001007			BIT	#SW8,#SWR	;LOAD MICRO-BREAK?
5896	027732	010346			BNE	28\$;BRANCH IF NO
5897	027734	117703	151176		MOV	R3,-(SP)	;SAVE R3
5898	027740	170127	000200		MOVB	#SWR,R3	
5899	027744	170003			LDFPS	#FD	;ENSURE FMH OFF
5900	027746	012603			LDUR		
5901	027750				MOV	(SP)+,R3	;RESTORE R3
5902	027750	010204			28\$:	MOV R2,R4	;PUT THE SHIFT COUNT IN R4
5903	027752	170004			MSN		;EXECUTE THE SHIFT
5904	027754	170005			STAO		;PUT AR INTO ACO
5905	027756	174037	001210		STD	ACO,STMP4	;SAVE AR DATA
5906	027762	042737	100000	001210	BIC	#BIT15,STMP4	;GET RID OF SIGN
5907	027770	170007			STQ0		;GET QR DATA
5908	027772	174037	001160		STD	ACO,\$REG0	;SAVE IT
5909	027776	042737	100000	001160	BIC	#BIT15,\$REG0	;GET RID OF SIGN
5910	030004	012704	001200		MOV	#STMP0,R4	;GET ADDRESS OF EXPECTED DATA
5911	030010	012705	001210		MOV	#STMP4,R5	;GET ADDRESS OF AP DATA
5912	030014	022425			CMP	(R4)+,(R5)+	;QUAD 3 OK?
5913	030016	001006			BNE	12\$;BRANCH IF NO
5914	030020	022425			CMP	(R4)+,(R5)+	;QUAD 2 OK?
5915	030022	001004			BNE	12\$;BRANCH IF NO
5916	030024	022425			CMP	(R4)+,(R5)+	;QUAD 1 OK?
5917	030026	001002			BNE	12\$;BRANCH IF NO
5918	030030	022425			CMP	(R4)+,(R5)+	;QUAD 0 OK?
5919	030032	001403			BEQ	13\$;BRANCH IF YES
5920	030034	010737	001170		17\$:	MOV R2,\$REG4	;SAVE SHIFT COUNT
5921	030040	104202			ERROR	202	;AR DATA BAD
5922	030042	012704	001200		13\$:	MOV #STMP0,R4	;GET ADDRESS OF EXPECTED DATA
5923	030046	012705	001160		MOV	#SREG0,R5	;GET ADDRESS OF RECEIVED DATA
5924	030052	022425			CMP	(R4)+,(R5)+	;QUAD 3 OK?
5925	030054	001006			BNE	14\$;BRANCH IF NO
5926	030056	022425			CMP	(R4)+,(R5)+	;QUAD 2 OK?
5927	030060	001004			BNE	14\$;BRANCH IF NO
5928	030062	022425			CMP	(R4)+,(R5)+	;QUAD 1 OK?
5929	030064	001002			BNE	14\$;BRANCH IF NO

5930	030066	022425				CMP	(R4)+,(R5)+	;QUAD 0 OK?
5931	030070	001403				BEQ	15\$;BRANCH IF YES
5932	030072	010237	001170		14\$:	MOV	R2,\$REG4	;SAVE SHIFT COUNT
5933	030076	104203				ERROR	203	;QR DATA BAD
5934	030100	105737	001103		15\$:	TSTB	\$ERFLG	;ANY ERRORS YET?
5935	030104	001261				BNE	26\$;BRANCH IF YES
5936	030106	014544				MOV	-(R5),-(R4)	;LOAD CHECK LOCATION
5937	030110	014544				MOV	-(R5),-(R4)	;WITH THE COMPLIMENT
5938	030112	014544				MOV	-(R5),-(R4)	;BIT PATTERN TO SHIFT.
5939	030114	014544				MOV	-(R5),-(R4)	;*
5940	030116	042714	177600			BIC	#177600,(R4)	;CLEAR EXPONENT
5941	030122	052714	077600			BIS	#77600,(R4)	;SET EXPONENT OF 377
5942	030126	005303				DEC	R3	;GO CHECK COMPLIMENT PATTERN
5943	030130	001402				BEQ	100\$	
5944	030132	000137	027430			JMP	8\$	
5945	030136	005300			100\$:	DEC	R0	;GO DO NEXT SHIFT COUNT
5946	030140	001402				BEQ	25\$	
5947	030142	000137	027410			JMP	7\$	
5948						;*****		
5949						;NOW CHECK THE LEFT SHIFT OPERATION		
5950	030146	012737	030256	000244	25\$:	MOV	#16\$,PPPVEC	;SET FP VECTOR
5951	030154	012737	030162	001110		MOV	#.+6,#\$LPERR	;SET ERROR LOOP
5952	030162	012702	000007			MOV	#7,R2	;PUT SHIFT COUNT IN R2
5953	030166	012703	001450			MOV	#LSH7,R3	;GET ADDRESS OF DATA TO LOAD
5954	030172	042713	000002			BIC	#BIT1,(R3)	;ADJUST DATA SO AN59 WON'T CLPAR
5955	030176	005037	001172			CLR	\$REG5	;INITIALIZE LOOP COUNTER
5956	030202	012737	000007	001170		MOV	#7,\$REG4	;SAVE SHIFT COUNT INCASE OF FAILURE
5957	030210	012700	001540			MOV	#LSH7CK,R0	;GET ADDRESS OF CHECK DATA
5958	030214	012701	001200			MOV	#STMP0,R1	;GET ADDRESS OF CHECK BUFFER
5959	030220	012021				MOV	(R0)+,(R1)+	;PUT CHECK DATA IN BUFFER
5960	030222	012021				MOV	(R0)+,(R1)+	;*
5961	030224	012021				MOV	(R0)+,(R1)+	;*
5962	030226	012021				MOV	(R0)+,(R1)+	;*
5963	030230	172513				LDD	(R3),AC1	;LOAD THE SRC & DST
5964	030232	172613				LDD	(R3),AC2	;WITH THE DATA TO SHIFT
5965	030234	176627	177677			LDEXP	#177677,AC2	;MAKE EXPONENTS DIFF. BY 300
5966	030240	012703	000275			MOV	#275,R3	
5967	030244	170003				LDUR		
5968	030246	170127	000220			LDFPS	#PM+PD	
5969	030252	172102				ADD	AC2,AC1	;EXECUTE PARTIAL ADD TO LOAD AR & QR
5970	030254	170000				CFCC		;WAIT FOR TRAP
5971	030256	022626			16\$:	CMP	(SP)+,(SP)+	;RESTORE THE SP
5972	030260	032777	000400	150650		BIT	#SWR,\$SWR	;SWITCH 8 ON?
5973	030266	001005				BNE	22\$;BRANCH IF YES
5974	030270	170127	000200			LDFPS	#PD	
5975	030274	117703	150636			MOV	@SWR,R3	
5976	030300	170003				LDUP		
5977	030302				22\$:			
5978	030302	010204				MOV	R2,R4	;PUT THE SHIFT COUNT IN R4
5979	030304	170004				MSN		;EXECUTE THE LEFT SHIFT
5980	030306	170005				STAO		;GET AR DATA
5981	030310	174037	001210			STD	AC0,STMP4	;SAVE IT
5982	030314	042737	100000	001710		BIC	#BIT15,STMP4	;GET RID OF SIGN
5983	030322	170007				STQO		;GET QR DATA
5984	030324	174037	001160			STD	AC0,\$REG0	;SAVE IT
5985	030330	042737	100000	001160		BIC	#BIT15,\$REG0	;GET RID OF SIGN

5986	030336	012701	001710		MOV	#STMP4,R1	;GET LAST ADR +2 OF CHECK BUFFER	
5987	030342	012700	001220		MOV	#STMP7+2,P0	;GET LAST ADDRESS +2 OF AR DATA	
5988	030346	024140			CMP	-(R1),-(R0)	;QUAD 0 OK?	
5989	030350	001006			BNE	17\$;BRANCH IF NO	
5990	030352	024140			CMP	-(R1),-(R0)	;QUAD 1 OK?	
5991	030354	001004			BNE	17\$;BRANCH IF NO	
5992	030356	024140			CMP	-(R1),-(R0)	;QUAD 2 OK?	
5993	030360	001007			BNE	17\$;BRANCH IF NO	
5994	030362	024140			CMP	-(R1),-(R0)	;QUAD 3 OK?	
5995	030364	001401			BEQ	18\$;BRANCH IF YES	
5996	030366	104207			ERROR	202	;AR DATA BAD	
5997	030370	012700	001160	17\$:	MOV	#SREG0,R0	;GET ADDRESS OF GR DATA	
5998	030374	022120		18\$:	CMP	(R1)+,(R0)+	;QUAD 3 OK?	
5999	030376	001006			BNE	19\$;BRANCH IF NO	
6000	030400	022120			CMP	(R1)+,(R0)+	;QUAD 2 OK?	
6001	030402	001004			BNE	19\$;BRANCH IF NO	
6002	030404	022120			CMP	(R1)+,(R0)+	;QUAD 1 OK?	
6003	030406	001002			BNE	19\$;BRANCH IF NO	
6004	030410	022120			CMP	(R1)+,(R0)+	;QUAD 0 OK?	
6005	030412	001401			BEQ	20\$;BRANCH IF YES	
6006	030414	104203			ERROR	203	;QR DATA BAD	
6007	030416	005737	001172	19\$:	TST	\$REG5	;LOOP COUNTER SET?	
6008	030422	001015		20\$:	BNE	21\$;BRANCH IF YES	
6009	030424	005237	001172		INC	\$REG5	;SET THE COUNTER	
6010	030430	012702	177776		MOV	#-2,R2	;PUT SHIFT COUNT IN R2	
6011	030434	010237	001170		MOV	R2,\$REG4	;SAVE IN CASE FAILURE	
6012	030440	012700	001560		MOV	#LRSHCK+10,R0	;GET ADDRESS OF CHECK DATA+10	
6013	030444	014041			MOV	-(R0),-(R1)	;PUT CHECK DATA IN CHECK BUFFER	
6014	030446	014041			MOV	-(R0),-(R1)	; *	
6015	030450	014041			MOV	-(R0),-(R1)	; *	
6016	030452	014041			MOV	-(R0),-(R1)	; *	
6017	030454	000717			BR	22\$;GO DO THE RIGHT SHIFT BY 2 & CHECK	
6018							;CHECK THAT AR59 GETS A ONE ON LEFT SHIPT	
6019	030456	022737	000002	001172	21\$:	CMP	#2,\$REG5	;COUNT AT 2 YET?
6020	030464	001467			BEQ	24\$;BRANCH IF YES	
6021	030466	005237	001172		INC	\$REG5	;INCREMENT THE COUNT	
6022	030472	012737	030566	000244	MOV	#23\$,PPPVEC	;SET PP VECTOR	
6023	030500	012737	030506	001110	MOV	#.+6,#\$LPERR	;SET ERROR LOOP	
6024	030506	012701	001710		MOV	#STMP4,R1		
6025	030512	052737	000002	001450	BIS	#BIT1,LSH7	;RESTORE LEFT SHIFT DATA	
6026	030520	012700	001570		MOV	#SLRSHCK+10,R0	;GET ADDRESS OF CHECK DATA+10	
6027	030524	014041			MOV	-(R0),-(R1)	;SAVE IN CHECK BUFFER	
6028	030526	014041			MOV	-(R0),-(R1)	; *	
6029	030530	014041			MOV	-(R0),-(R1)	; *	
6030	030532	014041			MOV	-(R0),-(R1)	; *	
6031	030534	012703	000275		MOV	#275,R3		
6032	030540	170003			LDUR			
6033	030542	170127	000220		LDPPS	#FD+PMM		
6034	030546	172537	001450		LDD	LSH7,AC1	;LOAD THE SHIFT DATA	
6035	030552	172637	001450		LDD	LSH7,AC7	;INTO THE SRC & DST ACC'S	
6036	030556	176627	177677		LDEXP	#177677,AC2	;MAKE EXPONENTS DIFF BY 300	
6037	030562	172102			ADDD	AC2,AC1	;EXECUTE PARTIAL ADD	
6038	030564	170000			CFCC		;WAIT FOR TRAP	
6039	030566	022626			CMP	(SP)+,(SP)+	;RESTORE THE SP	
6040	030570	032777	000400	150340	BIT	#SW8,ASWR	;SWITCH 8 ON?	
6041	030576	001005			BNE	99\$;BRANCH IF YES	


```

6098 030762 172427 040000 3S: LDF #040000,AC0 ;INITIALIZE THE DST
6099 030766 172527 140000 LDF #0140000,AC1 ;AND THE SRC
6100 030772 032777 001000 150136 ADD0: BIT #SM9,SMR ;SWITCH 9 ON?
6101 031000 001403 BEQ .+10 ;BRANCH IF NO
6102 031002 105737 001103 TSTB $ERFLG ;ANY ERRORS?
6103 031006 001027 BNE 4$ ;BRANCH IF YES
6104 031010 005005 CLR R5 ;USED TO CLEAR MAINTENANCE MODE
6105 031012 012737 031062 000244 MOV #1$,@FPPVEC ;SET INTERRUPT VECTOR
6106 031020 012700 031124 MOV #ADD1R0,R0 ;GET ADDRESS OF TABLE
6107 031024 112003 3S: MOVB (R0)+,R3 ;GET DATA FOR MICRO-BREAK REG
6108 031026 001420 BEQ 5$ ;BRANCH IF DONE
6109 031030 170127 000020 LDFPS #FNM ;LOAD FPS REGISTER
6110 031034 170003 LDUB ;LOAD MICRO-BREAK REGISTER
6111 031036 172001 ADDF AC1,AC0 ;EXECUTE INSTRUCTION
6112 031040 005001 CLR R1
6113 031042 005201 2S: INC R1 ;WAIT FOR FPP
6114 031044 001376 BNE 2$ ;TO FINISH IF NO INTERRUPT
6115 031046 170105 LDFPS R5 ;CLEAR MAINTENANCE MODE
6116 031050 114037 001200 MOVB -(R0),@STMP0 ;SAVE ADDRESS THAT FAILED
6117 031054 105037 001201 CLR #STMP0+1 ;GET RID OF SIGN EXT
6118 031060 104034 ERROR 34 ;FAILED TO TRAP ON THE ROM STATE
6119 031062 022626 1S: CMP (SP)+,(SP)+ ;RESTORE STACK POINTER
6120 031064 000757 BR 3$ ;GO TO NEXT ROM STATE
6121 031066 104000 4S: ERROR 0 ;ONLY EXECUTES WHILE LOOPING ON ERROR
6122 031070 170105 5S: LDFPS R5 ;CLEAR MAINTENANCE MODE
6123 031072 170437 001200 CLR #STMP0 ;SAVE EXPECTED DATA
6124 031076 104055 ERROR 5$ ;ROM FLOW OK, DATA BAD
6125 ;*****
6126 ;DATA OK, CHECK THE CC'S
6127 031100 022737 000024 001200 ADD5: CMP #FNM+FZ,$TMP0 ;CC'S OK?
6128 031106 001412 BEQ TST120 ;BRANCH IF YES
6129 031110 012737 000024 001202 MOV #FNM+FZ,$TMP1 ;SAVE EXPECTED DATA
6130 031116 104071 ERROR 71 ;CC'S BAD
6131 ;2F3 BRANCH FAILED
6132 031120 022626 ADD4: CMP (SP)+,(SP)+ ;RESTORE THE SP
6133 031122 104204 ERROR 204 ;FRMF SUB*SC<@ NOT GOING LOW
6134 031124 065 240 271 ADD1R0: .BYTE 65,240,271,74,263,311,0
6135 031127 074 263 311
6136 031132 000
6137 031134 .EVEN
6138
6139
6140 ;*****
6141 ;*TEST 120 SUBF*W0*(SRC=DST)*-(SS=SD)
6142 ;*
6143 ;* THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS THE 2F3 BRANCH.
6144 ;* THIS WILL ONLY FAIL IF FRMF SS XOR SD XOR SUB DOES NOT GO LOW.
6145 ;*
6146 ;* FPU ROM FLOW - 30, 65, 240, 271, 74, 263, 311
6147 ;*****
6148 031134 000004 TST120: SCOPE
6149 031136 012737 000120 001240 MOV #STN-1,$TFSTN ;;SET TEST NUMBER IN MAIL BOX
6150 031144 012737 031234 001222 MOV #TST121,$ESCAPE ;;ESCAPE TO TEST 121 ON ERROR
6151 031152 012737 031230 000744 MOV #2$,FPPVEC
6152 031160 012703 000275 MOV #275,R3 ;SET MICRO-BREAK TRAP TO
6153 031164 170003 LDUB ;CATCH 2F3 BRANCH FAILURE

```

```
6154 031166 170127 000020
6155 031172 012737 031200 001110
6156 031200 172527 040000
6157 031204 172401
6158 031206 173001
6159 031210 170500
6160 031212 170000
6161 031214 001407
6162 031216 174037 001204
6163 031222 170437 001200
6164 031226 104035
6165
6166 031230 022626
6167 031232 104205
6168
6169
6170
6171
6172
6173
6174
6175
6176
6177
6178
6179 031234 000004
6180 031236 012737 000121 001240
6181 031244 012737 031376 001222
6182 031252 012700 001200
6183 031256 012720 177777
6184 031262 012720 177777
6185 031266 012720 177777
6186 031272 012720 177777
6187 031276 012737 031304 001110
6188 031304 170127 000200
6189 031310 012700 001200
6190 031314 170400
6191 031316 172020
6192 031320 022700 001210
6193 031324 001401
6194 031326 104066
6195 031330 174037 001210
6196 031334 022737 177777 001210
6197 031342 001014
6198 031344 022737 177777 001212
6199 031352 001010
6200 031354 022737 177777 001214
6201 031362 001004
6202 031364 022737 177777 001216
6203 031372 001401
6204 031374 104125
6205
6206
6207
6208
6209
```

```
LDFPS #FMM
MOV #.+6,#SLPERR ;SET ERROR LOOP
LDF #040000,AC1 ;LOAD THE SRC
LDF AC1,ACO ;AND DST
1S: SUBF AC1,ACO ;EXECUTE INSTRUCTION UNDER TEST
TSTF ACO ;TEST ANSWER JUST TO BE SURE
CPCC
BEQ TST121 ;;BRANCH IF ANSWER OK
STF ACO,$TMP2 ;SAVE ANSWER
CLRF $TMP0 ;SAVE EXPECTED ANSWER
ERROR 35 ;DON'T KNOW WHAT HAPPENED
;2F3 BRANCH FAILED
2S: CMP (SP)+,(SP)+
ERROR 205 ;FRMF SS XOR SD XOR SUB NOT GOING LOW

;;*****
;*TEST 121 ADDD*-MO*-(SRC=0)*(DST=0)
;*
;* THIS TEST USES THE ADDF AND ADDD INSTRUCTIONS TO TEST THE
;* LDAC6*PD(1) FLOWS. THE DESTINATION WILL BE
;* SET TO ZERO SO THAT THE RESULT WILL BE THE SOURCE.
;*
;* FPU ROM FLOW-20,140,270,150,65,750,306
;;*****
TST121: SCOPE
MOV #STM-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX
MOV #TST122,$ESCAPE ;;ESCAPE TO TEST 122 ON ERROR
MOV #STMP0,P0 ;GET ADDRESS OF DATA BUFFER
MOV #-1,(R0)+ ;PUT SRC DATA IN BUFFER
MOV #-1,(R0)+ ; *
MOV #-1,(R0)+ ; *
MOV #-1,(R0)+ ; *
MOV #-1,(R0)+ ; *
MOV #-1,(R0)+ ; *
LDFPS #FD
MOV #STMP0,P0 ;GET ADDRESS OF DATA BUFFER
CLRD ACO ;ENSURE DST CLEAR
1S: ADDD (P0)+,ACO ;EXECUTE INSTRUCTION UNDER TEST
CMP #STMP4,R0 ;ADX ROM OK?
BEQ 2S ;BRANCH IF YES
ERROR 66 ;ADX ROM FAILED
2S: STD ACO,$TMP4 ;GET DATA BACK
CMP #-1,$TMP4 ;QUAD 3 OK?
BNE 3S ;BRANCH IF NO
CMP #-1,$TMP5 ;QUAD 2 OK?
BNE 3S ;BRANCH IF NO
CMP #-1,$TMP6 ;QUAD 1 OK?
BNE 3S ;BRANCH IF NO
CMP #-1,$TMP7 ;QUAD 0 OK?
BEQ TST122 ;;BRANCH IF YES
3S: ERROR 125 ;A QUADRANT DIDN'T GET LOADED PROPERLY

;;*****
;*TEST 122 CMPF*MO*(SRC=DST)
;*
```

6210
6211
6212
6213
6214 031376 000004
6215 031400 012737 000122 001240
6216 031406 012737 031462 001222
6217 031414 012737 047624 000244
6218 031422 172527 140000
6219 031426 172401
6220 031430 170127 000013
6221 031434 173401
6222 031436 170237 001200
6223 031442 022737 000004 001200
6224 031450 001404
6225 031452 012737 000004 001202
6226 031460 104071
6227
6228
6229
6230
6231
6232
6233
6234
6235
6236
6237
6238
6239
6240 031462 000004
6241 031464 012737 000123 001240
6242 031472 012737 031664 001222
6243 031500 172427 044177
6244 031504 012700 177777
6245 031510 170127 000012
6246 031514 175400
6247 031516 170237 001200
6248 031522 005700
6249 031524 001405
6250 031526 010037 001162
6251 031532 005037 001164
6252 031536 104141
6253 031540 022737 000005 001200
6254 031546 001404
6255 031550 012737 000005 001202
6256 031556 104071
6257
6258
6259 031560
6260 031560 012737 031566 001110
6261 031566 012737 031660 000244
6262 031574 012703 000374
6263 031600 170003
6264 031602 172427 050177
6265 031606 170127 000132

```
;* THE ONLY THING THAT SHOULD FAIL IS THE SUBTRACTION IN ROM STATE 255.
;*
;* FPU ROM FLOW - 30, 67, 320, 302, 255, 323, 135
;*****
TST122: SCOPE
MOV #STN-1,STESTM ;;SET TEST NUMBER IN MAIL BOX
MOV #TST123,$ESCAPE ;;ESCAPE TO TEST 123 ON ERROR
MOV #FPPSPUR,FPPVEC
LDF #0140000,AC1 ;LOAD THE SPC AND
LDF AC1,ACO ;DST ACCUMULATORS
LDFPS #13 ;LOAD COMPLIMENT CC'S
1S: CMPF AC1,ACO ;EXECUTE INSTRUCTION UNDER TEST
STFPS $TMP0 ;GET CC'S
CMP #FZ,$TMP0 ;CC'S OK?
BEQ TST123 ;;BRANCH IF YES
MOV #FZ,$TMP1 ;SAVE EXPECTED VALUE
ERROR 71 ;CC'S BAD

;*****
;*TEST 123 STCFI/STCPL*(EXP=2**16/2**32)*(SD=+)
;*
;* THE ONLY POSSIBLE FAILURE WITH THIS TEST IS THE SUBTRACTION IN ROM
;* STATE 51 OR 374 OR 370.
;*
;* INTEGER MODE IS TESTED FIRST, AND THEN INTEGER LONG. WHEN INTEGER LONG IS
;* TESTED, THE A BRANCH COULD FAIL.
;*
;* FPU ROM FLOW - 51, 374/370, 333, 372, 335, 345, 214/204
;*****
TST123: SCOPE
MOV #STN-1,STESTM ;;SET TEST NUMBER IN MAIL BOX
MOV #TST124,$ESCAPE ;;ESCAPE TO TEST 124 ON ERROR
LDF #044177,ACO ;LOAD THE SRC WITH EXP=2**16
MOV #-1,R0 ;ENSURE THE DST IN NOW-ZERO
LDFPS #12 ;LOAD COMPLIMENT CC'S
1S: STCFI ACO,R0 ;EXECUTE INSTRUCTION UNDER TEST
STFPS $TMP0 ;GET CC'S
TST R0 ;DID R0 GET THE DATA?
BEQ 2S ;BRANCH IF YES
MOV R0,$REG1 ;SAVE RECEIVED DATA
CLR $REG2 ;SAVE EXPECTED DATA
ERROR 141 ;DATA BAD
2S: CMP #FZ+FC,$TMP0 ;CC'S OK?
BEQ 3S ;BRANCH IF YES
MOV #5,$TMP1 ;SAVE EXPECTED VALUE
ERROR 71 ;CC'S BAD

;*****
;NOW CHECK INTEGER LONG
3S:
MOV #-6,#SLPERR ;SET ERROR LOOP
MOV #4,$FPPVEC ;SETUP FP VECTOR
MOV #374,R3 ;SET MICRO-BREAK TO CATCH
LDUR ;A 5F1 BRANCH FAILURE
LDF #050177,ACO ;LOCAL SPC WITH EXP=2**32
LDFPS #FL+FMH+12 ;LOAD COMPLIMENT CC'S
```

6266	031612	005000				CLR	R0		;ENSURE DST IS NON-ZERO
6267	031614	175400				STCFL	AC0,R0		;EXECUTE INSTRUCTION UNDER TEST
6268	031616	170237	001200			STFPS	\$TMP0		;GET CC'S
6269	031622	005700				TST	R0		;IS DST DATA OK?
6270	031624	001405				BEQ	5\$;BRANCH IF YES
6271	031626	010037	001160			MOV	R0,\$RFG0		;SAVE RECEIVED DATA
6272	031632	005037	001162			CLR	\$REG1		;SAVE EXPECTED DATA
6273	031636	104141				ERROR	141		;DATA BAD
6274	031640	022737	000125	001200	5\$:	CMP	\$FL+PMM+FZ+FC,\$TMP0		;CC'S OK?
6275	031646	001406				BEQ	TST124		;BRANCH IF YES
6276	031650	012737	000125	001202		MOV	\$FL+PMM+5,\$TMP1		;SAVE EXPECTED CC'S
6277	031656	104071				ERROR	71		;CC'S BAD
6278									;5F1 BRANCH FAILED
6279	031660	022626			4\$:	CMP	(SP)+,(SP)+		;RESTORE THE SP
6280	031662	104206				ERROR	206		;FRM J IL(0)H NOT GOING LOW OR NOT GETTING
6281									;TO FRMA RADO2 AS A HIGH
6282									
6283									
6284									
6285									
6286									
6287									
6288									
6289									
6290									
6291									
6292									
6293	031664	000004							
6294	031666	012737	000124	001240		TST124: SCOPE			
6295	031674	012737	032140	001222		MOV	\$STW-1,\$TESTN		;SET TEST NUMBER IN MAIL BOX
6296						MOV	\$TST125,\$ESCAPE		;ESCAPE TO TEST 125 ON ERROR
6297	031702	177027							;FIRST CHECK THE ADX ROM
6298	031704	000000			1\$:	LDCIF	(PC)+,AC0		;EXECUTE INSTRUCTION UNDER TEST
6299	031706	000403				.WOPD	0		;WILL HALT HERE IF ADX ROM FAILS
6300	031710	104066				BR	2\$		
6301	031712	104066				ERROR	66		;ADX ROM FAILED
6302	031714	104066				ERROR	66		; *
6303						ERROR	66		; *
6304	031716								;NOW CHECK THE ROM FLOW
6305	031716	032777	001000	147217	2\$:	STCIFO: BIT	\$SW9,\$SWR		;SWITCH 9 ON?
6306	031724	001403				BEQ	+.10		;BRANCH IF NO
6307	031726	105737	001103			TST	\$ERFLG		;ANY ERRORS?
6308	031732	001030				BNE	4\$;BRANCH IF YES
6309	031734	005005				CLR	R5		;USED TO CLEAR MAINTENANCE MODE
6310	031736	012737	032010	000244		MOV	\$I\$,\$FPPVEC		;SET INTERRUPT VECTOR
6311	031744	012700	032130			MOV	\$ROMSTC,R0		;GET ADDRESS OF TABLE
6312	031750	112003			3\$:	MOVR	(R0)+,R7		;GET DATA FOR MICRO-BREAK REG
6313	031752	001421				BEQ	5\$;BRANCH IF DONE
6314	031754	170127	000020			LDFPS	\$PMM		;LOAD FPS REGISTER
6315	031760	170003				LDUR			;LOAD MICRO-BREAK REGISTER
6316	031762	177027				LDCIF	(PC)+,AC0		;EXECUTE INSTRUCTION
6317	031764	000000				.WOPD	0		
6318	031766	005001				CLR	R1		
6319	031770	005201			2\$:	INC	R1		;WAIT FOR FPP
6320	031772	001376				BNE	2\$;TO FINISH IF NO INTERRUPT
6321	031774	170105				LDFPS	R5		;CLEAR MAINTENANCE MODE

6322 031776 114037 001200
6323 032002 105037 001201
6324 032006 104034
6325 032010 022626
6326 032012 000756
6327 032014 104000
6328 032016 170105
6329
6330
6331 032020 012737 032026 001110
6332 032026 012737 147777 001210
6333 032034 012737 177777 001212
6334 032042 172437 001210
6335 032046 170127 000013
6336 032052 177027
6337 032054 000000
6338 032056 170237 001200
6339 032062 174037 001204
6340 032066 005737 001204
6341 032072 001003
6342 032074 005737 001206
6343 032100 001403
6344 032102 170437 001200
6345 032106 104035
6346
6347 032110 022737 000004 001200
6348 032116 001410
6349 032120 012737 000004 001202
6350 032126 104071
6351 032130 174 213 314
6352 032133 312 253 245
6353 032136 000
6354 032140
6355
6356
6357
6358
6359
6360
6361
6362
6363
6364
6365
6366
6367
6368
6369
6370
6371
6372
6373
6374
6375
6376
6377

```
MOVB -(R0),@#STMP0 ;SAVE ADDRESS THAT FAILED
CLRR @#STMP0+1 ;GET RID OF SIGN EXT
ERROR 34 ;FAILED TO TRAP ON THE ROM STATE
1S: CMP (SP)+,(SP)+ ;RESTORE STACK POINTER
BR 3S ;GO TO NEXT ROM STATE
4S: ERROR 0 ;ONLY EXECUTES WHILE LOOPING ON ERROR
5S: LDFPS R5 ;CLEAR MAINTENANCE MODE
;*****
;NOW CHECK THE RESULT
MOV #.+6,@#SLPERR ;SET ERROR LOOP
MOV #147777,STMP4 ;PUT DATA TO INITIALIZE
MOV #-1,STMP5 ;DST INTO MEMORY
LDF STMP4,ACO ;LOAD THE DST
LDFPS #13 ;LOAD COMPLIMENT CC'S
LDCIF (PC)+,ACO ;EXECUTE INSTRUCTION UNDER TEST
.WORD 0
STFPS STMP0 ;SAVE CC'S
STF ACO,STMP2 ;GET DATA BACK
TST STMP2 ;QUAD 3 OK?
BWE 6S ;BRANCH IF NO
TST STMP3 ;QUAD 2 OK?
BEQ 7S ;BRANCH IF YES
6S: CLRF STMP0 ;SAVE EXPECTED VALUE
ERROR 35 ;DATA BAD
;NOW CHECK THE CC'S
7S: CMP #FZ,STMP0 ;CC'S OK?
BEQ TST125 ;;BRANCH IF YES
MOV #FZ,STMP1 ;SAVE EXPECTED VALUE
ERROR 71 ;CC'S BAD
ROMSTC: .BYTE 174,213,314,312,253,245,0
.EVEN
;*****
;*TEST 125 STCDF*M0*--(EXP=0)
;*
;* NONE OF THE MICRO-BRANCHES SHOULD FAIL.
;*
;* IF FRHC FALU 59 DOES NOT GET TO FRHK NORM POS ENCODER AND FRHL A CONTRCL 3
;* MUX AS A LOW, THE EXPONENT WILL INCREASE BY 1.
;*
;* IF FRHC FALU59 DOES NOT GET TO FRHK NORM POS ENCODER AS A LOW,
;* THE EXPONENT WILL DECREASE BY 7 AND THE FRACTION WILL BE SHIFTED LEFT BY 7.
;*
;* IF FRHC FALU59 DOES NOT GET TO FRHL ASHF CONT 3A(0) AS A LOW OR IF
;* FRHM SHIF CNT 3 DOES NOT GO LOW, THE EXPONENT WILL INCREASE BY 8.
;*
;* IF A BIT IN FRHK NORM POS ENCODER FAILS TO GO LOW OR FAILS TO GET
;* THROUGH TO FRHM SHIF CNT OR IF FRHM SHIF CNT FAILS TO GET TO THE
;* BMX ON FXP, THE EXPONENT WILL BE DECREASED BY A NUMBER BETWEEN 1 AND 7.
;*
;* IF FRHC FD(1) B L DOES NOT GO HIGH, DRIVEN BY FXPB STCF L, ROUNDING WILL
;* NOT OCCUR
;*
;* FPU ROM FLOW-44,35,166,121,360,377,157
```

```

6378                                     ;;*****
6379 032140 000004                       TST125: SCOPE
6380 032142 012737 000125 001240        MOV    #STN-1,STESTN    ;;SET TEST NUMBER IN MAIL BOX
6381 032150 012737 032452 001222        MOV    #TST126,SESCAPE ;;ESCAPE TO TEST 126 ON ERROR
6382 032156 012737 177777 001200        MOV    #-1,STMP0       ;PUT DATA IN
6383 032164 012737 177777 001202        MOV    #-1,STMP1       ;MEMORY TO
6384 032172 012737 177777 001204        MOV    #-1,STMP2       ;INITIALIZE THE DST
6385 032200 012737 040000 001210        MOV    #40000,STMP4    ;PUT DATA IN
6386 032206 005037 001217                CLR    STMP5           ;MEMORY TO
6387 032212 012737 100000 001214        MOV    #BIT15,STMP6    ;INITIALIZE THE SRC
6388 032220 012737 032226 001110        MOV    #-6,#$SLPERR    ;SET ERROR LOOP
6389 032226 170127 000200                LDFPS  #FD
6390 032232 172437 001200                LDD    STMP0,AC0       ;LOAD THE DST
6391 032236 172537 001210                LDD    STMP4,AC1       ;LOAD THE SRC
6392 032242 170127 000217                LDFPS  #FD+17         ;SET COMPLIMENT CC'S
6393 032246 176100                        1S:   STCDF  AC1,AC0       ;EXECUTE INSTRUCTION UNDER TEST
6394 032250 170200                        STFPS  R0              ;GET CC'S
6395 032252 170127 000000                LDFPS  #0              ;CLEAR THE FD BIT
6396 032256 174037 001204                STF    AC0,STMP2      ;GET THE DST BACK
6397 032262 012737 040000 001200        MOV    #40000,STMP0    ;SAVE THE EXPECTED
6398 032270 012737 000001 001202        MOV    #1,STMP1       ;VALUE OF THE DST
6399 032276 022737 040000 001204        CMP    #40000,STMP2   ;QUAD 3 OK?
6400 032304 001011                        BNE    2S              ;BRANCH IF NO
6401 032306 022737 000001 001206        CMP    #1,STMP3       ;QUAD 2 OK?
6402 032314 001445                        BEQ    3S              ;BRANCH IF YES
6403 032316 005737 001206                TST    STMP3           ;DID ROUND FAIL?
6404 032322 001001                        BNE    4S              ;BRANCH IF NO
6405 032324 104210                        ERROR  210             ;FRHC FD(1) B L DID NOT GO HIGH
6406 032326 104035                        4S:   ERROR  3S              ;DATA BAD
6407                                     ;QUAD 3 IS BAD-CHECK EXPONENT
6408 032330 013700 001204                2S:   MOV    STMP2,R0   ;GET QUAD 3 IN R0
6409 032334 042700 000177                BIC    #177,R0         ;CLEAR FRACTION BITS
6410 032340 022700 040000                CMP    #40000,R0      ;IS EXPONENT OK?
6411 032344 001001                        BNE    5S              ;BRANCH IF NO
6412 032346 104035                        ERROR  3S              ;EXPONENT OK-FRACTION BAD
6413 032350 022700 037600                5S:   CMP    #37600,R0   ;DID EXPONENT DECREASE BY ONE?
6414 032354 001001                        BNE    6S              ;BRANCH IF NO
6415 032356 104211                        ERROR  211             ;FRHC FALU59 NOT GETTING TO FRHK & FRHL
6416                                     ;AS A LOW.
6417 032360 022700 042000                6S:   CMP    #42000,R0   ;DID FRACTION INCREASE BY 8?
6418 032364 001001                        BNE    7S              ;BRANCH IF NO
6419 032366 104217                        ERROR  212             ;FRHC FALU59 NOT GETTING TO FRHL ASHF
6420                                     ;CONT 3A(0) AS A LOW.
6421 032370 022700 036200                7S:   CMP    #36200,R0   ;DID EXPONENT DECREASE BY 7?
6422 032374 001407                        BEQ    8S              ;BRANCH IF YES
6423 032376 002401                        BLT    9S              ;BRANCH IF EXPONENT DECREASED BY LESS THAN 7
6424 032400 104035                        ERROR  3S              ;EXPONENT DECREASED BY MORE THAN 7
6425 032402 022700 040000                9S:   CMP    #40000,R0   ;DID EXPONENT DECREASE BY MORE THAN
6426                                     ;ZERO AND LESS THAN 7?
6427 032406 003001                        BGT    10S            ;BRANCH IF YES
6428 032410 104035                        ERROR  3S              ;EXPONENT INCREASED BY MORE THAN 1
6429 032412 104213                10S:  ERROR  213         ;1 BIT OF FRHK NORM POS ENCODER DID
6430                                     ;NOT GO LOW OR DID NOT GET THPU
6431
6432                                     ;EXPONENT DECREASED BY 7-DETERMINE IF FALU59 FAILED.
6433 032414 022737 000200 001206        8S:   CMP    #200,STMP3 ;DID FRAC. SHFT LEFT BY 7?

```

```

6434 032422 001001      BNE      11$      ;BRANCH IF NO
6435 032424 104214      ERROR    214      ;FRHC FALU59 DOES NOT GET TO FRHK
6436                                     ;NORM POS ENCODER AS A LOW.
6437 032426 104213      11$:      ERROR    213      ;SEE 10$ ABOVE
6438                                     ;EXPONENT AND FRACTION OK-CHECK CC'S
6439 032430 042700 000200 3$:      BIC      #FD,RO    ;GET RID OF FD BIT
6440 032434 005700      TST      RO        ;CC'S OK?
6441 032436 001405      BEQ      TST126    ;BRANCH IF YES
6442 032440 010037 001200      MOV      RO,$TMP0  ;SAVE RECEIVED VALUE
6443 032444 005037 001202      CLR      $TMP1     ;SAVE EXPECTED VALUE
6444 032450 104071      ERROR    71        ;CC'S BAD
6445
6446
6447
6448                                     ;*****
6449                                     ;*TEST 126      STCDF*-MO*IMM
6450                                     ;*
6451                                     ;*      IF FRHL ASHP CONT 3A(0) DOES NOT GO HIGH THE EXPONENT WILL BE
6452                                     ;*      SMALLER BY 7.
6453                                     ;*
6454                                     ;*      IF A BIT OF FRHK NORM POS ENCODER DOES NOT GO HIGH THE EXPONENT
6455                                     ;*      WILL BE LARGER BY A NUMBER BETWEEN 2 AND 8.
6456                                     ;*
6457                                     ;*      FPU ROM FLOW-44,35,166,121,300,377,153,145
6458                                     ;*****
6458 032452 000004      TST126: SCOPE
6459 032454 012737 000126 001240      MOV      #STN-1,$TESTN ;SET TEST NUMBER IN MAIL BOX
6460 032462 012737 032646 001222      MOV      #TST127,$ESCAPE ;ESCAPE TO TEST 127 ON ERROR
6461 032470 012737 032642 000010      MOV      #3$,RESVEC    ;SET UP RESVEC TO CATCH ADX ROM FAILURE
6462 032476 005037 032544      CLR      2$          ;INITIALIZE LOCATION WHERE RESULT IS STORED
6463 032502 012737 007577 001210      MOV      #7577,$TMP4   ;PUT DATA TO LOAD INTO SRC
6464 032510 012737 177777 001212      MOV      #-1,$TMP5     ;IN MEMORY EXP=36 FRAC=ALL ONE'S
6465 032516 012737 100000 001214      MOV      #BIT15,$TMP6  ;MAKE SURE ROUND OCCURS
6466 032524 012737 032532 001110      MOV      #-+6,#$SLPERR ;SET ERROR LOOP
6467 032532 170127 000200      LDFPS   #FD
6468 032536 172437 001210      LDD     $TMP4,AC0     ;LOAD THE SRC
6469 032542 176027      1$:      STCDF   AC0,(PC)+    ;EXECUTE INSTRUCTION UNDER TEST
6470 032544 000000      2$:      .WORD  0          ;SHOULD GET CHANGED TO 7600
6471 032546 000403      BR      4$          ;IF ADX ROM FAILS A RESERVED TRAP WILL OCCUR
6472 032550 104066      ERROR   66         ;ADX ROM FAILED
6473 032552 104066      ERROR   66         ;
6474 032554 104066      ERROR   66         ;
6475 032556 022737 007600 032544 4$:      CMP     #7600,2$     ;DATA COME OUT OK?
6476 032564 001430      BEQ     TST127      ;BRANCH IF YES
6477 032566 013700 032544      MOV     2$,RO       ;PUT THE ANS IN RO
6478 032572 010037 001162      MOV     RO,$REG1    ;SAVE ANS FOR TYPEOUT
6479 032576 012737 007600 001164      MOV     #7600,$REG2 ;SAVE EXPECTED VALUE
6480 032604 042700 000177      BTC     #177,RO     ;GET RID OF FRACTION
6481 032610 022700 005600      CMP     #5600,RO    ;DID EXPONENT DECREASE BY 7?
6482 032614 001001      BNE     5$          ;BRANCH IF NO
6483 032616 104216      ERROR   216        ;FRHL ASHP CONT 3A(0) NOT GOING HIGH
6484                                     ;ON FRHC AR59 HIGH
6485 032620 022700 010000      5$:      CMP     #10000,RO   ;DID EXPONENT INCREASE BY 2 OR MORE?
6486 032624 003401      BLE     6$          ;BRANCH IF YES
6487 032626 104141      ERROR   141        ;DATA BAD
6488 032630 022700 010600      6$:      CMP     #10600,RO   ;DID EXP INCREASE BY MORE THAN 10?
6489 032634 002401      BLT     7$          ;BRANCH IF YES

```

```

6490 032636 104717 ERROR 217 ;A BIT OF FRHK WGRM POS ENCODER IS
6491 ;NOT GETTING TO THE BMX ON FXP
6492 ;AS A HIGH
6493 032640 104141 7S: ERROR 141 ;DATA BAD
6494 032642 072626 3S: CMP (SP)+,(SP)+ ;RESTORE THE SP
6495 032644 104066 ERROR 66 ;ADX PGM FATLFD
6496
6497
6498
6499
6500
6501
6502
6503
6504
6505

```

```

;;*****
;*TEST 127 STCFI*M0*(EXP=2**15)
;*
```

```

;* IF FXPJ EALU06 XOR EALU03 DOES NOT CAUSE EALU SWR TO GO LOW,
;* THE RESULT WILL BE 52525 SINCE ONLY A RIGHT SHIFT BY 1 WILL
;* GET EXECUTED.
;*
```

```

;* FPU ROM FLOW-51, 374, 333, 371, 375, 345, 214
;*
```

```

;;*****
```

```

6506 032646 000004 TST127: SCOPE
6507 032650 012737 000127 001240 MOV #STN-1,$TFSTN ;;SET TEST NUMBER IN MAIL BOX
6508 032656 012737 032777 001227 MOV #TST130,$ESCAPE ;;ESCAPE TO TEST 130 ON ERROR
6509 032664 012737 043600 001204 MOV #43600,$STMP2 ;PUT DATA TO INITIALIZE
6510 032677 012737 175252 001206 MOV #125252,$STMP3 ;THE SRC IN MEMORY
6511 032700 012737 032706 001110 MOV #.+6,$SLPERR ;SET ERROR LCCP
6512 032706 172437 001204 LDF STMP2,AC0 ;LOAD THE SPC
6513 032712 170127 000017 LDFPS #17 ;LOAD COMPLIMENT CC'S
6514 032716 175400 1S: STCFI AC0,R0 ;EXECUTE INSTRUCTION UNDER TEST
6515 032720 170737 001700 STFPS $STMP0 ;SAVE CC'S
6516 032724 022700 040125 CMP #40125,R0 ;IS DATA OK?
6517 032730 001412 BEQ 2S ;BRANCH IF YES
6518 032737 022700 052525 CMP #52525,R0 ;DID DATA ONLY GET SHIFTED 1 BIT?
6519 032736 001001 BNE 3S ;BRANCH IF NO
6520 032740 104220 ERROR 220 ;FXPJ EALU06 XOR EALU03 NOT
6521 ;GOING LOW
6522 032742 010037 001167 3S: MOV R0,$REG1 ;SAVE RECEIVED DATA
6523 032746 012737 040125 001164 MOV #40125,$REG2 ;SAVE EXPECTED DATA
6524 032754 104141 ERROR 141 ;DATA BAD
6525 ;DATA OK-CHECK THE CC'S
6526 032756 005737 001700 2S: TST $STMP0 ;CC'S OK?
6527 032762 001403 BEQ TST130 ;;BRANCH IF YES
6528 032764 005037 001202 CLR $STMP1 ;SAVE EXPECTED VALUE
6529 032770 104071 ERROR 71 ;CC'S BAD
6530
6531
6532
6533
6534
6535
6536
6537
6538
6539
6540
6541
6542

```

```

;;*****
;*TEST 130 STCFL*-M0*(EXP=2**31)
;*
```

```

;* THE ONLY THING THAT SHOULD FAIL IS THE SUBTRACTION IN ROM STATE 370
;* OR THE 7F1 BRANCH.
;*
```

```

;* IF FRMA IL+IMMEDIATE DCFS NOT GO LOW, EXECUTION WILL GO TO ROM
;* STATE 210 INSTEAD OF 200. THIS WOULD CAUSE THE CPU NOT TO STORE
;* THE SECOND WORD.
;*
```

```

;* FPU ROM FLOW-51, 370, 333, 371, 375, 345, 200, 66, 215
;*
```

```

;;*****
```

```

6543 032772 000004 TST130: SCOPE
6544 032774 012737 000130 001240 MOV #STN-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX
6545 033002 012737 033116 001222 MOV #TST131,$ESCAPE ;;ESCAPE TO TEST 131 ON ERROR

```

```

6546 033010 012737 047600 001204
6547 033016 012737 125252 001206
6548 033024 012737 033032 001110
6549 033032 172437 001204
6550 033036 170127 000100
6551 033042 012700 001204
6552 033046 175420
6553 033050 022700 001210
6554 033054 001401
6555 033056 104066
6556 033060 022737 040125 001204
6557 033066 001004
6558 033070 022737 052400 001206
6559 033076 001407
6560 033100 012737 040125 001700
6561 033106 012737 052400 001202
6562 033114 104221
6563
6564
6565
6566
6567
6568
6569
6570
6571
6572
6573
6574
6575
6576
6577
6578
6579
6580
6581
6582
6583
6584
6585
6586
6587
6588 033116 000004
6589 033120 012737 000131 001240
6590 033126 012737 033476 001222
6591 033134 012737 033244 000244
6592 033142 012703 000265
6593 033146 170003
6594 033150 012737 040000 001200
6595 033156 005037 001202
6596 033162 005037 001204
6597 033166 012737 000001 001206
6598 033174 170127 000220
6599 033200 172437 001200
6600 033204 172037 001200
6601 033710 170000
  
```

```

MOV #47600,STMP2 ;PUT THE SRC DATA IN MEMORY
MOV #125252,STMP3 ;EXP=237
MOV #.+6,0#SLPERR ;SET ERROR LOOP
LDF STMP2,ACO ;LOAD THE SPC
LDFPS #FL ;SET THE FL BIT
MOV #STMP2,PO ;SET UP ADDRESS OF DST IN RO
15: STCFL ACO,(RO)+ ;EXECUTE INSTRUCTION UNDER TEST
CMP #STMP4,PO ;ADX ROM OK?
BEQ 25 ;BRANCH IF YES
ERROR 66 ;ADX ROM FAILED
25: CMP #40125,STMP2 ;FIRST WORD OK?
BNE 35 ;BRANCH IF NO
CMP #52400,STMP3 ;SECOND WORD OK?
BEQ TST131 ;;BRANCH IF YES
35: MOV #40125,STMP0 ;SAVE EXPECTED
MOV #52400,STMP1 ;VALUE OF DATA
ERROR 221 ;DATA BAD
  
```

```

;*****
;*TEST 131 ADDD*-M0*IMM*(SRC=DST)
;*
;* THIS TEST DOES AN ADDD WITH THE EXPONENTS EQUAL AND THE
;* FRACTIONS EQUAL EXCEPT FOR THE LEAST SIGNIFICANT BIT. THIS IS DONE
;* TO TEST THE ROUND LOGIC AT BIT POSITION 2 OF THE FMX.
;*2F3 BRANCH
;* IF FRWF ADD*SC<8 DOES NOT GO LOW, EXECUTION WILL GO TO STATE
;* 271.
;*
;* IF FXPP OUT OF RANGE DOES NOT GO LOW OR DOES NOT
;* GET TO FXPJ EALU SWR AS A HIGH, EXECUTION
;* WILL GO TO STATE 275.
;*
;* THE 3F1 BRANCH, IN THE NORMALIZE FLOWS, SHOULD NOT FAIL SINCE IT
;* HAS ALREADY BEEN TESTED.
;* THE 5F2J BRANCH SHOULD NOT FAIL.
;*
;* NOTE: SYNC POINT NOT SELECTABLE DURING FIRST PART OF
;* TEST. DEFAULT=265
  
```

```

;* FPU ROM FLOW-20,141,65,740,265,313,317
;*****
TST131: SCOPE
MOV #STN-1,STESTN ;;SET TEST NUMBER IN MAIL BOX
MOV #TST132,$ESCAPE ;;ESCAPE TO TEST 132 ON ERROR
MOV #25,FPPVEC
MOV #265,R3 ;LOAD MICRO-BREAK TO TEST ROM FLOW
95: LDUR
MOV #40000,STMP0 ;PUT THE SRC
CLR STMP1 ;AND DST DATA
CLR STMP2 ;INTO MEMORY
MOV #BIT0,STMP3 ;
LDFPS #FMM+PD
LDD STMP0,ACO ;LOAD THE DST
ADDD STMP0,ACO ;EXECUTE INSTRUCTION
CFCC ;WAIT FOR INTERRUPT.
  
```

```

6602          ;FAILURE-NOT TAKING CORRECT ROM FLOW TRY THE 2 POSSIBLE 2P3 BRANCH FAILURES
6603 033212 022703 000275      CMP      #275,R3      ;TRIED 275 YET?
6604 033216 001403          BEQ      8$          ;BRANCH IF YES
6605 033220 012703 000275      MOV      #275,R3      ;GO TRY STATE
6606 033224 000750          BR       9$          ;275 FAILURE
6607 033226 022703 000271      8$:     CMP      #271,R3      ;TRIED 271 YET?
6608 033232 001403          BEQ      10$         ;BRANCH IF YES
6609 033234 012703 000271      MOV      #271,R3      ;TRY STATE
6610 033240 000742          BR       9$          ;271 FAILURE
6611          ;CAN'T TELL WHATS HAPPENING-GO FINISH REST OF TEST
6612 033242 000411      10$:    BP       11$
6613          ;*****
6614          ;INTERRUPT OCCURRED-FIND OUT IF ITS CORRECT
6615 033244 022626      2$:     CMP      (SP)+,(SP)+ ;RESTORE SP
6616 033246 022703 000265      CMP      #265,P3      ;MAKING CORRECT BRANCH?
6617 033252 001405          BEQ      11$         ;BRANCH IF YES
6618 033254 022703 000275      CMP      #275,R3      ;FATLING TO STATE 275?
6619 033260 001001          BNE     12$         ;BRANCH IF NO
6620 033262 104227          ERROR   277         ;FXPP OUT OF RANGE DID NOT GO LOW
6621 033264 104224      12$:    ERROR   274         ;FRNF SS XOR SD XOR SUB DID NOT GC HIGH
6622 033266          11$:
6623 033266 012737 033274 001110      MOV      #.+6,#SLPERR ;SET ERROR LOOP
6624 033274 012737 047624 000244      MOV      #FPPSPUR,FPPVEC ;SET FPP VECTOR
6625 033302 012700 177777          MOV      #-1,R0      ;INITIALIZE R0 TO CATCH ADX ROM FAILURE
6626 033306 170127 000200          LDPPS   #FD
6627 033312 172437 001200          LDF     $TMP0,AC0    ;LOAD THE DST
6628 033316 005037 001206          CLR     $TMP3
6629 033322 170127 000217          LDPPS   #FD+17      ;LOAD COMPLIMENT CC'S
6630 033326 172027      1$:     ADDD   (PC)+,AC0    ;EXECUTE INSTRUCTION UNDER TEST
6631 033330 040000          .WORD  40000       ;WILL EXECUTE AND CLR R0 IF ADX ROM FAILS
6632 033332 000403          BP      3$
6633 033334 104066          EPROR   66          ;ADX ROM FAILED
6634 033336 104066          EPROR   66          ;
6635 033340 104066          ERROR   66          ;
6636 033342 005700      3$:     TST     R0          ;DID R0 CLEAR?
6637 033344 001001          BNE     4$          ;BRANCH IF NO
6638 033346 104066          EPROR   66          ;ADX ROM FAILED
6639 033350 012737 040200 001200      4$:     MOV      #40200,$TMP0 ;SAVE EXPECTED VALUE OF QUAD 3
6640 033356 170200          STFPS   R0          ;SAVE CC'S
6641 033360 174037 001210          STD     AC0,$TMP4    ;GET DATA BACK
6642 033364 022737 040200 001210      CMP      #40200,$TMP4 ;QUAD 3 OK?
6643 033372 001020          BNE     5$          ;BRANCH IF NO
6644 033374 005737 001212          TST     $TMP5       ;QUAD 2 OK?
6645 033400 001025          BNE     6$          ;BRANCH IF NO
6646 033402 005737 001214          TST     $TMP6       ;QUAD 1 OK?
6647 033406 001405          BEQ     13$         ;BRANCH IF YES
6648 033410 022737 100000 001214      CMP      #PIT15,$TMP6 ;DID PMX 34 FAIL?
6649 033416 001016          BNE     6$          ;BRANCH IF NO
6650 033420 104240          ERROR   240         ;FKHC PMX34 DID NOT GO LOW
6651 033422 022737 000001 001216      13$:    CMP      #1,$TMP7    ;QUAD 0 OK?
6652 033430 001412          BEQ     7$          ;BRANCH IF YES
6653 033432 104222          EPROR   222         ;FRLC FMX02 DID NOT GO HIGH OR
6654          ;FRLC ASHFR02 DID NOT GO HIGH WITH
6655          ;AR03 ON A HIGH (RIGHT SHFT 1)
6656 033434 013700 001210      5$:     MOV      $TMP4,R0    ;SAVE $TMP4
6657 033440 042700 000177          BTC     #177,R0     ;GET RID OF FRACTION BITS
  
```

```
6658 033444 022700 040000      CMP      #40000,R0      ;DID EXPONENT FAIL TO INCREMENT?
6659 033450 001001              BNE      65           ;BRANCH IF NO
6660 033452 104223              ERROR    223          ;STATE 313 DID NOT SUBTRACT A MINUS ONE
6661                                ;FROM THE EXPONENT (ADD ONE)
6662 033454 104125              65:      ERROR    125          ;QUAD IS BAD
6663                                ;DATA OK-CHECK THE CC'S
6664 033456 042700 000200      75:      BIC      #FD,R0      ;CC'S OK?
6665 033462 001405              BEQ      TST132       ;BRANCH IF YES
6666 033464 010037 001200      MCV      R0,STMP0     ;SAVE RECEIVED DATA
6667 033470 005037 001202      CLR      STMP1        ;SAVE EXPECTED DATA
6668 033474 104071              ERROR    71           ;CC'S BAD
6669
6670                                ;*****
6671                                ;*TEST 132      SUBD*-MO*(SRC>DST*(-(SS=SD)
6672                                ;*
6673                                ;*2F3 BRANCH
6674                                ;*      IF FRMP SS XOR SD XOR SUB DOES NOT GO HIGH EXECUTION WILL GO TO STATE
6675                                ;*      273.  A MICRO-BREAK TRAP WILL BE SET TO CATCH THIS PAYLOAF.
6676                                ;*
6677                                ;*      IF THE ABS VAL ROM FAILS TO PUT 177 IN THE EALU, EXECUTION
6678                                ;*      WILL GO TO STATE 277.
6679                                ;*
6680                                ;*      THE FT BIT WILL BE SET TO TEST THAT FPLC FNX02 IS DISABLED.
6681                                ;*
6682                                ;*      FPU ROM FLOW-30,65,240,267,313,317
6683                                ;*****
6684 033476 000004              TST132: SCOPE
6685 033500 012737 000132 001240      MOV      #STN-1,STESTN ;SET TEST NUMBER IN MAIL BOX
6686 033506 012737 033754 001222      MOV      #TST133,$ESCAPE ;ESCAPE TO TEST 133 ON ERROR
6687 033514 012703 000273              MOV      #273,R3
6688 033520 170003              LDUB
6689 033522 012737 033530 001110      MOV      #.+6,#$SLPERR ;SET ERROR LOOP
6690 033530 012737 140000 001200      MCV      #140000,STMP0 ;PUT DST DATA
6691 033536 005037 001202              CLR      STMP1        ;IN MEMORY
6692 033547 005037 001204              CLR      STMP2        ;EXP=200,FRAC=.5+<0>=1
6693 033546 012737 000001 001206      MOV      #P10,STMP3   ;TO TEST TRUNCATE
6694 033554 170127 000200              LDFPS   #FD
6695 033560 172437 001200              LDD     STMP0,ACO     ;LOAD THE DST
6696 033564 012737 040200 001200      MOV      #40200,STMP0 ;PUT SRC DATA IN MEMORY
6697 033572 005037 001206              CLR      STMP3        ;EXP=201 AND FRAC=.5
6698 033576 012737 033750 000244      MOV      #25,FPPVEC   ;SET FPP VECTOR
6699 033604 012700 001200              MOV      #STMP0,F0    ;PUT ADDRESS OF SRC DATA IN R0
6700 033610 105737 001103              TSTR   SER*LG        ;ANY ERRORS?
6701 033614 001003              BNE      75           ;BRANCH IF YES
6702 033616 170127 000267              LDFPS   #FD+FT+FM*7  ;LOAD COMPLIMENT CC'S
6703 033622 000402              BP      1$
6704 033624 170127 000247              75:      LDFPS   #FD+FT+7
6705 033630 173020              15:      SUBD   (R0)+,ACO  ;EXFCUTE INSTRUCTION UNDER TEST
6706 033632 170201              STFPS   R1           ;GET CC'S
6707 033634 022700 001210      CMP      #STMP4,R0    ;ADY ROM OK?
6708 033640 001401              BEQ     3$           ;BRANCH IF YES
6709 033642 104066              EPROR   66          ;ADY ROM FAILED
6710 033644 174020              35:      STD    ACO,(R0)+   ;GET DATA BACK
6711 033646 022700 001220      CMP      #STMP7+2,R0 ;ADY ROM OK CN STD?
6712 033652 001401              BEQ     4$           ;BRANCH IF YES
6713 033654 104066              ERROR   66          ;ADY ROM FAILED
```

```

6714 033656 022737 140300 001210 4S:  CMP    #140300,STMP4    ;QUAD 3 OK?
6715 033664 001012                BNE    5$              ;BRANCH IF NO
6716 033666 005737 001212                TST    STMP5           ;QUAD 2 OK?
6717 033672 001007                BNE    5$              ;BRANCH IF NO
6718 033674 005737 001214                TST    STMP6           ;QUAD 1 OK?
6719 033700 001004                BNE    5$              ;BRANCH IF NO
6720 033702 005737 001216                TST    STMP7           ;QUAD 0 OK?
6721 033706 001405                BEQ    6$              ;BRANCH IF YES
6722 033710 104225                ERROR  275            ;PRLC FMX02 DID NOT GO LOW
6723 033712 012737 140300 001200 5S:  MOV    #140300,STMP0   ;SAVE EXPECT QUAD 3
6724 033720 104125                ERROR  125            ;A QUAD IS BAD
6725                                ;DATA OK-CHECK CC'S
6726 033722 042701 000260                6S:  BTC    #FD+FT+FMH,R1 ;GET CC'S ONLY
6727 033726 022701 000010                CMP    #FN,R1         ;CC'S OK?
6728 033732 001410                BEQ    TST133         ;;BRANCH IF YES
6729 033734 010137 001200                MOV    R1,STMP0       ;SAVE RECEIVED CC'S
6730 033740 012737 000010 001202                MOV    #FN,STMP1      ;SAVE EXPECTED CC'S
6731 033746 104071                ERROR  71             ;CC'S BAD
6732                                ;2P3 BRANCH FAILED
6733 033750 022626                2S:  CMP    (SP)+,(SP)+  ;RESTORE THE SP
6734 033752 104226                ERROR  226            ;PRMF SS XCH SD XCR SUB DID NOT GO HIGH
6735
6736
6737
6738
6739
6740
6741
6742
6743
6744
6745
6746
6747
6748
6749
6750
6751
6752

```

```

;;*****
;*TEST 133      ADDD*MO*(SRC<DST)*-(SS=SD)
;*
;*      THE 2P3 BRANCH WILL ONLY FAIL IF THE ABS VAL RCM FAILS. THIS
;*      WOULD CAUSE EXECUTION TO GO TO STATE 275 INSTEAD OF 271.
;*
;*      LIKewise, THE 3P0 BRANCH WILL ONLY FAIL IF THE ABS VAL RCM FAILS
;*      AND PUTS ZERO IN THE ER.
;*
;*      ANY OTHER FAILURE WOULD BE CAUSED BY CONTROL STORE SIGNALS IN STATES
;*      271 OR 75.
;*
;*      A MODE 0 INSTRUCTION IS USED TO TEST THE A-BRANCH MON.
;*
;*      FPU ROM FLOW-30,65,240,271,75,317
;;*****

```

```

6753 033754 000004                TST133: SCOPE
6754 033756 012737 000133 001240                MOV    #STN-1,STESTN  ;;SET TEST NUMBER IN MAIL BOX
6755 033764 012737 034200 001222                MOV    #TST134,$FSCAPE ;;ESCAPE TO TEST 134 ON ERROR
6756 033772 012737 034174 000244                MOV    #2$,FPPVEC     ;SET FP VECTOR
6757 034000 012703 000275                MOV    #275,R3        ;LOAD MICRO-BREAK
6758 034004 170003                LDUP                    ;TO CATCH ABS VAL RCM FAILURE
6759 034006 012737 034014 001110                MOV    #.+6,#$SLPERR  ;SET ERROR LOOP
6760 034014 012737 040000 001200                MOV    #40000,STMP0   ;PUT SRC DATA
6761 034022 005037 001202                CLR    STMP1          ;INTO MEMORY
6762 034026 005037 001204                CLR    STMP2          ;
6763 034032 005037 001206                CLR    STMP3          ;
6764 034036 170127 000200                LDPPS  #FD            ;
6765 034042 172537 001200                LDD    STMP0,AC1      ;LOAD THE SRC EXP=200 FRAC=.5
6766 034046 012737 140300 001700                MOV    #140300,STMP0  ;PUT DST DATA IN MEMORY
6767 034054 172437 001200                LDD    STMP0,AC0      ;LOAD THE DST EXP=201,FRAC=-.75
6768 034060 105737 001103                TSTR  $ERFLG         ;ANY ERRORS YET?
6769 034064 001003                BNE    5$              ;BRANCH IF YES

```


6770	034066	170127	000227		LDFPS	#PD+FMH+7	;LOAD COMPLIMENT CC'S
6771	034072	000407			BR	15	
6772	034074	170127	000207		5S: LDFPS	#PD+7	
6773	034100	172001			1S: ADDD	AC1,AC0	;EXECUTE INSTRUCTION UNDER TEST
6774	034102	170200			STFPS	R0	;GET CC'S
6775	034104	174037	001210		STD	AC0,\$TMP4	;GET DATA BACK
6776	034110	022737	140200	001210	CMP	#140200,\$TMP4	;QUAD 3 OK?
6777	034116	001011			BNE	35	;BRANCH IF NO
6778	034120	005737	001212		TST	\$TMP5	;QUAD 2 OK?
6779	034124	001006			BNE	35	;BRANCH IF NO
6780	034126	005737	001214		TST	\$TMP6	;QUAD 1 OK?
6781	034132	001003			BNE	35	;BRANCH IF NO
6782	034134	005737	001216		TST	\$TMP7	;QUAD 0 OK?
6783	034140	001404			BEQ	45	;BRANCH IF YES
6784	034142	012737	140200	001200	3S: MOV	#140200,\$TMP0	;SAVE EXPECTED DATA
6785	034150	104125			ERROR	125	;QUAD IS BAD
6786							;DATA OF-CHECK CC'S
6787	034152	022700	000730		4S: CMP	#PD+FMH+FN,R0	;CC'S OK?
6788	034156	001410			BEQ	TST134	;BRANCH IF YES
6789	034160	010037	001700		MOV	R0,\$TMP0	;SAVE RECEIVED VALUE
6790	034164	012737	000730	001202	MOV	#PD+FMH+FN,\$TMP1	;SAVE EXPECTED VALUE
6791	034172	104071			ERROR	71	;CC'S BAD
6792	034174	022626			2S: CMP	(SP)+,(SP)+	;RESTORE THE SP
6793	034176	104230			ERROR	230	;ABS VAL ROM FAILED
6794							
6795							
6796							
6797							
6798							
6799							
6800							
6801							
6802							
6803							
6804							
6805	034200	000004					
6806	034202	012737	000134	001240	MOV	#STN-1,\$TESTN	;SET TEST NUMBER IN MAIL BOX
6807	034210	012737	034406	001222	MOV	#TST135,\$ESCAPE	;ESCAPE TO TEST 135 ON ERROR
6808	034216	012700	001200		MOV	#STMP0,R0	;PUT ADDRESS OF BUFFER IN R0
6809	034222	012720	040000		MOV	#40000,(R0)+	;PUT DST DATA IN
6810	034226	012720	000001		MOV	#BIT0,(R0)+	;MEMORY
6811	034232	172437	001700		LDF	\$TMP0,AC0	;LOAD THE DST EXP=200,FRAC=.5+(0)=1
6812	034236	012737	140540	001200	MOV	#140540,\$TMP0	;PUT SRC DATA IN MEMORY
6813	034244	005037	001207		CLR	\$TMP1	;EXP=201,FRAC=-.111
6814	034250	012700	001700		MOV	#STMP0,R0	;PUT ADDRESS OF SRC DATA IN R0
6815	034254	170127	000147		LDFPS	#FL+FT+7	;LOAD COMPLIMENT CC'S
6816	034260	172020			1S: ADDD	(R0)+,AC0	;EXECUTE INSTRUCTION UNDER TEST
6817	034262	022700	001704		CMP	#STMP7,R0	;ADX ROM OK?
6818	034266	001401			BFC	45	;BRANCH IF YES
6819	034270	104066			ERROR	66	;ADX ROM FAILED
6820	034272	170201			4S: STFPS	R1	;GET CC'S
6821	034274	170127	000000		LDFPS	R0	;CLR THE FL BIT
6822	034300	170202			STFPS	R2	;SAVE FPS TO ENSURE A BRANCH FROM CK
6823	034302	174037	001204		STF	AC0,\$TMP2	;GET DATA BACK
6824	034306	012737	140477	001700	MOV	#140477,\$TMP0	;SAVE EXPECTED DATA
6825	034314	012737	177777	001202	MOV	#-1,\$TMP1	

6826	034322	022737	140477	001204	CMP	#140477,STMP2	;QUAD 3 OK?
6827	034330	001005			BNE	25	;BRANCH IF NO
6828	034332	022737	177777	001206	CMP	#-1,STMP3	;QUAD 2 OK?
6829	034340	001402			BEQ	35	;BRANCH IF YES
6830	034342	104731			ERROR	231	;PRMC FMX 34 FAILED TO GO LOW WITH FT(1)
6831	034344	104055			25: ERROR	55	;QUAD 3 WRONG
6832							
6833							
6834	034346	022701	000150		;DATA OR-NOW CHECK CC'S		
6835	034352	001406			35: CMP	#PL+FT+FN,R1	;CC'S OK ON ADDP?
6836	034354	012737	000150	001202	BEQ	55	;BRANCH IF YES
6837	034362	010137	001200		MOV	#PL+FT+FN,STMP1	;SAVE EXPECTED VALUE
6838	034366	104071			MOV	R1,STMP0	;SAVE RECEIVED VALUE
6839	034370	005702			ERROR	71	;CC'S FAILED
6840	034372	001405			55: TST	R2	;CC'S OK ON LDPPS?
6841	034374	005037	001202		BEQ	TST135	;BRANCH IF YES
6842	034400	010237	001200		CLR	STMP1	;SAVE EXPECTED VALUE
6843	034404	104071			MOV	R2,STMP0	;SAVE RECEIVED VALUE
6844					ERROR	71	;CC'S BAD AFTER LDPPS WITH PL BIT CN.

```

;*****
;TEST 135      NORMALIZATION SHIFT ENCODER
;
; THIS TEST IS COMPOSED OF THREE SUBTESTS.  THE FIRST SUBTEST RUNS
; A COUNT PATTERN THROUGH AR BITS <57:51> WITH AR BITS <59:58>00.
;
; THE SECOND SUBTEST RUNS A COUNT PATTERN THROUGH AR BITS <57:51>
; WITH AR BITS <59:58>01.
;
; THE THIRD SUBTEST TESTS THE UNIQUE DATA PATTERN WHERE AR BITS <59:51>
; ARE ALL ZERO.  THIS IS THE ONLY PATTERN THAT CAUSES PRMC NORM POS
; SWR TO GO LOW.
;
; IF AN ERROR IS DETECTED, LOOP ON ERROR WILL CAUSE THE FAILING
; DATA PATTERN TO BE LOCPED ON.  THE ERROR TIMEOUT INCLUDES
; THE AD DATA PATTERN (BITS <59:51>) THAT IS TRYING TO BE
; NORMALIZED.
;
; REFER TO THE FLOW CHART AT THE END OF THIS
; LISTING FOR FURTHER INFORMATION.
;
; FPU ROM FLOW-11,130,65,240,271,75,317
;*****

```

6869	034406	000004			TST135: SCOPE		
6870	034410	012737	000135	001240	MOV	#STN-1,STESTN	;SET TEST NUMBER IN MAIL BOX
6871					;SECTION 1		
6872	034416	172427	040200		LDF	#040200,AC0	;INITIALIZE THE DST TO 1.0
6873	034422	172500			LDF	AC0,AC1	;STORAGE TO REINITIALIZE DST
6874	034424	012737	040000	001210	MOV	#40000,STMP4	;INITIALIZE THE
6875	034432	005037	001212		CLR	STMP5	;SOURCE DATA
6876	034436	172637	001210		LDF	STMP4,AC2	;SAVE IN AC2
6877	034442	012737	036400	001200	MOV	#36400,STMP0	;INITIALIZE THE
6878	034450	005037	001202		CLR	STMP1	;CHECK DATA
6879	034454	012737	000376	001160	MOV	#376,COUNT	;INITIALIZE COUNT
6880	034462	012737	000002	001162	MOV	#2,STEP	;INITIALIZE STEP
6881	034470	012737	000200	001164	MOV	#200,OFFSET	;INITIALIZE OFFSET

```
6882 034476 012700 000177      MOV      #127.,R0      ;INITIALIZE LOOP COUNT
6883                                ;START OF THE LOOP
6884 034502 013701 001160      LOOP1:  MOV      COUNT,R1      ;GET COUNT
6885 034506 162701 000700      SUB      #200,R1      ;FRACTION=COUNT-200
6886 034512 100012                BPL      2S           ;BRANCH IF FRACTION NOT NEGATIVE
6887 034514 006337 001162      ASL      STEP        ;STEP=STEP X 2
6888 034520 162737 000200 001210  SUB      #BIT7,$TMP4    ;DECREMENT SRC EXPONENT
6889 034526 172637 001210      LDF      $TMP4,AC2    ;SAVE IN AC2
6890 034532 006337 001160      ASL      COUNT       ;COUNT=COUNT X 2
6891 034536 000761                BR       LOOP1       ;TRY AGAIN
6892 034540 050137 001210      2S:    BTS      R1,$TMP4  ;LOAD THE SRC FRACTION
6893 034544 173037 001210      LOOP2:  SUBF     $TMP4,AC0  ;EXECUTE THE SUBTRACT
6894 034550 173437 001200      CMPF    $TMP0,AC0    ;RESULT OK?
6895 034554 170000                CFCC
6896 034556 001042                BNE     $ERR        ;BRANCH IF NO
6897 034560 032777 001000 144350  BIT     #SW9,$SWR    ;LOOP ON ERROR
6898 034566 001403                BEQ     NOLOOP      ;BRANCH IF NO
6899 034570 105737 001103      TSTB    $ERRPLG     ;ANY ERRORS YET?
6900 034574 001033                BNE     $ERR        ;BRANCH IF YES
6901 034576 013701 001200      NOLOOP: MOV     $TMP0,R1  ;GET CHECK DATA
6902 034602 042701 177600      BTC     #177600,R1  ;FRACTION
6903 034606 042737 000177 001200  BIC     #177,$TMP0  ;CLEAR FRACTION IN CHECK DATA
6904 034614 063701 001164      ADD     OFFSET,R1  ;FRACT=FRACT+OFFSFT
6905 034620 105701                TSTR    R1         ;FRACTION OVERFLOW?
6906 034622 100006                BPL     3S         ;BRANCH IF NO
6907 034624 062737 000200 001200  ADD     #BIT7,$TMP0  ;INCREMENT EXPONENT
6908 034632 006237 001164      ASR     OFFSET     ;OFFSFT=OFFSET/2
6909 034636 000402                BR      4S
6910 034640 050137 001200      3S:    BTS      R1,$TMP0  ;INSERT NEW CHECK FRACTION
6911 034644 163737 001162 001160  4S:    SUB     STEP,COUNT  ;COUNT=COUNT-STEP
6912 034652 172401                LDF     AC1,AC0     ;REINITIALIZE DST
6913 034654 174237 001210      STF     AC2,$TMP4   ;REINITIALIZE SRC
6914 034660 077070                SOB     R0,LOOP1   ;CONTINUE
6915 034662 000437                BR     $ECT2       ;GO TO SECTION-2
6916 034664 012737 034746 001110  $ERR:  MOV     #75,$LPERP  ;SET ERROR LOOP
6917 034672 174037 001204      STF     AC0,$TMP2   ;SAVE RESULT
6918 034676 172401                LDF     AC1,AC0     ;RESTORE DST
6919 034700 012703 000075      MOV     #75,R3     ;SET MICRO-BREAK TRAP
6920 034704 170003                LDUR
6921 034706 012737 034726 000244  MOV     #15,$PPVEC  ;TO GET THE FAILING AR PATTERN
6922 034714 170127 000020      LDPPS  #PMM        ;SET FP VECTOR
6923 034720 173037 001210      SUBF   $TMP4,AC0   ;EXECUTE SUB ON DATA PATTERN
6924 034724 170000                CFCC
6925 034726 022626                1S:    CMP     (SP)+,(SP)+ ;WAIT FOR INTERRUPT
6926 034730 013704 177776      MOV     -2,R4     ;RESTORE THE SP
6927 034734 170004                MSH
6928 034736 170005                STAO
6929 034740 174037 001214      STF     AC0,$TMP6  ;PUT AR DATA INTO AC0
6930 034744 104232                EPROR  232        ;PUT AR DATA IN MEMCRV
6931 034746 032777 001000 144162  2S:    BIT     #SW9,$SWR  ;NORMALIZATION FAILED
6932 034754 001710                BEQ     NOLOOP     ;LOOP ON ERROR?
6933 034756 172401                LDF     AC1,AC0    ;BRANCH IF NO-CONTINUE TEST
6934 034760 000671                BR     LOOP2       ;RESTORE THE DST
6935                                ;GO TRY SAME DATA PATTERN
6936                                ;*****
6937                                ;SECTION 2
```

H14

6938	034767	012737	040000	001210	SECT2:	NOV	#40000,STMP4	;INITIALIZE THE
6939	034770	005037	001212			CLR	STMP5	;SRC DATA
6940	034774	012737	040300	001170		NOV	#40300,SREG4	;INITIALIZE THE
6941	035002	005037	001172			CLR	SREG5	;DST DATA
6942	035006	012737	040200	001700		NOV	#40200,STMP0	;INITIALIZE CHECK DATA
6943	035014	005037	001202			CLR	STMP1	
6944	035020	012737	000040	001162		NOV	#40,STEP	;INITIALIZE STEP
6945	035026	012737	000100	001160		NOV	#100,COUNT	;INITIALIZE COUNT
6946	035034	012700	000177			NOV	#127,R0	;SET LOOP COUNT
6947	035040	012701	000002			NOV	#7,R1	;SET LOOP COUNT
6948	035044	013737	001160	001164	LOOP4:	NOV	COUNT,OFFSET	
6949								
6950	035052	105737	001160					
6951	035056	100014			LOOP3:	TSTR	COUNT	;IS COUNT NEGATIVE?
6952	035060	162737	000200	001210		BPL	3S	;BRANCH IF NO
6953	035066	013737	001164	001160		SUB	#BIT7,STMP4	;DECREMENT SRC EXPONENT
6954	035074	063737	001162	001160		MOV	OFFSET,COUNT	;COUNT=OFFSET + STEP
6955	035102	006237	001162			ADD	STEP,COUNT	
6956	035106	000756				ASR	STEP	;STEP=STEP/2
6957	035110	042737	000177	001170	3S:	BR	LOOP4	;TRY AGAIN
6958	035116	053737	001160	001170		BIC	#177,SREG4	;CLEAR DST FRACTION
6959	035124	172437	001170		LOOP5:	BIS	COUNT,SREG4	;DST FRACT=COUNT
6960	035130	173037	001210			LDF	SREG4,AC0	;LOAD THE DESTINATION
6961	035134	173437	001200			SUBF	STMP4,AC0	;EXECUTE THE INSTRUCTION
6962	035140	170000				CMPF	STMP0,AC0	;RESULT OK?
6963	035142	001031				CFCC		
6964	035144	032777	001000	143764		BNE	\$ERR1	;BRANCH IF NO
6965	035152	001403				BIT	#SM9,MSR	;LOOP ON ERROR?
6966	035154	105737	001103			BEQ	NLOOP1	;BRANCH IF NO
6967	035160	001022				TSTR	\$ERP1G	;ANY ERPOPS YET?
6968	035162	005737	001700		NLOOP1:	BNE	\$ERR1	;BRANCH IF YES
6969	035166	005237	001160			IPC	STMP0	;CHECK FRACT=CHK FRACT + 1
6970	035172	077051				INC	COUNT	;COUNT=COUNT+1
6971	035174	012737	036200	001210		SOB	R0,LOOP3	;CONTINUE
6972	035202	012737	040377	001170		MOV	#36200,STMP4	;LOAD SRC FRACTION
6973	035210	012737	100000	001172		MOV	#40377,SREG4	;LOAD THE
6974	035216	012700	000001			MOV	#BIT15,SREG5	;DST FRACTION
6975	035222	077140				MOV	#1,R0	;FAKE OUT FIRST SOB LOOP
6976	035224	000437				SOB	R1,LOOP5	;CHECK LAST PATTERN
6977	035226	012737	035312	001110	\$ERP1:	BR	SECT3	;GO TO SECTION 3
6978	035234	174037	001704			MOV	#75,\$LPERF	;SET LOOP ADDRESS
6979	035240	172437	001170			STF	AC0,STMP2	;SAVE RESULT
6980	035244	012703	000075			LDF	SREG4,AC0	;RESTORE DST DATA
6981	035250	170003				MOV	#75,R3	;SET MICRO-BREAK TRAP
6982	035252	012737	035777	000744		LDUR		;TO GET FAILING AR PATTERN
6983	035260	170127	000020			MOV	#15,PPPVEC	;SET FPP VECTOR
6984	035264	173037	001210			LDFPS	#PMM	
6985	035270	170000				SUBF	STMP4,AC0	;EXECUTE INSTRUCTION
6986	035272	072626			1S:	CFCC		;WAIT FOR TRAP
6987	035274	012704	177776			CMP	(SP)+,(SP)+	;RESTORE THE SP
6988	035300	170004				MOV	#-2,R4	;PUT THE SHIFT COUNT IN R4
6989	035302	170005				MSN		;SHIFT THE AR TO GET BITS 59&58
6990	035304	174037	001214			STAC		;PUT AR IN AC0
6991	035310	104232				STF	AC0,STMP6	;PUT AR DATA IN MEMCRV
6992	035312	032777	001000	143616	2S:	EPROR	232	;NORMALIZATION FAILED
6993	035320	001720				BIT	#SM9,MSR	;LOOP ON ERPCR?
						BEQ	NLOOP1	;BRANCH IF NO

```

6994 035322 000700          BR          LOOPS          ;GO TRY SAME DATA PATTERN
6995
6996
6997
6998 035324
6999 035324 012737 035332 001110      MOV          #.+6,#SLPERR      ;SET ERROR LOOP
7000 035332 172427 040200          LDF          #040700,AC0      ;LOAD THE DST
7001 035336 012737 040177 001210      MOV          #40177,$TMP4     ;PUT THE SRC
7002 035344 005037 001212          CLR          $TMP5           ;DATA IN MEMORY
7003 035350 012737 036200 001200      MOV          #36200,$TMP0     ;LOAD CHECK
7004 035356 005037 001202          CLR          $TMP1           ;DATA INTO MEMORY
7005 035362 173037 001210          SUBF         $TMP4,AC0       ;EXECUTE SUBTRACT
7006 035366 173437 001200          CMPF        $TMP0,AC0       ;RESULT OK?
7007 035372 001405          BEQ          TST136         ;;BRANCH IF YES
7008 035374 005037 001214          CLR          $TMP6           ;SAVE AR DATA PATTERN
7009 035400 174037 001204          STF         AC0,$TMP2       ;SAVE RESULT
7010 035404 104233          ERROR        233           ;FRNK NORM POS SWR DID NOT GO LOW
7011
7012
7013
7014
7015
7016
7017
7018
7019
7020
7021
7022
7023
7024
7025
7026
7027
7028
7029
7030
7031
7032
7033
7034
7035
7036 035406 000004
7037 035410 012737 000136 001240      MOV          #STM-1,$TESTN    ;;SET TEST NUMBER IN MAIL BOX
7038 035416 012737 035740 001222      MOV          #TST137,$ESCAPE ;;ESCAPE TO TEST 137 ON ERROR
7039 035424 012737 000200 001202      MOV          #200,$TMP1      ;SAVE EXPECTED
7040 035432 005037 001204          CLR          $TMP2           ;VALUE OF
7041 035436 005037 001206          CLR          $TMP3           ;QUANT'S 3,2,1
7042 035442 012703 000265          MOV          #265,R3         ;LOAD THE MICROC-BREAK
7043 035446 170003          LDUR         ;TO CATCH 2F3 BRANCH FAILURE
7044 035450 012737 035554 000244      MOV          #25,$PPVEC      ;
7045 035456 012737 040000 001160      MOV          #40000,$REG0     ;PUT THE SRC DATA
7046 035464 005037 001162          CLR          $REG1           ;IN MEMORY
7047 035470 005037 001164          CLR          $REG2           ;EXP=200
7048 035474 005037 001166          CLR          $REG3           ;FRAC=.5
7049 035500 012737 035506 001110      MOV          #.+6,#SLPERR    ;SET ERROR LOOP
  
```

```

7050 035506 172527 046000      LDD      #046000,AC1      ;LOAD THE DST,EXP=230,FRAC=.5
7051 035512 105737 001103      TSTP     $ERFLG          ;ANY ERRORS YET?
7052 035516 001003              BNE      100$            ;BRANCH IF YES
7053 035520 170127 000220      LDFPS   #FD+FMN
7054 035524 000402              BR       1$
7055 035526 170127 000200      100$:   LDFPS   #FD
7056 035532 172137 001160      1$:     ADDD    $REG0,AC1      ;EXECUTE INSTRUCTION UNDER TEST
7057 035536 170000              CFCC                    ;WAIT FOR INTERRUPT
7058
7059 035540 173527 046000      ;ERROR-INTERUPT DID NOT OCCUP
7060 035544 170000              CMPD    #046000,AC1      ;DID DST CHANGE?
7061 035546 001001              CFCC
7062 035550 104234              BNE     3$              ;BRANCH IF NO
7063 035552 104000              ERROR   234            ;F1 BRANCH FAILED
7064
7065
7066
7067 035554 022626              3$:     ERROR   0          ;DON'T KNOW WHAT HAPPENED
7068 035556 170007              ;SHOULD HAVE TRAPPED
7069 035560 174037 001210      ;*****
7070 035564 042737 100000 001210  2$:     CMP      (SP)+,(SP)+      ;INTERUPT OCCURRED-CHECK THE QR DATA
7071 035572 022737 046000 001210      STQO
7072 035600 001005              STD     AC0,STMP4        ;GET DATA IN QR
7073 035602 022737 000200 001212  BIC     #BIT15,STMP4      ;PUT QR DATA IN MEMCRY
7074 035610 001412              BIC     #46000,STMP4     ;GET RID OF SIGN
7075 035612 000405              CMP     #200,STMP5       ;QUAD 3 OK?
7076 035614 022737 066000 001210  4$:     BNE     4$              ;BRANCH IF NO
7077 035622 001001              CMP     #200,STMP5       ;QUAD 2 OK?
7078 035624 104235              BEQ     5$              ;BRANCH IF YES
7079 035626 012737 046000 001200  6$:     BR      6$
7080 035634 104125              CMP     #66000,STMP4     ;DID 2F? BRANCH FAIL?
7081
7082 035636              BNE     6$              ;BRANCH IF NO
7083 035636 012737 035644 001110  5$:     ERROR   235            ;FXPJ EALU06 XOR EALU04 DID NOT GO LOW
7084 035644 012700 001160              MOV     #46000,STMP0     ;SAVE EXPECTED OF QUAD 3
7085 035650 170127 000200              ERROR   125            ;ALIGNMENT FAILED
7086 035654 172527 046000      ;BRANCHES ARE WORKING OK-CHECK RESULT
7087 035660 170127 000300      5$:     MOV     #.+6,#SLPEPR     ;SET ERROR LOOP
7088 035664 172120              MOV     #SREG0,R0        ;PUT ADDRESS OF SFC DATA IN R0
7089 035666 022700 001170      LDFPS   #FD
7090 035672 001401              LDD     #046000,AC1      ;RELOAD DST
7091 035674 104066              LDFPS   #FD+FL          ;SET FL BIT & FD
7092 035676 170127 000200      ADDD    (R0)+,AC1        ;EXECUTE INSTRUCTION UNDER TEST
7093 035702 012737 046000 001200  7$:     CMP     #SREG4,R0        ;ADX ROM OK?
7094 035710 005037 001202              BEQ     7$              ;BRANCH IF YES
7095 035714 012737 100000 001204  7$:     ERROR   66            ;ADX ROM FAILED
7096 035722 173537 001200      LDFPS   #FD              ;CLEAR FL BIT
7097 035726 170000              MOV     #46000,STMP0     ;SAVE EXPECTED VALUE OF QUAD 3
7098 035730 001403              CLR     STMP1
7099 035732 174137 001210      MOV     #BIT15,STMP2
7100 035736 104125              CMPD    STMP0,AC1        ;DST OK?
7101
7102
7103
7104
7105
;*****
;*TEST 137      ADDD*MO*(FXP DIFF=50)
;*
```

```

7106 ;* THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS FXPJ BALU06 XOR BALU05.
7107 ;* THIS WILL CAUSE THE 2F3 BRANCH TO GO TO STATE 265 INSTEAD OF 275.
7108 ;*
7109 ;* THE APS VAL ROM COULD ALSO FAIL AND NOT PUT 330 IN THE EA.
7110 ;*
7111 ;* A MODE 0 INSTRUCTION IS USED WITH THE IL BIT ON TO FURTHER TEST THE
7112 ;* A-BRANCH ROM.
7113 ;*
7114 ;* FPU ROM FLOW-30,65,240,5(275,100,225),265,313,317
7115 ;*****
7116 035740 000004 TST137: SCOPE
7117 035742 012737 000137 001240 MOV #STN-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX
7118 035750 012737 036766 001222 MOV #TST140,$ESCAPE ;;ESCAPE TO TEST 140 ON ERROR
7119 035756 012703 000265 MOV #265,R3 ;;SET MICRO-BREAK TRAP
7120 035762 170003 LDUR ;;TO CATCH 2F3 BRANCH FAILURE
7121 035764 012737 052000 001200 MOV #52000,$TMP0 ;;SAVE EXPECTED
7122 035772 005037 001202 CLR $TMP1 ;;RESULT
7123 035776 012737 000200 001204 MOV #BIT7,$TMP2 ;
7124 036004 005037 001206 CLR $TMP3 ;
7125 036010 012737 036106 000244 MOV #25,$PPVEC ;;SET FP VECTOR
7126 036016 012737 036024 001110 MOV #.+6,$SLPERR ;;SET ERROR LOOP
7127 036024 170127 000200 LDFPS #FD
7128 036030 172427 052000 LDF #*052000,ACO ;;LOAD THE DST EXP=250
7129 036034 172527 040000 LDF #*040000,AC1 ;;LOAD THE SRC EXP=200
7130 036040 105737 001103 TSTR $ERFLC ;;ANY ERRORS YET
7131 036044 001003 BNE 1005 ;;BRANCH IF YES
7132 036046 170127 000220 LDFPS #FD+PMH
7133 036052 000402 BR 15
7134 036054 170127 000200 1005: LDFPS #FD
7135 036060 172001 15: ADDD AC1,ACO ;;EXECUTE INSTRUCTION
7136 036062 170127 000200 LDFPS #FD ;;WAIT FOR TRAP
7137 ;FAILURE-TRAP DID NOT OCCUR
7138 036066 173427 052000 CMPD #*052000,ACO ;;DST REMAIN UNCHANGED?
7139 036072 170000 CFCC
7140 036074 001001 BNE 35 ;;BRANCH IF NO
7141 036076 104234 ERROR 234 ;;AVS VAL ROM MSB FAILED
7142 036100 174037 001210 35: STD ACO,$TMP4 ;;SAVE RESULT
7143 036104 104000 ERROR 0 ;;DON'T KNOW WHAT HAPPENED, SHOULD HAVE TRAPPED
7144 ;*****
7145 ;INTERRUPT OCCURRED-CHECK QR DATA
7146 036106 022626 25: CMP (SP)+,(SP)+ ;;RESTORE SP
7147 036110 170007 STQ0 ;;PUT QR IN ACO
7148 036112 174037 001210 STD ACO,$TMP4 ;;GET QR DATA
7149 036116 042737 100000 001210 BIC #BIT15,$TMP4 ;;GET RID OF SIGN
7150 036124 022737 052000 001210 CMP #52000,$TMP4 ;;QUAD 3 OK?
7151 036132 001012 BNE 65 ;;BRANCH IF NO
7152 036134 005737 001212 TST $TMP5 ;;QUAD 2 OK?
7153 036140 001007 BNE 65 ;;BRANCH IF NO
7154 036142 022737 000200 001214 CMP #BIT7,$TMP6 ;;QUAD 1 OK?
7155 036150 001003 BNE 65 ;;BRANCH IF NO
7156 036152 005737 001216 TST $TMP7 ;;QUAD 0 OK?
7157 036156 001413 BEQ 45 ;;BRANCH IF YES
7158 036160 005037 001204 65: CLR $TMP2
7159 036164 173437 001200 CMPD $TMP0,ACO ;;DID 2F3 BRANCH FAIL?
7160 036170 170000 CFCC
7161 036172 001001 BNE 55 ;;BRANCH IF NO

```

7162	036174	104236			5S:	ERROR	236		;FXPJ EALU06 XOR EALU05 DID NOT GO LOW
7163	036176	052737	000200	001204		BIS	#BIT7,\$TMP2		
7164	036204	104125				ERROR	125		;DATA BAD
7165									
7166									
7167	036206								
7168	036206	012737	036214	001110	4S:	MOV	#.+6,#SLPERR		;SET ERROR LOOP
7169	036214	170127	000200			LDFPS	#FD		
7170	036220	172427	052000			LDD	#*052000,AC0		;RELOAD THE DST
7171	036224	170127	000300			LDFPS	#FD+FL		;SET FL BIT TO TEST A-BRANCH ROM
7172	036230	172001				ADD	AC1,AC0		;EXECUTE INSTRUCTION UNDER TEST
7173	036232	170127	000200			LDFPS	#FD		
7174	036236	005037	001204			CLR	\$TMP2		
7175	036242	012737	100000	001206		MOV	#BIT15,\$TMP3		
7176	036250	173437	001200			CMPD	\$TMP0,AC0		;RESULT OK?
7177	036254	170000				CFCC			
7178	036256	001403				BEQ	TST140		;BRANCH IF YES
7179	036260	174037	001210			STD	AC0,\$TMP4		;SAVE RESULT
7180	036264	104125				ERROR	125		;RESULT IS BAD
7181									
7182									
7183									
7184									
7185									
7186									
7187									
7188									
7189									
7190									
7191									
7192									
7193									
7194									
7195	036266	000004							
7196	036270	012737	000140	001240					
7197	036276	012737	036434	001222					
7198	036304	172527	042200						
7199	036310	172427	040000						
7200	036314	170127	000100						
7201	036320	172001							
7202	036322	170127	000000		1S:	ADDF	AC1,AC0		;EXECUTE INSTRUCTION UNDER TEST
7203	036326	012737	042200	001200		LDFPS	#0		;SAVE
7204	036334	012737	040000	001202		MOV	#42200,\$TMP0		;EXPECTED VALUE
7205	036342	173437	001200			MOV	#BIT14,\$TMP1		;RESULT OK?
7206	036346	170000				CMPF	\$TMP0,AC0		
7207	036350	001431				CFCC			
7208	036352	012737	042300	001200		BEQ	TST141		;BRANCH IF YES
7209	036360	005037	001202			MOV	#42300,\$TMP0		
7210	036364	173437	001200			CLR	\$TMP1		
7211	036370	170000				CMPF	\$TMP0,AC0		;DID 2F3 BRANCH FAIL?
7212	036372	001417				CFCC			
7213	036374	012737	042200	001200		BEQ	2S		;BRANCH IF YES
7214	036402	173437	001200			MOV	#42200,\$TMP0		
7215	036406	170000				CMPF	\$TMP0,AC0		;DID 5F1 BRANCH FAIL?
7216	036410	001001				CFCC			
7217	036412	104237				BNE	3S		;BRANCH IF NO
						ERROR	237		;ABS VAL ROM MSB DID NOT GO HIGH


```

7218 036414 012737 042200 001200 35:
7219 036422 012737 040000 001702
7220 036430 104035
7221 036432 104230
7222
7223
7224
7225
7226
7227
7228
7229
7230 036434 000004
7231 036436 012737 000141 001240
7232 036444 012737 037034 001222
7233
7234 036452 012737 040000 001160
7235 036460 005037 001162
7236 036464 005037 001164
7237 036470 012737 000005 001166
7238 036476 012737 140537 001200
7239 036504 012737 177777 001202
7240 036512 012737 177777 001704
7241 036520 012737 177777 001206
7242 036526 012737 036534 001110
7243 036534 170011
7244 036536 172437 001160
7245 036542 173027 040600
7246 036546 173437 001200
7247 036552 170000
7248 036554 001403
7249 036556 174037 001210
7250 036562 104260
7251
7252
7253 036564 012737 000002 001166
7254 036572 012737 140540 001200
7255 036600 005037 001202
7256 036604 005037 001204
7257 036610 005037 001206
7258 036614 012737 036622 001110
7259 036622 170011
7260 036624 172437 001160
7261 036630 173027 040600
7262 036634 173437 001200
7263 036640 170000
7264 036642 001403
7265 036644 174037 001210
7266 036650 104260
7267
7268
7269 036652 012737 000005 001166
7270 036660 012737 040537 001200
7271 036666 012737 177777 001202
7272 036674 012737 177777 001204
7273 036702 012737 177777 001206
  
```

```

MOV #42200,STMP0
MOV #BIT14,STMP1
EPROR 35 ;DATA BAD
ERROR 230 ;RBS VAL ROM OUTPUT BAD
;ADDRESS 167 ON ROM

;;*****
;*TEST 141 GUARD BIT TEST
;*
;* THIS TEST CHECKS THE GUARD BITS IN THE AR AND QP DATA PATH
;;*****
TST141: SCOPE
MOV #STN-1,STESTN ;SET TEST NUMBER IN MAIL BOX
MOV #TST142,SESCAPE ;ESCAPE TO TEST 142 ON ERROR
;SECTION 1-AR<2:0>101
MOV #40000,SREG0 ;PUT DATA IN
CLR SREG1 ;MEMORY TO
CLR SREG2 ;INITIALIZE THE
MOV #5,SREG3 ;DST
MOV #140537,STMP0 ;SAVE EXPECTED
MOV #-1,STMP1 ;RESULT
MOV #-1,STMP2 ;
MOV #-1,STMP3 ;
MOV #.+6,#SLPERR ;SET ERROR LOOP
SFD
LDD SREG0,AC0 ;LOAD THE DST
15: SUBD #040600,AC0 ;EXECUTE THE FUNCTION
CMPD STMP0,AC0 ;RESULT OK?
CFCC
BEQ 25 ;BRANCH IF YES
STD AC0,STMP4 ;GET RESULT
ERROR 260 ;GUARD BITS FAILED IN AR
;;*****
;SECTION 2-AR<2:0>010
25: MOV #2,SREG3 ;PUT DATA FOR DST IN MEMORY
MOV #140540,STMP0 ;PUT EXPECTED
CLR STMP1 ;RESULT IN MEMORY
CLR STMP2 ;
CLR STMP3 ;
MOV #.+6,#SLPERR ;SET ERROR LOOP
SFD
LDD SREG0,AC0 ;LOAD THE DST
SUBD #040600,AC0 ;EXECUTE FUNCTION
CMPD STMP0,AC0 ;RESULT OK?
CFCC
BEQ 35 ;BRANCH IF YES
STD AC0,STMP4
ERROR 260 ;GUARD BITS FAILED IN AR
;;*****
;SECTION 3-AR<2:0>101
35: MOV #5,SREG3 ;PUT SRC DATA IN MEMORY
MOV #40537,STMP0 ;PUT EXPECTED
MOV #-1,STMP1 ;RESULT INTO
MOV #-1,STMP2 ;MEMORY
MOV #-1,STMP3 ;
  
```

```
7274 036710 012737 036716 001110 MOV #.+6,#$LPERR ;SET ERROR LOOP
7275 036716 170011 SETD
7276 036720 172427 040600 LDD #^040600,ACO ;LOAD THE DST
7277 036724 173037 001100 SUBD $REG0,ACO ;EXECUTE FUNCTION
7278 036730 173437 001200 CMPD $TMP0,ACO ;RESULT OK?
7279 036734 170000 CFCC
7280 036736 001403 BEQ 4S ;BRANCH IF YES
7281 036740 174037 001210 STD ACO,$TMP4
7282 036744 104261 ERROR 261 ;GUARD BITS FAILED IN QR
7283 ;*****
7284 ;SECTION 4-QR<2:0>010
7285 036746 012737 000002 001166 4S: MOV #2,$REG3 ;PUT SRC DATA IN MEMORY
7286 036754 012737 040540 001200 MOV #40540,$TMP0 ;PUT EXPECTED
7287 036762 005037 001202 CLR $TMP1 ;RESULT IN
7288 036766 005037 001204 CLR $TMP2 ;MEMORY
7289 036772 005037 001206 CLR $TMP3 ;
7290 036776 012737 037004 001110 MOV #.+6,#$LPERR ;SET ERROR LOOP
7291 037004 170011 SETD
7292 037006 172427 040600 LDD #^040600,ACO ;LOAD THE DST
7293 037012 173037 001160 SUBD $REG0,ACO ;EXECUTE FUNCTION
7294 037016 173437 001200 CMPD $TMP0,ACO ;RESULT OK?
7295 037027 170000 CFCC
7296 037024 001403 BEQ TST142 ;BRANCH IF YES
7297 037026 174037 001210 STD ACO,$TMP4
7298 037032 104261 ERROR 261 ;GUARD BITS FAILED IN QR
7299 ;*****
7300 ;*TEST 142 LDCIF*-M0*(INT=+)
7301 ;*
7302 ;*
7303 ;* THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS CONTROL STORE SIGNALS
7304 ;* IN ROM STATES 253, 247, OR THE EXPONENT LOAD IN STATE 314.
7305 ;*
7306 ;* FPU ROM FLOW-41,174,213,314,312,253,247,317
7307 ;*****
7308 037034 000004 TST142: SCOPE
7309 037036 012737 000142 001740 MOV #STN-1,$TESTN ;SET TEST NUMBER IN MAIL BOX
7310 037044 012737 037126 001222 MOV #TST143,$ESCAPE ;ESCAPE TO TST 143 ON ERROR
7311 037052 012737 000001 001200 MOV #1,$TMP0 ;PUT INTEGER TO LOAD INTO MEMORY
7312 037060 012700 001700 MOV #TMP0,R0 ;PUT ADDRESS OF INTEGER IN R0
7313 037064 177020 1S: LDCTF (R0)+,ACO ;EXECUTE INSTRUCTION UNDER TEST
7314 037066 022700 001202 CMP #TMP1,R0 ;ADX ROM OK?
7315 037072 001401 BEQ 2S ;BRANCH IF YES
7316 037074 104066 ERROR 66 ;ADX ROM FAILED
7317 037076 174037 001204 2S: STF ACO,$TMP2 ;GET RESULT BACK
7318 037102 173427 040200 CMPF #^040200,ACO ;RESULT OK?
7319 037106 170000 CFCC
7320 037110 001406 BEQ TST143 ;BRANCH IF YES
7321 037112 012737 040200 001200 MOV #40200,$TMP0 ;SAVE EXPECTED
7322 037120 005037 001202 CLR $TMP1 ;VALUE OF DATA
7323 037124 104035 ERROR 35 ;DATA BAD
7324 ;*****
7325 ;*TEST 143 LDCIF*M0*(INT=-)
7326 ;*
7327 ;*
7328 ;* THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS THE CONTROL STORE
7329 ;* SIGNALS IN STATE 251 OR THE A OR NO-MEM ROMS.
```

```
7330 ;*
7331 ;* FPU ROM FLOW-30,40,174,213,314,312,251,313,317
7332 ;*****
7333 037126 000004 TST143: SCOPE
7334 037130 012737 000143 001240 MOV #STN-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX
7335 037136 012737 037242 001222 MOV #TST144,$ESCAPE ;;ESCAPE TO TEST 144 ON ERROR
7336 037144 170400 CLR# ACO ;INITIALIZE ACO
7337 037146 012700 177777 MOV #1,R0 ;PUT NUMBER TO CONVERT IN R0
7338 037152 170127 000007 LDFPS #7 ;LOAD COMPLIMENT CC'S
7339 037156 177000 1$: LDCTF R0,ACO ;EXECUTE INSTRUCTION UNDER TEST
7340 037160 170237 001200 STFPS $TMP0 ;GET CC'S BACK
7341 037164 174037 001204 STF ACO,$TMP2 ;GET DST DATA
7342 037170 022737 140200 001204 CMP #140200,$TMP2 ;QUAD 3 OK?
7343 037176 001003 BNE 2$ ;BRANCH IF NO
7344 037200 005737 001206 TST $TMP3 ;QUAD 2 OK?
7345 037204 001406 BEQ 3$ ;BRANCH IF YES
7346 037206 012737 140200 001200 2$: MOV #140200,$TMP0 ;SAVE EXPECTED
7347 037214 005037 001702 CLR $TMP1 ;VALUES
7348 037220 104035 ERROR 35 ;DATA BAD
7349 037222 022737 000010 001200 3$: CMP #FN,$TMP0 ;CC'S OK?
7350 037230 001404 BEQ TST144 ;;BRANCH IF YES
7351 037232 012737 000010 001202 MOV #FN,$TMP1 ;SAVE EXPECTED VALUE
7352 037240 104071 ERROR 71 ;CC'S BAD
7353
7354 ;*****
7355 ;*TEST 144 LDCLF*-MO*(INT=+)
7356 ;*
7357 ;* THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS ROM STATE 310 OR
7358 ;* THE A-BRANCH ROM OR THE ADX ROM.
7359 ;*
7360 ;* FPU ROM FLOW-41,164,174,213,310,312,253,247,317
7361 ;*****
7362 037242 000004 TST144: SCOPE
7363 037244 012737 000144 001240 MOV #STN-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX
7364 037252 012737 037360 001222 MOV #TST145,$ESCAPE ;;ESCAPE TO TEST 145 ON ERROR
7365 037260 012700 001210 MOV #TMP4,R0 ;PUT ADDRESS OF INTEGER IN R0
7366 037264 012720 000400 MOV #400,(R0)+ ;PUT INTEGER IN
7367 037270 012710 000001 MOV #1,(R0) ;MEMORY = 2**24+1
7368 037274 005740 TST -(R0) ;READJUST R0
7369 037276 012737 046200 001200 MOV #46200,$TMP0 ;SAVE EXPECTED
7370 037304 012737 000001 001202 MOV #1,$TMP1 ;RESULT
7371 037312 170012 SETL ;SET INTEGER LONG MODE
7372 037314 177020 1$: LDCLF (R0)+,ACO ;EXECUTE INSTRUCTION UNDER TEST
7373 037316 022700 001214 CMP #TMP6,R0 ;ADX ROM OK?
7374 037322 001401 BEQ 2$ ;BRANCH IF YES
7375 037324 104066 ERROR 66 ;ADX ROM FAILED
7376 037326 170002 2$: SETI ;RETURN TO INTEGER MODE
7377 037330 174037 001204 STF ACO,$TMP2 ;GET DST DATA
7378 037334 022737 046200 001204 CMP #46200,$TMP2 ;QUAD 3 OK?
7379 037342 001005 BNE 3$ ;BRANCH IF NO
7380 037344 022737 000001 001206 CMP #1,$TMP3 ;QUAD 2 OK?
7381 037352 001402 BEQ TST145 ;;BRANCH IF YES
7382 037354 104241 ERROR 241 ;STATE 247 DID NOT ROUND
7383 037356 104035 3$: ERROR 35 ;DATA BAD
7384
7385 ;*****
```

```
7386 ;*TEST 145 STCDF*-M0
7387 ;*
7388 ;* THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS THE NORMALIZE
7389 ;* SHIFT IN STATE 121 OR STATE 144.
7390 ;*
7391 ;* FPU ROM FLOW-44,35,166,121,300,377,153,144,154
7392 ;*
7393 037360 000004 TST145: SCOPE
7394 037362 012737 000145 001240 MOV #STN-1,STESTN ;;SET TEST NUMBER IN MAIL BOX
7395 037370 012737 037504 001222 MOV #TST146,$ESCAPE ;;ESCAPE TO TEST 146 ON ERROR
7396 037376 012700 001160 MOV #SREG0,R0 ;GET ADDRESS OF DATA BUFFER
7397 037402 012720 040377 MOV #40377,(R0)+ ;PUT DATA TO
7398 037406 012720 177777 MOV #-1,(R0)+ ;INITIALIZE THE
7399 037412 012720 100000 MOV #BIT15,(R0)+ ;SRC INTO
7400 037416 005010 CLR (R0) ;MEMORY
7401 037420 012737 037426 001110 MOV #.+6,#SLPERR ;SET ERROR LOOP
7402 037426 012700 001204 MOV #STMP2,R0 ;PUT ADDRESS OF DST IN R0
7403 037432 170011 SETD
7404 037434 172437 001160 LDD $REG0,AC0 ;LOAD THE SRC
7405 037440 176020 1S: STCDF AC0,(R0)+ ;EXECUTE INSTRUCTION UNDER TEST
7406 037442 022700 001210 CMP #STMP4,R0 ;ADX ROW OK?
7407 037446 001401 BEQ 2S ;BRANCH IF YES
7408 037450 104066 ERROR 66 ;ADX ROW FAILED
7409 037452 022737 040400 001204 2S: CMP #40400,STMP2 ;QUAD 3 OK?
7410 037460 001003 BNE 3S ;BRANCH IF NO
7411 037462 005737 001206 TST $TMP3 ;QUAD 2 OK?
7412 037466 001406 BEQ TST146 ;;BRANCH IF YES
7413 037470 012737 040400 001200 3S: MOV #40400,STMP0 ;SAVE EXPECTED
7414 037476 005037 001202 CLR $TMP1 ;VALUE OF DATA
7415 037502 104242 ERROR 242 ;STATE 121 DID NOT NORMALIZE PROPERLY
7416
7417 ;*
7418 ;*TEST 146 STCDF*-M0
7419 ;*
7420 ;* THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS THE "CLEAR AR LCM"
7421 ;* SIGNAL IN STATE 167 OR STATES 155 OR 156.
7422 ;*
7423 ;* FPU ROM FLOW-44,35,167,123,153,144,155,156
7424 ;*
7425 037504 000004 TST146: SCOPE
7426 037506 012737 000146 001240 MOV #STN-1,STESTN ;;SET TEST NUMBER IN MAIL BOX
7427 037514 012737 037650 001222 MOV #TST147,$ESCAPE ;;ESCAPE TO TEST 147 ON ERROR
7428 ;*
7429 037522 012737 177777 001164 MOV #-1,$REG2 ;INITIALIZE DATA TO LOAD INTO
7430 037530 170011 SETD ;THE SRC REG0 & 1 ALREADY SET
7431 037532 172437 001160 LDD $REG0,AC0 ;LOAD THE SRC
7432 037536 012737 040377 001200 MOV #40377,STMP0 ;SAVE EXPECTED
7433 037544 012737 177777 001202 MOV #-1,STMP1 ;VALUES
7434 037552 005037 001204 CLR STMP2
7435 037556 005037 001206 CLR STMP3
7436 037562 012737 037570 001110 MOV #.+6,#SLPERR ;SET ERROR LOOP
7437 037570 012700 001210 MOV #STMP4,R0 ;PUT ADDRESS OF DST IN R0
7438 037574 170001 SETF ;GO TO FLOATING MODE
7439 037576 176020 1S: STCDF AC0,(R0)+ ;EXECUTE INSTRUCTION UNDER TEST
7440 037600 022700 001220 CMP #STMP7+2,R0 ;ADX ROW OK?
7441 037604 001401 BEQ 2S ;BRANCH IF YES
```

```

7442 037606 104066          EPROR 66          ;ADX ROM FAILED
7443 037610 022737 040377 001210 2S:  CMP  #40377,STMP4  ;QUAD 3 OK?
7444 037616 001013          BNE 3$          ;BRANCH IF NO
7445 037620 022737 177777 001212  CMP  #-1,STMP5  ;QUAD 2 OK?
7446 037626 001007          BNE 3$          ;BRANCH IF NO
7447 037630 005737 001214  TST  STMP6      ;QUAD 1 OK?
7448 037634 001003          BNE 4$          ;BRANCH IF NO
7449 037636 005737 001216  TST  STMP7      ;QUAD 0 OK?
7450 037642 001402          BEQ  TST147     ;;BRANCH IF YES
7451 037644 104243          4S:  ERKOR 243    ;STATE 167 DID NOT CLEAR AR LOW
7452 037646 104125          3S:  ERROR 125   ;DATA BAD
7453                                     ;;*****
7454                                     ;*TEST 147 STCDF*-M0*FV
7455                                     ;*
7456                                     ;*SFO BRANCH
7457                                     ;*
7458                                     ;* IF FXPJ EALU 08 DOES NOT GET TO FRM# 800 AS A HIGH OR
7459                                     ;* FRMP 800 DOES NOT CAUSE FRMA 800+RZ TO GO LOW OR IF
7460                                     ;* FRMA 800+BZ DOES NOT GET TO RADO0 AS A HIGH, EXECUTION
7461                                     ;* WILL GO TO STATE 377. THIS WILL CAUSE THE RESULT TO BE
7462                                     ;* STORED INSTEAD OF ZEROED.
7463                                     ;*
7464                                     ;* THE IL BIT IS SET TO TEST THE A-BRANCH & ADX ROMS.
7465                                     ;*
7466                                     ;* FPU ROM FLOW-44,35,166,121,300,376,127,125,153,144,154
7467                                     ;;*****
7467 037650 000004          TST147: SCOPE
7468 037652 012737 000147 001240  MOV  #STN-1,STESTM ;;SET TEST NUMBER IN MAIL BOX
7469 037660 012737 040010 001222  MOV  #TST150,$FSCAPE ;;ESCAPE TO TEST 150 ON EPROR
7470 037666 012737 077777 001160  MOV  #77777,$REGC  ;PUT DATA IN MEMORY TO INITIALIZE
7471 037674 170011          SETD
7472 037676 172437 001160  LDD  $REG0,ACO    ;LOAD THE SRC
7473 037702 005037 001200  CLR  STMP0        ;SAVE EXPECTED
7474 037706 005037 001202  CLR  STMP1        ;DATA
7475 037717 012737 037720 001110  MOV  #-6,$SLPERR ;SET ERROR LOOP
7476 037720 012700 001204  MOV  #STMP2,R0    ;PUT ADDRESS OF DST IN R0
7477 037724 170127 000311  LDFPS #FD+FL+11  ;LOAD COMPLIMENT CC'S
7478 037730 176020          1S:  STCDF ACO,(R0)+ ;EXECUTE INSTRUCTION UNDER TEST
7479 037732 170201          STFPS R1         ;GET CC'S
7480 037734 170127 000000  LDFPS #0
7481 037740 022700 001210  CMP  #STMP4,R0    ;ADX ROM OK?
7482 037744 001401          BEQ 2$          ;BRANCH IF YES
7483 037746 104066          ERKOR 66        ;ADX ROM FAILED
7484 037750 005737 001204  2S:  TST  STMP2      ;QUAD 3 OK?
7485 037754 001003          BNE 3$          ;BRANCH IF NO
7486 037756 005737 001206  TST  STMP3      ;QUAD 2 OK?
7487 037762 001401          BEQ 4$          ;BRANCH IF YES
7488 037764 104035          3S:  ERKOR 35    ;DATA BAD
7489 037766 022701 000306  4S:  CMP  #FD+FL+FV+FZ,R1 ;CC'S OK?
7490 037772 001406          BEQ  TST150     ;;BRANCH IF YES
7491 037774 010137 001200  MOV  R1,STMP0    ;SAVE RECEIVED CC'S
7492 040000 012737 000306 001202  MOV  #306,STMP1  ;SAVE EXPECTED CC'S
7493 040006 104244          ERROR 244      ;EITHER SFO BRANCH FAILED OR
7494                                     ;CC'S DID NOT LOAD PROPERLY
7495                                     ;;*****
7496                                     ;*TEST 150 LDCDF*-M0*AR59
7497

```

```

7498 ;*
7499 ;*
7500 ;* THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS THE EXPONENT AND
7501 ;* SHIFT CONTROL SUBTRACTION IN STATE 231 OR STATE 161.
7502 ;*
7503 ;* THE IL BIT IS SET TO TEST THE A-BRANCH AND ADX ROMS.
7504 ;*
7505 ;* FPU ROM FLOW-17,220,47,46,137,231,161
7506 ;* *****
7506 040010 000004 TST150: SCOPE
7507 040012 012737 000150 001240 MOV #STN-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX
7508 040020 012737 040130 001222 MOV #TST151,$ESCAPE ;;ESCAPE TO TEST 151 ON ERROR
7509 040026 012737 040377 001160 MOV #40377,$REGO ;PUT SRC DATA
7510 040034 012737 177777 001162 MOV #-1,$REG1 ;IN MEMORY TO CAUSE AR59
7511 040042 012737 100000 001164 MOV #BIT15,$REG2 ;WHEN ROUND OCCURS
7512 040050 012737 040056 001110 MOV #-6,#SLPERR ;SET ERROR LOOP
7513 040056 012700 001160 MOV $REGO,R0 ;PUT ADDRESS OF SRC IN R0
7514 040062 170012 SETL
7515 040064 177420 1S: LDCDF (R0)+,AC0 ;EXECUTE INSTRUCTION UNDER TEST
7516 040066 170001 SETP
7517 040070 022700 001170 CMP #$REG4,R0 ;ADX ROM OK?
7518 040074 001401 BEQ 2S ;BRANCH IF YES
7519 040076 104066 ERROR 66 ;ADX ROM FAILED
7520 040100 173427 040400 2S: CMPF #040400,AC0 ;RESULT OK?
7521 040104 170000 CFCC
7522 040106 001410 BFG TST151 ;;BRANCH IF YES
7523 040110 174037 001204 STF AC0,$TMP2 ;SAVE RECEIVED DATA
7524 040114 012737 040000 001200 MOV #40000,$TMP0 ;SAVE EXPECTED
7525 040122 005037 001202 CLR $TMP1 ;DATA
7526 040126 104035 ERROR 3S ;DATA BAD
7527
7528 ;* *****
7529 ;* TEST 151 LDCDF*MO*ROU
7530 ;*
7531 ;* THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS ROM STATES
7532 ;* 162,243.
7533 ;*
7534 ;* THE IL BIT IS SET TO TEST THE A-BRANCH ROM.
7535 ;*
7536 ;* FPU ROM FLOW-30,46,137,231,161,162,243,70,173
7537 ;* *****
7538 040130 000004 TST151: SCOPE
7539 040132 012737 000151 001240 MOV #STN-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX
7540 040140 012737 040224 001222 MOV #TST152,$ESCAPE ;;ESCAPE TO TEST 152 ON ERROR
7541 040146 012737 077777 001160 MOV #77777,$REGO ;PUT DATA TO INITIALIZE THE SRC
7542 040154 170011 SETD ;IN MEMORY REG1 & 2 ALREADY SETUP
7543 040156 172537 001160 LDD $REGO,AC1 ;LOAD THE SRC
7544 040162 012737 040170 001110 MOV #-6,#SLPERR ;SET ERROR LOOP
7545 040170 170127 000115 LDFPS #FL+15 ;LOAD COMPLIMENT CC'S
7546 040174 177401 1S: LDCDF AC1,AC0 ;EXECUTE INSTRUCTION UNDER TEST
7547 040176 170000 CFCC ;GET CC'S
7548 040200 102411 BVS TST152 ;;BRANCH IF OVERFLOW
7549 040202 170127 000000 LDFPS #0
7550 040206 174037 001204 STF AC0,$TMP2 ;GET DST DATA
7551 040212 005037 001200 CLR $TMP0 ;SAVE EXPECTED DATA
7552 040216 005037 001202 CLR $TMP1 ;
7553 040222 104245 ERROR 24S ;OVERFLOW DID NOT OCCUR
  
```

```

7554
7555
7556
7557
7558
7559
7560
7561
7562
7563 040224 000004
7564 040226 012737 000152 001240
7565 040234 012737 040324 001222
7566 040242 012737 077777 001160
7567 040250 172437 001160
7568 040254 174001
7569 040256 012737 077400 001164
7570 040264 012737 000001 001166
7571 040272 012737 040300 001110
7572 040300 172037 001164
7573 040304 170000
7574 040306 102406
7575 040310 174037 001204
7576 040314 172401
7577 040316 170437 001700
7578 040322 104245
7579
7580
7581
7582
7583
7584
7585
7586
7587
7588
7589 040324 000004
7590 040326 012737 040504 001222
7591 040334 012737 000377 001160
7592 040342 012737 177777 001162
7593 040350 012737 000300 001164
7594 040356 005037 001166
7595 040362 172437 001160
7596 040366 174001
7597 040370 012737 040376 001110
7598 040376 170127 000111
7599 040402 012700 001164
7600 040406 173020
7601 040410 170201
7602 040412 170002
7603 040414 022700 001170
7604 040420 001402
7605 040422 172401
7606 040424 104066
7607 040426 174037 001204
7608 040432 170000
7609 040434 001402

;;*****
;*TEST 157      ADDF*-MO*800
;*
;*      THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS THE ROUND IN
;*      STATE 313 OF THE EXPONENT MINUS SHIFT CONTROL IN STATE 317.
;*
;*      FPU ROM FLOW-11,130,65,240,265,313,317,161,162,243,70,173
;;*****
TST157: SCOPE
MOV      #STW-1,$TESTN      ;;SET TEST NUMBER IN MAIL BOX
MOV      #TST153,$ESCAPE    ;;ESCAPE TO TEST 153 ON ERROR
MOV      #77777,$REG0       ;PUT DST DATA IN MEMORY
LDF      $REG0,AC0          ;REG1 ALREADY SETUP WITH -1
STF      AC0,AC1            ;SAVE IN AC1 IN CASE ERROR
MOV      #77400,$REG2       ;PUT SRC DATA IN MEMORY TO
MOV      #1,$REG3           ;CAUSE OVERFLOW ON ROUND.
MOV      #.+6,#SLPERR       ;SET ERROR LOOP
15:      ADDF      $REG2,AC0   ;EXECUTE INSTRUCTION UNDER TEST
        CFC      CFC
        BVS      TST153      ;;BRANCH IF OVERFLOW
        STF      AC0,$TMP2    ;SAVE RESULT
        LDF      AC1,AC0     ;RESTORE DST FOR LOOPING
        CLRF     $TMP0       ;SAVE EXPECTED DATA
        ERROR    245         ;OVERFLOW DID NOT OCCUR

;;*****
;*TEST 153      SUBF*-MO*(RESULT=0)
;*
;*      THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS STATE 160.
;*
;*      THE IL PIT IS SET TO TEST THE A-BRANCH AND ADX ROMS.
;*
;*      FPU ROM FLOW-11,130,65,240,271,74,263,313,317,160,162,243
;;*****
TST153: SCOPE
MOV      #TST154,$ESCAPE    ;;ESCAPE TO TEST 154 ON ERROR
MOV      #377,$REG0         ;PUT DST DATA IN MEMORY
MOV      #-1,$REG1          ;EXP=1 FRAC=.1111....
MOV      #300,$REG2        ;PUT SRC DATA IN MEMORY
CLR      $REG3              ;EXP=1 FRACT=.11000....
LDF      $REG0,AC0         ;LOAD THE DST
STF      AC0,AC1           ;SAVE IN AC1 IN CASE ERROR
MOV      #.+6,#SLPERR       ;SET ERROR LOOP
LDFPS    #FL+11             ;LOAD COMPLIMENT CC'S
MOV      #SREG2,R0         ;PUT ADDRESS OF SRC IN R0
15:      SUBF     (R0)+,AC0   ;EXECUTE INSTRUCTION UNDER TEST
        STFPS   R1
        SETI    #SREG4,R0    ;ADX ROM OK?
        BEQ     25           ;BRANCH IF YES
        LDF     AC1,AC0     ;RESTORE DST
        EPROR   66          ;ADX ROM FAILED
25:      STF      AC0,$TMP2   ;GET RESULT
        CFC      CFC
        BEQ     35           ;BRANCH IF RESULT ZERO
  
```

```
7610 040436 172401 LDF AC1,AC0 ;RESTORE DST
7611 040440 104035 ERROR 35 ;EXPONENT NON-ZERO
7612 040442 172401 3S: LDF AC1,AC0 ;RESTORE DST INCASE ERROR
7613 040444 005737 001204 TST $TMP2 ;IS FPACTON ALSO ZERO?
7614 040450 001003 BNE 4S ;BRANCH IF NO
7615 040452 005737 001206 TST $TMP3 ;CHECK QUAD 2 EQUAL ZERO
7616 040456 001401 BEQ 5S ;BRANCH IF YES
7617 040460 104246 4S: ERROR 246 ;PRMA BOU+8Z DID NOT GO LOW ON 8Z
7618 040462 022701 000104 5S: CMP #FL+FZ,R1 ;CC'S OK?
7619 040466 001406 BEQ TST154 ;)BRANCH IF YES
7620 040470 010137 001200 MOV R1,$TMP0 ;SAVE RECEIVED CC'S
7621 040474 012737 000104 001202 MOV #FL+FZ,$TMP1 ;SAVE EXPECTED CC'S
7622 040502 104747 ERROR 247 ;STATE 162 DID NOT ADD 1 TO THE EXP
7623
7624 ;*****
7625 ;*TEST 154 SUBD*-MO*TMH*-AR59*BOU
7626 ;*
7627 ;* THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS THE 3F2 BRANCH
7628 ;* AFTER STATE 162.
7629 ;*
7630 ;* FPU ROM FLOW-20,140,270,150,65,240,271,74,263,313,317,162,241,207
7631 ;*****
7632 TST154: SCOPE
7633 040504 000004 MOV #STN-1,$STESTN ;)SET TEST NUMBER IN MAIL BOX
7634 040506 012737 000154 001240 MOV #TST155,$ESCAPE ;)ESCAPE TO TEST 155 ON ERROR
7635 040514 012737 040640 001222 LDF #0377,AC0 ;LOAD THE DST
7636 040522 172427 000377 STF AC0,AC1 ;SAVE IN AC1 INCASE OF ERROR
7637 040526 174001 MOV #25,$RESVEC ;SET RESVEC INCASE ADX ROM FAILS
7638 040530 012737 040632 000010 MOV #.+6,$SLPERR ;SET ERROR LOOP
7639 040536 012737 040544 001110 LDFPS #FD+13 ;LOAD COMPLIMENT CC'S
7640 040544 170127 000713 1S: SUBP (PC)+,AC0 ;EXECUTE INSTRUCTION UNDER TEST
7641 040552 000340 .WORD 340 ;WILL EXECUTE THIS & TRAP IF ADX ROM FAILS
7642 040554 000403 BR 3S
7643 040556 104066 ERROR 66 ;ADX ROM FAILED
7644 040560 104066 ERROR 66 ;
7645 040562 104066 ERROR 66 ;
7646 040564 170700 3S: STFPS R0 ;GET CC'S
7647 040566 174037 001210 STD AC0,$TMP4 ;GET RESULT
7648 040572 170000 CFCC
7649 040574 001404 BEQ 4S ;BRANCH IF RESULT ZERO
7650 040576 172401 LDD AC1,AC0 ;RELOAD THE DST
7651 040600 170437 001200 CLPD $TMP0 ;SAVE EXPECTED VALUE
7652 040604 104125 ERROR 125 ;DATA BAD
7653 040606 172401 4S: LDD AC1,AC0 ;RELOAD THE DST INCASE OF ERROR
7654 040610 032700 000007 BIT #FV,R0 ;IS V BIT CLEAR?
7655 040614 001411 BEQ TST155 ;)BRANCH IF YES
7656 040616 010037 001200 MOV R0,$TMP0 ;SAVE RECEIVED CC'S
7657 040622 012737 000004 001207 MOV #FZ,$TMP1 ;SAVE EXPECTED.
7658 040630 104071 ERROR 71 ;CC'S BAD
7659 ;ADX ROM FAILED
7660 040632 022626 2S: CMP (SP)+,(SP)+ ;RESTORE SP
7661 040634 174100 STD AC1,AC0 ;RESTORE THE DST
7662 040636 104066 ERROR 66 ;ADX ROM FAILED
7663
7664 ;*****
7665 ;*TEST 155 STCPI*-MO*(INT=-1)
```



```

7666 ;*
7667 ;* THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS ROM STATES 347 OR
7668 ;* 103 OR FMX BIT 35 OR FXM BIT 19.
7669 ;*
7670 ;* FPU ROM FLOW-51,374,333,371,375,347,103,210,215
7671 ;*****
7672 040640 000004 TST155: SCOPE
7673 040642 012737 000155 001240 MOV #STN-1,STESTN ;;SET TEST NUMBER IN MAIL BOX
7674 040650 012737 040760 001222 MOV #TST156,$ESCAPE ;;ESCAPE TO TEST 156 ON ERROR
7675 040656 012737 177777 001164 MOV #-1,$SREG2 ;SAVE EXPECTED VALUE
7676 040664 172427 140200 LDF #*0140200,ACO ;LOAD THE SRC WITH A -1
7677 040670 170127 000007 LDFPS #7 ;LOAD COMPLIMENT CC'S
7678 040674 175437 001162 1$: STCFI ACO,$REG1 ;EXECUTE INSTRUCTION UNDER TEST
7679 040700 022737 177777 001162 CMP #-1,$SREG1 ;DATA OK?
7680 040706 001412 BEQ 2$ ;BRANCH IF YES
7681 040710 022737 177776 001162 CMP #-2,$REG1 ;FMX35 FAIL?
7682 040716 001001 BNE 3$ ;BRANCH IF NO
7683 040720 104250 ERROR 250 ;PRRC FMX35 NOT GOING HIGH
7684 040722 005737 001162 3$: TST $SREG1 ;FMX19 FAIL?
7685 040726 001001 BNE 4$ ;BRANCH IF NO
7686 040730 104251 ERROR 251 ;PRLC FMX19 NOT GOING LOW
7687 040732 104252 4$: ERROR 252 ;DATA BAD
7688 ;CHECK THE CC'S
7689 040734 170237 001200 2$: STFPS $TMP0 ;GET CC'S
7690 040740 022737 000010 001700 CMP #FN,$TMP0 ;CC'S OK?
7691 040746 001404 BEQ TST156 ;;BRANCH IF YES
7692 040750 012737 000010 001202 MOV #FN,$TMP1 ;SAVE EXPECTED VALUE
7693 040756 104071 ERROR 71 ;CC'S BAD
7694
7695 ;*****
7696 ;*TEST 156 STCFI*-M0*(INT=-1)
7697 ;*
7698 ;* THE ONLY THING THAT SHOULD FAIL IN THIS TEST IS FMX35 OR
7699 ;* FMX19.
7700 ;*
7701 ;* FPU ROM FLOW-51,370,333,371,375,347,103,345,200,66,215
7702 ;*****
7703 040760 000004 TST156: SCOPE
7704 040762 012737 000156 001240 MOV #STN-1,STESTN ;;SET TEST NUMBER IN MAIL BOX
7705 040770 012737 041076 001222 MOV #TST157,$ESCAPE ;;ESCAPE TO TEST 157 ON ERROR
7706 040776 012737 177777 001200 MOV #-1,$TMP0 ;SAVE EXPECTED
7707 041004 012737 177777 001202 MOV #-1,$TMP1 ;VALUE OF DATA
7708 041012 012737 041020 001110 MOV #-6,@#SLPERR ;SET ERROR LOOP
7709 041020 172427 140200 LDF #*0140200,ACO ;LOAD THE SRC WITH -1
7710 041024 170012 SETL
7711 041026 175437 001204 1$: STCFI ACO,$TMP2 ;EXECUTE INSTRUCTION UNDER TEST
7712 041032 022737 177777 001206 CMP #-1,$TMP3 ;LOW ORDER BITS OK?
7713 041040 001406 BEQ 2$ ;BRANCH IF YES
7714 041042 022737 177776 001206 CMP #-2,$TMP3 ;FMX19 FAIL?
7715 041050 001001 BNE 3$ ;BRANCH IF NO
7716 041052 104252 ERROR 253 ;PRRC FMX19 NOT GOING HIGH
7717 041054 104221 3$: ERROR 221 ;DATA BAD
7718 041056 022737 177777 001204 2$: CMP #-1,$TMP2 ;HIGH ORDER WORD OK?
7719 041064 001404 BEQ TST157 ;;BRANCH IF YES
7720 041066 005737 001204 TST $TMP2 ;DID FMX35 FAIL?
7721 041072 001370 BNE 3$ ;BRANCH IF NO

```

```

7722 041074 104126          ERROR 176          ;FRHC FWX35 DID NOT GO LOW
7723
7724
7725
7726
7727
7728
7729
7730
7731
7732
7733
7734
7735
7736
7737 041076 000004          ;*****
7738 041100 012737 000050 001220          ;*TEST 157      STCFI*M0*(EXP=2**16)*(SD=-)
7739 041106 012737 000157 001240          ;*
7740
7741 041114 012701 077777          ;* THIS TEST FIRST CHECKS THE QR FILL LOGIC. THIS IS DONE
7742 041120 012737 144000 001200          ;* BY FLOATING A "ONE" FROM BIT POSITION
7743 041126 012737 001000 001202          ;* 43 TO BIT 57. IF QR FILL DOES NOT FILL PROPERLY THE
7744 041134 005037 001164          ;* RESULT WILL NOT BE ZERO.
7745 041140 012737 041146 001110          ;*
7746 041146 172437 001200          ;* A ONE IS THEN FLOATED FROM BIT 35 TO 42 TO CHECK THE
7747 041152 175400          ;* LOGICAL AND FUNCTION OF THE ALU (SINGLE PRECISION).
7748 041154 005700          ;*
7749 041156 001016          ;* FPU ROM FLOW-51,374,333,372,337,256,151,357,322,316,103,345,214
7750 041160 032777 001000 137750          ;*****
7751 041166 001403          TST157: SCOPE
7752 041170 105737 001103          MOV #50,$TIMES ;DO 50 ITERATIONS
7753 041174 001364          MOV $SW-1,$TESTN ;SET TEST NUMBER IN MAIL BOX
7754 041176 062737 001000 001202 45:   MOV #77777,R1 ;SET SOB LOOP COUNT
7755 041204 105537 001200          MOV #144000,$TMP0 ;INITIALIZE DATA TO LOAD
7756 041210 077122          MOV #BIT9,$TMP1 ;INTO SRC ACCUMULATOR
7757 041212 000404          CLR $REG2 ;SAVE EXPECTED DATA
7758 041214 010037 001162 35:   MOV #.+6,$SLPERR ;SET ERROR LOOP
7759 041220 104754          LDF $TMP0,ACO ;LOAD THE SRC
7760 041222 000765          STCFI ACO,R0 ;EXECUTE INSTRUCTION UNDER TEST
7761
7762
7763 041224 012701 000010          TST R0 ;RESULT OK?
7764 041230 012737 144000 001200          BNE 35 ;BRANCH IF NO
7765 041236 012737 000001 001202          BIT #SW9,$SWH ;LOOP ON ERROR?
7766 041244 012737 100000 001164          BEQ 45 ;BRANCH IF NO
7767 041252 012737 041260 001110          TSTP $ERFLG ;ANY ERRORS YET?
7768 041260 172437 001200          BNE 25 ;BRANCH IF YES
7769 041264 175400          ADD #BIT9,$TMP1 ;SELECT THE NEXT DATA PATTERN
7770 041266 022700 100000          ADCB $TMP0
7771 041272 001013          SOB R1,25 ;CONTINUE
7772 041274 032777 001000 137634 35:   BR 55
7773 041302 001403          MOV R0,$REG1 ;SAVE RECEIVED DATA
7774 041304 105737 001103          ERROR 254 ;FRHC Q FILL FAILED TO FILL 8 BITS
7775 041310 001363          BR 45
7776 041312 006337 001202          ;*****
7777 041316 077120          ;Q FILL WORKS - NOW TEST THE LOGICAL AND FUNCTION
7778
7779
7780
7781
7782
7783
7784
7785
7786
7787
7788
7789
7790
7791
7792
7793
7794
7795
7796
7797
7798
7799
7800
7801
7802
7803
7804
7805
7806
7807
7808
7809
7810
7811
7812
7813
7814
7815
7816
7817
7818
7819
7820
7821
7822
7823
7824
7825
7826
7827
7828
7829
7830
7831
7832
7833
7834
7835
7836
7837
7838
7839
7840
7841
7842
7843
7844
7845
7846
7847
7848
7849
7850
7851
7852
7853
7854
7855
7856
7857
7858
7859
7860
7861
7862
7863
7864
7865
7866
7867
7868
7869
7870
7871
7872
7873
7874
7875
7876
7877
7878
7879
7880
7881
7882
7883
7884
7885
7886
7887
7888
7889
7890
7891
7892
7893
7894
7895
7896
7897
7898
7899
7900
7901
7902
7903
7904
7905
7906
7907
7908
7909
7910
7911
7912
7913
7914
7915
7916
7917
7918
7919
7920
7921
7922
7923
7924
7925
7926
7927
7928
7929
7930
7931
7932
7933
7934
7935
7936
7937
7938
7939
7940
7941
7942
7943
7944
7945
7946
7947
7948
7949
7950
7951
7952
7953
7954
7955
7956
7957
7958
7959
7960
7961
7962
7963
7964
7965
7966
7967
7968
7969
7970
7971
7972
7973
7974
7975
7976
7977
7978
7979
7980
7981
7982
7983
7984
7985
7986
7987
7988
7989
7990
7991
7992
7993
7994
7995
7996
7997
7998
7999
8000

```

```
7778 041320 000404
7779 041322 010037 001162
7780 041326 104255
7781 041330 000770
7782
7783
7784
7785
7786
7787
7788
7789
7790
7791
7792
7793 041332 000004
7794 041334 012737 000025 001220
7795 041342 012737 000160 001240
7796 041350 170011
7797 041352 170012
7798 041354 172427 044422
7799 041360 012700 001204
7800 041364 175420
7801 041366 022700 001210
7802 041372 001401
7803 041374 104066
7804 041376 012701 177777
7805 041402 012737 150000 001160
7806 041410 005037 001162
7807 041414 012737 000400 001164
7808 041422 005037 001166
7809 041426 005037 001200
7810 041432 005037 001202
7811 041436 012737 041444 001110
7812 041444 170011
7813 041446 170012
7814 041450 172437 001160
7815 041454 175437 001204
7816 041460 005737 001204
7817 041464 001021
7818 041466 005737 001206
7819 041472 001016
7820 041474 032777 001000 137434
7821 041502 001403
7822 041504 105737 001103
7823 041510 001355
7824 041512 062737 000400 001164
7825 041520 005537 001162
7826 041524 077131
7827 041526 000402
7828 041530 104256
7829 041532 000767
7830
7831
7832 041534 012701 000030
7833 041540 005037 001162

7S: BR TST160 ;GO TO NEXT TEST
MOV R0,$R0G1 ;SAVE RECEIVED DATA
ERROR 255 ;LOGICAL AND FAILED
BR 8S

;*****
;*TEST 160 STCDL*-M0*(EXP=2**32)*(SD=-)
;*
;* THIS TEST FIRST ENSURES THAT THE ADX ROM IS OK. IT THEN
;* FLOATS A "ONE" FROM BIT 27 TO 47 TO ENSURE THAT THE QR MASK
;* WAS FORMED PROPERLY. IT THEN FLOATS A ONE FROM BIT 3 TO
;* BIT 26 TO TEST THE LOGICAL AND OF THE FALU.
;*
;* FPU ROM FLOW-51,370,333,372,337,252,353,256,151,357,322,316,103,345,200,66,215
;*****
TST160: SCOPE
MOV #25,$TIMES ;DO 25 ITERATIONS
MOV #STN-1,$TESTN ;SET TEST NUMBER IN MAIL BOX
SETD
SETL
LDD #150000,ACO ;PUT THE NUMBER IN THE SRC ACC.
MOV #STMP2,R0 ;PUT ADDRESS OF DST IN R0
1S: STCDL ACO,(R0)+ ;EXECUTE INSTRUCTION UNDER TEST
CMP #STMP4,R0 ;ADX ROM OK?
BEQ 2S ;BRANCH IF YES
ERROR 66 ;ADX ROM FAILED
2S: MOV #-1,R1 ;SET SOB LOOP
MOV #150000,$REG0 ;INITIALIZE DATA
CLR $REG1 ;TO LOAD INTO
MOV #BIT8,$REG2 ;THE SRC
CLR $REG3 ;
CLR $TMP0 ;SAVE EXPECTED
CLR $TMP1 ;VALUE OF RESULT
3S: MOV #.+6,$NSLPERR ;SET ERROR LOOP
SFTD
SETL
LDD $REG0,ACO ;LOAD THE SRC
STCDL ACO,$TMP2 ;EXECUTE THE INSTRUCTION
TST $TMP2 ;FIRST WORD OK?
BNE 4S ;BRANCH IF NO
TST $TMP3 ;SECOND WORD OK?
BNE 4S ;BRANCH IF NO
BIT #SW9,$SWR ;LOOP CN ERROR?
BEQ 5S ;BRANCH IF NO
TSTR $KFLG ;ANY ERRORS YET?
BNE 3S ;BRANCH IF YES
5S: ADD #BIT8,$REG2 ;SELECT NEXT
ADC $REG1 ;DATA PATTERN
SOB R1,3S ;CONTINUE TEST
BR 6S
4S: ERROR 256 ;QR FYLL FAILED CN INTEGER LONG
BR 5S

;*****
;QR FYLL OK - NO CHECK LOGICAL AND IN FALU
6S: MOV #30,R1 ;SET SOB COUNT
CLR $REG1 ;INITIALIZE DATA
```

K15

```
7834 041544 005037 001164 CLR $REG2 ;TO LOAD INTO THE
7835 041550 012737 000001 001166 MOV #BIT0,$REG3 ;SRC ACC.
7836 041556 012737 100000 001200 MOV #BIT15,STMP0 ;SAVE EXPECTED RESULT
7837 041564 012737 041572 001110 MOV #.+6,@$SLPERR ;SET ERROR LOOP
7838 041572 170011 ;S: SETD
7839 041574 170012 SETL
7840 041576 172437 001160 LDD $REG0,ACO ;LOAD THE SRC
7841 041602 175437 001704 STCDL ACO,STMP2 ;EXECUTE INSTRUCTION
7842 041606 022737 100000 001204 CMP #BIT15,STMP2 ;FIRST WORK OK?
7843 041614 001021 BNE 8$ ;BRANCH IF NO
7844 041616 005737 001206 TST STMP3 ;SECOND WORD OK?
7845 041622 001016 BNE 8$ ;BRANCH IF NO
7846 041624 032777 001000 137304 BIT #SW9,@SWR ;LOOP ON ERROR?
7847 041632 001403 BEQ 9$ ;BRANCH IF NO
7848 041634 105737 001103 TSTR $ERFLG ;ANY ERRORS YET?
7849 041640 001354 BNE 7$ ;BRANCH IF YES
7850 041642 000241 ;S: CLC
7851 041644 006137 001166 ROL $REG3 ;ROTATE THE DATA
7852 041650 006137 001164 ROL $REG2 ;PATTERN
7853 041654 077137 SOB R1,7$ ;CONTINUE LOOP
7854 041656 000402 BR TST161 ;GO TO NEXT TEST
7855 041660 104257 ;S: ERROR 257 ;FALU LOGICAL AND FAILED
7856 041662 000767 BR 9$
7857
7858 ;*****
7859 ;*TEST 161 STCDL*(INT=-2)
7860 ;*
7861 ;* THIS TEST INSURES THAT FRLC FMX02 GOES LOW WITH FXPR
7862 ;* INTEGER INCREMENT ON A LOW, PD(1) HIGH, AND FT(1) LOW.
7863 ;*****
7864 041664 000004 TST161: SCOPE
7865 041666 012737 000161 001240 MOV #STN-1,STESTN ;SET TEST NUMBER IN MAIL BOX
7866 041674 012737 041764 001222 MOV #TST162,$FSCAPE ;ESCAPE TO TEST 162 ON ERROR
7867 041702 012737 177777 001200 MOV #-1,STMP0 ;SAVE EXPECTED
7868 041710 012737 177776 001202 MOV #-2,STMP1 ;VALUE OF RESULT
7869 041716 012737 041724 001110 MOV #.+6,@$SLPERR ;SET ERROR LOOP
7870 041724 170127 000300 LDFPS #FD+FL
7871 041730 172427 140400 LDD #Q140400,ACO ;PUT -2 IN ACO
7872 041734 175437 001204 ;S: STCDL ACO,STMP2 ;EXEC INSTRUCTION UNDER TEST
7873 041740 022737 177776 001206 CMP #-2,STMP3 ;LOW ORDER BITS OK?
7874 041746 001406 BEQ TST162 ;BRANCH IF YES
7875 041750 022737 177777 001206 CMP #-1,STMP3 ;DID FMX02 CAUSE RESULT TO BE -1?
7876 041756 001001 BNE 2$ ;BRANCH IF NO
7877 041760 104274 ERROR 274 ;FRLC FMX02 NOT GOING LOW
7878 041762 104221 ;S: ERROR 221 ;DON'T KNOW WHAT HAPPENED
7879
7880 ;*****
7881 ;*TEST 162 ABS VAL ROM (EXP DIFF = +)
7882 ;*
7883 ;* THIS TEST CHECKS THE CONTENTS OF THE ABS VAL ROM FOR ADDRESSES
7884 ;* 001 THRU 077 AND 201 THRU 277.
7885 ;*
7886 ;* IF A FAILURE IS DETECTED THE ROM ADDRESS BEING TESTED AND
7887 ;* THE EXPECTED CONTENTS OF THE ROM ARE TYPED OUT.
7888 ;*****
7889 041764 000004 TST162: SCOPE
```

```

7890 041766 012737 000162 001240      MOV      #STW-1,STSTN      ;;SET TEST NUMBER IN MAIL BOX
7891                                ;SECTION 1 - PD(0) ADDRESSES 001 THRU 030.
7892 041774 012737 040200 001160      MOV      #40200,$REG0     ;INITIALIZE THE
7893 042002 005037 001162                CLR      $REG1           ;DESTINATION DATA
7894 042006 012737 040300 001200      MOV      #40300,$TMP0     ;INITIALIZE THE
7895 042014 005037 001202                CLR      $TMP1           ;CHECK DATA
7896 042020 012737 000001 001170      MOV      #1,$REG4         ;INITIALIZE THE ADDRESS
7897 042026 012737 000377 001172      MOV      #377,$REG5       ;INITIALIZE THE ROMOUT DATA
7898 042034 012700 000030                MOV      #30,R0          ;INITIALIZE LOOP COUNT
7899 042040 012737 042046 001110      MOV      #.+6,#$LPERR     ;SET ERROR LOOP
7900 042046 172437 001160      1$:    LDF      $REG0,AC0       ;LOAD THE DST
7901 042052 172027 040000                ADDF     #-040000,AC0     ;EXECUTE THE ADD
7902 042056 173437 001200                CMPF     $TMP0,AC0       ;RESULT OK?
7903 042062 170000                CFCC
7904 042064 001040                BNE      2$              ;BRANCH IF NO
7905 042066 032777 001000 137042      BIT      #SW9,#SWR        ;LOOP ON ERROR?
7906 042074 001403                BEQ      3$              ;BRANCH IF NO
7907 042076 105737 001103                TSTR     $ERFLG          ;ANY ERRORS?
7908 042102 001361                BNE      1$              ;BRANCH IF YES
7909 042104 005237 001170      3$:    INC      $REG4         ;INCREMENT ADDRESS
7910 042110 005337 001172                DEC      $REG5           ;DECREMENT ROMOUT DATA
7911 042114 062737 000200 001160      ADD      #BIT7,$REG0       ;INCREMENT DST EXPONENT
7912 042122 042737 177600 001200      BIC      #177600,$TMP0     ;CLEAR THE EXPONENT
7913 042130 106037 001200                RORB     $TMP0           ;ADJUST
7914 042134 006037 001202                ROR      $TMP1           ;THE CHECK
7915 042140 053737 001160 001200      BIS      $REG0,$TMP0       ;CHECK EXP = DST EXP
7916 042146 022700 000002                CMP      #2,R0           ;SOB COUNT AT 2?
7917 042152 001003                BNE      4$              ;BRANCH IF NO
7918 042154 052737 000001 001202      BIS      #BIT0,$TMP1       ;FIX UP CHECK FRACTION
7919 042162 077047      4$:    SOB      R0,1$         ;CONTINUE
7920 042164 000404                BR
7921 042166 174037 001204      2$:    STF      AC0,$TMP2     ;GET RESULT
7922 042172 104262                ERROR    262             ;ABS VAL ROM FAILED
7923 042174 000743                BR      3$
7924                                ;*****
7925                                ;SECTION 2 - PD(0) ADDRESSES 31 THRU 77
7926 042176 013737 001160 001200      5$:    MOV      $REG0,$TMP0     ;INITIALIZE THE CHECK
7927 042204 105037 001103                CLRR     $ERFLG          ;DATA
7928 042210 005037 001202                CLR      $TMP1           ;SET SOB LOOP
7929 042214 012700 000047                MOV      #47,R0          ;INITIALIZE ROMOUT DATA
7930 042220 005037 001172                CLR      $REG5           ;SET ERROR LOOP
7931 042224 012737 042232 001110      MOV      #.+6,#$LPERR     ;INITIALIZE ROMOUT DATA
7932 042232 172437 001160      6$:    LDF      $REG0,AC0       ;LOAD THE DST
7933 042236 172027 040000                ADDF     #-040000,AC0     ;EXECUTE THE ADD
7934 042242 173437 001200                CMPF     $TMP0,AC0       ;RESULT OK?
7935 042246 170000                CFCC
7936 042250 001021                BNE      7$              ;BRANCH IF NO
7937 042252 032777 001000 136656      BIT      #SW9,#SWR        ;LOOP ON ERROR?
7938 042260 001403                BEQ      8$              ;BRANCH IF NO
7939 042262 105737 001103                TSTR     $ERFLG          ;ANY ERRORS?
7940 042266 001361                BNE      6$              ;BRANCH IF YES
7941 042270 062737 000200 001160      8$:    ADD      #BIT7,$REG0       ;INCREMENT DST EXPONENT
7942 042276 005237 001210                INC      $TMP4           ;INCREMENT ROM ADDRESS
7943 042302 062737 000200 001200      ADD      #BIT7,$TMP0       ;INCREMENT CHECK EXPONENT
7944 042310 077030                SOB      R0,6$           ;CONTINUE
7945 042312 000404                BR      9$

```

```

7946 042314 174037 001204 75:   STF      ACO,STMP2      ;GET RESULT
7947 042320 104262          ERROR    262          ;ABS VAL POW FAILED
7948 042322 000767          BR       85
7949                                     ;*****
7950                                     ;SECTION 3 - FD(1) ADDRESSES 201 THRU 270
7951 042324 012737 040200 001160 95:   MOV      #40200,SREG0    ;INITIALIZE
7952 042332 105037 001103          CLR     SERFLG
7953 042336 005037 001162          CLR     SREG1          ;THE DST
7954 042342 005037 001164          CLR     SREG2          ;DATA
7955 042346 005037 001166          CLR     SREG3          ;
7956 042352 012737 040300 001200 MOV     #40300,STMP0    ;INITIALIZE
7957 042360 005037 001202          CLR     STMP1         ;THE CHECK
7958 042364 005037 001204          CLR     STMP2         ;DATA
7959 042370 005037 001206          CLR     STMP3         ;
7960 042374 012737 000201 001170 MOV     #201,SREG4     ;INITIALIZE ROM ADDRESS
7961 042402 012737 000377 001172 MOV     #377,SREG5     ;INIT ROMOUT DATA
7962 042410 012700 000070          MOV     #70,R0        ;SET LOOP COUNT
7963 042414 170011          10$:  SFTD
7964 042416 172437 001160          LDD     SREG0,ACO     ;LOAD THE DST
7965 042422 172027 040000          ADDD   #040000,ACO   ;EXECUTE THE ADD
7966 042426 173437 001200          CMPD   STMP0,ACO     ;RESULT OK?
7967 042432 170000          CFCC
7968 042434 001044          BNE     11$          ;BRANCH IF NO
7969 042436 032777 001000 136472 BIT     #SW9,#SWR     ;LOOP ON ERPOP?
7970 042444 001403          BEQ     12$          ;BRANCH IF NO
7971 042446 105737 001103          ISTP   SERFLG        ;ANY ERRORS?
7972 042452 001360          BNE     10$          ;BRANCH IF YES
7973 042454 005237 001210          12$:  INC     STMP4        ;INCREMENT POW ADDRESS
7974 042460 005337 001212          DEC     STMP5        ;DECREMENT ROMOUT DATA
7975 042464 062737 000700 001160 ADD     #BIT7,SREG0    ;INCREMENT DST EXP
7976 042472 042737 177600 001200 BIC     #177600,STMP0 ;CLEAR THE EXPONENT
7977 042500 106037 001200          RORB   STMP0         ;ADJUST
7978 042504 006037 001202          ROR    STMP1         ;CHECK
7979 042510 006037 001204          ROR    STMP2         ;FRACTION
7980 042514 006037 001206          ROR    STMP3         ;
7981 042520 053737 001160 001200 BIC     SREG0,STMP0   ;CHECK EXP = DST EXP
7982 042526 022700 000002          CMP    #7,R0         ;SOB COUNT AT 2?
7983 042537 001003          BNE     13$          ;BRANCH IF NO
7984 042534 052737 000001 001206 BIC     #BIT0,STMP3   ;FIX UP CHECK FRACTION
7985 042542 077054          13$:  SOB    R0,10$      ;CONTINUE
7986 042544 000404          BR     14$
7987 042546 174037 001210          11$:  STD     ACO,STMP4    ;GET RESULT
7988 042552 104263          EPROR  263          ;ABS VAL POW FAILED
7989 042554 000737          BR     12$
7990                                     ;*****
7991                                     ;SECTION 4 - FD(1) ADDRESSES 271 THRU 277
7992 042556 013737 001160 001200 14$:  MOV     SREG0,STMP0    ;INITIALIZE
7993 042564 105037 001103          CLR     SERFLG
7994 042570 005037 001206          CLR     STMP3        ;THE CHECK DATA
7995 042574 012700 000007          MOV     #7,R0        ;SET SOB COUNT
7996 042600 005037 001172          CLR     SREG5        ;INITIALIZE ROMOUT DATA
7997 042604 170011          15$:  SETD
7998 042606 172437 001160          LDD     SREG0,ACO     ;LOAD THE DST
7999 042612 172027 040000          ADDD   #040000,ACO   ;EXECUTE THE ADD
8000 042616 173437 001200          CMPD   STMP0,ACO     ;RESULT OK?
8001 042622 170000          CFCC

```

```
8002 042624 001021          BNE      16$          ;BRANCH IF NO
8003 042626 032777 001000 136302 BIT      #SW9,#SWR    ;LOOP ON ERROR?
8004 042634 001403          BEQ      17$          ;BRANCH IF NO
8005 042636 105737 001103 TSTP    $ERFLG       ;ANY ERRORS?
8006 042642 001360          BNE      15$          ;BRANCH IF YES
8007 042644 062737 000200 001160 17$: ADD    #BIT7,$REG0    ;INCREMENT DST EXPONENT
8008 042652 062737 000200 001200 ADD    #BIT7,$TMP0    ;INCREMENT CHECK EXP
8009 042660 005237 001170 INC     $REG4         ;INCREMENT ROM ADDRESS
8010 042664 077031          SOB     R0,15$       ;CONTINUE
8011 042666 000404          BR      TST163       ;;GO TO NEXT TEST
8012 042670 174037 001210 16$: STD   ACO,$TMP4     ;GET RESULT
8013 042674 104263          ERROR   263         ;ABS VAL ROM FAILED
8014 042676 000762          BR      17$
8015
8016
8017
8018
8019
8020
8021
8022 042700 000004          ;;*****
;*TEST 163      ABS VAL ROM (EXP DIFF = -)
;*
;* THIS TEST CHECKS THE CONTENTS OF THE ABS VAL ROM FOR
;* ADDRESSES 100 THRU 177 AND 300 THRU 377
;*****
TST163: SCOPE
8023 042702 012737 000163 001240 MOV     #STN-1,$TESTN ;;SET TEST NUMBER IN MAIL BOX
8024
;SECTION 1 - FD(0) ADDRESSES 177 THRU 150
8025 042710 012737 040200 001160 MOV     #40200,$REG0  ;INITIALIZE
8026 042716 005037 001162 CLR     $REG1         ;THE SRC DATA
8027 042722 012737 040300 001200 MOV     #40300,$TMP0  ;INITIALIZE
8028 042730 005037 001202 CLR     $TMP1         ;THE CHECK DATA
8029 042734 012737 000177 001170 MOV     #177,$REG4   ;INITIALIZE THE ADDRESS
8030 042742 012737 000377 001172 MOV     #377,$REG5   ;INIT THE ROMOUT DATA
8031 042750 012700 000030 MOV     #30,R0       ;INIT LOOP COUNT
8032 042754 012737 042762 001110 MOV     #.+6,#SLPERR ;SET ERROR LOOP
8033 042762 172427 040000 15$: LDF    #-040000,ACO ;LOAD THE DST
8034 042766 172037 001160 ADDP   $REG0,ACO     ;EXECUTE THE ADD
8035 042772 173437 001200 CMPF   $TMP0,ACO     ;RESULT OK?
8036 042776 170000          CFCC
8037 043000 001040          BNE     2$          ;BRANCH IF NO
8038 043002 032777 001000 136126 BIT     #SW9,#SWR    ;LOOP ON ERROR?
8039 043010 001403          BEQ     3$          ;BRANCH IF NO
8040 043012 105737 001103 TSTR   $ERFLG       ;ANY ERRORS?
8041 043016 001361          BNE     1$          ;BRANCH IF YES
8042 043020 005337 001210 3$: DEC   $TMP4         ;DEC ROM ADDRESS
8043 043024 005337 001212 DEC     $TMP5         ;DEC ROMOUT DATA
8044 043030 062737 000200 001160 ADD    #BIT7,$REG0    ;INCREMENT SRC EXP
8045 043036 042737 177600 001200 BTC    #177600,$TMP0 ;CLEAR THE EXPONENT
8046 043044 106037 001200 RORB   $TMP0         ;ADJUST THE
8047 043050 006037 001202 ROR    $TMP1         ;CHECK FRACTION
8048 043054 053737 001160 001200 BIS    $REG0,$TMP0    ;EXPONENT
8049 043062 022700 000002 CMP     #2,R0        ;LOOP COUNT = 2?
8050 043066 001003          BNE     4$          ;BRANCH IF NO
8051 043070 052737 000001 001202 BIS    #BIT0,$TMP1    ;FIX UP CHECK FRACTION
8052 043076 077047          SOB     R0,15$       ;CONTINUE
8053 043100 000404          BR      5$
8054 043102 174037 001204 2$: STF   ACO,$TMP2     ;GET RESULT
8055 043106 104262          ERROR   262         ;ABS VAL ROM FAILED
8056 043110 000743          BR      3$
8057
;*****
```

```

;SECTION 2 - PD(0) ADDRESSES 147 THRU 100
8058
8059 043112 105037 001103 5$: CLR  $ERFLG ;
8060 043116 013737 001160 001200 MOV  $REG0,$TMP0 ;INIT CHECK
8061 043124 005037 001202 CLR  $TMP1 ;DATA
8062 043130 005037 001217 CLR  $TMP5 ;INIT ROMOUT DATA
8063 043134 012700 000050 MOV  #50,R0 ;SET SOB LOOP COUNT
8064 043140 012737 043146 001110 MOV  #-+6,#$SLPERR ;SET ERROR LOOP
8065 043146 172427 040000 6$: LDF  #-040000,ACO ;LOAD THE DST
8066 043152 172037 001160 ADDP $REG0,ACO ;EXECUTE THE ADD
8067 043156 173437 001200 CMPF $TMP0,ACO ;RESULT OK?
8068 043162 170000 CFCC
8069 043164 001021 BNE  7$ ;BRANCH IF NO
8070 043166 032777 001000 135742 BIT  #$SW9,#SWR ;LOOP ON ERROR?
8071 043174 001403 BEQ  8$ ;BRANCH IF NO
8072 043176 105737 001103 TSTR $ERFLG ;ANY ERRORS?
8073 043202 001361 BNE  6$ ;BRANCH IF YES
8074 043204 062737 000200 001160 8$: ADD  #BIT7,$REG0 ;INCREMENT SRC EXP
8075 043212 062737 000200 001200 ADD  #BIT7,$TMP0 ;INCREMENT CHECK EXP
8076 043220 005337 001210 DEC  $TMP4 ;DEC ROM ADDRESS
8077 043224 077030 SOB  R0,6$ ;CONTINUE
8078 043226 000404 BR   9$ ;
8079 043230 174037 001204 7$: STF  ACO,$TMP2 ;GET RESULT
8080 043234 104262 ERROR 262 ;ABS VAL ROM FAILED
8081 043236 000762 BR   8$ ;
8082
;*****
8083 ;SECTION 3 - PD(1) ADDRESSES 377 THRU 310
8084 043240 105037 001103 9$: CLR  $ERFLG ;
8085 043244 012737 040200 001160 MOV  #40200,$REG0 ;INIT SRC
8086 043252 005037 001162 CLR  $REG1 ;DATA
8087 043256 005037 001164 CLR  $REG2 ; *
8088 043262 005037 001166 CLR  $REG3 ; *
8089 043266 012737 040300 001200 MOV  #40300,$TMP0 ;INIT CHECK
8090 043274 005037 001202 CLR  $TMP1 ;DATA
8091 043300 005037 001204 CLR  $TMP2 ; *
8092 043304 005037 001206 CLR  $TMP3 ; *
8093 043310 012737 000377 001210 MOV  #377,$TMP4 ;INIT ROM ADDRESS
8094 043316 012737 000377 001212 MOV  #377,$TMP5 ;INIT ROMOUT DATA
8095 043324 012700 000070 MOV  #70,R0 ;SET SOB LOOP COUNT
8096 043330 012737 043336 001110 MOV  #-+6,#$SLPERR ;SET ERROR LOOP
8097 043336 170011 10$: SETD
8098 043340 172427 040000 LDD  #-040000,ACO ;LOAD THE DST
8099 043344 172037 001160 ADDD $REG0,ACO ;EXECUTE THE ADD
8100 043350 173437 001200 CMPD $TMP0,ACO ;RESULT OK?
8101 043354 170000 CFCC
8102 043356 001044 BNE  11$ ;BRANCH IF NO
8103 043360 032777 001000 135550 BIT  #$SW9,#SWR ;LOOP ON ERROR?
8104 043366 001403 BEQ  12$ ;BRANCH IF NO
8105 043370 105737 001103 TSTR $ERFLG ;ANY ERRORS?
8106 043374 001360 BNE  10$ ;BRANCH IF YES
8107 043376 062737 000200 001160 12$: ADD  #BIT7,$REG0 ;INCREMENT SRC EXP
8108 043404 042737 177600 001200 BIC  #177600,$TMP0 ;CLEAR THE EXPONENT
8109 043412 106037 001200 RORB $TMP0 ;ADJUST
8110 043416 006037 001202 ROR  $TMP1 ;THE
8111 043422 006037 001204 ROR  $TMP2 ;CHECK
8112 043426 006037 001206 ROR  $TMP3 ;FRACTION & EXP
8113 043432 053737 001160 001200 BTS  $REG0,$TMP0 ;

```



```

8114 043440 005337 001210          DFC      $TMP4          ;DEC ROM ADDRESS
8115 043444 005337 001212          DEC      $TMP5          ;DEC ROMOUT DATA
8116 043450 022700 000007          CMP      #7,RO         ;LOOP COUNT = 2
8117 043454 001003                    BNE      13$           ;BRANCH IF NC
8118 043456 052737 000001 001206   BIS      #BIT0,$TMP3   ;FIX UP CHECK FRACTION
8119 043464 077054                    SOB      RO,10$        ;CONTINUE
8120 043466 000404                    BR       14$
8121 043470 174037 001210          11$:    STD      ACO,$TMP4   ;SAVE RESULT
8122 043474 104263                    ERROR   263           ;ABS VAL ROM FAILED
8123 043476 000737                    BR       12$
8124
8125                                ;;*****
;SECTION 4 - PD(1) ADDRESSES 307 THPU 300
8126 043500 105037 001103          14$:    CLR      $ERFLG
8127 043504 005037 001212          CLR      $TMP5          ;INIT ROMOUT DATA
8128 043510 013737 001160 001200   MOV      $REG0,$TMP0   ;INIT THE
8129 043516 005037 001206          CLR      $TMP3          ;CHECK DATA
8130 043522 012700 000010          MOV      #10,RO        ;SET LOOP COUNT
8131 043526 170011                    15$:    SETD
8132 043530 172427 040000          LDD      #040000,ACO   ;LOAD THE DST
8133 043534 172037 001160          ADDD     $REG0,ACO     ;EXECUTE THE ADD
8134 043540 173437 001200          CMPD     $TMP0,ACO     ;RESULT OK?
8135 043544 170000          CFCC
8136 043546 001021                    BNE      16$           ;BRANCH IF NO
8137 043550 032777 001000 135360   BIT      #SW9,@SWR     ;LOOP ON ERROR?
8138 043556 001403                    BEQ      17$           ;BRANCH IF NO
8139 043560 105737 001103          TSTR     $ERFLG        ;ANY ERRORS?
8140 043564 001360                    BNE      15$           ;BRANCH IF YES
8141 043566 062737 000200 001160 17$:    ADD      #BIT7,$REG0   ;INC SRC EXP
8142 043574 013737 001160 001700   MOV      $REG0,$TMP0   ;AND CHECK EXP
8143 043602 005337 001210          DEC      $TMP4          ;DEC ROM ADDRESS
8144 043606 077031                    SOB      RO,15$        ;CONTINUE
8145 043610 000404                    BR       $EOP          ;;GO TO NEXT TEST
8146 043612 174037 001210          16$:    STD      ACO,$TMP4   ;GET RESULT
8147 043616 104263                    ERROR   263           ;ABS VAL ROM FAILED
8148 043620 000762                    BR       17$
8149
8150                                .SBTTL  END OF PASS ROUTINE
8151
8152                                ;;*****
8153                                ;*INCREMENT THE PASS NUMBER ($PASS)
8154                                ;*TYPE "END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYY"
8155                                ;*WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS
8156                                ;*IF THERES A MONITOR GO TO IT
8157                                ;*IF THERF ISN'T JUMP TO LOOP
8158
8159                                $EOP:
8160                                SCOPE
8161 043622 000004          CLR      $STNM          ;;ZERO THE TEST NUMBER
8162 043624 005037 001102          CLR      $TIMES        ;;ZERO THE NUMBER OF ITERATIONS
8163 043630 005037 001220          INC      $PASS         ;;INCEMENT THE PASS NUMBER
8164 043634 005237 001242          BIC      #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
8165 043640 042737 100000 001242   DEC      (PC)+         ;;LOOP?
8166 043646 005327          $EOPCT: .WORD 1
8167 043650 000001          BGT      $DOAGN        ;;YES
8168 043652 003063          MOV      (PC)+,@(PC)+ ;;RESTORE COUNTER
8169 043654 012737          SENDCT: .WORD 1
8169 043656 000001

```

```

8170 043660 043650
8171 043662 104400 043670
8172 043666 000407
8173
8174 043706
8175 043706 013746 001242
8176
8177 043712 104404
8178 043714 104400 043722
8179 043720 000421
8180
8181 043764
8182 043764 013746 001112
8183
8184 043770 104404
8185 043772 104400 001231
8186 043776 005037 001112
8187 044002 013700 000042
8188 044006 001405
8189 044010 000005
8190 044012 004710
8191 044014 000240
8192 044016 000240
8193 044020 000240
8194 044022
8195 044022 000137 005254
8196 044026 377 377 000
8197 044032
8198
8199
8200
8201
8202
8203
8204
8205
8206
8207
8208
8209
8210
8211
8212
8213 044032
8214 044032 105737 001102
8215 044036 001423
8216 044040 010346
8217 044042 005003
8218 044044 170103
8219 044046 012603
8220 044050 032777 000400 135060
8221 044056 001013
8222 044060 010346
8223 044062 117703 135050
8224 044066 170003
8225 044070 012603

SEOPCT
TYPE ,65$ ;;TYPE ASCIZ STRING
BR 64$ ;;GET OVER THE ASCIZ
;;65$: .ASCIZ <12><15>/END PASS #/
64$:
MOV $PASS,-(SP) ;;SAVE $PASS FOR TYPEOUT
;;TYPE PASS NUMBER
TYPDS
TYPE ,67$ ;;GO TYPE--DECIMAL ASCII WITH SIGN
BR 66$ ;;TYPE ASCIZ STRING
;;67$: .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
66$:
MOV $ERTTL,-(SP) ;;SAVE $ERTTL FOR TYPEOUT
;;TOTAL NUMBER OF ERRORS
TYPDS
TYPE ,SCRLF ;;GO TYPE--DECIMAL ASCII WITH SIGN
CLR $ERTTL ;;TYPE CARRIAGE RETURN, LINE FEED
$GET42: MOV @#42,R0 ;;CLEAR ERROR TOTAL
BEQ $DOAGN ;;GET MONITOR ADDRESS
RSET ;;BRANCH IF NO MONITOR
SENDAD: JSR PC,(R0) ;;CLEAR THE WORLD
NOP ;;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 SURTESTS. IT WILL INCREMENT
;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
;*AND LOAD THE ERROR FLAG ($ENFLG) 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
$SCOPE:
TSTP $TSTNM ;FIRST TEST?
BEQ 1$ ;BRANCH IF YES
MOV R3,-(SP) ;SAVE R3
CLR R3
LDFPS R3
MOV (SP)+,R3
BIT #SW8,#SWR ;SWITCH 8 ON?
BNE 1$ ;BRANCH IF YES
MOV R3,-(SP) ;SAVE R3
MOVR @SWR,R3 ;GET LOWER SWITCHES
LDUP ;PUT IN MICRO-BREAK REGISTER
MOV (SP)+,R3 ;RESTORE R3

```

```

8226 044072 012737 047504 000004      MOV      #CPSUR,EPRVEC
8227 044100 012737 047624 000244      MOV      #FPSUR,PPPVEC
8228 044106 032777 040000 135022 1S:    BIT      #BIT14,SWR      ;;LOOP ON PRESENT TEST?
8229 044114 001114                BNE      $OVER        ;;YES IF SW14=1
8230                ;#####START OF CODE FOR THE XOR TESTER#####
8231 044116 000416                SXTSTR: BR      6S      ;;IF RUNNING ON THE "XOR" TESTER CHANGE
8232                ;THIS INSTRUCTION TO A "NOP" (NOP=240)
8233 044120 013746 000004      MOV      @EPRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
8234 044124 012737 044144 000004      MOV      #5,$@EPRVEC  ;;SET FOR TIMEOUT
8235 044132 005737 177060                TST      @177060      ;;TIME OUT ON XOR?
8236 044136 012637 000004      MOV      (SP)+,@EPRVEC ;;RESTORE THE ERROR VECTOR
8237 044142 000463                BP       $SVLAD      ;;GO TO THE NEXT TEST
8238 044144 022626                5S:    CMP      (SP)+,(SP)+ ;;CLEAR THE STACK AFTER A TIME OUT
8239 044146 012637 000004      MOV      (SP)+,@EPRVEC ;;RESTORE THE ERROR VECTOR
8240 044152 000423                BR       7S          ;;LOOP ON THE PRESENT TEST
8241 044154                6S:;#####END OF CODE FOR THE XOR TESTER#####
8242 044154 032777 000400 134754      BIT      #BIT08,SWR    ;;LOOP ON SPEC. TEST?
8243 044162 001404                BEQ      2S          ;;BR IF NO
8244 044164 127737 134746 001102      CMPP    @SWR,$STSTN   ;;ON THE RIGHT TEST? SWR<7:0>
8245 044172 001465                BEQ      $OVER      ;;BR IF YES
8246 044174 105737 001103                2S:    TSTR    $ERFLG     ;;HAS AN ERROR OCCURRED?
8247 044200 001421                BEQ      3S          ;;BR IF NO
8248 044202 123737 001115 001103      CMPB    $ERMAX,$ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
8249 044210 101015                BHI     3S          ;;BR IF NO
8250 044212 032777 001000 134716      BIT      #BIT09,SWR    ;;LOOP ON ERROR?
8251 044220 001404                BEQ      4S          ;;BR IF NO
8252 044222 013737 001110 001106      7S:    MOV      $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
8253 044230 000446                BR       $OVER
8254 044232 105037 001103                4S:    CLRB    $ERFLG     ;;ZERO THE ERROR FLAG
8255 044236 005037 001220                CLR     $TIMES      ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
8256 044242 000415                BP       1S          ;;ESCAPE TO THE NEXT TEST
8257 044244 032777 004000 134664      3S:    BIT      #BIT11,SWR   ;;INHIBIT ITERATIONS?
8258 044252 001011                BNE     1S          ;;BR IF YES
8259 044254 005737 001242                TST    $PASS       ;;IF FIRST PASS OF PROGRAM
8260 044260 001406                BEQ    1S          ;;      INHIBIT ITERATIONS
8261 044262 005237 001104                INC    $ICNT      ;;INCREMENT ITERATION COUNT
8262 044266 023737 001220 001104      CMP     $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
8263 044274 002024                BGE    $OVER      ;;BR IF MORE ITERATION REQUIRED
8264 044276 012737 000001 001104      1S:    MOV     #1,$ICNT  ;;REINITIALIZE THE ITERATION COUNTER
8265 044304 013737 044367 001220      MOV     $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
8266 044312 105237 001102                $SVLAD: INCR    $STSTN ;;COUNT TEST NUMBERS
8267 044316 113737 001102 001240      MOVB   $STSTN,$TESTN ;;SET TEST NUMBER IN APT MAILBOX
8268 044324 011637 001106                MOV    (SP),$LPADR  ;;SAVE SCOPE LOOP ADDRESS
8269 044330 011637 001110                MOV    (SP),$LPERR  ;;SAVE ERROR LOOP ADDRESS
8270 044334 005037 001222                CLR    $ESCAPE     ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
8271 044340 112737 000001 001115      MOVB   #1,$ERMAX   ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
8272 044346 013777 001102 134564      $OVER: MOV    $STSTN,$DISPLAY ;;DISPLAY TEST NUMBER
8273 044354 013716 001106                MOV    $LPADR,(SP)  ;;PUDGE RETURN ADDRESS
8274 044360 000002                RTI                    ;;FIXES PS
8275 044362 003720                $MXCNT: 2000.        ;;MAX. NUMBER OF ITERATIONS
8276
8277                .SBTTL  ERROR HANDLER ROUTINE
8278
8279                ;*****
8280                ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
8281                ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL

```

```

8282                   ;*AND GO TO $ERRTYP ON ERROR
8283                   ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
8284                   ;*SW15=1            HALT ON ERROR
8285                   ;*SW13=1            INHIBIT ERROR TYPEOUTS
8286                   ;*SW10=1            BELL ON ERROR
8287                   ;*SW09=1            LOOP ON ERROR
8288                   ;*CALL
8289                   ;*            ERROR    N            ;;ERROR=EMT AND N=ERROR ITEM NUMBER
8290
8291                   $ERROR:
8292   044364   122737   000001   001102            CMPB    #1,$STSTNM            ;FIRST TEST?
8293   044372   001413                            BEQ       7$                    ;BRANCH IF YES
8294   044374   032777   000400   134534            BIT       #SWR,@SWR            ;SWITCH 8 ON?
8295   044402   001007                            BNE       7$                    ;BRANCH IF YES
8296   044404   010346                            MOV       R3,-(SP)            ;SAVE R3
8297   044406   005003                            CLR       R3
8298   044410   170103                            LDFPS    R3                    ;CLEAR THE FPS REGISTER
8299   044412   117703   134520                    MOVB     @SWR,R3            ;GET LOWER SWITCHES
8300   044416   170003                            LDUB                         ;PUT IN MICRO-BREAK REG
8301   044420   012603                            MOV       (SP)+,R3            ;RESTORE R3
8302   044422   105737   001103            7$:    INCB     $ERRFLG            ;;SET THE ERROR FLAG
8303   044426   001775                            BEQ       7$                    ;;DON'T LET THE FLAG GO TO ZERO
8304   044430   013777   001102   134502            MOV       $STSTNM,@DISPLAY   ;;DISPLAY TEST NUMBER AND ERROR FLAG
8305   044436   032777   002000   134472            BIT       #BIT10,@SWR       ;;BELL ON ERROR?
8306   044444   001402                            BEQ       1$                    ;;NO - SKIP
8307   044446   104400   001224                    TYPE     ,$BELL            ;;RING BELL
8308   044452   005237   001112            1$:    INC       $RPTL            ;;COUNT THE NUMBER OF ERRORS
8309   044456   011637   001116                    MOV       (SP),$ERRPC       ;;GET ADDRESS OF ERROR INSTRUCTION
8310   044462   162737   000002   001116            SUB       #2,$ERRPC
8311   044470   117737   134422   001114            MOVB     @ERRPC,$ITEMB       ;;STRIP AND SAVE THE ERORR ITEM CODE
8312   044476   032777   020000   134432            BIT       #BIT13,@SWR       ;;SKIP TYPEOUT IF SET
8313   044504   001004                            BNE       20$                 ;;SKIP TYPEOUTS
8314   044506   004737   046274                    JSR       PC,$ERRTYP        ;;GO TO USER ERROR ROUTINE
8315   044512   104400   001231                    TYPE     ,$SCLF
8316   044516                                       20$:
8317   044516   122737   000001   001254            CMPB     #APTENV,$ENV        ;;RUNNING IN APT MODE
8318   044524   001007                            BNE       2$                    ;;NO,SKIP APT ERROR REPORT
8319   044526   113737   001114   044540            MOVB     $ITEMB,21$         ;;SET ITEM NUMBER AS ERROR NUMBER
8320   044534   004737   046044                    JSR       PC,$ATY4            ;;REPORT FATAL ERROR TO APT
8321   044540       000                            21$:    .BYTE    0
8322   044541       000                            .BYTE    0
8323   044542   000777                            22$:    BR       22$            ;;APT ERROR LOOP
8324   044544   005777   134366            2$:    TST       @SWR            ;;HALT ON ERROR
8325   044550   100001                            BPL       3$                    ;;SKIP IF CONTINUE
8326   044552   000000                            HALT                         ;;HALT ON ERROR!
8327   044554   032777   001000   134354   3$:    BIT       #BIT09,@SWR        ;;LOOP ON ERROR SWITCH SET?
8328   044562   001402                            BEQ       4$                    ;;BR IF NO
8329   044564   013716   001110                    MOV       $LPERR,(SP)        ;;FUDGE RETURN FOR LOOPING
8330   044570   005737   001222            4$:    TST       $ESCAPE            ;;CHECK FOR AN ESCAPE ADDRESS
8331   044574   001402                            BEQ       5$                    ;;BR IF NONE
8332   044576   013716   001222                    MOV       $ESCAPE,(SP)       ;;FUDGE RETURN ADDRESS FOR ESCAPE
8333   044602                                       5$:
8334   044602   022737   044012   000047            CMP       #SENDAD,@#42        ;;ACT-11 AUTO-ACCEPT?
8335   044610   001001                            BNE       6$                    ;;BRANCH IF NO
8336   044612   000000                            HALT                         ;;YES
8337   044614                                       6$:

```

```

8338 044614 000002
8339
8340
8341
8342
8343
8344
8345
8346
8347
8348
8349
8350
8351
8352
8353
8354
8355
8356 044616 104405
8357 044620 017600 000000
8358 044624 062716 000002
8359 044630 016001 000002
8360 044634 011000
8361 044636 012704 001333
8362 044642 112744 000000
8363 044646 012705 000005
8364 044652 010103
8365 044654 042703 177770
8366 044660 062703 000060
8367 044664 110344
8368 044666 073027 177775
8369 044672 077511
8370 044674 010103
8371 044676 042703 177776
8372 044702 062703 000060
8373 044706 110344
8374 044710 112744 000040
8375 044714 073027 177777
8376 044720 012705 000002
8377 044724 010103
8378 044726 042703 177770
8379 044732 062703 000060
8380 044736 110344
8381 044740 073027 177775
8382 044744 077511
8383 044746 010103
8384 044750 042703 177776
8385 044754 062703 000060
8386 044760 110344
8387 044762 112744 000040
8388 044766 112744 000040
8389 044772 072127 177777
8390 044776 012705 000002
8391 045002 010103
8392 045004 042703 177770
8393 045010 062703 000060
  
```

```

RTI ;;RETURN
;*****
.SBTTL CONVERT FLOATING BINAPY TO OCTAL ASCIZ
;*
;*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.
;*****
SPL20: SAVREG
MOV      0(SP),R0      ;GET ADDRESS OF DATA
ADD      #2,(SP)      ;ADJUST RETURN PC
MOV      2(R0),R1      ;PUT SECOND DATA WORD IN R1
MOV      (R0),R0       ;PUT FIRST DATA WORD IN R0
MOV      #SPLBUFF+73,R4 ;GET ADDRESS OF BUFFER END IN R4
MOVR    #0,-(R4)      ;PUT TERMINATOR IN BUFFER
MOV      #5,R5        ;SET SOB COUNT FOR FRACTION DIGITS
15:      MOV      R1,R3      ;GET LSB'S OF FRACTION
BIC      #^C7,R3      ;SAVE LS 3 BITS
ADD      #60,R3       ;MAKE THEM ASCII
MOVR    R3,-(R4)      ;STORE IN BUFFER
ASHC    #-3,R0        ;SHFT NUMBER TO NEXT 3 BITS
SOB     R5,15         ;CONTINUE FOR 7 DIGITS
MOV      R1,R3        ;GET NEXT DIGITS
BIC      #^C1,R3      ;ONLY WANT 1 BIT
ADD      #60,R3       ;MAKE THEM ASCII
MOVB    R3,-(R4)      ;STORE IN BUFFER
MOVR    #40,-(R4)     ;PUT SPACE IN BUFFER
ASHC    #-1,R0
MOV      #2,R5        ;SET SOB COUNT
35:      MOV      R1,R3      ;GET LOW WORD
BIC      #^C7,R3      ;MASK 3 BITS
ADD      #60,R3       ;MAKE THEM ASCII
MOVB    R3,-(R4)      ;PUT IN BUFFER
ASHC    #-3,R0        ;GET NEXT 3 BITS
SOB     R5,35         ;CONVERT THEM
MOV      R1,R3
BIC      #^C1,R3      ;ONLY WANT 1 BIT
ADD      #60,R3       ;MAKE IT ASCII
MOVB    R3,-(R4)      ;PUT IN BUFFER
MOVR    #40,-(R4)     ;PUT SPACE IN BUFFER
MOVR    #40,-(R4)
ASH     #-1,R1        ;GET FIRST 3 BITS OF EXPONENT
MOV      #2,R5        ;SET SOB COUNT FOR 2 DIGITS
25:      MOV      R1,R3      ;GET LSB'S OF EXPONENT
BIC      #^C7,R3      ;SAVE 3 BITS
ADD      #60,R3       ;MAKE THEM ASCII
  
```

8394	045014	110344		
8395	045016	072127	177775	
8396	045022	077511		
8397	045024	010103		
8398	045026	042703	177774	
8399	045032	062703	000060	
8400	045036	110344		
8401	045040	112744	000040	
8402	045044	112744	000040	
8403	045050	042700	177776	
8404	045054	062700	000060	
8405	045060	110044		
8406	045062	104406		
8407	045064	011646		
8408	045066	016666	000004	000002
8409	045074	012766	001310	000004
8410	045102	000006		
8411				
8412				
8413				
8414				
8415				
8416				
8417				
8418				
8419				
8420				
8421				
8422				
8423				
8424				
8425				
8426				
8427				
8428				
8429				
8430				
8431	045104	104405		
8432	045106	017637	000000	045122
8433	045114	062716	000002	
8434	045120	104407		
8435	045122	000000		
8436	045124	012600		
8437	045126	010037	001354	
8438	045132	062700	000041	
8439	045136	105040		
8440	045140	013701	045122	
8441	045144	062701	000004	
8442	045150	012102		
8443	045152	012103		
8444	045154	012701	000002	
8445	045160	012704	000005	
8446	045164	010305		
8447	045166	042705	177770	
8448	045172	062705	000060	
8449	045176	110540		

```

MOV B R3,-(R4) ;STORE IN BUFFER
ASH #3,R1 ;GET NEXT 3 BITS
SOB R5,Z$ ;CONTINUE
MOV R1,R3 ;GET LAST 2 BITS OF EXPONENT
BIC #C3,R3 ;MAKE SURE ONLY 2 BITS
ADD #60,R3 ;MAKE THEM ASCII
MOV B R3,-(R4) ;STORE IN BUFFER
MOV B #40,-(R4) ;PUT SPACE IN BUFFER
MOV B #40,-(R4)
BIC #C1,R0 ;GET SIGN BIT (IT WAS EXTENDED)
ADD #60,R0 ;MAKE IT ASCII
MOV B R0,-(R4) ;PUT IT IN THE BUFFER
RESREG
MOV (SP),-(SP) ;SAVE RETURN PC
MOV 4(SP),2(SP) ;AND RETURN PSW
MOV #SFLBUFF,4(SP) ;PUT BUFFER ADDRESS ON STACK
RTT ;RETURN
.EVEN

```

```

*****
.SBTTL CONVERT FLOATING DOUBLE BINARY TO OCTAL ASCII
;*
;*THIS ROUTINE CONVERTS A 64 BIT FLOATING NUMBER TO AN OCTAL
;*ASCII STRING IN THE FOLLOWING FORMAT:
;*
;*      U VVV WWW XXXXX YYYYY ZZZZZ
;*
;*      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 ASCII STRING ON THE STACK.
*****

```

```

SFLD20: SAVREG
MOV @ (SP),15 ;GET ADDRESS OF DATA TO CONVERT
ADD #7,(SP) ;ADJUST RETURN PC
FL20 ;CONVERT MS 32 BITS
15: .WORD
MOV (SP)+,R0 ;GET ADDRESS OF CONVERTED DATA
MOV R0,$BUFF ;SAVE IT
ADD #41,R0 ;ADJUST TO END OF BUFFER
CLRB -(R0) ;PUT TERMINATOR IN BUFFER
MOV 15,R1 ;GET ADDRESS OF DATA TO CONVERT
ADD #4,R1 ;ADJUST TO LOWER 32 BITS
MOV (R1)+,R2 ;SAVE THE DATA
MOV (R1)+,R3
MOV #2,R1 ;SET LOOP COUNT
35: MOV #5,R4 ;SET LOOP COUNT
45: MOV R3,R5 ;GET LS 32 BITS OF DATA
BIC #C7,R5 ;MASK 3 BITS
ADD #60,R5 ;MAKE THEM ASCII
MOV B R5,-(R0) ;PUT IN BUFFER

```

8450	045200	073227	177775		ASHC	#-3,R7		;GET NEXT 3 BITS
8451	045204	077411			SOB	R4,4\$;CONTINUE
8452	045206	010305			MOV	R3,R5		;GET LS 32 BITS
8453	045210	042705	177776		BIC	#C1,R5		;ONLY WANT 1 BIT
8454	045214	062705	000060		ADD	#60,R5		;MAKE IT ASCII
8455	045220	110540			MOVB	R5,-(P0)		;PUT IN TABLE
8456	045222	112740	000040		MOVB	#40,-(R0)		;PUT SPACE IN TABLE
8457	045226	073227	177777		ASHC	#-1,R7		
8458	045232	077126			SOB	R1,3\$;CONVERT NEXT 16 BITS
8459	045234	104406			RESREG			
8460	045236	011646			MOV	(SP),-(SP)		;ADJUST STACK
8461	045240	016666	000004	000007	MOV	4(SP),2(SP)		;TO RETURN WITH ADDRESS
8462	045246	013766	001354	000004	MOV	\$BUFF,4(SP)		;OF BUFFER ON STACK
8463	045254	000006			RTT			;RETURN
8464								
8465								
8466								
8467								
8468								
8469								
8470								
8471	045256	012737	045272	000004	GETTIK: MOV	#2\$,ERRVEC		
8472	045264	005737	177546		TST	LKSTAT		;LINE CLOCK INSTALLED?
8473	045270	000441			BR	3\$;BRANCH IF YES
8474	045272	012706	001100		2\$: MOV	#STACK,SP		
8475	045276	005737	001242		TST	\$PASS		;FIRST PASS?
8476	045302	001032			BNE	4\$;BRANCH IF NO
8477	045304	005227	177777		INC	#-1		
8478	045310	001027			BNE	4\$		
8479	045312	104400	045320		TYPE	,65\$;TYPE ASCII STRING
8480	045316	000424			BR	64\$;GET OVER THE ASCII
8481					;;65\$: .ASCII	"SHOULD HAVE LINE CLOCK INSTALLED!!!!!!"		
8482	045370				64\$:			
8483	045370	000177	133626		4\$: JMP	@\$ESCAPE		
8484	045374	013746	000100		3\$: MOV	LKVEC,-(SP)		;SAVE CONTENTS OF VECTOR
8485	045400	013746	177776		MOV	PSW,-(SP)		;SAVE PSW
8486	045404	005037	177776		CLR	PSW		
8487	045410	012737	045426	000100	MOV	#1\$,LKVEC		;SETUP THE VECTOR
8488	045416	012737	000100	177546	MOV	#BIT6,LKSTAT		;START THE INTERRUPT
8489	045424	000001			WAIT			;WAIT FOR IT TO HAPPEN
8490	045426	022626			1\$: CMP	(SP)+,(SP)+		;RESTORE THE STACK
8491	045430	012637	177776		MOV	(SP)+,PSW		;RESTORE THE PSW
8492	045434	012637	000100		MOV	(SP)+,LKVEC		;RESTORE THE VECTOR
8493	045440	012737	000100	177546	MOV	#BIT6,LKSTAT		;START THE CLOCK AGAIN
8494	045446	000207			RTS	PC		;RETURN
8495								
8496								
8497								
8498								
8499								
8500								
8501								
8502								
8503								
8504								
8505								

```

8506 ;* +2---(+1P)
8507 ;* +4---R5
8508 ;* +6---R4
8509 ;* +8---R3
8510 ;*+10---R2
8511 ;*+12---R1
8512 ;*+14---R0
8513
8514 045450 $SAVREG:
8515 045450 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
8516 045452 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
8517 045454 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
8518 045456 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
8519 045460 010446 MOV R4,-(SP) ;;PUSH R4 ON STACK
8520 045462 010546 MOV R5,-(SP) ;;PUSH R5 ON STACK
8521 045464 016646 000022 MOV 22(SP),-(SP) ;;SAVE PS OF MAIN FLOW
8522 045470 016646 000022 MOV 22(SP),-(SP) ;;SAVE PC OF MAIN FLOW
8523 045474 016646 000022 MOV 22(SP),-(SP) ;;SAVE PS OF CALL
8524 045500 016646 000022 MOV 27(SP),-(SP) ;;SAVE PC OF CALL
8525 045504 000002 RTI
8526
8527 ;*RESTORE R0-R5
8528 ;*CALL:
8529 ;* RESPEG
8530 045506 $RESREG:
8531 045506 012666 000022 MOV (SP)+,22(SP) ;;RESTORE PC OF CALL
8532 045512 012666 000022 MOV (SP)+,27(SP) ;;RESTORE PS OF CALL
8533 045516 012666 000022 MOV (SP)+,22(SP) ;;RESTORE PC OF MAIN FLOW
8534 045522 012666 000022 MOV (SP)+,22(SP) ;;RESTORE PS OF MAIN FLOW
8535 045526 012605 MOV (SP)+,R5 ;;POP STACK INTO R5
8536 045530 012604 MOV (SP)+,R4 ;;POP STACK INTO R4
8537 045532 012603 MOV (SP)+,R3 ;;POP STACK INTO R3
8538 045534 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
8539 045536 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
8540 045540 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
8541 045542 000002 RTI
8542
8543 .SBTTL TYPE ROUTINE
8544
8545 ;;*****
8546 ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
8547 ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
8548 ;*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
8549 ;*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
8550 ;*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
8551 ;*
8552 ;*CALL:
8553 ;*1) USING A TRAP INSTRUCTION
8554 ;* TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
8555 ;*OR
8556 ;* TYPE
8557 ;* MESADR
8558 ;*
8559
8560 045544 105737 001155 $TYPE: TSTP $TPFLG ;;IS THERE A TERMINAL?
8561 045550 100002 BPL IS ;;BP IF YES
  
```


8562	045552	000000			HALT		;;HALT HERE IF NO TERMINAL
8563	045554	000430			BR	3\$;;LEAVE
8564	045556	010046			MOV	RO,-(SP)	;;SAVE RO
8565	045560	017600	000002		MOV	02(SP),RO	;;GET ADDRESS OF ASCIZ STRING
8566	045564	122737	000001	001254	CMPB	#APTEW,\$ENV	;;RUNNING IN APT MODE
8567	045572	001011			BNE	62\$;;NO,GO CHECK FOR APT CONSOLE
8568	045574	132737	000100	001255	BITB	#APTSPOOL,\$ENVM	;;SPOOL MESSAGE TO APT
8569	045602	001405			BEQ	62\$;;NO,GO CHECK FOR CONSOLE
8570	045604	010037	045614		MOV	RO,61\$;;SETUP MESSAGE ADDRESS FOR APT
8571	045610	004737	046034		JSR	PC,\$ATY3	;;SPOOL MESSAGE TO APT
8572	045614	000000			.WORD	0	;;MESSAGE ADDRESS
8573	045616	132737	000040	001255	BITB	#APTCSUP,\$ENVM	;;APT CONSOLE SUPPRESSED
8574	045624	001003			BNE	60\$;;YES,SKIP TYPE OUT
8575	045626	112046			NOVB	(RO)+,-(SP)	;;PUSH CHARACTER TO BE TYPED ONTO STACK
8576	045630	001005			BNE	4\$;;BR IF IT ISN'T THE TERMINATOR
8577	045632	005726			TST	(SP)+	;;IF TERMINATOR POP IT OFF THE STACK
8578	045634	012600			MOV	(SP)+,RO	;;RESTORE RO
8579	045636	062716	000002		ADD	#2,(SP)	;;ADJUST RETURN PC
8580	045642	000002			RTI		;;RETURN
8581	045644	122716	000011		CMPB	#HT,(SP)	;;BRANCH IF <HT>
8582	045650	001430			BEQ	8\$	
8583	045652	122716	000200		CMPB	#CRLF,(SP)	;;BRANCH IF NOT <CRLF>
8584	045656	001006			BNE	5\$	
8585	045660	005726			TST	(SP)+	;;POP <CR><LF> EQUIV
8586	045662	104400			TYPE		;;TYPE A CR AND LF
8587	045664	001231			\$CRLF		
8588	045666	105037	046022		CLRB	\$CHARCNT	;;CLEAR CHARACTER COUNT
8589	045672	000755			BR	2\$;;GET NEXT CHARACTER
8590	045674	004737	045756		JSR	PC,\$TYPEC	;;GO TYPE THIS CHARACTER
8591	045700	123726	001154		CMPB	\$FILLC,(SP)+	;;IS IT TIME FOR FILLER CHARS.?
8592	045704	001350			BNE	2\$;;IF NO GO GET NEXT CHAR.
8593	045706	013746	001152		MOV	\$NULL,-(SP)	;;GET # OF FILLER CHARS. NEEDED
8594							;;AND THE NULL CHAR.
8595	045712	105366	000001		DECB	1(SP)	;;DOES A NULL NEED TO BE TYPED?
8596	045716	002770			BLT	6\$;;BR IF NO--GO POP THE NULL OFF OF STACK
8597	045720	004737	045756		JSR	PC,\$TYPEC	;;GO TYPE A NULL
8598	045724	105337	046022		DECB	\$CHARCNT	;;DO NOT COUNT AS A COUNT
8599	045730	000770			BR	7\$;;LOOP
8600							
8601							
8602							
8603	045732	112716	000040		NOVB	# ,(SP)	;;REPLACE TAB WITH SPACE
8604	045736	004737	045756		JSR	PC,\$TYPEC	;;TYPE A SPACE
8605	045742	132737	000007	046022	BITB	#7,\$CHARCNT	;;BRANCH IF NOT AT
8606	045750	001372			BNE	9\$;;TAB STOP
8607	045752	005726			TST	(SP)+	;;POP SPACE OFF STACK
8608	045754	000724			BR	2\$;;GET NEXT CHARACTER
8609	045756	105777	133164		\$TYPEC:	TSTP	;;WAIT UNTIL PRINTER IS READY
8610	045762	100375			BPL	\$TYPEC	
8611	045764	116677	000002	133156	NOVB	2(SP),@STPB	;;LOAD CHAR TO BE TYPED INTO DATA REG.
8612	045772	122766	000015	000002	CMPB	#CR,2(SP)	;;IS CHARACTER A CARRIAGE RETURN?
8613	046000	001003			BNE	1\$;;BRANCH IF NO
8614	046002	105037	046022		CLRB	\$CHARCNT	;;YES--CLEAR CHARACTER COUNT
8615	046006	000406			BR	\$TYPEC	;;EXIT
8616	046010	122766	000012	000002	1\$:	CMPB	#LF,2(SP)
8617	046016	001402			BEQ	\$TYPEC	;;IS CHARACTER A LINE FEED?
							;;BRANCH IF YES

TYPE ROUTINE

8618	046020	105227								
8619	046022	000000								
8620	046024	000207								
8621										
8622										
8623										
8624										
8625										

INCB (PC)+ ; ;COUNT THE CHARACTER
 SCHARCNT: .WORD 0 ; ;CHARACTER COUNT STORAGE
 \$TYPEX: RTS PC

.SBTTL APT COMMUNICATIONS ROUTINE

8626	046026	112737	000001	046272						
8627	046034	112737	000001	046270						
8628	046042	000403								
8629	046044	112737	000001	046272						
8630	046052									
8631	046052	010046								
8632	046054	010146								
8633	046056	105737		046270						
8634	046062	001450								
8635	046064	122737	000001	001254						
8636	046072	001031								
8637	046074	132737	000100	001255						
8638	046102	001425								
8639	046104	017600	000004							
8640	046110	062766	000002	000004						
8641	046116	005737	001234							
8642	046122	001375			1\$:					
8643	046124	010037	001250							
8644	046130	105720			2\$:					
8645	046132	001376								
8646	046134	163700	001250							
8647	046140	006200								
8648	046147	010037	001752							
8649	046146	012737	000004	001234						
8650	046154	000413								
8651	046156	017637	000004	046202	3\$:					
8652	046164	062766	000002	000004						
8653	046172	013746	177776							
8654	046176	004737	045544							
8655	046207	000000			4\$:					
8656	046204				5\$:					
8657	046204	105737	046272		10\$:					
8658	046210	001416								
8659	046212	005737	001254							
8660	046216	001413								
8661	046220	005737	001234		11\$:					
8662	046224	001375								
8663	046226	017637	000004	001236						
8664	046234	062766	000002	000004						
8665	046242	005237	001234							
8666	046246	105037	046272		12\$:					
8667	046252	105037	046271							
8668	046256	105037	046270							
8669	046262	012601								
8670	046264	012600								
8671	046266	000707								
8672	046270	000								
8673	046271	000								

\$MFLG: .BYTE 0 ; ;MESSG. FLAG
 \$LFLG: .BYTE 0 ; ;LOG FLAG

8674 046272 000
 8675 046274
 8676 000200
 8677 000001
 8678 000100
 8679 000040
 8680
 8681
 8682
 8683
 8684
 8685
 8686
 8687
 8688
 8689
 8690
 8691
 8692
 8693 046274 113737 001102 001766
 8694 046302 104400 001231
 8695 046306 010046
 8696 046310 005000
 8697 046312 153700 001114
 8698 046316 001004
 8699
 8700 046320 013746 001116
 8701
 8702 046324 104401
 8703 046326 000562
 8704 046330 005300
 8705 046332 006300
 8706 046334 006300
 8707 046336 006300
 8708 046340 062700 001570
 8709 046344 012037 046354
 8710 046350 001404
 8711 046352 104400
 8712 046354 000000
 8713 046356 104400 001231
 8714 046362 012037 046440
 8715 046366 001427
 8716 046370 010146
 8717 046372 016001 000002
 8718 046376 001416
 8719 046400 105711
 8720 046402 001414
 8721 046404 104400 050142
 8722 046410 013746 001116
 8723 046414 104401
 8724 046416 104400 046706
 8725 046422 013746 001266
 8726 046426 104401
 8727 046430 104400 001231
 8728 046434 012601
 8729 046436 104400

```

SFPLG: .BYTE 0          ;;FATAL FLAG
        .EVEN
APTSIZE=200
APTENV=001
APTSPOOL=100
APTCSUP=040
;;*****

.SBTTL  ERROR MESSAGE TYPEOUT ROUTINE

;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" (SITEMB) TO DETERMINE WHICH
;*ERROR IS TO BE REPOPTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" (SERRTB),
;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.

;*THE DATA FORMATS FOR THIS PROGRAM ARE:
;* 0      16-BIT BINARY TO 6 DIGIT OCTAL
;* 1      32-BIT FLOATING TO 13-DIGIT OCTAL
;* 2      64-BIT FLOATING TO 25-DIGIT OCTAL
;* 3      BITS <15:7> TO 3-DIGIT OCTAL

SERRTVP:MOVB $SSTNM,$SSTNM
        TYPE ,SCLRF          ;;"CARRIAGE RETURN" & "LINE FEED"
        MOV RO,-(SP)        ;;SAVE #0
        CLR RO              ;;PICKUP THE ITEM INDEX
        BISP @SITEMB,RO
        BNE IS              ;;IF ITEM NUMBER IS ZERO, JUST
                            ;;TYPE THE PC OF THE ERROR
                            ;;SAVE SERRPC FOR TYPEOUT
                            ;;ERROR ADDRESS
                            ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
                            ;;GET OUT
                            ;;ADJUST THE INDEX SO THAT IT WILL
                            ;; WORK FOR THE ERROR TABLE

        MOV SERRPC,-(SP)

        TYPOC
        BR 10$
1$:     DEC RO
        ASL RO
        ASL RO
        ASL RO
        ADD #SERRTB,RO      ;;FORM TABLE POINTER
        MOV (RO)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
        BEQ 3$              ;;SKIP TYPEOUT IF NO POINTER
        TYPE                ;;TYPE THE "ERROR MESSAGE"
        .WOPD 0             ;;"ERROR MESSAGE" POINTER GOES HERE
        TYPE ,SCLRF        ;;"CARRIAGE RETURN" & "LINE FEED"
3$:     MOV (RO)+,4$        ;;PICKUP "DATA HEADER" POINTER
        BEQ 5$              ;;SKIP TYPEOUT IF 0
        MOV R1,-(SP)        ;;SAVE R1
        MOV 2(RO),R1        ;;GET DATA FORMAT POINTER
        BEQ 127$           ;;BRANCH IF ZERO
        TSTB (R1)          ;;DATA FORMAT ZERO?
        BEQ 127$           ;;BRANCH IF YES
        TYPE ,MSG1
        MOV SERRPC,-(SP)    ;;GET ERROR PC
        TYPOC              ;;TYPE IT
        TYPE ,11$          ;;TYPE 2 SPACES
        MOV $SSTNM,-(SP)    ;;GET TEST NUMBER
        TYPOC              ;;TYPE IT
127$:   MOV (SP)+,R1        ;;RESTORE R1
        TYPE                ;;TYPE THE "DATA HEADER"
  
```

```

8730 046440 000000      4S:  .WORD  0          ;; "DATA HEADER" POINTER GOES HERE
8731 046442 104400 001231  TYPE  , $CRLF      ;; "CARRIAGE RETURN" & "LINE FEED"
8732 046446 010146      5S:  MOV    R1, -(SP)   ;; SAVE R1
8733 046450 012001      MOV    (R0)+, R1    ;; PICKUP "DATA TABLE" POINTER
8734 046452 001507      BEQ    9S           ;; BR IF NO DATA TO BE TYPED
8735 046454 012000      MOV    (R0)+, R0    ;; PICKUP "DATA FORMAT" POINTER
8736 046456 005700      6S:  TST    R0       ; IS FORMAT POINTER 0?
8737 046460 001411      BEQ    12S        ; BRANCH IF YES
8738 046462 122710 000001  CMPB  #1, (R0)    ; IS IT 0, 1, OR >1?
8739 046466 003005      BGT    20S        ; BRANCH IF ZERO
8740 046470 001410      BEQ    30S        ; BRANCH IF 1
8741 046472 122710 000003  CMPB  #2, (R0)    ; IS IT 2, 3, OR >3?
8742 046476 003017      BGT    40S        ; BRANCH IF 2
8743 046500 001430      BEQ    50S
8744
8745      ; DATA FORMAT 0-6 DIGIT OCTAL
8746 046502 005200      20S:  INC    R0       ; ADJUST FORMAT POINTER
8747 046504      12S:
8748 046504 013146      MOV    @(R1)+, -(SP) ; ; SAVE @(R1)+ FOR TYPEOUT
8749 046506 104401      TYPOC ; ; GO TYPE--OCTAL ASCII(ALL DIGITS)
8750 046510 000463      BR    8S
8751
8752      ; DATA FORMAT 1-TYPE A FLOATING NUMBER
8753 046512 005200      30S:  INC    R0       ; ADJUST FORMAT POINTER
8754 046514 012137 046522  MOV    (R1)+, 31S   ; GET ADDRESS OF NUMBER
8755 046520 104407      FL20   ; CONVERT IT TO OCTAL
8756 046522 000000      31S:  .WORD
8757 046524 012637 046532  MOV    (SP)+, 32S   ; GET ADDRESS OF ASCII
8758 046530 104400      TYPE   ; TYPE THE NUMBER
8759 046532 000000      32S:  .WORD
8760 046534 000451      BR    8S
8761
8762      ; DATA FORMAT 2-TYPE A FLOATING DOUBLE NUMBER
8763 046536 005200      40S:  INC    R0       ; ADJUST FORMAT POINTER
8764 046540 012137 046546  MOV    (R1)+, 41S   ; GET ADDRESS OF NUMBER
8765 046544 104410      FLD20 ; CONVERT IT TO ASCII
8766 046546 000000      41S:  .WORD
8767 046550 012637 046556  MOV    (SP)+, 42S   ; GET ADDRESS OF ASCII
8768 046554 104400      TYPE   ; TYPE THE NUMBER
8769 046556 000000      42S:  .WORD
8770 046560 000437      BR    8S
8771
8772      ; DATA FORMAT 3-TYPE AN BITS<59:51>
8773 046562 005200      50S:  INC    R0       ; ADJUST FORMAT POINTER
8774 046564 012146      MOV    (R1)+, -(SP) ; PUT ADDR OF NUMBER ON STACK
8775 046566 011646      MOV    (SP), -(SP) ; AND AGAIN
8776 046570 062716 000007  ADD    #2, (SP)    ; MAKE ADDR OF NUMBER+2
8777 046574 006176 000000  ROL    @(SP)       ; PUT ALL THE
8778 046600 006176 000002  ROL    @2(SP)      ; BITS AP<59:51>
8779 046604 000241      CLC
8780 046606 006176 000000  ROL    @(SP)       ; INTO THE FIRST
8781 046612 006176 000002  ROL    @2(SP)      ; WORD
8782 046616 005726      TST    (SP)+       ; POP THE SECOND ADDRESS
8783 046620 042776 177000 000000  BIC    #177000, @(SP) ; GET RID OF EXPONENT BITS
8784 046626 004737 047364  JSR    PC, $DB20   ; CONVERT TO ASCII
8785 046632 062716 000010  ADD    #10, (SP)   ; ONLY WANT 3 DIGITS
  
```

```

8786 046636 104400 046704      TYPE      ,13$      ;TYPE A SPACE
8787 046642 012637 046650      MOV        (SP)+,51$ ;GET ADDRESS OF DATA
8788 046646 104400              TYPE              ;TYPE THE DATA
8789 046650 000000              51$: .WORD
8790 046652 104400 046706      TYPE      ,11$      ;TYPE TWO SPACES
8791 046656 000400              BR          8$
8792 046660 005711              8$:  TST      (R1)      ;IS THERE ANOTHER NUMBER?
8793 046662 001403              BEQ        9$         ;BR IF NO
8794 046664 104400 046706      TYPE      ,11$      ;TYPE TWO(?) SPACES
8795 046670 000672              BR          6$         ;LOCP
8796
8797 046672 012601              9$:  MOV      (SP)+,R1   ;RESTORE R1
8798 046674 012600              10$: MOV     (SP)+,R0   ;RESTORE R0
8799 046676 104400 001231      TYPE      ,SCRLF     ;"CARRIAGE RETURN" & "LINE FEED"
8800 046702 000207              RTS        PC         ;RETURN
8801 046704 000040              13$: .ASCIZ  / /
8802 046706 020040 000          11$: .ASCIZ  / /      ;TWO(2) SPACES
8803 046712
8804
8805
8806
8807
8808
8809
8810
8811
8812
8813
8814
8815
8816
8817
8818
8819
8820
8821
8822
8823
8824
8825
8826
8827
8828
8829
8830 046712 017646 000000      .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
8831 046716 116637 000001 047135 ;*****
8832 046724 112637 047137 ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
8833 046730 062716 000002 ;*OCTAL (ASCII) NUMBER AND TYPE IT.
8834 046734 000406 ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
8835 046736 112737 000001 047135 ;*CALL:
8836 046744 112737 000006 047137 ;*      MOV      NUM,-(SP)      ;NUMBER TO BE TYPED
8837 046752 112737 000005 047134 ;*      TYPOS      ;CALL FOR TYPEOUT
8838 046760 010346 ;*      .BYTE    M              ;M=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
8839 046762 010446 ;*      .BYTE    M              ;M=1 OR 0
8840 046764 010546 ;*                               ;1=TYPE LEADING ZEROS
8841 046766 113704 047137 ;*                               ;0=SUPPRESS LEADING ZEROS
;*$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
;*$TYPOS OR $TYPOC
;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
;*$CALL:
;*      MOV      NUM,-(SP)      ;NUMBER TO BE TYPED
;*      TYPON      ;CALL FOR TYPEOUT
;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
;*$CALL:
;*      MOV      NUM,-(SP)      ;NUMBER TO BE TYPED
;*      TYPOC      ;CALL FOR TYPEOUT
;*$TYPOS: MOV      0(SP),-(SP)      ;PICKUP THE MODE
;*$TYPOC: MOV      1(SP),SOFILL    ;LOAD ZERO FILL SWITCH
;*$TYPOS: MOV      (SP)+,SOMODE+1 ;NUMBER OF DIGITS TO TYPE
;*$TYPOC: ADD      #7,(SP)        ;ADJUST RETURN ADDRESS
;*$TYPOS: BR          STYPON
;*$TYPOC: MOV      #1,SOFILL      ;SET THE ZERO FILL SWITCH
;*$TYPOS: MOV      #6,SOMODE+1    ;SET FOR SIX(6) DIGITS
;*$TYPOC: MOV      #5,SOCNT      ;SET THE ITERATION COUNT
;*$TYPOS: MOV      R3,-(SP)      ;SAVE R3
;*$TYPOC: MOV      R4,-(SP)      ;SAVE R4
;*$TYPOS: MOV      R5,-(SP)      ;SAVE R5
;*$TYPOC: MOV      SOMODE+1,R4    ;GET THE NUMBER OF DIGITS TO TYPE

```

```

8842 046772 005404          NEG      R4
8843 046774 062704 000006  ADD     #6,R4          ;;SUBTRACT IT FOR MAX. ALLOWED
8844 047000 110437 047136  MOVB   R4,$OMODE     ;;SAVE IT FOR USE
8845 047004 113704 047135  MOVB   $OFILL,R4    ;;GET THE ZERO FILL SWITCH
8846 047010 016605 000012  MOV    12(SP),R5    ;;PICKUP THE INPUT NUMBER
8847 047014 005003          CLR     R3          ;;CLEAR THE OUTPUT WORD
8848 047016 006105          1$:    ROL     R5          ;;ROTATE MSB INTO "C"
8849 047020 000404          BR     3$          ;;GO DO MSB
8850 047022 006105          2$:    ROL     R5          ;;FORM THIS DIGIT
8851 047024 006105          ROL     R5
8852 047026 006105          ROL     R5
8853 047030 010503          MOV    R5,R3
8854 047032 006103          3$:    ROL     R3          ;;GET LSB OF THIS DIGIT
8855 047034 105337 047136  DECB  $OMODE     ;;TYPE THIS DIGIT?
8856 047040 100016          BPL    7$          ;;BR IF NO
8857 047042 042703 177770  BIC   #177770,R3   ;;GET RID OF JUNK
8858 047046 001002          BNE    4$          ;;TEST FOR 0
8859 047050 005704          TST   R4          ;;SUPPRESS THIS 0?
8860 047052 001403          BEQ   5$          ;;BR IF YES
8861 047054 005204          4$:    INC    R4          ;;DON'T SUPPRESS ANYMORE 0'S
8862 047056 052703 000060  BIS   #'0,R3      ;;MAKE THIS DIGIT ASCII
8863 047062 052703 000040  5$:    BIS   #' ,R3    ;;MAKE ASCII IF NOT ALREADY
8864 047066 110337 047132  MOVB  R3,$S          ;;SAVE FOR TYPING
8865 047072 104400 047132  TYPE  ,8$          ;;GO TYPE THIS DIGIT
8866 047076 105337 047134  7$:    DECB  $OCNT   ;;COUNT BY 1
8867 047102 003347          BGT   2$          ;;BR IF MORE TO DO
8868 047104 002402          BLT   6$          ;;BR IF DONE
8869 047106 005204          INC   R4          ;;INSURE LAST DIGIT ISN'T A BLANK
8870 047110 000744          BR    2$          ;;GO DO THE LAST DIGIT
8871 047112 012605          6$:    MOV   (SP)+,R5   ;;RESTORE R5
8872 047114 012604          MOV   (SP)+,R4   ;;RESTORE R4
8873 047116 012603          MOV   (SP)+,R3   ;;RESTORE R3
8874 047120 016666 000002 000004  MOV   2(SP),4(SP) ;;SET THE STACK FOR RETURNING
8875 047126 012616          MOV   (SP)+,(SP)
8876 047130 000002          RTI                    ;;RETURN
8877 047132 000          8$:    .PYTE  0          ;;STORAGE FOR ASCII DIGIT
8878 047133 000          .BYTE  0          ;;TERMINATOR FOR TYPE ROUTINE
8879 047134 000          $OCNT: .BYTE  0    ;;OCTAL DIGIT COUNTER
8880 047135 000          $OFILL: .BYTE  0   ;;ZERO FILL SWITCH
8881 047136 000000          $OMODE: .WORD  0   ;;NUMBER OF DIGITS TO TYPE
8882
8883          .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
8884
8885          ;;*****
8886          ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
8887          ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
8888          ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
8889          ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
8890          ;*REPLACED WITH SPACES.
8891          ;*CALL:
8892          ;*      MOV    NUM,-(SP)          ;;PUT THE BINARY NUMBER ON THE STACK
8893          ;*      TYPDS          ;;GO TO THE ROUTINE
8894
8895          $TYPDS:
8896          MOV    R0,-(SP)          ;;PUSH R0 ON STACK
8897          MOV    R1,-(SP)          ;;PUSH R1 ON STACK

```

8898	047144	010246			MOV	R2,-(SP)	;;PUSH R2 ON STACK
8899	047146	010346			MOV	R3,-(SP)	;;PUSH R3 ON STACK
8900	047150	010546			MOV	R5,-(SP)	;;PUSH R5 ON STACK
8901	047152	012746	020200		MOV	#20200,-(SP)	;;SET BLANK SWITCH AND SIGN
8902	047156	016605	000020		MOV	20(SP),R5	;;GET THE INPUT NUMBER
8903	047162	100004			BPL	1\$;;BR IF INPUT IS POS.
8904	047164	005405			NEG	R5	;;MAKE THE BINARY NUMBER POS.
8905	047166	112766	000055	000001	MOVB	#'-,1(SP)	;;MAKE THE ASCII NUMBER NEG.
8906	047174	005000			1\$: CLR	R0	;;ZERO THE CONSTANTS INDEX
8907	047176	012703	047354		MOV	#\$DBLK,R3	;;SETUP THE OUTPUT POINTER
8908	047202	112723	000040		MOVB	#' ,(R3)+	;;SET THE FIRST CHARACTER TO A BLANK
8909	047206	005002			2\$: CLR	R2	;;CLEAR THE BCD NUMBER
8910	047210	016001	047344		MOV	\$DTBL(R0),R1	;;GET THE CONSTANT
8911	047214	160105			3\$: SUB	R1,R5	;;FORM THIS BCD DIGIT
8912	047216	002402			BLT	4\$;;BR IF DONE
8913	047220	005202			INC	R2	;;INCREASE THE BCD DIGIT BY 1
8914	047222	000774			BR	3\$	
8915	047224	060105			4\$: ADD	R1,R5	;;ADD BACK THE CONSTANT
8916	047226	005702			TST	R2	;;CHECK IF BCD DIGIT=0
8917	047230	001002			BNE	5\$;;FALL THROUGH IF 0
8918	047232	105716			TSTR	(SP)	;;STILL DOING LEADING 0'S?
8919	047234	100407			BMI	7\$;;BR IF YES
8920	047236	106316			5\$: ASLB	(SP)	;;MSD?
8921	047240	103003			BCC	6\$;;BR IF NO
8922	047242	116663	000001	177777	MOVB	1(SP),-1(R3)	;;YES--SET THE SIGN
8923	047250	052702	000060		6\$: BIS	#'0,R2	;;MAKE THE BCD DIGIT ASCII
8924	047254	052707	000040		7\$: BIS	#' ,R2	;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
8925	047260	110223			MOVB	R2,(R3)+	;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
8926	047262	005720			TST	(R0)+	;;JUST INCREMENTING
8927	047264	020027	000010		CMP	R0,#10	;;CHECK THE TABLE INDEX
8928	047270	002746			BLT	2\$;;GO DO THE NEXT DIGIT
8929	047272	003002			BGT	8\$;;GO TO EXIT
8930	047274	010502			MOV	R5,R2	;;GET THE LSD
8931	047276	000764			BR	6\$;;GO CHANGE TO ASCII
8932	047300	105726			8\$: TSTR	(SP)+	;;WAS THE LSD THE FIRST NON-ZERO?
8933	047302	100003			BPL	9\$;;BR IF NO
8934	047304	116663	177777	177776	MOVB	-1(SP),-2(R3)	;;YES--SET THE SIGN FOR TYPING
8935	047312	105013			9\$: CLRB	(R3)	;;SET THE TERMINATOR
8936	047314	012605			MOV	(SP)+,R5	;;POP STACK INTO R5
8937	047316	012603			MOV	(SP)+,R3	;;POP STACK INTO R3
8938	047320	012602			MOV	(SP)+,R2	;;POP STACK INTO R2
8939	047322	012601			MOV	(SP)+,R1	;;POP STACK INTO R1
8940	047324	012600			MOV	(SP)+,R0	;;POP STACK INTO R0
8941	047326	104400	047354		TYPE	, \$DBLK	;;NOW TYPE THE NUMBER
8942	047332	016666	000002	000004	MOV	2(SP),4(SP)	;;ADJUST THE STACK
8943	047340	012616			MOV	(SP)+,(SP)	
8944	047342	000002			RTI		;;RETURN TO USER
8945	047344	023420			\$DTBL:	10000.	
8946	047346	001750				1000.	
8947	047350	000144				100.	
8948	047352	000012				10.	
8949	047354	000004			\$DBLK:	.BLKW 4	
8950							
8951					.SBTTL	DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE	
8952							
8953							

```

8954 ;*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
8955 ;*UNSIGNED OCTAL ASCII NUMBER.
8956 ;*CALL
8957 ;*      NOV      #PNTR,-(SP)      ;;POINTER TO LOW WORD OF BINARY NUMBER
8958 ;*      JSR      PC,#$DB20      ;;CALL THE ROUTINE
8959 ;*      RETURN                      ;;THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK
8960
8961
8962 047364 104405      $DB20: SAVREC                      ;;SAVE ALL REGISTERS
8963 047366 016601 000002      NOV      2(SP),R1                      ;;PICKUP THE POINTER TO LOW WORD
8964 047372 012705 047503      NOV      #SOCTVL+13.,R5          ;;POINTER TO DATA TABLE
8965 047376 012704 000014      NOV      #12.,R4                      ;;DO ELEVEN CHARACTERS
8966 047407 012703 177770      NOV      #C7,R3                      ;;MASK
8967 047406 012100      NOV      (R1)+,R0                      ;;LOWER WORD
8968 047410 012101      NOV      (R1)+,R1                      ;;HIGH WORD
8969 047412 005002      CLR      R7                          ;;TERMINATOR
8970 047414 110245      15:  NOVB     R2,-(R5)                      ;;PUT CHARACTER IN DATA TABLE
8971 047416 010002      NOV      R0,R2                          ;;GET THIS DIGIT
8972 047420 005304      DEC      #4                          ;;COUNT THIS CHARACTER
8973 047422 003007      BGT      3$                          ;;BR IF NOT THE LAST DIGIT
8974 047424 001405      BEQ      2$                          ;;BR IF IT IS THE LAST DIGIT
8975 047426 005205      INC      R5                          ;;ALL DIGITS DONE-ADJUST POINTER FOR FIRST
8976 047430 010566 000002      NOV      R5,2(SP)                      ;;ASCII CHAR. & PUT IT ON THE STACK
8977 047434 104406      RESPEC                      ;;RESTORE ALL REGISTERS
8978 047436 000207      RTS      PC                          ;;RETURN TO USER
8979 047440 006203      2$:  ASR      #3                          ;;POSITION THE MASK FOR THE LAST DIGIT
8980 047442 006001      3$:  ROR      R1                          ;;POSITION THE BINARY NUMBER FOR
8981 047444 006000      ROR      R0                          ;;
8982 047446 006001      ROR      R1                          ;;
8983 047450 006000      ROR      R0                          ;;
8984 047452 006001      ROR      R1                          ;;
8985 047454 006000      ROR      R0                          ;;
8986 047456 040302      BIC      #3,R2                      ;;MASK OUT ALL JUNK
8987 047460 062702 000060      ADD      #0,R2                      ;;MAKE THIS CHAR. ASCII
8988 047464 000753      BR       1$                          ;;GO PUT IT IN THE DATA TABLE
8989 047466 000016      SOCTVL: .BLKB 14.                      ;;RESERVE DATA TABLE
8990 ;*****
8991 ;SBTTL UNEXPECTED TRAP TO 4 ROUTINE
8992
8993 047504 011637 001200      CPSPUR: NOV      (SP),STMP0          ;;SAVE PC OF TRAP
8994 047510 012706 001100      NOV      #STACK,SP                ;;RESTORE THE SP
8995 047514 005737 001406      TST      $CPUERR                    ;;11/70?
8996 047520 001410      BEQ      1$                          ;;BRANCH IF NO
8997 047522 013737 177766 001202      NOV      CPUERR,STMP1              ;;SAVE ERROR REG
8998 047530 005037 177766      CLR      CPUERR                      ;;CLEAN THE ERROR
8999 047534 104275      ERROR   275                          ;;UNEXPECTED TRAP TO 4
9000 047536 000137 005254      JMP      LOOP                          ;;PESTART
9001 047542 104276      1$:  ERROR   276                          ;;UNEXPECTED TRAP TO 4
9002 047544 000137 005254      JMP      LOOP                          ;;RESTART
9003
9004 ;*****
9005 ;SBTTL UNEXPECTED TRAP TO 114 ROUTINE
9006
9007 047550 011637 001200      CACHSPU: NOV      (SP),STMP0        ;;SAVE PC OF TPAP
9008 047554 012706 001100      NOV      #STACK,SP                ;;INIT THE SP
9009 047560 005737 001406      TST      $CPUERR                    ;;11/70?
  
```


9010 047564 001414
9011 047566 013737 177744 001202
9012 047574 013737 177740 001204
9013 047602 013737 177742 001206
9014 047610 104277
9015 047612 000137 005254
9016 047616 104300
9017 047620 000137 005254

BEQ 15 ;BRANCH IF NO
MOV MEMERR,\$TMP1 ;SAVE ERROR REGISTER
MOV LOADRS,\$TMP2 ;SAVE ERROR
MOV HIADRS,\$TMP3 ;ADDRESS REGISTER
ERROR 277 ;UNEXPECTED TRAP TO 114
JMP LOOP ;RESTART
15: ERROR 300 ;UNEXPECTED TRAP TO 114
JMP LOOP ;RESTART

;;*****

9018
9019
9020
9021
9022 047624 011637 001200
9023 047630 012706 001100
9024 047634 170337 001202
9025 047640 104301
9026 047642 000137 005254

.SBTTL UNEXPECTED TRAP TO 244 ROUTINE
FPSPUR: MOV (SP),\$TMP0 ;SAVE ADDRESS OF TRAP
MOV \$STACK,SP ;RESTORE THE SP
STST \$TMP1 ;GET PEC AND PEA
ERROR 301 ;UNEXPECTED TRAP TO 244
JMP LOOP ;RESTART

9027
9028
9029
9030
9031
9032
9033
9034
9035

.SBTTL TRAP DECODER
;;*****
;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
;*GO TO THAT ROUTINE.

9036 047646 010046
9037 047650 016600 000002
9038 047654 005740
9039 047656 111000
9040 047660 006300
9041 047662 016000 047670
9042 047666 000200

STRAP: MOV R0,-(SP) ;;SAVE R0
MOV 2(SP),R0 ;;GET TRAP ADDRESS
TST -(R0) ;;BACKUP BY 2
NOVB (R0),R0 ;;GET RIGHT BYTE OF TRAP
ASL R0 ;;POSITION FOR INDEXING
MOV \$TRPAD(R0),R0 ;;INDEX TO TABLE
RTS R0 ;;GO TO ROUTINE

9043
9044
9045
9046
9047
9048
9049

.SBTTL TRAP TABLE
;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
;*BY THE "TRAP" INSTRUCTION.

9050
9051
9052 047670
9053 047670 045544
9054 047672 046736
9055 047674 046712
9056 047676 046752
9057 047700 047140
9058 047702 045450
9059 047704 045506
9060 047706 044616
9061 047710 045104

; ROUTINE
; -----
\$TRPAD: \$TYPE ;;CALL=TYPE TRAP+0(104400) TTY TYPEOUT ROUTINE
\$TYPOC ;;CALL=TYPOC TRAP+1(104401) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
\$TYPOS ;;CALL=TYPOS TRAP+2(104402) TYPE OCTAL NUMBER (NO LEADING ZEROS)
\$TYPON ;;CALL=TYPON TRAP+3(104403) TYPE OCTAL NUMBER (AS PER LAST CALL)
\$TYPDS ;;CALL=TYPDS TRAP+4(104404) TYPE DECIMAL NUMBER (WITH SIGN)
\$SAVREG ;;CALL=SAVREG TRAP+5(104405) SAVE R0-R5 ROUTINE
\$RESREG ;;CALL=RESREG TRAP+6(104406) RESTORE R0-R5 ROUTINE
\$FL20 ;;CALL=FL20 TRAP+7(104407)
\$FLD20 ;;CALL=FLD20 TRAP+10(104410)

9062
9063
9064
9065

.SBTTL POWER DOWN AND UP ROUTINES
;;*****

```

9066                                     ;POWER DOWN ROUTINE
9067 047712 012737 050124 000024 SPWRDN: MOV    #SILLUP,@SPWRVEC ;;SET FOR FAST UP
9068 047720 012737 000340 000026          MOV    #340,@SPWRVEC+2 ;;PRIO:7
9069 047726 010046          MOV    R0,-(SP) ;;PUSH R0 ON STACK
9070 047730 010146          MOV    R1,-(SP) ;;PUSH R1 ON STACK
9071 047732 010246          MOV    R2,-(SP) ;;PUSH R2 ON STACK
9072 047734 010346          MOV    R3,-(SP) ;;PUSH R3 ON STACK
9073 047736 010446          MOV    R4,-(SP) ;;PUSH R4 ON STACK
9074 047740 010546          MOV    R5,-(SP) ;;PUSH R5 ON STACK
9075 047742 170200          STFPS  R0 ;;GET FPS
9076 047744 170127 000200          LDFPS  #FD
9077 047750 010046          MOV    R0,-(SP) ;;PUSH ON STACK
9078 047752 174046          STD    AC0,-(SP)
9079 047754 174146          STD    AC1,-(SP)
9080 047756 174246          STD    AC2,-(SP)
9081 047760 174346          STD    AC3,-(SP)
9082 047762 172404          LDD    AC4,AC0
9083 047764 174046          STD    AC0,-(SP)
9084 047766 172405          LDD    AC5,AC0
9085 047770 174046          STD    AC0,-(SP)
9086 047772 010637 050130          MOV    SP,$SAVR6 ;;SAVE SP
9087 047776 012737 050010 000024          MOV    #SPWRUP,@SPWRVEC ;;SET UP VECTOR
9088 050004 000000          HALT
9089 050006 000775          BR    -2 ;;HANG UP
9090
9091                                     ;;*****
9092                                     ;POWER UP ROUTINE
9093 050010 012737 050124 000024 SPWRUP: MOV    #SILLUP,@SPWRVEC ;;SET FOR FAST DOWN
9094 050016 013706 050130          MOV    $SAVR6,SP ;;GET SP
9095 050022 005037 050130          CLR    $SAVR6 ;;WAIT LOOP FOR THE TTY
9096 050026 005237 050130          1$: INC    $SAVR6 ;;WAIT FOR THE INC
9097 050032 001375          BWE    1$ ;;OF WORD
9098 050034 170127 000200          LDFPS  #FD
9099 050040 172426          LDD    (SP)+,AC0
9100 050042 174005          STD    AC0,AC5
9101 050044 172426          LDD    (SP)+,AC0
9102 050046 174004          STD    AC0,AC4
9103 050050 172726          LDD    (SP)+,AC3
9104 050052 172626          LDD    (SP)+,AC2
9105 050054 172526          LDD    (SP)+,AC1
9106 050056 172426          LDD    (SP)+,AC0
9107 050060 170126          LDFPS  (SP)+
9108 050062 012605          MOV    (SP)+,R5 ;;POP STACK INTO R5
9109 050064 012604          MOV    (SP)+,R4 ;;POP STACK INTO R4
9110 050065 012603          MOV    (SP)+,R3 ;;POP STACK INTO R3
9111 050070 012602          MOV    (SP)+,R2 ;;POP STACK INTO R2
9112 050072 012601          MOV    (SP)+,R1 ;;POP STACK INTO R1
9113 050074 012600          MOV    (SP)+,R0 ;;POP STACK INTO R0
9114 050076 012737 047712 000024          MOV    #SPWRDN,@SPWRVEC ;;SET UP THE POWER DOWN VECTOR
9115 050104 012737 000340 000026          MOV    #340,@SPWRVEC+2 ;;PRIO:7
9116 050112 104400          TYPE   ;;REPORT THE POWER FAILURE
9117 050114 050132          SPWRNG: .WORD $POWER ;;POWER FAIL MESSAGE POINTER
9118 050116 013716          MOV    @(PC)+,(SP) ;;RESTART AT $ESCAPE
9119 050120 001222          SPWRAD: .WORD $ESCAPE ;;RESTART ADDRESS
9120 050122 000002          RTI
9121 050124 000000          $ILLUP: HALT ;;THE POWER UP SEQUENCE WAS STARTED

```

```

9122 050126 000776          BR      -2      ;) BEFORE THE POWER DOWN WAS COMPLETE
9123 050130 000000          $SAVR6: 0      ;)PUT THE SP HERE
9124 050132 005015 047520 042527 $POWER: .ASCIZ <15><12>"POWER"
9125 050140 000122
9126
9127          .EVEN
9128 050142 051105 050122 004503 MSG1: .ASCIZ /ERRPC TEST NO/<CRLF>
9129 050150 042524 052123 047040
9130 050156 100117      000
9131 050161      111 041522 020103 EM1: .ASCIZ /TRCC FCLASS DOES NOT GET TO E40(A0) OR E40(A0) BAD/
9132 050166 041506 040514 051523
9133 050174 042040 042517 020123
9134 050202 047516 020124 042507
9135 050210 020124 047524 042440
9136 050216 030064 040450 024460
9137 050224 047440 020122 032105
9138 050232 024060 030101 020051
9139 050240 040502 000104
9140 050244 051105 050122 020103 DM1: .ASCIZ /ERRPC TEST NO/
9141 050252 020040 042524 052123
9142 050260 047040 000117
9143
9144 050264 001116 001266 000000 DT1: .EVEN
9145 050272 051111 041503 043040 EM2: .WORD $ERRPC,$STSTNM,0
9146 050300 046103 051501 020123 .ASCIZ /TRCC FCLASS DOES NOT GET TO E21 OR E21 BAD/
9147 050306 047504 051505 047040
9148 050314 052117 043440 052105
9149 050322 052040 020117 031105
9150 050330 020061 051117 042440
9151 050336 030462 041040 042101
9152 050344      000
9153 050345      122 041501 020113 EM3: .ASCII /RACK BE75 DOES NOT GET "FP REG HT" AS A LOW/<CRLF>
9154 050352 042502 032467 042040
9155 050360 042517 020123 047516
9156 050366 020124 042507 020124
9157 050374 043042 020120 042522
9158 050402 020107 052127 020047
9159 050410 051501 040440 046040
9160 050416 053517      200
9161 050421      122 041501 020110 EM4: .ASCIZ /RACK A2 PAR00 DID NOT GO LOW/
9162 050426 031101 051040 041101
9163 050434 030060 042040 042111
9164 050442 047040 052117 043440
9165 050450 020117 047514 000127
9166 050456 051117 046440 030470 .ASCIZ /OR MR132-YA NOT INSTALLED/
9167 050464 031063 054455 020101
9168 050472 047516 020124 047111
9169 050500 052123 046101 042514
9170 050506 000104
9171 050510 040522 045503 042440 EM5: .ASCIZ /RACK E64(A0) DID NOT GO HIGH/
9172 050516 032066 040450 024460
9173 050524 042040 042111 047040
9174 050532 052117 043440 020117
9175 050540 044510 044107      000
9176 050545      111 041522 020103 EM6: .ASCIZ /TRCC FCLASS DID NOT GO LOW/
9177 050552 041506 040514 051523
  
```

9178	050560	042040	042111	047040		
9179	050566	052117	043440	020117		
9180	050574	047514	000127			
9181	050600	040522	045503	051040	EM7:	.ASCII /RACK RIP+PP SYNC DID NOT CAUSE E50(A0) TO GO H GR/<CRLF>
9182	050606	050111	043053	020120		
9183	050614	054523	041516	042040		
9184	050622	042111	047040	052117		
9185	050630	041440	052501	042523		
9186	050636	042440	030065	040450		
9187	050644	024460	052040	020117		
9188	050652	047507	044040	047440		
9189	050660	100122				
9190	050662	051111	041503	042040		.ASCIZ /IRCC DSTMO DID NOT GO LOW/
9191	050670	052123	030115	042040		
9192	050676	042111	047040	052117		
9193	050704	043440	020117	047514		
9194	050712	000127				
9195	050714	044505	044124	051105	EM10:	.ASCII /EITHER LDPPS OR STPPS DID NOT TRANSFER THE DATA/<CRLF>
9196	050722	046040	043104	051520		
9197	050730	047440	020122	052123		
9198	050736	050106	020123	044504		
9199	050744	020104	047516	020124		
9200	050752	051124	047101	043123		
9201	050760	051105	052040	042510		
9202	050766	042040	052101	100101		
9203	050774	051117	041040	047105		.ASCIZ /OR BEN 7 FAILED/
9204	051002	033440	043040	044501		
9205	051010	042514	000104			
9206	051014	044502	020124	052123	EM11:	.ASCIZ /PIT STUCK IN FPS DATA PATH/
9207	051022	041525	020113	047111		
9208	051030	043040	051520	042040		
9209	051036	052101	020101	040520		
9210	051044	044124	000			
9211	051047	105	051122	041520	DM11:	.ASCII /ERPPC DATA TEST NO/<CRLF>
9212	051054	020040	020040	020040		
9213	051062	020040	040504	040524		
9214	051070	020040	020040	020040		
9215	051076	020040	042524	052123		
9216	051104	047040	100117			
9217	051110	042411	050130	041505		.ASCIZ / EXPECT ACTUAL/
9218	051116	020124	040440	052103		
9219	051124	040525	000114			
9220						.EVEN
9221	051130	001116	001164	001162	DM11:	.WORD \$ERPPC,\$REG2,\$REG1,\$STSTN,0
9222	051136	001266	000000			
9223	051142	044515	051103	026517	EM12:	.ASCIZ /MICRO-BREAK TRAP DID NOT TRAP/
9224	051150	051102	040505	020113		
9225	051156	051124	050101	042040		
9226	051164	042111	047040	052117		
9227	051172	052040	040522	000120		
9228	051200	044505	044124	051105	EM13:	.ASCII /EITHER FXPR FIRA00 OR FIRA01 IS STUCK LOW OR/<CRLF>
9229	051206	043040	050130	020107		
9230	051214	044506	040522	030060		
9231	051222	047440	020122	044506		
9232	051230	040522	030460	044440		
9233	051236	020123	052123	041525		

9234	051244	020113	047514	020127	
9235	051252	051117	200		
9236	051255	116	052117	043440	.ASCIZ /NOT GETTING TO FXPC E94 OR E94 BAD/
9237	051262	052105	044524	043516	
9238	051270	052040	020117	054106	
9239	051276	041520	042440	032071	
9240	051304	047440	020122	034505	
9241	051312	020064	040502	000104	
9242	051320	050106	020125	051124	EM14: .ASCII /FPU TRAPPED TO 4 INSTEAD OF 224/<CRLF>
9243	051326	050101	042520	020104	
9244	051334	047524	032040	044440	
9245	051342	051516	042524	042101	
9246	051350	047440	020106	031062	
9247	051356	100064			
9248	051360	044505	044124	051105	.ASCIZ /EITHER TMCB FPTRAP NOT GOING LOW OR NOT GETTING TO DAPE/
9249	051366	052040	041515	020102	
9250	051374	050106	051124	050101	
9251	051402	047040	052117	043440	
9252	051410	044517	043516	046040	
9253	051416	053517	047440	020122	
9254	051424	047516	020124	042507	
9255	051432	052124	047111	020107	
9256	051440	047524	042040	050101	
9257	051446	000105			
9258	051450	054106	041120	043040	EM15: .ASCII /FXPB FIRA03 STUCK LOW OR NOT GETTING TO FRMJ PD MUX OR/<CRLF>
9259	051456	051111	030101	020063	
9260	051464	052123	041525	020113	
9261	051472	047514	020127	051117	
9262	051500	047040	052117	043440	
9263	051506	052105	044524	043516	
9264	051514	052040	020117	046506	
9265	051522	045122	043040	020104	
9266	051530	052515	020130	051117	
9267	051536	200			
9268	051537	106	046522	020117	.ASCIZ /FRMJ PD MUX BAD OR PD CLCK BAD/
9269	051544	042106	046440	054125	
9270	051552	041040	042101	047440	
9271	051560	020122	042106	041440	
9272	051566	041514	070113	040502	
9273	051574	000104			
9274	051576	051105	050122	020103	DM15: .ASCII /ERPPC FPS TEST NO/<CRLF>
9275	051604	020040	020040	020040	
9276	051612	043040	051520	070040	
9277	051620	020040	020040	020040	
9278	051626	042524	052123	047040	
9279	051634	100117			
9280	051636	042411	050130	041505	.ASCIZ / EXPECT ACTUAL/
9281	051644	020124	040440	052103	
9282	051652	040525	000114		
9283					.EVEN
9284	051656	001116	001202	001200	DT15: .WORD \$ERRPC,\$STMP1,\$STMP0,\$STSTNM,0
9285	051664	001266	000000		
9286	051670	051106	045115	043040	EM16: .ASCIZ /FRMJ FL MUX BAD OR FL CLOCK BAD/
9287	051676	020114	052515	020130	
9288	051704	040502	020104	051117	
9289	051712	043040	020114	046103	

9290	051720	041517	020113	040502					
9291	051726	000104							
9292	051730	051106	045115	043040	EM17:	.ASCIZ	/FRMJ PD MUX INPUT FLOATING?/		
9293	051736	020104	052515	020130					
9294	051744	047111	052520	020124					
9295	051752	046106	040517	044524					
9296	051760	043516	000077						
9297	051764	051106	045115	043040	EM20:	.ASCIZ	/FRMJ PL MUX INPUT FLOATING?/		
9298	051772	020114	052515	020130					
9299	052000	047111	052520	020124					
9300	052006	046106	040517	044524					
9301	052014	043516	000077						
9302	052020	044505	044124	051105	EM21:	.ASCII	/EITHER FXPC FCLD EN NOT GOING LOW OR/<CRLF>		
9303	052026	043040	050130	020103					
9304	052034	041506	042114	042440					
9305	052042	020116	047516	020124					
9306	052050	047507	047111	020107					
9307	052056	047514	020127	051117					
9308	052064	200							
9309	052065	116	052117	043440		.ASCIZ	/NOT GETTING TO IRCE OR IRCE CC < BR NOT GOING HIGH/		
9310	052072	052105	044524	043516					
9311	052100	052040	020117	051111					
9312	052106	042503	047440	020122					
9313	052114	051111	042503	041440					
9314	052122	020103	020074	051102					
9315	052130	047040	052117	043440					
9316	052136	044517	043516	044040					
9317	052144	043511	000110						
9318	052150	051105	050122	020103	DN21:	.ASCII	/ERRPC	PSM	TEST NO/<CRLF>
9319	052156	020040	020040	020040					
9320	052164	050040	053523	020040					
9321	052172	020040	020040	020040					
9322	052200	042524	052123	047040					
9323	052206	100117							
9324	052210	042411	050130	041505		.ASCIZ	/	EXPECT	ACTUAL/
9325	052216	020124	040440	052103					
9326	052224	040525	000114						
9327	052230	051106	043115	043040	EM22:	.ASCIZ	/FRMF FPREQ DOES NOT GET TO RACK AS A LOW/		
9328	052236	051120	050505	042040					
9329	052244	042517	020123	047516					
9330	052252	020124	042507	020124					
9331	052260	047524	051040	041501					
9332	052266	020113	051501	040440					
9333	052274	046040	053517	000					
9334	052301	122	041501	020113	EM23:	.ASCIZ	/RACK BRCAB05 NOT GOING LOW ON BRQ*(T+COMP)/		
9335	052306	051102	040503	030102					
9336	052314	020065	047516	020124					
9337	052322	047507	047111	020107					
9338	052330	047514	020127	047117					
9339	052336	041040	050522	024052					
9340	052344	025524	047503	043116					
9341	052352	000051							
9342	052354	051124	050101	042520	EM24:	.ASCIZ	/TRAPPED TO 4 DURING LDF/		
9343	052362	020104	047524	032040					
9344	052370	042040	051125	047111					
9345	052376	020107	042114	000106					

9346	052404	051105	050122	020103	DN24:	.ASCIZ	/ERRPC	ERRRPC	TEST NO/
9347	052412	020040	051105	051122					
9348	052420	043505	020040	042524					
9349	052426	052123	047040	000117					
9350									
9351	052434	001116	001200	001266	DT24:	.EVEN			
9352	052442	000000				.WORD	\$ERRPC,\$TMP0,\$STSTNM,0		
9353	052444	052123	041501	042513	EM25:	.ASCIZ	/STACKED PC IS WRONG/		
9354	052452	020104	041520	044440					
9355	052460	020123	051127	047117					
9356	052466	000107							
9357	052470	051105	050122	004503	DN25:	.ASCII	/ERRPC	PC	TEST NO/<CRLF>
9358	052476	020040	020040	020040					
9359	052504	041520	020040	020040					
9360	052512	020040	020040	042524					
9361	052520	052123	047040	100117					
9362	052526	042411	050130	041505		.ASCIZ	/	EXPECT	ACTUAL/
9363	052534	020124	040440	052103					
9364	052542	040525	000114						
9365	052546	051111	042103	042040	EM26:	.ASCIZ	/TRCD DSTCON<2	JT GOING HIGH/	
9366	052554	052123	047503	036116					
9367	052562	020062	047516	020124					
9368	052570	047507	047111	020107					
9369	052576	044519	044107	000					
9370	052603	106	050130	020102	EM27:	.ASCIZ	/FXPB A BRANCH MUX(A3) DOES NOT GO HIGH/		
9371	052610	020101	051102	047101					
9372	052616	044103	046440	054125					
9373	052624	040450	024463	042040					
9374	052632	042517	020123	047516					
9375	052640	020124	047507	044040					
9376	052646	043511	000110						
9377	052652	054106	041120	043040	EM30:	.ASCIZ	/FXPB FIRA10(0)H DOES NOT GO LOW/		
9378	052660	051111	030501	024060					
9379	052666	024460	020110	047504					
9380	052674	051505	047040	052117					
9381	052702	043440	020117	047514					
9382	052710	000127							
9383	052712	051106	043115	043040	EM31:	.ASCIZ	/FRMF FPREQ NOT GOING HIGH ON IMMEDIATE/		
9384	052720	051120	050505	047040					
9385	052726	052117	043440	044517					
9386	052734	043516	044040	043511					
9387	052742	020110	047117	044440					
9388	052750	046515	042105	040511					
9389	052756	042524	000						
9390	052761	106	050130	020101	EM32:	.ASCIZ	/FXPA AD2 DID NOT GO HIGH AT ROW ADR 210/		
9391	052766	042101	020062	044504					
9392	052774	020104	047516	020124					
9393	053002	047507	044040	043511					
9394	053010	020110	052101	051040					
9395	053016	046517	040440	051104					
9396	053024	031040	030061	000					
9397	053031	106	050130	020102	EM33:	.ASCIZ	/FXPB FIRA10(0)H DID NOT GO HIGH/		
9398	053036	044506	040522	030061					
9399	053044	030050	044051	042040					
9400	053052	042111	047040	052117					
9401	053060	043440	020117	044510					

9402	053066	044107	000					
9403	053071	103	047117	051124	EM34:	.ASCIZ	/CONTROL STORE BRANCH FAILED/	
9404	053076	046117	051440	047524				
9405	053104	042522	041040	040522				
9406	053112	041516	070110	040506				
9407	053120	046111	042105	000				
9408	053125	105	051122	041520	DH34:	.ASCIZ	/ERRPC POWADR TEST NO/	
9409	053132	020040	051040	046517				
9410	053140	042101	020127	052040				
9411	053146	051505	020124	047516				
9412	053154	000						
9413	053155	122	046517	043040	EM35:	.ASCIZ	/ROM FLOW OR PUT DATA IS BAD/	
9414	053162	047514	020127	045517				
9415	053170	041040	052125	042040				
9416	053176	052101	020101	051511				
9417	053204	041040	042101	000				
9418	053211	105	051122	041520	DH35:	.ASCII	/ERRPC	DATA TEST NO/<CRLF>
9419	053216	004411	020040	020040				
9420	053224	004440	040504	040524				
9421	053232	004411	052011	051505				
9422	053240	020124	047516	200				
9423	053245	011	020040	020040		.ASCIZ	/	EXPECT ACTUAL/
9424	053252	020040	054105	042520				
9425	053260	052103	020011	020040				
9426	053266	040411	052103	040525				
9427	053274	000114						
9428						.EVEN		
9429	053276	001116	001200	001204	DT35:	.WOPD	\$ERRPC,\$TMP0,\$TMP2,\$STSTNM,0	
9430	053304	001266	000000					
9431	053310	000	001	001	DF35:	.RYTE	0,1,1,0	
9432	053313	000						
9433	053314	051111	042103	042040	EM36:	.ASCIZ	/IKCD DSTCON<4 DID NOT GO HIGH/	
9434	053327	052123	047503	036116				
9435	053330	020064	044504	020104				
9436	053336	047516	020124	047507				
9437	053344	044040	043511	000110				
9438	053352	051111	042103	042040	EM37:	.ASCIZ	/IKCD DSTCON<10 DID NOT GO LOW/	
9439	053360	052123	047503	036116				
9440	053366	030061	042040	042111				
9441	053374	047040	052117	043440				
9442	053402	020117	047514	000127				
9443	053410	051504	020124	042522	EM40:	.ASCIZ	/DST PEG DID NOT INCREMENT CORRECTLY/	
9444	053416	020107	044504	020104				
9445	053424	047516	020124	047111				
9446	053437	051103	046505	047105				
9447	053440	020124	047503	051122				
9448	053446	041505	046124	000131				
9449	053454	051106	045115	043040	EM41:	.ASCIZ	/FRMJ FD(1)H NOT GETTING TO FRM AS A LOW/	
9450	053462	024104	024461	020110				
9451	053470	047516	020124	042507				
9452	053476	052124	047111	020107				
9453	053504	047524	043040	046522				
9454	053512	040440	020127	070101				
9455	053520	047514	000127					
9456	053524	042114	020106	045517	EM42:	.ASCIZ	/LDF OK BUT STF FAILED AUTO-INCREMENT/	
9457	053532	041040	052125	051440				

9459	053540	043124	043040	044501		
9459	053546	042514	020104	052501		
9460	053554	047524	044455	041516		
9461	053562	042522	042515	052116		
9462	053570	000				
9463	053571	105	052111	042510	EM43:	.ASCIZ /EITHER FIRST OR SECOND WORD FAILED TO LOAD OR STORE/
9464	053576	020122	044506	051522		
9465	053604	020124	051117	051440		
9466	053612	041505	047117	020104		
9467	053620	047527	042122	043040		
9468	053626	044501	042514	020104		
9469	053634	047524	046040	040517		
9470	053642	020104	051117	051440		
9471	053650	047524	042522	000		
9472	053655	102	052111	051440	EM44:	.ASCIZ /BIT STUCK IN DATA PATH IN FPP/
9473	053662	052524	045503	044440		
9474	053670	020116	040504	040524		
9475	053676	050040	052101	020110		
9476	053704	047111	043040	050120		
9477	053712	000				
9478	053713	104	040525	020114	EM45:	.ASCIZ /DUAL ADDRESSING IN QUAD(3,2) ACC'S/
9479	053720	042101	051104	051505		
9480	053726	044523	043516	044440		
9481	053734	020116	052521	042101		
9482	053742	031533	031054	020135		
9483	053750	041501	023503	000123		
9484	053756	042114	050106	025123	EM46:	.ASCIZ /LDPPS*-NO FAILED (CONT STORE)/
9485	053764	046455	020060	040506		
9486	053772	046111	042105	024040		
9487	054000	047503	052116	051440		
9488	054006	047524	042522	000051		
9489	054014	052123	050106	025123	EM47:	.ASCIZ /STPPS*-NO FAILED (CONT STORE)/
9490	054022	046455	020060	040506		
9491	054030	046111	042105	024040		
9492	054036	047503	052116	051440		
9493	054044	047524	042522	000051		
9494	054052	041503	051447	041040	EM50:	.ASCIZ /CC'S BAD ON LOAD FLOAT/
9495	054060	042101	047440	020116		
9496	054066	047514	042101	043040		
9497	054074	047514	052101	000		
9498		054102				
9499	054102	001116	001210	001212	DT50:	.EVFN .WORD \$ERRPC,STMP4,STMP5,\$STSTM,0
9500	054110	001266	000000			
9501	054114	054105	020120	046103	EM51:	.ASCIZ /EXP CLEARED, FRAC DIDN'T/
9502	054122	040505	042522	026104		
9503	054130	043040	040522	020103		
9504	054136	044504	047104	052047		
9505	054144	000				
9506	054145	106	040522	020103	EM52:	.ASCIZ /FRAC CLEARED, EXP DIDN'T/
9507	054152	046103	040505	042522		
9508	054160	026104	042440	050130		
9509	054166	042040	042111	023516		
9510	054174	000124				
9511	054176	040502	020104	041503	EM53:	.ASCIZ /BAD CC'S ON CLEAR FLOAT (CONT STORE)/
9512	054204	051447	047440	020116		
9513	054212	046103	040505	020122		

9514	054220	046106	040517	020124	
9515	054226	041450	047117	020124	
9516	054234	052123	051117	024505	
9517	054242	000			
9518	054243	102	042101	041440	EM54: .ASCIZ /BAD CC'S ON STORE FLOAT (CONT STORE)/
9519	054250	023503	020123	047117	
9520	054256	051440	047524	042522	
9521	054264	043040	047514	052101	
9522	054272	024040	047503	052116	
9523	054300	051440	047524	042522	
9524	054306	000051			
9525	054310	047522	020115	046106	EM55: .ASCIZ /ROM FLOW OK, BUT DATA BAD/
9526	054316	053517	047440	026113	
9527	054324	041040	052125	042040	
9528	054332	052101	020101	040502	
9529	054340	000104			
9530	054342	041503	051447	041040	EM56: .ASCIZ /CC'S BAD ON TSTP, IF Z BIT CHECK FXPL ACMX EQ 0 NOT GOING HIGH/
9531	054350	042101	047440	020116	
9532	054356	051524	043124	020054	
9533	054364	043111	055040	041040	
9534	054372	052111	041440	042510	
9535	054400	045503	043040	050130	
9536	054406	020114	041501	054115	
9537	054414	042440	020121	020060	
9538	054422	047516	020124	047507	
9539	054430	047111	020107	044510	
9540	054436	044107	000		
9541	054441	106	046522	020106	EM57: .ASCIZ /PRMF SD MUX NOT GOING LOW WHEN FXPM ACMXC(1) IS LOW/
9542	054446	042123	046440	054125	
9543	054454	047040	052117	043440	
9544	054462	044517	043516	046040	
9545	054470	053517	053440	042510	
9546	054476	020116	054106	046520	
9547	054504	040440	046503	041530	
9548	054512	030450	020051	051511	
9549	054520	046040	053517	000	
9550	054525	111	041522	020102	EM60: .ASCIZ /TRCB BO PAR03 DID NOT GO LOW/
9551	054532	047502	051040	041101	
9552	054540	031460	042040	042111	
9553	054546	047040	052117	043440	
9554	054554	020117	047514	000127	
9555	054562	051111	041103	041040	EM61: .ASCIZ /TRCB BO PAR01 DID NOT GO LOW/
9556	054570	020117	040522	030102	
9557	054576	020061	044504	020104	
9558	054604	047516	020124	047507	
9559	054612	046040	053517	000	
9560	054617	102	052111	051440	EM62: .ASCIZ "BIT STUCK IN CPU/FPU INTERFACE OR FDR"
9561	054624	052524	045503	044440	
9562	054632	020116	050103	027525	
9563	054640	050106	020125	047111	
9564	054646	042524	043122	041501	
9565	054654	020105	051117	043040	
9566	054662	051104	000		
9567	054665	123	043511	020116	EM63: .ASCIZ /SIGN BIT DID NOT CLEAR/
9568	054672	044502	020124	044504	
9569	054700	020104	047516	020124	

9570	054706	046103	040505	000122		
9571	054714	054105	047520	042516	EM64:	.ASCIZ /EXPONENT DID NOT CLEAR/
9572	054722	052116	042040	042111		
9573	054730	047040	052117	041440		
9574	054736	042514	051101	000		
9575	054743	106	040522	052103	EM65:	.ASCIZ /FRACTION DID NOT CLEAR/
9576	054750	047511	020116	044504		
9577	054756	020104	047516	020124		
9578	054764	046103	040505	000122		
9579	054772	044124	020105	042101	EM66:	.ASCIZ /THE ADX ROM FAILED/
9580	055000	020130	047522	020115		
9581	055006	040506	046111	042105		
9582	055014	000				
9583	055015	105	052111	042510	EM67:	.ASCIZ /EITHER FIRB06(1)H NOT GOING LOW OR NOT GETTING TO PRMA RADQ03/
9584	055022	020122	044506	041122		
9585	055030	033060	030450	044051		
9586	055036	047040	052117	043440		
9587	055044	044517	043516	046040		
9588	055052	053517	047440	020122		
9589	055060	047516	020124	042507		
9590	055066	052124	047111	020107		
9591	055074	047524	043040	046522		
9592	055102	020101	040522	051104		
9593	055110	031460	000			
9594	055113	105	052111	042510	EM70:	.ASCIZ /EITHER PRMB BZ(0)H NOT GOING HIGH OR NOT GETTING TO PRMA RADP00/
9595	055120	020122	051106	041115		
9596	055126	041040	024132	024460		
9597	055134	020110	047516	020124		
9598	055142	047507	047111	020107		
9599	055150	044510	044107	047440		
9600	055156	020122	047516	020124		
9601	055164	042507	052124	047111		
9602	055172	020107	047524	043040		
9603	055200	046522	020101	040522		
9604	055206	051104	030060	000		
9605	055213	103	047117	044504	EM71:	.ASCIZ /CONDITION CODES RAD/
9606	055220	044524	047117	041440		
9607	055226	042117	051505	041040		
9608	055234	042101	000			
9609	055237	105	052111	042510	EM72:	.ASCIZ /EITHER FIRB06(1)H NOT GOING HIGH OR NOT GETTING TO PRMA RADQ03/
9610	055244	020122	044506	041122		
9611	055252	033060	030450	044051		
9612	055260	047040	052117	043440		
9613	055266	044517	043516	044040		
9614	055274	043511	020110	051117		
9615	055302	047040	052117	043440		
9616	055310	052105	044524	043516		
9617	055316	052040	020117	051106		
9618	055324	040515	051040	042101		
9619	055332	030122	000063			
9620	055336	040504	040524	041440	EM73:	.ASCIZ /DATA CHANGED/
9621	055344	040510	043516	042105		
9622	055352	000				
9623	055353	105	052111	042510	EM74:	.ASCIZ /EITHER FRMJ EXPA0(1)H NOT GOING LOW OR NOT GETTING TO PRMA RADQ03/
9624	055360	020122	051106	045115		
9625	055366	042440	050130	030101		

9626	055374	030450	044051	047040	
9627	055402	052117	043440	044517	
9628	055410	043516	046040	053517	
9629	055416	047440	020122	047516	
9630	055424	020124	042507	052124	
9631	055432	047111	020107	047524	
9632	055440	043040	046522	020101	
9633	055446	040522	030104	000063	
9634	055454	044505	044124	051105	EM75: .ASCIZ /EITHER FRNJ EXPB0(1)H NOT GOING HIGH OR NOT GETTING TO FRNA RAD02/
9635	055462	043040	046522	020112	
9636	055470	054105	041120	024060	
9637	055476	024461	020110	047516	
9638	055504	020124	047507	047111	
9639	055512	020107	044510	044107	
9640	055520	047440	020122	047516	
9641	055526	020124	042507	052124	
9642	055534	047111	020107	047524	
9643	055542	043040	046522	020101	
9644	055550	040522	030104	000062	
9645	055556	044505	044124	051105	EM76: .ASCIZ /EITHER FRNJ EXPA0(1)H NOT GOING HIGH OR NOT GETTING TO FRNA RAD03/
9646	055564	043040	046522	020112	
9647	055572	054105	040520	024060	
9648	055600	024461	020110	047516	
9649	055606	020124	047507	047111	
9650	055614	020107	044510	044107	
9651	055622	047440	020122	047516	
9652	055630	020124	042507	052124	
9653	055636	047111	020107	047524	
9654	055644	043040	046522	020101	
9655	055652	040522	030104	000063	
9656	055660	047522	020115	046106	EM100: .ASCIZ /ROM FLOW OK, BUT CC'S BAD/
9657	055666	053517	047440	026113	
9658	055674	041040	052125	041440	
9659	055702	023503	020123	040502	
9660	055710	000104			
9661	055712	047522	020115	052123	EM101: .ASCIZ /ROM STATE 334 DID NOT CLR DST FRAC/
9662	055720	052101	020105	031463	
9663	055726	020064	044504	020104	
9664	055734	047516	020124	046103	
9665	055742	020122	051504	020124	
9666	055750	051106	041501	000	
9667	055755	105	052111	042510	EM102: .ASCIZ /EITHER FRMB RZ(0)H NOT GOING LOW OR NOT GETTING TO FRNA RAD00 AS A H/
9668	055762	020122	051106	041115	
9669	055770	041040	024132	024460	
9670	055776	020110	047516	020124	
9671	056004	047507	047111	020107	
9672	056012	047514	020127	051117	
9673	056020	047040	052117	043440	
9674	056026	052105	044524	043516	
9675	056034	052040	020117	051106	
9676	056042	040515	051040	042101	
9677	056050	030060	040440	020123	
9678	056056	020101	000110		
9679	056062	044505	044124	051105	EM104: .ASCIZ /EITHER STATE 105 OR 362 DID NOT LOAD FRACTION PROPERLY/
9680	056070	051440	040524	042524	
9681	056076	030440	032460	047440	

9682	056104	020122	033063	020062
9683	056112	044504	020104	047516
9684	056120	020124	047514	042101
9685	056126	043040	040522	052103
9686	056134	047511	020116	051120
9687	056142	050117	051105	054514
9688	056150	000		
9689	056151	105	050130	047117
9690	056156	047105	020124	044504
9691	056164	020104	047516	020124
9692	056172	047514	042101	050040
9693	056200	047522	042520	046122
9694	056206	000131		
9695	056210	044505	044124	051105
9696	056216	043040	046522	020102
9697	056224	047102	030050	044051
9698	056232	042040	042517	020123
9699	056240	047516	020124	047507
9700	056246	044040	043511	020110
9701	056254	051117	047040	052117
9702	056262	043440	052105	044524
9703	056270	043516	052040	020117
9704	056276	051106	040515	051040
9705	056304	042101	030460	040440
9706	056312	020123	020101	000114
9707	056320	044507	020124	040506
9708	056326	046111	042105	044440
9709	056334	020116	052123	050105
9710	056342	041440	052517	052116
9711	056350	051105	000	
9712	056353	123	040524	042524
9713	056360	031040	030065	047440
9714	056366	020122	033463	020067
9715	056374	040506	046111	042105
9716	056402	052040	020117	047514
9717	056410	042101	042440	050130
9718	056416	047440	020122	051106
9719	056424	041501	000	
9720	056427	106	046522	020106
9721	056434	051523	030050	044051
9722	056447	054040	051117	051440
9723	056450	041125	047040	052117
9724	056456	043440	044517	043516
9725	056464	046040	053517	047440
9726	056472	020122	047516	020124
9727	056500	042507	052124	047111
9728	056506	020107	047524	051440
9729	056514	020104	046106	050117
9730	056522	000		
9731	056523	106	046522	020106
9732	056530	051523	030050	044051
9733	056536	054040	051117	051440
9734	056544	041125	047040	052117
9735	056552	043440	044517	043516
9736	056560	044040	043511	020110
9737	056566	051117	047040	052117

EM105: .ASCIZ /EXPONENT DID NOT LOAD PROPERLY/

EM106: .ASCIZ /EITHER FRMB BN(0)H DOES NOT GO HIGH OR NOT GETTING TO FRMA RAD01 AS A L

EM107: .ASCIZ /BIT FAILED IN STEP COUNTER/

EM110: .ASCIZ /STATE 250 OR 377 FAILED TO LOAD EXP OR FRAC/

EM111: .ASCIZ /FRMF SS(0)H XOR SUB NOT GOING LOW OR NOT GETTING TO SD FLOP/

EM112: .ASCIZ /FRMF SS(0)H XOR SUB NOT GOING HIGH OR NOT GETTING TO SD FLOP/

9738	056574	043440	052105	044524	
9739	056602	043516	052040	020117	
9740	056610	042123	043040	047514	
9741	056616	000120			
9742	056620	054106	046120	041040	EM113: .ASCIZ /FXPL BCM NOT GOING HIGH WITH EALU09 LOW/
9743	056626	047103	047040	052117	
9744	056634	043440	044517	043516	
9745	056642	044040	043511	020110	
9746	056650	044527	044124	042440	
9747	056656	046101	030125	020071	
9748	056664	047514	000127		
9749	056670	051106	043115	051440	EM114: .ASCIZ /PRMF SS XOR SD NOT GOING LOW OR NOT GETTING TO SD FLOP/
9750	056676	020123	047530	020122	
9751	056704	042123	047040	052117	
9752	056712	043440	044517	043516	
9753	056720	046040	053517	047440	
9754	056726	020122	047516	020124	
9755	056734	042507	052124	047111	
9756	056742	020107	047524	051440	
9757	056750	020104	046106	050117	
9758	056756	000			
9759	056757	106	046522	020106	EM115: .ASCIZ /PRMF SS XOR SD NOT DOING WRIGHT THING WITH HR OR HL INPUT/
9760	056764	051523	054040	051117	
9761	056772	051440	020104	047516	
9762	057000	020124	047504	047111	
9763	057006	020107	051127	043511	
9764	057014	052110	052040	044510	
9765	057022	043516	053440	052111	
9766	057030	020110	044110	047440	
9767	057036	020122	046110	044440	
9769	057044	050116	052125	000	
9769	057051	105	052111	042510	EM116: .ASCIZ /EITHER FXPL BCM NOT GOING LOW WITH EALU09 H OR/<CRLF>
9770	057056	020122	054106	046120	
9771	057064	041040	047103	047040	
9772	057072	052117	043440	044517	
9773	057100	043516	046040	053517	
9774	057106	053440	052111	020110	
9775	057114	040505	052514	034460	
9776	057122	044040	047440	100122	
9777	057130	051106	043115	051440	.ASCIZ /PRMF SS(0)H NOT GETTING THRU SD MUX AS A HIGH/
9778	057136	024123	024460	020110	
9779	057144	047516	020124	042507	
9780	057152	052124	047111	020107	
9781	057160	044124	052522	051440	
9782	057166	020104	052515	020130	
9783	057174	051501	040440	044040	
9784	057202	043511	000110		
9785	057206	051106	043115	051440	EM117: .ASCIZ /PRMF SS(0)H NOT GETTING THRU SD MUX AS A LOW/
9786	057214	024123	024460	020110	
9787	057222	047516	020124	042507	
9788	057230	052124	047111	020107	
9789	057236	044124	052522	051440	
9790	057244	020104	052515	020130	
9791	057252	051501	040440	046040	
9792	057260	053517	000		
9793	057263	111	041522	020104	EM170: .ASCIZ /TRCD DSTCON<10 NOT GOING HIGH/

9794	057270	051504	041524	047117				
9795	057276	030474	020060	047516				
9796	057304	020124	047507	047111				
9797	057312	020107	044510	044107				
9798	057320	000						
9799	057321	122	046517	043040	EM121:	.ASCIZ	/ROM FLOW OK, DATA PAD/	
9800	057326	047514	020127	045517				
9801	057334	020054	040504	040524				
9802	057342	041040	042101	000				
9803	057347	011	004411	042011	DM121:	.ASCII	/	DATA/<CRLF>
9804	057354	052101	100101					
9805	057360	020011	020040	054105		.ASCIZ	/	EXPECT
9806	057366	042520	052103	004411				ACTUAL/
9807	057374	020011	020040	020040				
9808	057402	020040	041501	052524				
9809	057410	046101	000					
9810		057414						
9811	057414	001200	001210	000000	DT121:	.EVEN		
9812	057427	002	002		DF121:	.WORD	\$TMP0,\$TMP4,0	
9813						.BYTE	2,2	
9814	057424	052504	046101	040440	EM123:	.EVEN		
9815	057432	042104	042522	051523		.ASCIZ	/DUAL ADDRESSING IN QUAD1,03 ACC'S/	
9816	057440	047111	020107	047111				
9817	057446	050440	040525	055504				
9818	057454	026061	056460	040440				
9819	057462	041503	051447	000				
9820	057467	122	046517	051440	EM124:	.ASCIZ	/ROM STATE 143 FAILED/	
9821	057474	040524	042524	030440				
9822	057502	031464	043040	044501				
9823	057510	042514	000104					
9824	057514	052521	042101	041040	EM125:	.ASCIZ	/QUAD BAD/	
9825	057522	042101	000					
9826	057525	106	044122	020103	EM126:	.ASCIZ	/PRNC PNX 35 NOT GOING LOW/	
9827	057532	046506	020130	032463				
9828	057540	047040	052117	043440				
9829	057546	044517	043516	046040				
9830	057554	053517	000					
9831	057557	106	044122	020105	EM127:	.ASCIZ	/PRHE AP59(1)L NOT GOING HIGH OR NOT GETTING TO PRMA RAD01 AS A L/	
9832	057564	051101	034465	030450				
9833	057572	046051	047040	052117				
9834	057600	043440	044517	043516				
9835	057606	044040	043511	020110				
9836	057614	051117	047040	052117				
9837	057622	043440	052105	044524				
9838	057630	043516	052040	020117				
9839	057636	051106	040515	051040				
9840	057644	042101	030460	040440				
9841	057652	020123	020101	000114				
9842	057660	051106	041510	043040	EM130:	.ASCIZ	/PRHC PNX 34 NOT GOING HIGH OR PALU BAD/	
9843	057666	054115	031440	020064				
9844	057674	047516	020124	047507				
9845	057702	047111	020107	044510				
9846	057710	044107	047440	020122				
9847	057716	040506	052514	041040				
9848	057724	042101	000					
9849	057727	123	042524	050130	EM131:	.ASCIZ	/STEXP FAILED/	

9850	057734	043040	044501	042514	
9851	057742	000104			
9852	057744	051106	043115	042440	EM132: .ASCIZ /FRWF EALD CIN DID NOT GO LOW/
9853	057752	046101	020125	044503	
9854	057760	020116	044504	020104	
9855	057766	047516	020124	047507	
9856	057774	046040	053517	000	
9857	060001	106	050130	020103	EM133: .ASCIZ /FXPC FCLD EM DID NOT GO LOW/
9858	060006	041506	042114	042440	
9859	060014	020116	044504	020104	
9860	060022	047516	020124	047507	
9861	060030	046040	053517	000	
9862	060035	106	050130	020106	EM134: .ASCIZ /FXPF ESXT DID NOT GO HIGH/
9863	060042	051505	052130	042040	
9864	060050	042111	047040	052117	
9865	060056	043440	020117	044510	
9866	060064	044107	000		
9867	060067	103	051114	020104	EM135: .ASCIZ /CLRD FAILED/
9868	060074	040506	046111	042105	
9869	060102	000			
9870	060103	101	051502	020106	EM136: .ASCIZ /ABSF FAILED TO STORE CORRECT DATA/
9871	060110	040506	046111	042105	
9872	060116	052040	020117	052123	
9873	060124	051117	020105	047503	
9874	060132	051122	041505	020124	
9875	060140	040504	040524	000	
9876	060145	116	043505	020106	EM137: .ASCIZ /NEGF FAILED TO STORE CORRECT DATA/
9877	060152	040506	046111	042105	
9878	060160	052040	020117	052123	
9879	060166	051117	020105	047503	
9880	060174	051122	041505	020124	
9881	060202	040504	040524	000	
9882	060207	104	052101	020101	EM140: .ASCIZ /DATA BAD ON ADDF*-W0/
9883	060214	040507	020104	047117	
9884	060222	040440	042104	025106	
9885	060230	046455	000060		
9886	060234	040504	040524	041040	EM141: .ASCIZ /DATA BAD/
9887	060242	042101	000		
9888	060245	123	040524	042524	EM142: .ASCIZ /STATE 51 DID NOT SET BN/
9889	060252	032440	020061	044504	
9890	060260	020104	047516	020124	
9891	060266	042523	020124	047107	
9892	060274	000			
9893	060275	122	046517	051440	EM143: .ASCIZ /POM STATE 361 OR 237 FAILED/
9894	060302	040524	042524	031440	
9895	060310	030466	047440	020122	
9896	060316	031462	020067	040506	
9897	060324	046111	042105	000	
9898	060331	114	042504	050130	EM144: .ASCIZ /LDXP DID NOT CLR DST ON UNDERFLOW/
9899	060336	042040	042111	047040	
9900	060344	052117	041440	051114	
9901	060352	042040	052123	047440	
9902	060360	020116	047125	042504	
9903	060366	043122	047514	000127	
9904	060374	052123	052101	020105	EM145: .ASCIZ /STATE 363 FAILED TO LOAD DST/
9905	060402	033063	020063	040506	

9906	060410	046111	042105	052040	
9907	060416	020117	047514	042101	
9908	060424	042040	052123	000	
9909	060431	106	050130	020112	EM146: .ASCIZ /FXPJ OVF NOT GOING HIGH OR NOT GETTING TO FRLN/
9910	060436	053117	020106	047516	
9911	060444	020124	047507	047111	
9912	060452	020107	044510	044107	
9913	060460	047440	020122	047516	
9914	060466	020124	042507	052124	
9915	060474	047111	020107	047524	
9916	060502	043040	046122	000116	
9917	060510	052521	042101	031440	EM147: .ASCIZ /QUAD 3 DID NOT CLEAR/
9918	060516	042040	042111	047040	
9919	060524	052117	041440	042514	
9920	060532	051101	000		
9921	060535	106	046522	020106	EM150: .ASCIZ /FRMF ADD*SC<R DID NOT GO HIGH/
9922	060542	042101	025104	041523	
9923	060550	034074	042040	042111	
9924	060556	047040	052117	043440	
9925	060564	020117	044510	044107	
9926	060572	000			
9927	060573	106	046522	020106	EM151: .ASCIZ /FRMF SUB*SC<R DID NOT GO HIGH/
9928	060600	052523	025102	041523	
9929	060606	034074	042040	042111	
9930	060614	047040	052117	043440	
9931	060622	020117	044510	044107	
9932	060630	000			
9933	060631	106	050130	020120	EM152: .ASCIZ /FXPP OUT OF RANGE NOT GETTING TO FRMA AS A HIGH/
9934	060636	052517	020124	043117	
9935	060644	051040	047101	042507	
9936	060652	047040	052117	043440	
9937	060660	052105	044524	043516	
9938	060666	052040	020117	051106	
9939	060674	040515	040440	020123	
9940	060702	020101	044510	044107	
9941	060710	000			
9942	060711	106	050130	020120	EM153: .ASCIZ /FXPP SC09 XOR SC07 DID NOT GO LOW/
9943	060716	041523	034460	054040	
9944	060724	051117	051440	030103	
9945	060732	020067	044504	020104	
9946	060740	047516	020124	047507	
9947	060746	046040	053517	000	
9948	060753	106	050130	020120	EM154: .ASCIZ /FXPP SC09 XOR SC06 DID NOT GO LOW/
9949	060760	041523	034460	054040	
9950	060766	051117	051440	030103	
9951	060774	020066	044504	020104	
9952	061002	047516	020124	047507	
9953	061010	046040	053517	000	
9954	061015	106	050130	020120	EM155: .ASCIZ /FXPP SC09 DID NOT GO LOW/
9955	061022	041523	034460	042040	
9956	061030	042111	047040	052117	
9957	061036	043440	020117	047514	
9958	061044	000127			
9959	061046	054106	050120	051440	EM156: .ASCIZ /FXPP SC09 NOT GETTING TO FRMA AS A HIGH/
9960	061054	030103	020071	047516	
9961	061062	020124	042507	052124	

9962	061070	047111	020107	047524	
9963	061076	043040	046522	020101	
9964	061104	051501	040440	044040	
9965	061112	043511	000110		
9966	061116	052123	052101	020105	EM157: .ASCIZ /STATE 274 FAILED TO LOAD DST/
9967	061124	033462	020064	040506	
9968	061132	046111	042105	052040	
9969	061140	020117	047514	042101	
9970	061146	042040	052123	000	
9971	061153	106	046522	020106	EM160: .ASCIZ /PRMF SD(1) NOT GETTING TO PRMA RADO1 AS A HIGH/
9972	061160	042123	030450	020051	
9973	061166	047516	020124	042507	
9974	061174	052124	047111	020107	
9975	061202	047524	043040	046522	
9976	061210	020101	040522	030104	
9977	061216	020061	051501	040440	
9978	061224	044040	043511	000110	
9979	061232	051106	043115	051440	EM161: .ASCIZ /PRMF SD(1) NOT GETTING TO PRMA RADO1 AS A LOW/
9980	061240	024104	024461	047040	
9981	061246	052117	043440	052105	
9982	061254	044524	043516	052040	
9983	061262	020117	051106	040515	
9984	061270	051040	042101	030460	
9985	061276	040440	020123	020101	
9986	061304	047514	000127		
9987	061310	051106	042510	040440	EM162: .ASCIZ /PRHE AP59(1) NOT GOING L OR NOT GETTING TO PRMA RADO1 AS A B/<CRLF>
9988	061316	032522	024071	024461	
9989	061324	047040	052117	043440	
9990	061332	044517	043516	046040	
9991	061340	047440	020122	047516	
9992	061346	020124	042507	052124	
9993	061354	047111	020107	047524	
9994	061362	043040	046522	020101	
9995	061370	040522	030104	020061	
9996	061376	051501	040440	044040	
9997	061404	200			
9998	061405	117	020122	054106	.ASCIZ /OR FXPW ACS0 DID NOT GO HIGH IN STATE 302/
9999	061412	047120	040440	051503	
10000	061420	020060	044504	020104	
10001	061426	047516	020124	047507	
10002	061434	044040	043511	020110	
10003	061442	047111	051440	040524	
10004	061450	042524	031440	031060	
10005	061456	000			
10006	061457	106	046522	020101	EM163: .ASCIZ /PRMA PTU(0) NOT GOING LOW/
10007	061464	044506	024125	024460	
10008	061472	047040	052117	043440	
10009	061500	044517	043516	046040	
10010	061506	053517	000		
10011	061511	106	050130	020120	EM164: .ASCIZ /FXPP SC09 NOT GETTING THRU SD MUX AS A LOW/
10012	061516	041523	034460	047040	
10013	061524	052117	043440	052105	
10014	061532	044524	043516	052040	
10015	061540	051110	020125	042123	
10016	061546	046440	054125	040440	
10017	061554	020123	020101	047514	

10018	061562	000127			
10019	061564	054106	050120	051440	EM165: .ASCIZ /FXPP SC09 NOT GETTING THRU SD MUX AS A HIGH/
10020	061572	030103	020071	047516	
10021	061600	020124	042507	052124	
10022	061606	047111	020107	044124	
10023	061614	052522	051440	020104	
10024	061622	052515	020130	051501	
10025	061630	040440	044040	043511	
10026	061636	000110			
10027	061640	051106	042510	040440	EM166: .ASCIZ /FRNE AR59(1)L DID NOT GO H OR FXPM ACS1 DID NOT/<CPLP>
10028	061646	032522	024071	024461	
10029	061654	020114	044504	020104	
10030	061662	047516	020124	047507	
10031	061670	044040	047440	020122	
10032	061676	054106	047120	040440	
10033	061704	051503	020061	044504	
10034	061712	020104	047516	100124	
10035	061720	047507	044040	043511	.ASCIZ /GO HIGH IN STATE 302/
10036	061726	020110	047111	051440	
10037	061734	040524	042524	031440	
10038	061742	031060	000		
10039	061745	104	052101	020101	EM167: .ASCIZ /DATA BAD, CHECK ROM FLOW OF 7F1 BRANCH/
10040	061752	040502	026104	041440	
10041	061760	042510	045503	051040	
10042	061766	046517	043040	047514	
10043	061774	020127	043117	033440	
10044	062002	030506	041040	040522	
10045	062010	041516	000110		
10046	062014	052523	020102	047111	EM170: .ASCIZ /SUB IN STATE 374 DID NOT CLEAR BN AND BZ/
10047	062022	051440	040524	042524	
10048	062030	031440	032067	042040	
10049	062036	042111	047040	052117	
10050	062044	041440	042514	051101	
10051	062052	041040	020116	047101	
10052	062060	020104	055102	000	
10053	062065	123	040524	042524	EM171: .ASCIZ /STATE 373 DID NOT CLEAR AC6/
10054	062072	031440	031467	042040	
10055	062100	042111	047040	052117	
10056	062106	041440	042514	051101	
10057	062114	040440	033103	000	
10058	062121	106	046522	020105	EM172: .ASCIZ /FRNE FCC1(1) NOT GETTING TO PRLN C(1) AS A H/
10059	062126	041506	030503	030450	
10060	062134	020051	047516	020124	
10061	062142	042507	052124	047111	
10062	062150	020107	047524	043040	
10063	062156	046122	020116	024103	
10064	062164	024461	040440	020123	
10065	062172	020101	000110		
10066	062176	052123	052101	020105	EM173: .ASCIZ /STATE 167 DID NOT CLEAR AR LOW/
10067	062204	033061	020067	044504	
10068	062212	020104	047516	020124	
10069	062220	046103	040505	020122	
10070	062226	051101	046040	053517	
10071	062234	000			
10072	062235	123	040524	042524	EM174: .ASCIZ /STATE 122 DID NOT CLEAR QUAD [3:03/
10073	062242	030440	031062	042040	

10074	062250	042111	047040	052117					
10075	062256	041440	042514	051101					
10076	062264	050440	040525	020104					
10077	062272	031533	030072	000135					
10078	062300	052123	052101	020105	EM175:	.ASCIZ	/STATE 123 FAILED/		
10079	062306	031061	020063	040506					
10080	062314	046111	042105	000					
10081	062321	106	046122	020120	EM176:	.ASCIZ	/PRLP FVINT DID NOT GO W OR NOT GETTING TO PRMA RAD01 AS A L/		
10082	062326	053106	047111	020124					
10083	062334	044504	020104	047516					
10084	062342	020124	047507	044040					
10085	062350	047440	020122	047516					
10086	062356	020124	042507	052124					
10087	062364	047111	020107	047524					
10088	062372	043040	046522	020101					
10089	062400	040522	030104	020061					
10090	062406	051501	040440	046040					
10091	062414	000							
10092	062415	123	040524	020060	EM177:	.ASCIZ	/STAO DID NOT STORE ALL QUAD'S PROPERLY/		
10093	062422	044504	020104	047516					
10094	062430	020124	052123	051117					
10095	062436	020105	046101	020114					
10096	062444	052521	042101	051447					
10097	062452	050040	047522	042520					
10098	062460	046122	000131						
10099									
10100	062464	001210	001160	000000	DT177:	.EVEN			
10101	062472	052123	030121	042040	EM200:	.WORD	\$TMP4,\$REG0,0		
10102	062500	042111	047040	052117		.ASCIZ	/STQ0 DID NOT STORE ALL QUADS PROPERLY/		
10103	062506	051440	047524	042522					
10104	062514	040440	046114	050440					
10105	062522	040525	051504	050040					
10106	062530	047522	042520	046122					
10107	062536	000131							
10108									
10109	062540	001200	001160	000000	DT200:	.EVEN			
10110	062546	052123	052101	020105	EM201:	.WORD	\$TMP0,\$REG0,0		
10111	062554	033461	020063	044504		.ASCIZ	/STATE 173 DID NOT CLEAR DST/		
10112	062562	020104	047516	020124					
10113	062570	046103	040505	020122					
10114	062576	051504	000124						
10115	062602	051101	042040	052101	EM202:	.ASCIZ	/AR DATA RAD ON SHIFT/		
10116	062610	020101	040502	020104					
10117	062616	047117	051440	044510					
10118	062624	052106	000						
10119	062627	011	004411	042011	DH202:	.ASCII	/	DATA	SRPT
10120	062634	052101	004501	004411					
10121	062642	020011	020040	051440					
10122	062650	043110	041524	100124					
10123	062656	020011	020040	054105		.ASCIZ	/	EXPECT	ACTUAL/
10124	062664	042520	052103	004411					
10125	062672	020011	020040	020040					
10126	062700	020040	041501	052524					
10127	062706	046101	000						
10128		062712							
10129	062712	001200	001210	001170	DT202:	.EVEN			
						.WORD	\$TMP0,\$TMP4,\$REG4,0		

10130	062720	000000				
10131	062722	002	002	000	DF202:	.PYTE 2,2,0
10132	062725	121	020122	040504	EM203:	.ASCIZ /QR DATA BAD ON SHIFT/
10133	062732	040524	041040	042101		
10134	062740	047440	070116	044123		
10135	062746	043111	000124			
10136						
10137	062752	001200	001160	001170	DT203:	.EVEN .WORD STMP0,\$REG0,\$REG4,0
10138	062760	000000				
10139	062762	051106	043115	051440	EM204:	.ASCIZ /FRMF SUB*SC<8 NOT GOING LOW OR NOT GETTING TO FRMA RAO2 AS A W/
10140	062770	041125	051452	036103		
10141	062776	020070	047516	020124		
10142	063004	047507	047111	020107		
10143	063012	047514	020127	051117		
10144	063020	047040	052117	043440		
10145	063026	052105	044524	043516		
10146	063034	052040	020117	051106		
10147	063042	040515	051040	042101		
10148	063050	031060	040440	020123		
10149	063056	020101	000110			
10150	063062	051106	043115	051440	EM205:	.ASCIZ /FRMF SS XOR SD XOR SUB NOT GOING LOW/
10151	063070	020123	047530	020122		
10152	063076	042123	054040	051117		
10153	063104	051440	041125	047040		
10154	063117	052117	043440	044517		
10155	063120	043516	046040	053517		
10156	063126	000				
10157	063127	106	046522	020112	EM206:	.ASCIZ /FRMJ IL(0)H NOT GOING LOW OR NOT GETTING TO FRMA RAO2 AS A W/
10158	063134	046111	030050	044051		
10159	063142	047040	052117	043440		
10160	063150	044517	043516	046040		
10161	063156	053517	047440	020122		
10162	063164	047516	020124	042507		
10163	063172	052124	047111	020107		
10164	063200	047524	043040	046522		
10165	063206	020101	040522	030104		
10166	063214	020062	051501	040440		
10167	063222	044040	000			
10168	063225	106	050130	020120	EM207:	.ASCIZ /FXPP ABS VAL ROM OUT NOT GETTING TO OUT OF RANGE GATE AS A LOW/
10169	063232	041101	020123	040526		
10170	063240	020114	047522	020115		
10171	063246	052517	020124	047516		
10172	063254	020124	042507	052124		
10173	063262	047111	020107	047524		
10174	063270	047440	052125	047440		
10175	063276	020106	040522	043516		
10176	063304	020105	040507	042524		
10177	063312	040440	020123	020101		
10178	063320	047514	000127			
10179	063324	051106	041510	043040	EM210:	.ASCIZ /FRHC FD(1)R L NOT GOING HIGH/
10180	063332	024104	024461	020102		
10181	063340	020114	047516	020124		
10182	063346	047507	047111	020107		
10183	063354	044510	044107	000		
10184	063361	106	044122	020103	EM211:	.ASCIZ /FRHC FALU59 NOT GETTING TO FRHC AND FRRL AS A LOW/
10185	063366	040506	052514	034465		

10186	063374	047040	052117	043440
10187	063402	052105	044524	043516
10188	063410	052040	020117	051106
10189	063416	045510	040440	042116
10190	063424	043040	044122	020114
10191	063432	051501	040440	046040
10192	063440	053517	000	
10193	063443	106	044122	020103
10194	063450	040506	052514	034465
10195	063456	047040	052117	043440
10196	063464	052105	044524	043516
10197	063472	052040	020117	051106
10198	063500	046110	040440	044123
10199	063506	020106	047503	052116
10200	063514	031440	024101	024460
10201	063522	040440	020123	020101
10202	063530	047514	000127	
10203	063534	020101	044502	020124
10204	063542	043117	043040	044122
10205	063550	020113	047516	046522
10206	063556	050040	051517	042440
10207	063564	041516	042117	051105
10208	063572	042040	042111	047040
10209	063600	052117	043440	020117
10210	063606	047514	020127	051117
10211	063614	100040		
10212	063616	044504	020104	047516
10213	063624	020124	042507	020124
10214	063632	044124	052522	052040
10215	063640	042510	041040	054115
10216	063646	047440	020116	054106
10217	063654	020120	051501	040440
10218	063662	046040	053517	000
10219	063667	106	044122	020103
10220	063674	040506	052514	034465
10221	063702	042040	042517	020123
10222	063710	047516	020124	042507
10223	063716	020124	047524	043040
10224	063724	044122	020113	047516
10225	063732	046522	050040	051517
10226	063740	042440	041516	042117
10227	063746	051105	040440	020123
10228	063754	020101	047514	000127
10229	063762	051106	043115	043040
10230	063770	020120	042522	020121
10231	063776	047503	047125	042524
10232	064004	020122	044504	020104
10233	064012	047516	020124	047514
10234	064020	042101	044440	020116
10235	064026	047522	020115	052123
10236	064034	052101	020105	033461
10237	064042	000066		
10238	064044	051106	046110	040440
10239	064052	044123	020106	047503
10240	064060	052116	031440	024101
10241	064066	024460	047040	052117

EM212: .ASCIZ /FRHC PAL059 NOT GETTING TO FRHL ASHF CONT 3A(0) AS A LOW/

EM213: .ASCII /A BIT OF FRHK NORM POS ENCODER DID NOT GO LOW ON /<CRLF>

.ASCIZ /DID NOT GET THRU THE BNX ON PXP AS A LOW/

EM214: .ASCIZ /FRHC PAL059 DOES NOT GET TO FRHK NORM POS ENCODER AS A LOW/

EM215: .ASCIZ /FRMF FP REQ COUNTER DID NOT LOAD IN ROM STATE 176/

EM216: .ASCIZ /FRHL ASHF CONT 3A(0) NOT GETTING TO BNX AS A H WHEN PAL059 H/

10242 064074 043440 052105 044524
 10243 064102 043516 052040 070117
 10244 064110 046502 020130 051501
 10245 064116 040440 044040 053440
 10246 064124 042510 020116 040506
 10247 064132 052514 034465 044040
 10248 064140 000
 10249 064141 101 041040 052111
 10250 064146 047440 020106 051106
 10251 064154 045510 047040 051117
 10252 064162 020115 047520 020123
 10253 064170 047105 047503 042504
 10254 064176 020122 044504 020104
 10255 064204 047516 020124 042507
 10256 064212 020124 047524 052040
 10257 064220 042510 041040 054115
 10258 064226 040440 020123 020101
 10259 064234 044510 044107 000
 10260 064241 106 050130 020112
 10261 064246 040505 052514 033060
 10262 064254 054040 051117 042440
 10263 064262 046101 030125 020063
 10264 064270 047516 020124 047507
 10265 064276 047111 020107 047514
 10266 064304 000127
 10267
 10268
 10269 064306 051105 050122 004503
 10270 064314 020040 020040 004440
 10271 064322 020040 020040 070040
 10272 064330 020040 042040 052101
 10273 064336 004501 004411 042524
 10274 064344 052123 047040 100117
 10275 064352 020011 020040 054105
 10276 064360 042520 052103 004411
 10277 064366 020040 040440 052103
 10278 064374 040525 000114
 10279
 10280 064400 001116 001200 001202
 10281 064406 001204 001206 001266
 10282 064414 000000
 10283 064416 051106 041514 043040
 10284 064424 054115 031060 042040
 10285 064432 042111 047040 052117
 10286 064440 043440 020117 044510
 10287 064446 044107 047440 020122
 10288 064454 051106 042514 040440
 10289 064462 044123 030106 020062
 10290 064470 044504 020104 047516
 10291 064476 020124 047507 044040
 10292 064504 043511 020110 044527
 10293 064512 044124 200
 10294 064515 101 030122 020063
 10295 064522 047117 040440 044040
 10296 064530 043511 020110 051117
 10297 064536 021040 051106 044115

EM217: .ASCIZ /A BIT OF FRHK NORM POS ENCODER DID NOT GET TO THE RMX AS A HIGH/

EM220: .ASCIZ /FXPJ EALU06 XOR EALU03 NOT GOING LOW/

DH221: .ASCIZ /ERRPC DATA TEST NO/<CRLF>

.ASCIZ / EXPECT ACTUAL/

DT221: .EVEN
 .WORD SEROPC,STMP0,STMP1,STMP2,STMP3,\$STSTN,0

EM222: .ASCIZ /FRLC FMX02 DID NOT GO HIGH OR FRLC ASHP02 DID NOT GO HIGH WITH/<CRLF>

.ASCIZ /AR03 ON A HIGH UP "FRMR FORCE A" DID NOT GO HIGH/

10298	064544	043040	051117	042503
10299	064552	040440	020042	044504
10300	064560	020104	047516	020124
10301	064566	047507	044040	043511
10302	064574	000110		
10303	064576	052123	052101	020105
10304	064604	030463	020063	044504
10305	064612	020104	047516	020124
10306	064620	052523	052102	040522
10307	064626	052103	040440	046440
10308	064634	047111	051525	047440
10309	064642	042516	043040	047522
10310	064650	020115	054105	047520
10311	064656	042516	052116	000
10312	064663	106	046522	020106
10313	064670	051523	054040	051117
10314	064676	051440	020104	047530
10315	064704	020122	052523	020102
10316	064712	044504	020104	047516
10317	064720	020124	047507	044040
10318	064726	043511	020110	051117
10319	064734	040440	042104	051452
10320	064742	036103	020070	044504
10321	064750	020104	047516	020124
10322	064756	047507	046040	053517
10323	064764	047440	100122	
10324	064770	044504	020104	047516
10325	064776	020124	042507	020124
10326	065004	047524	043040	046522
10327	065012	020101	040522	030104
10328	065020	020063	051501	040440
10329	065026	044040	043511	000110
10330	065034	051106	041514	043040
10331	065042	054115	031060	042040
10332	065050	042111	047040	052117
10333	065056	043440	020117	047514
10334	065064	020127	044527	044124
10335	065072	043040	024124	024461
10336	065100	044040	043511	000110
10337	065106	051106	043115	051440
10338	065114	020123	047530	020122
10339	065122	042123	054040	051117
10340	065130	051440	041125	042040
10341	065136	042111	047040	052117
10342	065144	043440	020117	044510
10343	065152	044107	000	
10344	065155	106	050130	020120
10345	065162	052517	020124	043117
10346	065170	051040	047101	042507
10347	065176	047040	052117	043440
10348	065204	044517	043516	044040
10349	065212	043511	020110	051117
10350	065220	047040	052117	043440
10351	065226	052105	044524	043516
10352	065234	200		
10353	065235	124	020117	054106

EM223: .ASCIZ /STATE 313 DID NOT SUBTRACT A MINUS ONE FROM EXPONENT/

EM224: .ASCII /PRMF SS XOR SD XOR SUB DID NOT GO HIGH OR ADD*SC<8 DID NOT GO LOW OR/<<

.ASCIZ /DID NOT GET TO PRMA RAD03 AS A HIGH/

EM225: .ASCII /FRLC FNX02 DID NOT GO LOW WITH FT(1) HIGH/

EM226: .ASCIZ /PRMF SS XOR SD XOR SUB DID NOT GO HIGH/

EM227: .ASCII /FXPP OUT OF RANGE NOT GOING HIGH OR NOT GETTING/<<CRLF>

.ASCIZ /TU FXPJ EALU SWR AS A HIGH/

10354	065242	045120	042440	046101				
10355	065250	020125	053523	020122				
10356	065256	051501	040440	044040				
10357	065264	043511	000110					
10358	065270	041101	020123	040526	EM230:	.ASCII	/ABS VAL ROM FAILED/	
10359	065276	020114	047522	020115				
10360	065304	040506	046111	042105				
10361	065312	000						
10362	065313	106	044122	020103	EM231:	.ASCII	/PRHC PMX34 DID NOT GO LOW WITH FT(1)/	
10363	065320	046506	031530	020064				
10364	065326	044504	020104	047516				
10365	065334	020124	047507	046040				
10366	065342	053517	053440	052111				
10367	065350	020110	052106	030450				
10368	065356	000051						
10369	065360	051106	045510	047040	EM232:	.ASCII	/PRHK NORM POS ENCODER FAILED/	
10370	065366	051117	020115	047520				
10371	065374	020123	047105	047503				
10372	065402	042504	020122	040506				
10373	065410	046111	042105	000				
10374	065415	105	051122	041520	DM232:	.ASCII	/ERRPC	DATA AR(59:51) TEST WC/<<CBLP>
10375	065422	004411	020040	020040				
10376	065430	004440	042040	052101				
10377	065436	004501	020011	020040				
10378	065444	020040	040440	036122				
10379	065452	034465	032472	037061				
10380	065460	052040	051505	020124				
10381	065466	047516	200					
10382	065471	011	020040	020040		.ASCII	/	EXPECT ACTUAL/
10383	065476	020040	054105	042520				
10384	065504	052103	004411	020040				
10385	065512	040440	052103	040525				
10386	065520	000114						
10387								
10388	065522	001116	001200	001204	DT232:	.EVEN		
10389	065530	001214	001266	000000		.WORD	\$ERRPC,STMP0,STMP2,STMP6,\$STSTNM,0	
10390	065536	000	001	001	DF232:	.BYTE	0,1,1,3,0	
10391	065541	003	000					
10392	065543	106	044122	020113	EM233:	.ASCII	/PRHK NORM POS SWR DID NOT GO LOW/	
10393	065550	047516	046522	050040				
10394	065556	051517	051440	051127				
10395	065564	042040	042111	047040				
10396	065572	052117	043440	020117				
10397	065600	047514	000127					
10398	065604	054106	050120	047440	EM234:	.ASCII	/FXPP OUT OF RANGE NOT GETTING TO FRMA AS A LOW OR RADO3 AS A HIGH/	
10399	065612	052125	047440	020106				
10400	065620	040522	043516	020105				
10401	065626	047516	020124	042507				
10402	065634	052124	047111	020107				
10403	065642	047524	043040	046522				
10404	065650	020101	051501	040440				
10405	065656	046040	053517	047440				
10406	065664	020122	040522	030104				
10407	065672	020063	051501	040440				
10408	065700	044040	043511	000110				
10409	065706	054106	045120	042440	EM235:	.ASCII	/FXPJ EALU06 XOR EALU04 DID NOT GO LOW/	

10410	065714	046101	030125	020066	
10411	065722	047530	020122	040505	
10412	065730	052514	032060	042040	
10413	065736	042111	047040	052117	
10414	065744	043440	020117	047514	
10415	065752	000127			
10416	065754	054106	045120	042440	EM236: .ASCIZ /FXPJ EALU06 XOR EALU05 DID NOT GO LOW/
10417	065762	046101	030125	020066	
10418	065770	047530	020122	040505	
10419	065776	052514	032460	042040	
10420	066004	042111	047040	052117	
10421	066012	043440	020117	047514	
10422	066020	000127			
10423	066022	041101	020123	040526	EM237: .ASCIZ /ABS VAL ROM NSR DID NOT GO HIGH/
10424	066030	020114	047522	020115	
10425	066036	051515	020102	044504	
10426	066044	020104	047516	020124	
10427	066052	047507	044040	043511	
10428	066060	000110			
10429	066062	051106	041510	043040	EM240: .ASCIZ /PRHC PMX34 NOT GOING LOW WITH PD(1)H/
10430	066070	054115	032063	047040	
10431	066076	052117	043440	044517	
10432	066104	043516	046040	053517	
10433	066112	053440	052111	020110	
10434	066120	042106	030450	044051	
10435	066126	000			
10436	066127	123	040524	042524	EM241: .ASCIZ /STATE 747 DID NOT ROUND/
10437	066134	031040	033464	042040	
10438	066142	042111	047040	052117	
10439	066150	051040	052517	042116	
10440	066156	000			
10441	066157	123	040524	042524	EM242: .ASCIZ /STATE 121 DID NOT NORMALIZE PROPERLY/
10442	066164	030440	030462	042040	
10443	066172	042111	047040	052117	
10444	066200	047040	051117	040515	
10445	066206	044514	042532	050040	
10446	066214	047522	042520	046122	
10447	066222	000131			
10448	066224	052123	052101	020105	EM243: .ASCIZ /STATE 167 DID NOT CLEAR AR LOW/
10449	066232	033061	020067	044504	
10450	066240	020104	047516	020124	
10451	066246	046103	040505	020122	
10452	066254	051101	046040	053517	
10453	066262	000			
10454	066263	105	052111	042510	EM244: .ASCII /EITHER FXPJ EALU08 NOT GETTING TO PRMB AS A R OR/<CRLF>
10455	066270	020122	054106	045120	
10456	066276	042440	046101	030125	
10457	066304	020070	047516	020124	
10458	066312	042507	052124	047111	
10459	066320	020107	047524	043040	
10460	066326	046522	020102	051501	
10461	066334	040440	044040	047440	
10462	066342	100122			
10463	066344	051106	040515	041040	.ASCIZ /PRMA BOU+BZ NOT GETTING TO RAD00 AS A R OR CC'S FAILED/
10464	066352	052517	041053	020132	
10465	066360	047516	020124	042507	

10466	066366	052124	047111	020107	
10467	066374	047524	051040	042101	
10468	066402	030060	040440	020123	
10469	066410	020101	020110	051117	
10470	066416	041440	023503	020123	
10471	066424	040506	046111	042105	
10472	066432	000			
10473	066433	117	042526	043122	EM245: .ASCIZ /OVERFLOW DID NOT OCCUR/
10474	066440	047514	020127	044504	
10475	066446	020104	047516	020124	
10476	066454	041517	052503	000122	
10477	066462	051106	040515	041040	EM246: .ASCIZ /PRMA BOU+8Z DID NOT GO LOW ON RZ/
10478	066470	052517	041053	020132	
10479	066476	044504	020104	047516	
10480	066504	020124	047507	046040	
10481	066512	053517	047440	020116	
10482	066520	055102	000		
10483	066523	123	040524	042524	EM247: .ASCIZ /STATE 162 DID NOT ADD 1 TO THE EXP/
10484	066530	030440	031066	042040	
10485	066536	042111	047040	052117	
10486	066544	040440	042104	030440	
10487	066552	052040	020117	044124	
10488	066560	020105	054105	000120	
10489	066566	051106	041510	043040	EM250: .ASCIZ /PRNC PMX34 NOT GOING HIGH/
10490	066574	054115	032063	047040	
10491	066602	052117	043440	044517	
10492	066610	043516	044040	043511	
10493	066616	000110			
10494	066620	051106	041514	043040	EM251: .ASCIZ /PRLC PMX19 NOT GOING LOW/
10495	066626	054115	034461	047040	
10496	066634	052117	043440	044517	
10497	066642	043516	046040	053517	
10498	066650	000			
10499	066651	104	052101	020101	EM252: .ASCIZ /DATA BAD ON STCFI*INT=-1/
10500	066656	040507	020104	047117	
10501	066664	051440	041524	044506	
10502	066672	044452	052116	026475	
10503	066700	000061			
10504	066702	051106	041514	043040	EM253: .ASCIZ /PRLC PMX19 NOT GOING HIGH/
10505	066710	054115	034461	047040	
10506	066716	052117	043440	044517	
10507	066724	043516	044040	043511	
10508	066732	000110			
10509	066734	051106	044110	050440	EM254: .ASCIZ /PRRH Q FILL DID NOT FILL PROPERLY/
10510	066742	043040	046111	020114	
10511	066750	044504	020104	047516	
10512	066756	020124	044506	046114	
10513	066764	050040	047522	042520	
10514	066772	046122	000131		
10515	066776	040506	052514	042040	EM255: .ASCIZ /FALU DID NOT TAKE LOGICAL AND PROPERLY/
10516	067004	042111	047040	052117	
10517	067012	052040	045501	020105	
10518	067020	047514	044507	040503	
10519	067026	020114	047101	020104	
10520	067034	051120	050117	051105	
10521	067042	054514	000		

10522	067045	106	044122	020110	EM256:	.ASCIZ	/PRHH Q FILL FAILED ON INTEGER LONG/				
10523	067052	020121	044506	046114							
10524	067060	043040	044501	042514							
10525	067066	020104	047117	044440							
10526	067074	052116	043505	051105							
10527	067102	046040	047117	000107							
10528	067110	040506	052514	046040	EM257:	.ASCIZ	/FALU LOGICAL AND FAILED IN LOW 24 BITS/				
10529	067116	043517	041511	046101							
10530	067124	040440	042116	043040							
10531	067132	044501	042514	020104							
10532	067140	047111	046040	053517							
10533	067146	031040	020064	044502							
10534	067154	051524	000								
10535	067157	107	040525	042122	EM260:	.ASCIZ	/GUARD BITS FAILED IN AR/				
10536	067164	041040	052111	020123							
10537	067172	040506	046111	042105							
10538	067200	044440	020116	051101							
10539	067206	000									
10540	067207	107	040525	042122	EM261:	.ASCIZ	/GUARD BITS FAILED IN QR/				
10541	067214	041040	052111	020123							
10542	067222	040506	046111	042105							
10543	067230	044440	020116	051121							
10544	067236	000									
10545	067237	101	051502	053040	EM262:	.ASCIZ	/ABS VAL ROM FAILED/				
10546	067244	046101	051040	046517							
10547	067252	043040	044501	042514							
10548	067260	000104									
10549	067262	051105	050122	004503	DM262:	.ASCII	/ERRPC	DATA	ABS VAL ROM	TEST NO/	
10550	067270	004411	040504	040524							
10551	067276	004411	040411	051502							
10552	067304	053040	046101	051040							
10553	067312	046517	052011	051505							
10554	067320	020124	047516	200							
10555	067325	011	020040	042440		.ASCIZ	/	EXPECT	ACTUAL	ADDRS	ROMOUT/
10556	067332	050130	041505	004524							
10557	067340	040411	052103	040525							
10558	067346	004514	040411	042104							
10559	067354	042522	020123	051040							
10560	067362	046517	052517	000124							
10561						.EVEN					
10562	067370	001116	001200	001204	DT262:	.WORD	\$ERRPC,\$TMP0,\$TMP7,\$REG4,\$REG5,\$\$TSTN,0				
10563	067376	001170	001172	001266							
10564	067404	000000									
10565	067406	000	001	001	DF262:	.BYTE	0,1,1,0,0,0				
10566	067411	000	000	000							
10567	067414	004411	004411	040504	DM263:	.ASCII	/	DATA		ABS	
10568	067422	040524	004411	004411							
10569	067430	020040	020040	041101							
10570	067436	020123	040526	020114							
10571	067444	047522	100115								
10572	067450	020011	020040	054105		.ASCIZ	/	EXPECT	ACTUAL	ADDR	
10573	067456	042520	052103	004411							
10574	067464	020011	020040	041501							
10575	067472	052524	046101	004411							
10576	067500	020040	020040	042101							
10577	067506	051104	051505	051040							

10578	067514	046517	052517	000124		
10579						
10580	067522	001200	001210	001170	DT263:	.EVEN
10581	067530	001172	000000			.WORD STMP0,\$TMP4,\$REG4,\$REG5,0
10582	067534	002	002	000	DP263:	.BYTE 2,2,0,0
10583	067537	000				
10584	067540	044507	020124	052123	EM264:	.ASCIZ /BIT STUCK IN QR DATA PATH/
10585	067546	041525	020113	047111		
10586	067554	050440	020122	040504		
10587	067562	040524	050040	052101		
10588	067570	000110				
10589	067572	051111	041503	042040	EM265:	.ASCIZ /IRCC DSTMO L NOT GOING LOW--CHECK FP11C JUMPER/
10590	067600	052123	030115	046040		
10591	067606	047040	052117	043440		
10592	067614	044517	043516	046040		
10593	067622	053517	026455	044103		
10594	067630	041505	020113	050106		
10595	067636	030461	020103	052512		
10596	067644	050115	051105	000		
10597	067651	106	046522	020110	EM266:	.ASCII /FRMH WRITE AC01 OR WRITE AC00 NOT GOING HIGH/<CRLF>
10598	067656	051127	052111	020105		
10599	067664	041501	030504	047440		
10600	067672	020122	051127	052111		
10601	067700	020105	041501	030104		
10602	067706	047040	052117	043440		
10603	067714	044517	043516	044040		
10604	067722	043511	100110			
10605	067726	044527	044124	043040		.ASCIZ /WITH PD(1) LOW/
10606	067734	024104	024461	046040		
10607	067742	053517	000			
10608	067745	106	050130	020103	EM267:	.ASCIZ /FXPC FCLD EN STUCK LOW/
10609	067752	041506	042114	042440		
10610	067760	020116	052123	041525		
10611	067766	020113	047514	000127		
10612	067774	051106	042110	042040	EM270:	.ASCIZ /PRHD DISABLE LOW FMX OR DISABLE ROUND FMX NOT GOING HIGH/
10613	070002	051511	041101	042514		
10614	070010	046040	053517	043040		
10615	070016	054115	047440	020122		
10616	070024	044504	040523	046102		
10617	070032	020105	047522	047125		
10618	070040	020104	046506	020130		
10619	070046	047516	020124	047507		
10620	070054	047111	020107	044510		
10621	070062	044107	000			
10622	070065	106	044122	020104	EM271:	.ASCIZ /PRHD OR PRLA DISABLE LOW QMX NOT GOING HIGH/
10623	070072	051117	043040	046122		
10624	070100	020101	044504	040523		
10625	070106	046102	020105	047514		
10626	070114	020127	046521	020130		
10627	070122	047516	020124	047507		
10628	070130	047111	020107	044510		
10629	070136	044107	000			
10630	070141	105	052111	042510	EM272:	.ASCII /EITHER PRHD OR PRLC DISABLE LOW FMX NOT GOING HIGH OR/<CRLF>
10631	070146	020122	051106	042110		
10632	070154	047440	020127	051106		
10633	070162	041514	042040	051511		

10634	070170	041101	042514	046040	
10635	070176	053517	043040	054115	
10636	070204	047040	052117	043440	
10637	070212	044517	043516	044040	
10638	070220	043511	020110	051117	
10639	070226	200			
10640	070227	106	046122	020103	.ASCIZ /PRLC FMX02 NOT GOING LOW OR PALU BAD/
10641	070234	046506	030130	020062	
10642	070242	047516	020124	047507	
10643	070250	047111	020107	047514	
10644	070256	020127	051117	043040	
10645	070264	046101	020125	040502	
10646	070272	000104			
10647	070274	044505	044124	051105	EM273: .ASCII /EITHER FRHD DISABLE FI FMX OR FMX34 NOT GOING HIGH OR/<CRLF>
10648	070302	043040	044122	020104	
10649	070310	044504	040523	046102	
10650	070316	020105	044510	043040	
10651	070324	054115	047440	020122	
10652	070332	046506	031530	020064	
10653	070340	047516	020124	047507	
10654	070346	047111	020107	044510	
10655	070354	044107	047440	100122	
10656	070362	040506	052514	041040	.ASCIZ /PALU BAD/
10657	070370	042101	000		
10658	070373	106	046122	020103	EM274: .ASCIZ /PRLC FMX02 NOT GOING LOW WITH INTEGER INC ON A LOW/
10659	070400	046506	030130	020062	
10660	070406	047516	020124	047507	
10661	070414	047111	020107	047514	
10662	070422	020127	044527	044124	
10663	070430	044440	052116	043505	
10664	070436	051105	044440	041516	
10665	070444	047440	020116	020101	
10666	070452	047514	000127		
10667	070456	047125	054105	042520	EM275: .ASCIZ /UNEXPECTED TRAP TO 4/
10668	070464	052103	042105	052040	
10669	070472	040522	020120	047524	
10670	070500	032040	000		
10671	070503	105	051122	041520	DM275: .ASCIZ /ERRPC PCOPTRP ERRREG TEST NO/
10672	070510	050011	047503	052106	
10673	070516	050122	042411	051122	
10674	070524	042522	004507	042524	
10675	070532	052123	047040	000117	
10676					.EVEN
10677	070540	001116	001200	001202	DT275: .WORD \$ERRPC,\$TMPO,\$TMP1,\$STSTMW,0
10678	070546	001266	000000		
10679	070552	047125	054105	042520	EM277: .ASCIZ /UNEXPECTED TRAP TO 114/
10680	070560	052103	042105	052040	
10681	070566	040522	020120	047524	
10682	070574	030440	032061	000	
10683	070601	105	051122	041520	DM277: .ASCIZ /ERRPC PCOPTRP ERRREG LOADRS HIADRS TEST NO/
10684	070606	050011	047503	052106	
10685	070614	050122	042411	051122	
10686	070622	042522	004507	047514	
10687	070630	042101	051522	044011	
10688	070636	040511	051104	004523	
10689	070644	042524	052123	047040	

10690	070652	000117								
10691										
10692	070654	001116	001200	001202	DT277:	.EVEN				
10693	070662	001204	001206	001266		.WORD	\$ERRPC,STMP0,STMP1,STMP2,STMP3,\$STSTNM,0			
10694	070670	000000								
10695	070672	047125	054105	042520	EM301:	.ASCIZ	/UNEXPECTED TRAP TO 244/			
10696	070700	052103	042105	052040						
10697	070706	040522	020120	047524						
10698	070714	031040	032064	000						
10699	070721	105	051122	041520	DH301:	.ASCIZ	/ERRPC PCOFTRP FEC FFA TEST NO/			
10700	070726	050011	047503	052106						
10701	070734	050122	020011	043040						
10702	070742	041505	020011	043040						
10703	070750	040505	052011	051505						
10704	070756	020124	047516	000						
10705		070764								
10706	070764	001116	001200	001202	DT301:	.EVEN				
10707	070772	001204	001266	000000		.WORD	\$ERRPC,STMP0,STMP1,STMP2,\$STSTNM,0			
10708	071000	054106	046120	042440	EM302:	.ASCIZ	/FXPL EXPA EQ 0 DID NOT GO LOW/			
10709	071006	050130	020101	050505						
10710	071014	030040	042040	042111						
10711	071022	047040	052117	043440						
10712	071030	020117	047514	000127						
10713	071036	051105	050122	004503	DH302:	.ASCIZ	/ERRPC EXPA DATA TEST NO/			
10714	071044	020040	020040	042440						
10715	071052	050130	020101	040504						
10716	071060	040524	020040	020040						
10717	071066	020040	042524	052123						
10718	071074	047040	000117							
10719										
10720	071100	001116	001200	001266	DT302:	.EVEN				
10721	071106	000000				.WORD	\$ERRPC,STMP0,\$STSTNM,0			
10722	071110	000	001	000	DF302:	.BYTE	0,1,0			
10723	071113	106	050130	020114	EM303:	.ASCIZ	/FXPL EXPR EQ 0 NOT GOING LOW/			
10724	071120	054105	041120	042440						
10725	071126	020121	020060	047516						
10726	071134	020124	047507	047111						
10727	071142	020107	047514	000127						
10728	071150	051105	050122	004503	DH303:	.ASCIZ	/ERRPC EXPB DATA TEST NO/			
10729	071156	020040	020040	042440						
10730	071164	050130	020102	040504						
10731	071172	040524	020040	020040						
10732	071200	020040	042524	052123						
10733	071206	047040	000117							

PDP 11/45/55/70...FP11-C DIAGNOSTIC PART 1 MACY11 30(1046) 23-NOV-77 14:32 PAGE 198
DEPPAA.P11 18-JAN-76 10:17 POWER DOWN AND UP ROUTINES

10734

000001

.END

ABASE = 000000	579			
ACDW1 = 000000	579			
ACDW2 = 000000	579			
ACPUOP= 000000	579	594		
ADD 024444	5200#			
ADDW0 = 000000	579			
ADDW1 = 000000	579			
ADDW10= 000000	579			
ADDW11= 000000	579			
ADDW12= 000000	579			
ADDW13= 000000	579			
ADDW14= 000000	579			
ADDW15= 000000	579			
ADDW2 = 000000	579			
ADDW3 = 000000	579			
ADDW4 = 000000	579			
ADDW5 = 000000	579			
ADDW6 = 000000	579			
ADDW7 = 000000	579			
ADDW8 = 000000	579			
ADDW9 = 000000	579			
ADD0 030772	6100#			
ADD1 024560	5187	5228#		
ADD1R0 031124	6106	6134#		
ADD2 024602	5171	5235#		
ADD3 024606	5230	5243#		
ADD4 031120	6080	6132#		
ADD5 031100	6090	6177#		
ADEVCT= 000000	579	585		
ADEVH = 000000	579			
AENV = 000000	579	590		
AENVH = 000000	579	591		
AFATAL= 000000	579	582		
ANADR1= 000000	579			
ANADR2= 000000	579			
ANADR3= 000000	579			
ANADR4= 000000	579			
ANAMS1= 000000	579			
ANAMS2= 000000	579			
ANAMS3= 000000	579			
ANAMS4= 000000	579			
AMSGAD= 000000	579	587		
AMSGLC= 000000	579	588		
AMSGTY= 000000	579	581		
AMTYP1= 000000	579			
AMTYP2= 000000	579			
AMTYP3= 000000	579			
AMTYP4= 000000	579			
APASS = 000000	579	584		
APRIOR= 000000	579			
APTCSU= 000040	8573	8679#		
APTENV= 000001	8317	8566	8635	8677#
APTSIZ= 000200	1936	8676#		
APTSPO= 000100	8568	8637	8678#	
ASMREG= 000000	579	592		
ATESTN= 000000	579	583		

AUNIT = 000000	579	586												
AUSWR = 000000	579	593												
AVECT1= 000000	579													
AVECT2= 000000	579													
BIT0 = 000001	167#	462	5452	6597	6693	6810	7765	7835	7918	7984	8051	8118		
BIT00 = 000001	157#	167												
BIT01 = 000002	156#	166												
BIT02 = 000004	155#	165												
BIT03 = 000010	154#	164												
BIT04 = 000020	153#	163												
BIT05 = 000040	152#	162												
BIT06 = 000100	151#	161												
BIT07 = 000200	150#	160												
BIT08 = 000400	149#	159	8242											
BIT09 = 001000	148#	158	8250	8327										
BIT1 = 000002	166#	463	478	2775	3886	5820	5954	6025						
BIT10 = 002000	147#	472	8305											
BIT11 = 004000	146#	473	8257											
BIT12 = 010000	145#													
BIT13 = 020000	144#	8312												
BIT14 = 040000	143#	474	7204	7219	8228									
BIT15 = 100000	142#	475	2083	2112	2133	3439	5505	5906	5909	5982	5985	6387	6465	
	6648	6973	7070	7095	7149	7175	7399	7511	7766	7770	7836	7842		
BIT2 = 000004	165#	464	479											
BIT3 = 000010	164#	465												
BIT4 = 000020	163#	466	2412											
BIT5 = 000040	162#	467												
BIT6 = 000100	161#	468	2027	2030	8488	8493								
BIT7 = 000200	160#	469	2699	2702	2922	3409	3436	5844	6888	6907	6952	7173	7154	
	7163	7911	7941	7943	7975	8007	8008	8044	8074	8075	8107	8141		
BIT8 = 000400	159#	470	7807	7824										
BIT9 = 001000	158#	471	7743	7754										
BPTVEC= 000014	174#													
B0 007432	2524#													
B1 007560	2521	2553#												
B2 007620	2567#													
CACHSP 047550	1956	9007#												
CACHVE= 000114	182#	1956*	1957*											
CONTRL= 177746	192#													
COUNT = 001160	457#	6879*	6884	6890*	6911*	6945*	6948	6950	6953*	6954*	6958	6969*		
CPSPUR 047504	1954	2109	2460	8226	8993#									
CPUERR= 177766	205#	1950	1951	8997	8998*									
CR = 000015	79#	8612	8622											
CRLF = 000200	80#	1948	8583	8622	9128	9153	9181	9195	9211	9228	9242	9258	9274	
	9302	9318	9357	9418	9769	9803	9987	10077	10119	10203	10269	10283	10312	
	10344	10374	10454	10549	10567	10597	10630	10647						
CO 010720	2800	2803#												
DATA1 007262	2462*	2469#	2477	2479	2484									
DATA2 007420	2512*	2519#	2520	2549										
DATA3 007606	2555*	2562#	2563	2592										
DDISP = 177570	73#	541	1925											
DF121 057422	1190	1196	1202	1208	1214	1232	1280	1442	1448	1466	1472	1579	1585	
	1681	1759	1765	1784	1796	1808	1814	1820	1826	9812#				
DF202 062722	1484	1490	10131#											
DF232 065536	1627	1633	10390#											
DF262 067406	1772	10565#												

DP263	067534	1778	10582#											
DP302	071110	1868	1874	10722#										
DP35	053310	878	914	920	926	950	956	974	1262	1304	1310	1370	1520	1526
		1532	1538	1544	1669	1675	1693	1699	9431#					
DH1	050244	703	709	715	771	777	733	739	745	758	764	770	808	814
		833	839	845	851	857	863	882	888	900	906	990	996	1026
		1032	1038	1050	1062	1068	1074	1098	1122	1152	1158	1164	1170	1176
		1218	1224	1242	1248	1254	1290	1314	1376	1372	1338	1344	1350	1356
		1362	1374	1380	1386	1392	1398	1404	1410	1422	1428	1434	1458	1494
		1500	1506	1512	1548	1565	1589	1595	1601	1607	1613	1619	1637	1643
		1649	1655	1661	1788	1800	1842	1854	9140#					
DH11	051047	751	894	1002	1008	1014	1020	1056	1080	1092	1104	1110	1116	1128
		1134	1140	1146	1182	1236	1266	1272	1284	1296	1320	1416	1452	1476
		1554	1560	1709	1715	1771	1733	1739	9211#					
DH121	057347	1188	1194	1200	1206	1212	1230	1278	1440	1446	1464	1470	1577	1583
		1679	1757	1763	1782	1794	1806	1812	1818	1824	9803#			
DH15	051576	776	783	789	795	930	936	942	960	966	978	984	1044	1086
		1685	1703	9274#										
DH202	062627	1482	1488	10119#										
DH21	052150	801	9318#											
DH221	064306	1571	1727	1745	1751	1830	10269#							
DH232	065415	1625	1631	10374#										
DH24	052404	820	9346#											
DH25	052470	826	9357#											
DH262	067267	1769	10549#											
DH263	067414	1776	10567#											
DH275	070503	1836	10671#											
DH277	070601	1848	10683#											
DH301	070721	1860	10699#											
DH302	071036	1866	10713#											
DH303	071150	1872	10728#											
DH34	053125	869	9408#											
DH35	053211	875	912	918	924	948	954	972	1260	1302	1308	1368	1516	1524
		1530	1536	1542	1667	1673	1691	1697	9418#					
DISPLA	001140	541#	1925*	1931*	8272*	8304*								
DISPRE	000174	498#	1931											
DSMR =	177570	72#	540	1924										
DT1	050264	704	710	716	722	728	734	740	746	759	765	771	809	815
		834	840	846	852	858	864	883	889	901	907	991	997	1027
		1033	1039	1051	1063	1069	1075	1099	1123	1153	1159	1165	1171	1177
		1219	1225	1243	1249	1255	1291	1315	1327	1333	1339	1345	1351	1357
		1363	1375	1381	1387	1393	1399	1405	1411	1423	1429	1435	1459	1495
		1501	1507	1513	1549	1566	1590	1596	1602	1608	1614	1620	1638	1644
		1650	1656	1662	1789	1801	1843	1855	9144#					
DT11	051130	753	1003	1057	1081	1093	1105	1111	1117	1129	1135	1141	1147	1183
		1237	1267	1273	1285	1297	1321	1417	1453	1477	1555	1561	1710	1716
		1722	1734	1740	9221#									
DT121	057414	1189	1195	1201	1207	1213	1231	1279	1441	1447	1578	1584	1680	1758
		1764	1783	1795	1807	1813	1819	1875	9811#					
DT15	051656	778	784	790	796	803	828	895	931	937	979	985	1009	1015
		1021	1045	1087	1686	1704	9284#							
DT177	062464	1465	10100#											
DT200	062540	1471	10109#											
DT202	062712	1483	10129#											
DT203	062752	1489	10137#											
DT221	064400	1572	1728	1746	1752	1831	10280#							

DT232	065522	1626	1632	10388#															
DT24	052434	821	870	9351#															
DT262	067370	1771	10562#																
DT263	067522	1777	105#0#																
DT275	070540	1837	10677#																
DT277	070654	1849	10692#																
DT301	070764	1861	10706#																
DT302	071100	1867	1873	10720#															
DT35	053276	877	913	919	925	949	955	973	1261	1303	1309	1369	1519	1525					
		1531	1537	1543	1668	1674	1692	1698	9429#										
DT50	054102	943	961	967	9499#														
DO	011230	2864	28#1#																
EMTVEC=	000030	177#	1908*	1909*															
EM1	050161	702	9131#																
EM10	050714	744	9195#																
EM100	055660	1085	9656#																
EM101	055712	1091	9661#																
EM102	055755	1097	9667#																
EM104	056062	1109	9679#																
EM105	056151	1115	9689#																
EM106	056210	1121	9695#																
EM107	056320	1127	9707#																
EM11	051014	750	9206#																
EM110	056353	1133	9712#																
EM111	056427	1139	9770#																
EM112	056523	1145	9731#																
EM113	056620	1151	9742#																
EM114	056670	1157	1570	9749#															
EM115	056757	1163	9759#																
EM116	057051	1169	9769#																
EM117	057206	1175	9785#																
EM12	051142	757	9223#																
EM120	057263	1181	9793#																
EM121	057321	1187	9799#																
EM123	057424	1199	9814#																
EM124	057467	1205	9820#																
EM125	057514	1211	9824#																
EM126	057525	1217	9826#																
EM127	057557	1223	9831#																
EM13	051200	763	9228#																
EM130	057660	1279	9842#																
EM131	057727	1235	9849#																
EM132	057744	1241	9852#																
EM133	060001	1247	9857#																
EM134	060035	1253	9862#																
EM135	060067	1259	9867#																
EM136	060103	1265	9870#																
EM137	060145	1271	9876#																
EM14	051320	769	9242#																
EM140	060207	1277	98#2#																
EM141	060234	1283	98#6#																
EM142	060245	1289	9888#																
EM143	060275	1295	9893#																
EM144	060331	1301	9898#																
EM145	060374	1307	9904#																
EM146	060431	1313	9909#																

EM147	060510	1319	9917#
EM15	051450	775	9258#
EM150	060535	1325	9921#
EM151	060573	1331	9927#
EM152	060631	1337	9933#
EM153	060711	1343	9942#
EM154	060753	1349	9948#
EM155	061015	1355	9954#
EM156	061046	1361	9959#
EM157	061116	1367	9966#
EM16	051670	782	9286#
EM160	061153	1373	9971#
EM161	061232	1379	9979#
EM162	061310	1385	9987#
EM163	061457	1391	10006#
EM164	061511	1397	10011#
EM165	061564	1403	10019#
EM166	061640	1409	10027#
EM167	061745	1415	10039#
EM17	051730	788	9292#
EM170	062014	1421	10046#
EM171	062065	1427	10053#
EM172	062121	1433	10058#
EM173	062176	1439	10066#
EM174	062235	1445	10072#
EM175	062300	1451	10078#
EM176	062321	1457	10081#
EM177	062415	1463	10092#
EM2	050272	788	9145#
EM20	051764	794	9297#
EM200	062472	1469	10101#
EM201	062546	1475	10110#
EM202	062607	1481	10115#
EM203	062725	1487	10132#
EM204	062762	1493	10139#
EM205	063062	1499	10150#
EM206	063127	1505	10157#
EM207	063225	1511	10168#
EM21	052020	800	9302#
EM210	063324	1517	10179#
EM211	063361	1523	10184#
EM212	063443	1529	10193#
EM213	063534	1535	10203#
EM214	063667	1541	10219#
EM215	063762	1547	10229#
EM216	064044	1553	10238#
EM217	064141	1559	10249#
EM22	052230	807	9327#
EM220	064741	1564	10260#
EM222	064416	1576	10283#
EM223	064576	1582	10303#
EM224	064663	1588	10312#
EM225	065034	1594	10330#
EM226	065106	1600	10337#
EM227	065155	1606	10344#
EM23	052301	813	9334#

EM230	065270	1612	10358#	
EM231	065313	1618	10362#	
EM232	065360	1674	10369#	
EM233	065543	1630	10392#	
EM234	065604	1636	10398#	
EM235	065706	1642	10409#	
EM236	065754	1648	10416#	
EM237	066022	1654	10423#	
EM24	052354	819	9342#	
EM240	066062	1660	10429#	
EM241	066127	1666	10436#	
EM242	066157	1672	10441#	
EM243	066224	1678	10448#	
EM244	066263	1684	10454#	
EM245	066433	1690	10473#	
EM246	066462	1696	10477#	
EM247	066523	1702	10483#	
EM25	052444	825	9353#	
EM250	066566	1708	10489#	
EM251	066620	1714	10494#	
EM252	066651	1720	10499#	
EM253	066702	1726	10504#	
EM254	066734	1732	10509#	
EM255	066776	1738	10515#	
EM256	067045	1744	10522#	
EM257	067110	1750	10528#	
EM26	052546	832	9365#	
EM260	067157	1756	10535#	
EM261	067207	1762	10540#	
EM262	067237	1768	1775	10545#
EM264	067540	1781	10584#	
EM265	067572	1787	10589#	
EM266	067651	1793	10597#	
EM267	067745	1799	10608#	
EM27	052603	838	9370#	
EM270	067774	1805	10612#	
EM271	070065	1811	10622#	
EM272	070141	1817	10630#	
EM273	070274	1823	10647#	
EM274	070373	1829	10658#	
EM275	070456	1835	1841	10667#
EM277	070552	1847	1853	10679#
EM3	050345	714	9153#	
EM30	052652	844	9377#	
EM301	070672	1859	10695#	
EM302	071000	1865	10708#	
EM303	071113	1871	10723#	
EM31	052712	850	9383#	
EM32	052761	856	9390#	
EM33	053031	862	9397#	
EM34	053071	868	9403#	
EM35	053155	874	9413#	
EM36	053314	891	9433#	
EM37	053352	887	9478#	
EM4	050421	720	9161#	
EM40	053410	893	9443#	

PLD20 = 104410	8765	90618												
PL20 = 104407	8434	8755	90608											
PNM = 000020	4668	2320	2533	2576	2890	3058	3155	3159	3235	3283	3376	3399	3629	
	3637	3686	3694	3929	3999	4050	4170	4163	4417	4425	4472	4491	4919	
	4939	4941	4971	4982	4984	5179	5209	5228	5249	5260	5311	5324	5337	
	5367	5375	5377	5409	5418	5422	5455	5459	5809	5882	5968	6033	6083	
	6109	6177	6179	6154	6265	6274	6276	6314	6598	6702	6726	6770	6787	
	6790	6922	6983	7053	7132									
FN = 000010	4658	2941	2943	4255	4257	5375	5377	5423	5425	5460	6777	6730	6787	
	6790	6834	6836	7349	7351	7690	7692							
FO = 000010	4818													
FOCE = 000002	4788													
PPPVEC= 000244	4878	1952*	1953*	2234*	2317*	2529*	2572*	2886*	3054*	3156*	3191*	3231*	3279*	
	3371*	3441*	3678*	3683*	3913*	3994*	4045*	4105*	4157*	4412*	4421*	4466*	4528*	
	4905*	4966*	5006*	5044*	5171*	5205*	5243*	5256*	5307*	5320*	5333*	5362*	5406*	
	5415*	5447*	5715*	5805*	5827*	5950*	6022*	6051*	6080*	6105*	6151*	6217*	6261*	
	6310*	6591*	6624*	6698*	6756*	6921*	6982*	7044*	7175*	8277*				
PPSPUR 047624	1952	2234	3191	3441	4528	5006	6051	6217	6624	8227	90228			
FT = 000040	4678	6702	6704	6726	6815	6834	6836							
FU = 000012	4828													
FUV = 000014	4838													
FV = 000002	4638	7489	7654											
FZ = 000004	4648	2962	2964	4982	4984	5125	5127	5547	5549	5584	5586	5589	5631	
	5633	5698	5700	6177	6179	6273	6275	6253	6274	6347	6349	7489	7618	
	7621	7657												
FO 012670	32268													
GETTIR 045256	2002	84718												
GWS = ***** U	497	1947	8173	8180	8481	9053	9054	9055	9056	9057	9058	9059	9060	
	9061													
GO 013050	32748													
HIADRS= 177742	1908	9013												
HITMIS= 177752	1948													
HT = 000011	778	8581	8622											
IOTVEC= 000020	1758	1906*	1907*											
KDPAR0= 172360	3488													
KDPAR1= 172362	3498													
KDPAR2= 172364	3508													
KDPAR3= 172366	3518													
KDPAR4= 172370	3528													
KDPAR5= 172372	3538													
KDPAR6= 172374	3548													
KDPAR7= 172376	3558													
KDPDR0= 172320	3268													
KDPDR1= 172322	3278													
KDPDR2= 172324	3288													
KDPDR3= 172326	3298													
KDPDR4= 172330	3308													
KDPDR5= 172332	3318													
KDPDR6= 172334	3328													
KDPDR7= 172336	3338													
KERSTK= 001100	638													
KIPAR0= 172340	3378													
KIPAR1= 172342	3388													
KIPAR2= 172344	3398													
KIPAR3= 172346	3408													
KIPAR4= 172350	3418													

KIPAR5= 172352	342#								
KIPAR6= 172354	343#								
KIPAR7= 172356	344#								
KIPDR0= 172300	315#								
KIPDR1= 172302	316#								
KIPDR2= 172304	317#								
KIPDR3= 172306	318#								
KIPDR4= 172310	319#								
KIPDR5= 172312	320#								
KIPDR6= 172314	321#								
KIPDR7= 172316	322#								
LF = 000012	78#	8616	8622						
LKS = 177546	74#								
LKSTAT= 177546	489#	2027*	2030*	2031	8472	8488*	8493*		
LKVEC = 000100	181#	488#	1997*	8484	8487*	8492*			
LOADRS= 177740	189#	9012							
LOOP 005254	1949	1952#	8195	9000	9002	9015	9017	9026	
LOOP1 034502	6884#	6891	6914						
LOOP2 034544	6893#	6934							
LOOP3 035052	6950#	6970							
LOOP4 035044	6948#	6956							
LOOP5 035124	6959#	6975	6994						
LSHCK 001550	675#	6012							
LSH7 001450	642#	5953	6025*	6034	6035				
LSH7CK 001540	671#	5957							
MAINT = 177750	193#								
NAPH0 = 170202	432#								
NAPH00= 170202	368#	432							
NAPH01= 170206	370#	434							
NAPH02= 170212	372#	436							
NAPH03= 170216	374#	438							
NAPH04= 170222	376#	440							
NAPH05= 170226	378#	442							
NAPH06= 170232	380#	444							
NAPH07= 170236	382#	446							
NAPH1 = 170206	434#								
NAPH10= 170242	384#								
NAPH11= 170246	386#								
NAPH12= 170252	388#								
NAPH13= 170256	390#								
NAPH14= 170262	392#								
NAPH15= 170266	394#								
NAPH16= 170272	396#								
NAPH17= 170276	398#								
NAPH2 = 170212	436#								
NAPH20= 170302	400#								
NAPH21= 170306	402#								
NAPH22= 170312	404#								
NAPH23= 170316	406#								
NAPH24= 170320	408#								
NAPH25= 170326	410#								
NAPH26= 170332	412#								
NAPH27= 170336	414#								
NAPH3 = 170216	438#								
NAPH30= 170342	416#								
NAPH31= 170346	418#								

MAPH32=	170352	420#	
MAPH33=	170356	422#	
MAPH34=	170362	424#	
MAPH35=	170366	426#	
MAPH36=	170372	428#	
MAPH37=	170376	430#	
MAPH4 =	170222	440#	
MAPH5 =	170226	442#	
MAPH6 =	170232	444#	
MAPH7 =	170236	446#	
MAPL0 =	170200	431#	
MAPL00=	170200	367#	431
MAPL01=	170204	369#	433
MAPL02=	170210	371#	435
MAPL03=	170214	373#	437
MAPL04=	170220	375#	439
MAPL05=	170224	377#	441
MAPL06=	170230	379#	443
MAPL07=	170234	381#	445
MAPL1 =	170204	433#	
MAPL10=	170240	383#	
MAPL11=	170244	385#	
MAPL12=	170250	387#	
MAPL13=	170254	389#	
MAPL14=	170260	391#	
MAPL15=	170264	393#	
MAPL16=	170270	395#	
MAPL17=	170274	397#	
MAPL2 =	170210	435#	
MAPL20=	170300	399#	
MAPL21=	170304	401#	
MAPL22=	170310	403#	
MAPL23=	170314	405#	
MAPL24=	170320	407#	
MAPL25=	170324	409#	
MAPL26=	170330	411#	
MAPL27=	170334	413#	
MAPL3 =	170214	437#	
MAPL30=	170340	415#	
MAPL31=	170344	417#	
MAPL32=	170350	419#	
MAPL33=	170354	421#	
MAPL34=	170360	423#	
MAPL35=	170364	425#	
MAPL36=	170370	427#	
MAPL37=	170374	429#	
MAPL4 =	170220	439#	
MAPL5 =	170224	441#	
MAPL6 =	170230	443#	
MAPL7 =	170234	445#	
MEMERR=	177744	191#	9011
NMR0 =	177572	216#	270
NMR1 =	177574	217#	221
NMR2 =	177576	218#	222
NMR3 =	172516	219#	223
NHVEC =	000750	184#	

SW4 = 000020	135#													
SW5 = 000040	134#													
SW6 = 000100	133#													
SW7 = 000200	132#													
SW8 = 000400	131#	5894	5972	6040	8220	8294								
SW9 = 001000	130#	2524	2567	2881	3049	3226	3274	4020	4071	5200	5800	5884	6100	
	6305	6897	6931	6964	6992	7750	7772	7820	7846	7905	7937	7969	8003	
	8038	8070	8103	8137										
SYSTID= 177764	204#													
TAPE1 = ***** U	6845													
TAPE2 = ***** U	1	6845												
TBITVE= 000014	172#													
TKVEC = 000060	179#													
TPVEC = 000064	180#													
TRAPVE= 000034	178#	1910*	1911*											
TRTVEC= 000014	173#													
TST1 005320	1991#													
TST10 007200	2390	2403	2456#											
TST100 024266	5110	5126	5168#											
TST101 024734	5170	5265	5285#											
TST102 025210	5287	5341	5357#											
TST103 025332	5359	5376	5398#											
TST104 025512	5400	5424	5444#											
TST105 025644	5446	5461	5476#											
TST106 025736	5478	5489	5500#											
TST107 026042	5502	5514	5528#											
TST11 007354	2458	2478	2508#											
TST110 026204	5530	5553	5566#											
TST111 026340	5568	5585	5600#											
TST112 026550	5602	5632	5644#											
TST113 026642	5646	5656	5668#											
TST114 027024	5670	5699	5711#											
TST115 027124	5713	5726	5741#											
TST116 027242	5743	5757	5796#											
TST117 030646	5798	6052	6074#											
TST12 007754	2510	2564	2630#											
TST120 031134	6076	6128	6148#											
TST121 031234	6150	6161	6179#											
TST122 031376	6181	6203	6214#											
TST123 031462	6216	6224	6240#											
TST124 031664	6242	6275	6293#											
TST125 032140	6295	6348	6379#											
TST126 032452	6381	6441	6458#											
TST127 032646	6460	6476	6506#											
TST13 010154	2632	2671	2681#											
TST130 032772	6508	6527	6543#											
TST131 033116	6545	6559	6588#											
TST132 033476	6590	6665	6684#											
TST133 033754	6686	6728	6753#											
TST134 034200	6755	6788	6805#											
TST135 034406	6807	6840	6869#											
TST136 035406	7007	7036#												
TST137 035740	7038	7098	7116#											
TST14 010456	2730	2742#												
TST140 036266	7118	7178	7195#											
TST141 036434	7197	7207	7230#											

TST142	037034	7232	7296	7308#
TST143	037126	7310	7320	7333#
TST144	037242	7335	7350	7362#
TST145	037360	7364	7381	7393#
TST146	037504	7395	7412	7425#
TST147	037650	7427	7450	7467#
TST15	010660	2772	2794#	
TST150	040010	7469	7490	7506#
TST151	040130	7508	7522	7538#
TST152	040224	7540	7548	7563#
TST153	040324	7565	7574	7589#
TST154	040504	7590	7619	7632#
TST155	040640	7634	7655	7672#
TST156	040760	7674	7691	7703#
TST157	041076	7705	7719	7737#
TST16	010772	2832#		
TST160	041332	7778	7793#	
TST161	041664	7854	7864#	
TST162	041764	7866	7874	7889#
TST163	042700	8011	8022#	
TST17	011334	2846	2877	2918#
TST2	005536	1994	2017	2074#
TST20	011504	2920	2942	2953#
TST21	011636	2955	2967	2989#
TST22	011736	2991	3005	3024#
TST23	012212	3076	3082	3097#
TST24	012356	3099	3125	3148#
TST25	012524	3150	3171	3189#
TST26	012626	3192	3203	3215#
TST27	013002	3217	3223	3262#
TST3	006150	2076	2167#	
TST30	013162	3264	3271	3310#
TST31	013234	3318	3330#	
TST32	013334	3332	3344	3366#
TST33	013744	3368	3442	3456#
TST34	014034	3458	3468	3480#
TST35	014100	3482	3487	3502#
TST36	014250	3504	3531	3542#
TST37	014370	3544	3561	3579#
TST4	006402	2169	2232#	
TST40	014660	3581	3658#	
TST41	015056	3660	3720#	
TST42	015262	3722	3754	3764#
TST43	015520	3805	3818#	
TST44	016042	3881	3910#	
TST45	016360	3912	3963	3983#
TST46	017156	3985	4087	4102#
TST47	017350	4104	4131	4146#
TST5	006650	2290	2312#	
TST50	017542	4148	4174	4184#
TST51	017702	4186	4206	4219#
TST52	020070	4221	4256	4270#
TST53	020250	4272	4301	4314#
TST54	020422	4316	4345	4358#
TST55	020562	4360	4383	4401#
TST56	021030	4403	4445	4463#

TST57	021320	4465	4513	4525#																
TST6	006760	2314	2347#																	
TST60	021442	4527	4543	4556#																
TST61	021602	4559	4576	4593#																
TST62	021644	4600	4616#																	
TST63	022140	4619	4669	4683#																
TST64	022252	4685	4701	4721#																
TST65	022360	4723	4737	4749#																
TST66	022472	4751	4765	4777#																
TST67	022632	4779	4798	4811#																
TST7	007024	2355	2388#																	
TST70	022734	4813	4827	4840#																
TST71	023074	4842	4861	4872#																
TST72	023140	4874	4880	4902#																
TST73	023372	4904	4940	4960#																
TST74	023562	4962	4988	5003#																
TST75	023700	5005	5074	5041#																
TST76	024036	5043	5062	5076#																
TST77	024154	5078	5091	5108#																
TYPDS =	104404	8177	8184	9057#																
TYPE =	104400	1945	8171	8178	8185	8307	8315	8479	8586	8694	8711	8713	8721	8724						
		8727	8729	8731	8758	8768	8786	8788	8790	8794	8799	8865	8941	9053#						
		9116																		
TYPOC =	104401	8702	8723	8726	8749	9054#														
TYPON =	104403	9056#																		
TYPOS =	104402	9055#																		
UDPAR0=	177660	260#																		
UDPAR1=	177662	261#																		
UDPAR2=	177664	262#																		
UDPAR3=	177666	263#																		
UDPAR4=	177670	264#																		
UDPAR5=	177672	265#																		
UDPAR6=	177674	266#																		
UDPAR7=	177676	267#																		
UDPDR0=	177620	238#																		
UDPDR1=	177622	239#																		
UDPDR2=	177624	240#																		
UDPDR3=	177626	241#																		
UDPDR4=	177630	242#																		
UDPDR5=	177632	243#																		
UDPDR6=	177634	244#																		
UDPDR7=	177636	245#																		
UIPAR0=	177640	249#																		
UIPAR1=	177642	250#																		
UIPAR2=	177644	251#																		
UIPAR3=	177646	252#																		
UIPAR4=	177650	253#																		
UIPAR5=	177652	254#																		
UIPAR6=	177654	255#																		
UIPAR7=	177656	256#																		
UIPDR0=	177600	227#																		
UIPDR1=	177602	228#																		
UIPDR2=	177604	229#																		
UIPDR3=	177606	230#																		
UIPDR4=	177610	231#																		
UIPDR5=	177612	232#																		

\$ESCAP	001222	569#	1916*	1994*	2076*	2169*	2314*	2390*	2458*	2510*	2632*	2846*	2920*	2955*
		2991*	3026*	3099*	3150*	3192*	3217*	3264*	3332*	3368*	3458*	3482*	3504*	3544*
		3581*	3660*	3722*	3912*	3985*	4104*	4148*	4186*	4221*	4272*	4316*	4360*	4403*
		4465*	4527*	4558*	4618*	4685*	4723*	4751*	4779*	4813*	4842*	4874*	4904*	4962*
		5005*	5043*	5078*	5110*	5170*	5287*	5359*	5400*	5446*	5478*	5502*	5530*	5568*
		5602*	5646*	5670*	5713*	5743*	5798*	6076*	6150*	6181*	6216*	6242*	6295*	6381*
		6460*	6508*	6545*	6590*	6686*	6755*	6807*	7038*	7118*	7197*	7232*	7310*	7335*
		7364*	7395*	7427*	7469*	7508*	7540*	7565*	7590*	7634*	7674*	7705*	7866*	8270*
		8330	8332	8339	8483	9119								
\$ETABL	001254	589#												
\$ETEND	001264	601#	1897											
\$FATAL	001236	5#2#	8663*											
\$FD	001276	609#												
\$FEA	001272	606#												
\$FEC	001270	605#												
\$FFLG	046277	8626*	8629*	8657	8666*	8674#								
\$FILLC	001154	548#	8591	8622										
\$FILLS	001153	547#	8622											
\$FL	001275	608#												
\$FLBUF	001310	611#	8361	8409										
\$FLD20	045104	8431#	9061											
\$FLT	001300	610#												
\$FL20	044616	8356#	9060											
\$FT	001274	607#												
\$GDADR	001120	533#												
\$GDDAT	001124	535#												
\$GET42	044002	8187#												
\$HD	= 000000	30												
\$HIBTS	004620	1892#												
\$ICNT	001104	526#	8261*	8262	8264*	8275								
\$ILLUP	050124	9067	9093	9121#										
\$ITEMB	001114	530#	8311*	8319	8339	8697								
\$LF	001232	573#	8339	8622										
\$LFLG	046271	8667*	8673#											
\$LPADR	001106	527#	1918*	8252*	8268*	8273	8275							
\$LPERR	001110	528#	1919*	1998*	2082*	2111*	2132*	2185*	2204*	2236*	2252*	2264*	2277*	2321*
		2393*	2461*	2511*	2554*	2639*	2687*	2717*	2804*	2812*	2847*	2923*	2936*	3077*
		3120*	3372*	3410*	3427*	3525*	3555*	3597*	3607*	3616*	3630*	3674*	3687*	3771*
		3791*	3845*	3996*	4047*	4114*	4158*	4195*	4228*	4278*	4422*	4473*	4503*	4652*
		4967*	5174*	5244*	5257*	5308*	5321*	5334*	5363*	5403*	5419*	5456*	5878*	5951*
		6023*	6155*	6187*	6260*	6331*	6388*	6466*	6511*	6548*	6623*	6689*	6759*	6916*
		6977*	6999*	7049*	7083*	7126*	7168*	7242*	7258*	7274*	7290*	7401*	7436*	7475*
		7512*	7544*	7571*	7597*	7638*	7708*	7745*	7767*	7811*	7837*	7869*	7899*	7931*
		8032*	8064*	8096*	8252	8269*	8275	8329						
\$LRSHC	001560	679#	6026											
\$MAIL	001234	580#	1893	1897	1935	1993	2075	2168	2233	2313	2348	2389	2457	2509
		2631	2682	2743	2795	2833	2919	2954	2990	3025	3098	3149	3190	3216
		3263	3311	3331	3367	3457	3481	3503	3543	3580	3659	3721	3765	3819
		3911	3984	4103	4147	4185	4220	4271	4315	4359	4402	4464	4526	4557
		4594	4617	4684	4722	4750	4778	4812	4841	4873	4903	4961	5004	5042
		5077	5109	5169	5286	5358	5399	5445	5477	5501	5529	5567	5601	5645
		5669	5712	5742	5797	6075	6149	6180	6215	6241	6294	6380	6459	6507
		6544	6589	6685	6754	6806	6870	7037	7117	7196	7231	7309	7334	7363
		7394	7426	7468	7507	7539	7564	7633	7673	7704	7739	7795	7865	7890
		8023	8267	8317	8566									
\$MBADR	004622	1893#												

\$MFLG	046270	8627*	8633	8668*	8672#									
\$MSGAD	001250	587#	8643*	8646										
\$MSGLG	001252	588#	8648*											
\$MSGTY	001234	581#	8641	8649*	8661	8665*								
\$MXCNT	044362	8265	8275#											
\$NULL	001152	546#	8593	8622										
\$NWTST=	000001	1961#	1963	2036#	2038	2149#	2151	2224#	2226	2294#	2296	2338#	2340	2358#
		2360	2423#	2425	2488#	2490	2599#	2601	2675#	2677	2733#	2735	2785#	2787
		2819#	2821	2908#	2910	2946#	2948	2981#	2983	3008#	3010	3088#	3090	3129#
		3131	3179#	3181	3207#	3209	3254#	3256	3302#	3304	3322#	3324	3351#	3353
		3446#	3448	3472#	3474	3491#	3493	3535#	3537	3565#	3567	3646#	3648	3703#
		3705	3758#	3760	3809#	3811	3893#	3895	3967#	3969	4090#	4092	4136#	4138
		4178#	4180	4210#	4212	4261#	4263	4307#	4309	4351#	4353	4389#	4391	4449#
		4451	4518#	4520	4549#	4551	4585#	4587	4605#	4607	4676#	4678	4710#	4712
		4742#	4744	4770#	4772	4804#	4806	4833#	4835	4867#	4869	4886#	4888	4948#
		4950	4996#	4998	5029#	5031	5069#	5071	5100#	5102	5131#	5133	5273#	5275
		5347#	5349	5383#	5385	5433#	5435	5469#	5471	5493#	5495	5519#	5521	5557#
		5559	5593#	5595	5636#	5638	5661#	5663	5704#	5706	5733#	5735	5763#	5765
		6057#	6059	6140#	6142	6170#	6172	6207#	6209	6229#	6231	6284#	6286	6355#
		6357	6447#	6449	6497#	6499	6531#	6533	6565#	6567	6670#	6672	6737#	6739
		6795#	6797	6846#	6848	7012#	7014	7103#	7105	7183#	7185	7225#	7227	7300#
		7302	7325#	7327	7354#	7356	7385#	7387	7417#	7419	7453#	7455	7496#	7498
		7528#	7530	7555#	7557	7580#	7582	7624#	7626	7664#	7666	7695#	7697	7724#
		7726	7783#	7785	7858#	7860	7880#	7882	8016#	8018				
\$OCNT	047134	8837*	8866*	8879#										
\$OCTVL	047466	8964	8989#											
\$OMODE	047136	8832*	8836*	8841	8844*	8855*	8881#							
\$OVER	044346	8279	8245	8253	8263	8272#								
\$PASS	001242	584#	1935*	8163*	8164*	8175	8196	8259	8276	8475				
\$PASTM	004626	1895#												
\$POWER	050132	2171	9117	9124#										
\$PWRAD	050120	9119#												
\$PWRDN	047712	1912	2077	9067#	9114									
\$PWRMG	050114	9117#												
\$PWRUP	050010	9087	9093#											
\$QUES	001230	571#	8339	8622										
\$RDCHR=	***** U	9058												
\$RDDEC=	***** U	9058												
\$RDLIN=	***** U	9058												
\$RDOCT=	***** U	9058												
\$REGAD	001156	550#												
\$REGO	001160	457	552#	3939*	3940*	3941	3954*	3955*	3956	3986*	4000	4027*	4037*	4051
		4079*	4106*	4116	4149*	4162	4191*	4197	4404*	4416	5851*	5854*	5857	5858*
		5865	5908*	5909*	5973	5984*	5985*	5997	6271*	7045*	7056	7084	7234*	7244
		7260	7277	7293	7396	7404	7431	7470*	7472	7509*	7513	7541*	7543	7566*
		7567	7591*	7595	7805*	7814	7840	7882*	7900	7911*	7915	7926	7932	7941*
		7951*	7964	7975*	7981	7992	7998	8007*	8025*	8034	8044*	8048	8060	8066
		8074*	8085*	8099	8107*	8113	8128	8133	8141*	8142	10100	10109	10137	
\$REG1	001162	458	553#	2124*	2190*	2210*	3033*	3034	3036	3080*	3081	3105*	3106	3123*
		3174	3161*	3162	3164	3198*	3199	3271*	3272	3338*	3339	3385*	3386	3389
		3412*	3413	3429*	3430	3463*	3464	3485*	3486	3510*	3511	3529*	3530	3548*
		3549	3559*	3560	3736*	3743*	3943	3958	3987*	4026*	4038*	4078*	4107*	4150*
		4192*	4405*	4598*	4599	4632*	4654	4661	4663	4774*	4775	4731	4752*	4753
		4759	4979*	5018*	5019	5122*	5532	5540	5581*	5657*	5724*	5725	5852*	5855*
		5859*	5863	5866*	5868	6250*	6272*	6478*	6522*	7046*	7235*	7510*	7592*	7678*
		7679	7681	7684	7758*	7779*	7806*	7825*	7833*	7893*	7953*	8026*	8086*	9221

\$REG2	001164	459	554*	2107*	2116	2121*	2129*	2137	2143*	2191*	2211*	3072*	3083*	3111*
		3126*	3167*	3249*	3341*	3388*	3443*	3488*	3514*	3532*	3551*	3562*	3737*	3744*
		3945	3960	3988*	4025*	4039*	4077*	4108*	4151*	4193*	4406*	4601*	4633*	4657
		4665*	4733*	4761*	4980*	5021*	5046	5123*	5542*	5544	5582*	5658*	5727*	5862*
		5863*	5864	6251*	6479*	6523*	7047*	7236*	7429*	7511*	7569*	7572	7593*	7599
		7675*	7744*	7766*	7807*	7824*	7834*	7852*	7954*	8087*	9271			
\$REG3	001166	555*	3947	3962	3989*	4024*	4040*	4076*	4109*	4152*	4194*	4407*	4728	4756
		5050	7048*	7237*	7253*	7269*	7285*	7570*	7594*	7808*	7835*	7851*	7955*	8088*
\$REG4	001170	556*	5920*	5932*	5956*	6011*	6049*	6940*	6957*	6958*	6959	6972*	6979	7089
		7517	7603	7896*	7909*	7960*	8009*	8029*	10129	10137	10562	10580		
\$REG5	001172	557*	5955*	6007	6009*	6019	6021*	6941*	6973*	7897*	7910*	7930*	7961*	7996*
		8030*	10562	10580										
\$REG6	001174	558*												
\$REG7	001176	559*												
\$RESRE	045506	8530*	9059											
\$R2A = ***** U		9060												
\$SAVRE	045450	8514*	9058											
\$SAVR6	050130	9086*	9094	9095*	9096*	9123*								
\$SCOPE	044032	1906	8213*											
\$SETUP=	000037	699*	1905	1906	1908	1910	1912	1914	1915	1916	1918	1943	8161	8334
\$STUP =	177777	699*												
\$SVLAD	044312	8237	8266*											
\$SVPC =	000204	508*	513											
\$SWR =	167777	1*	30	50	51	52	53	54	55	56	568	569	570	1915
		1916	1918	1919	1992	2075	2168	2233	2313	2348	2389	2457	2509	2631
		2682	2743	2795	2833	2919	2954	2990	3025	3098	3149	3190	3216	3263
		3311	3331	3367	3457	3481	3503	3543	3580	3659	3721	3765	3819	3911
		3984	4103	4147	4185	4220	4271	4315	4359	4402	4464	4526	4557	4594
		4617	4684	4722	4750	4778	4812	4841	4873	4903	4961	5004	5042	5077
		5109	5169	5286	5358	5399	5445	5477	5501	5529	5567	5601	5645	5669
		5712	5742	5797	6075	6149	6180	6215	6241	6294	6380	6459	6507	6544
		6589	6685	6754	6806	6870	7037	7117	7196	7231	7309	7334	7363	7394
		7426	7468	7507	7539	7564	7590	7633	7673	7704	7738	7794	7865	7890
		8023	8156	8162	8189	8195	8196	8205	8206	8207	8208	8209	8228	8240
		8242	8243	8246	8247	8248	8255	8256	8257	8269	8272	8275	8283	8284
		8285	8286	8287	8305	8312	8324	8327	8339	9120				
\$SWREG	001256	592*	1938											
\$SWRMK=	000000	56	57	8209	8210	8244								
\$TESTW	001240	583*	1993*	2075*	2168*	2233*	2313*	2348*	2389*	2457*	2509*	2631*	2682*	2743*
		2795*	2833*	2919*	2954*	2990*	3025*	3098*	3149*	3190*	3216*	3263*	3311*	3331*
		3367*	3457*	3481*	3503*	3543*	3580*	3659*	3721*	3765*	3819*	3911*	3984*	4103*
		4147*	4185*	4220*	4271*	4315*	4359*	4402*	4464*	4526*	4557*	4594*	4617*	4684*
		4722*	4750*	4778*	4812*	4841*	4873*	4903*	4961*	5004*	5042*	5077*	5109*	5169*
		5286*	5358*	5399*	5445*	5477*	5501*	5529*	5567*	5601*	5645*	5669*	5712*	5742*
		5797*	6075*	6149*	6180*	6215*	6241*	6294*	6380*	6459*	6507*	6544*	6589*	6685*
		6754*	6806*	6870*	7037*	7117*	7196*	7231*	7309*	7334*	7363*	7394*	7426*	7468*
		7507*	7539*	7564*	7633*	7673*	7704*	7739*	7795*	7865*	7890*	8023*	8267*	
\$TIMES	001220	568*	1915*	1992*	7738*	7794*	8162*	8255*	8262	8265*	8275			
\$TKB	001144	543*												
\$TKS	001142	542*												
\$TMP0	001200	560*	2091*	2245*	2260*	2272*	2286*	2353*	2354*	2416*	2482*	2540*	2541*	2547*
		2583*	2584*	2590*	2634	2640	2644	2647	2651	2654*	2655	2683*	2688	2690
		2696*	2699*	2701*	2702*	2704	2706	2714*	2718	2720	2728*	2752*	2756	2764*
		2767	2774	2801*	2805*	2806*	2807	2835*	2837	2840	2897*	2898*	2927*	2978
		2940*	2941	2968*	2994*	3032*	3043	3065*	3066*	3104*	3114	3158*	3159*	3170
		3197*	3202	3242*	3243*	3269*	3270	3290*	3291*	3316*	3317	3337*	3343	3380*

	3399*	3400	3462*	3467	3509*	3518	3586*	3587	3590	3639*	3665*	3666	3696*
	3723*	3727	3766*	3773	3793	3882*	3916	3928	3949*	3990*	4010	4031*	4041*
	4061	4084*	4085*	4110*	4153*	4187*	4222	4230	4238*	4255	4273	4280	4281
	4302*	4317	4323	4324	4346*	4361	4367	4368	4384*	4408*	4474*	4476	4487*
	4493*	4508	4529*	4531	4544*	4559*	4561	4562*	4564	4627*	4637	4660*	4668
	4670	4702	4735*	4726	4763*	4764	4782*	4785	4828*	4846*	4849	4908	4917
	4921*	4933*	4939	4976*	4982	5017*	5023	5053*	5058*	5061	5083*	5087*	5098
	5092	5119*	5125	5181*	5216*	5217*	5223*	5228*	5229	5298*	5374*	5375	5421*
	5422*	5423	5458*	5459*	5460	5480*	5482	5487*	5488	5503*	5506	5539*	5547
	5569*	5571	5575*	5584	5586	5603*	5610	5614*	5625*	5631	5671	5679	5682*
	5692*	5698	5759*	5817*	5818*	5831	5837	5838	5843*	5844*	5871	5874*	5918
	5922	5958	6085*	6116*	6117*	6123*	6127	6163*	6182	6189	6222*	6223	6247*
	6253	6268*	6274	6322*	6323*	6338*	6344*	6347	6382*	6390	6397*	6442*	6515*
	6526	6560*	6594*	6599	6600	6627	6639*	6666*	6690*	6695	6696*	6699	6723*
	6729*	6760*	6765	6766*	6767	6784*	6789*	6808	6811	6812*	6814	6824*	6837*
	6842*	6877*	6894	6901	6903*	6907*	6910*	6942*	6961	6968*	7003*	7006	7079*
	7093*	7096	7121*	7159	7176	7203*	7205	7208*	7210	7213*	7214	7218*	7238*
	7246	7254*	7262	7270*	7278	7286*	7294	7311*	7312	7321*	7340*	7346*	7349
	7369*	7413*	7432*	7473*	7491*	7524*	7551*	7577*	7620*	7651*	7656*	7689*	7698
	7706*	7742*	7746	7755*	7764*	7768	7809*	7836*	7867*	7894*	7902	7912*	7913*
	7915*	7926*	7934	7943*	7956*	7966	7976*	7977*	7981*	7992*	8000	8008*	8027*
	8035	8045*	8046*	8048*	8060*	8067	8075*	8089*	8100	8108*	8109*	8113*	8128*
	8134	8142*	8993*	9007*	9022*	9284	9351	9429	9811	10109	10129	10137	10280
	10388	10562	10580	10677	10692	10706	10720						
\$TMP1 001202	5618	2090*	2244*	2259*	2266*	2356*	2483*	2548*	2591*	2655*	2684*	2692	2697*
	2708	2715*	2722	2727*	2796*	2797	2836*	2838	2930*	2943*	2969*	2995*	3045*
	3116*	3172*	3204*	3297*	3319*	3345*	3402*	3469*	3520*	3589*	3640*	3697*	3724*
	3767*	3883*	3991*	4030*	4042*	4083*	4111*	4154*	4188*	4257*	4303*	4347*	4385*
	4409*	4475*	4488*	4494*	4505*	4530*	4545*	4560*	4563*	4639	4643*	4673*	4738*
	4766*	4783*	4829*	4847*	4934*	4941*	4984*	5025*	5059*	5063*	5088*	5094*	5127*
	5224*	5231*	5299*	5377*	5425*	5462*	5490*	5504*	5515*	5549*	5570*	5589*	5604*
	5626*	5633*	5693*	5700*	5760*	5861	5872	5875*	6129*	6225*	6255*	6276*	6349*
	6383*	6398*	6443*	6528*	6561*	6595*	6667*	6691*	6730*	6761*	6790*	6813*	6825*
	6836*	6841*	6878*	6943*	7004*	7039*	7094*	7122*	7204*	7209*	7219*	7239*	7255*
	7271*	7287*	7314	7322*	7347*	7351*	7370*	7414*	7433*	7474*	7492*	7525*	7552*
	7621*	7657*	7692*	7707*	7743*	7754*	7765*	7776*	7810*	7868*	7895*	7914*	7918*
	7928*	7957*	7978*	8028*	8047*	8051*	8061*	8090*	8110*	8997*	9011*	9024*	9284
	10280	10677	10692	10706									
\$TMP2 001204	5628	7484*	7549*	2592*	2642	2658	2668	2689*	2690	7705*	2706	2719*	2720
	2753*	2765*	2837*	2852*	2854	2856	2922*	2924	2934*	2992*	2996	3001	3377*
	3378	3381	3396	3406*	3411	3417*	3424*	3428	3435*	3436*	3725*	3768*	3775
	3782*	3788*	3795	3803*	3889*	3992*	4029*	4043*	4082*	4112*	4155*	4189*	4297*
	4372*	4376	4410*	4479*	4480	4506*	4514*	4539*	4540	4567	4577*	4784*	4788
	4799*	4823*	4824	4848*	4851	4862*	4935*	5055*	5056	5084*	5085	5183*	5184
	5190	5195	5291*	5292	5294	5451*	5453	5454	5484	5505*	5512*	5605*	5627*
	5694*	5845	5869*	6086*	6087	6091	6162*	6339*	6340	6384*	6396*	6399	6408
	6509*	6512	6546*	6549	6551	6556	6596*	6692*	6762*	6817	6823*	6826	6917*
	6978*	7009*	7040*	7095*	7123*	7158*	7163*	7174*	7240*	7256*	7272*	7288*	7317*
	7341*	7342	7377*	7378	7402	7409	7434*	7476	7484	7523*	7550*	7575*	7607*
	7613	7711*	7718	7720	7799	7815*	7816	7841*	7842	7872*	7921*	7946*	7958*
	7979*	8054*	8079*	8091*	8111*	9012*	9429	10280	10388	10562	10692	10706	
\$TMP3 001206	5638	7485*	2550*	2593*	2670	2692	2708	2722	2838*	2860	2993*	3004	3382
	3726*	3769*	3777	3781*	3789*	3797	3802*	3890*	3993*	4028*	4044*	4081*	4113*
	4156*	4190*	4378	4411*	4482	4484	4507*	4542	4578*	4800*	4826	4863*	4936*
	5186	5452*	5513	5606*	5628*	5695*	5846	5870*	6089	6342	6401	6403	6433
	6510*	6547*	6558	6597*	6628*	6693*	6697*	6763*	6828	7041*	7124*	7175*	7241*

		7257*	7273*	7289*	7344	7380	7411	7435*	7486	7615	7712	7714	7818	7844
		7873	7875	7959*	7980*	7984*	7994*	8092*	8112*	8118*	8179*	9013*	10280	10692
STMP4	001210	564#	2660	2754*	2778	2874*	2964*	2998	3728*	3734	3737	3747	3774*	3794*
		3821	3827	3831	3835	3838	3841	3844	3849*	3854*	3859*	3864*	3871*	3877*
		3925	3964*	4008*	4009*	4011	4059*	4060*	4062	4126*	4127*	4169*	4170*	4202*
		4239*	4240	4285*	4328*	4333	4433*	4434*	4442	4511*	4570*	4571	4686	4693
		4792*	4793	4855*	4856	4918	4923*	4924	5509	5607*	5611	5616*	5617	5683*
		5684	5744	5745*	5752	5905*	5906*	5911	5981*	5982*	5986	6024	6192	6195*
		6196	6332*	6334	6385*	6391	6463*	6468	6553	6641*	6642	6656	6787	6714
		6775*	6776	6874*	6876	6888*	6889	6892*	6893	6913*	6923	6938*	6952*	6960
		6971*	6984	7001*	7005	7069*	7070*	7071	7076	7099*	7142*	7148*	7149*	7150
		7179*	7249*	7265*	7281*	7297*	7365	7406	7437	7443	7481	7647*	7801	7942*
		7973*	7987*	8012*	8042*	8076*	8093*	8114*	8121*	8143*	8146*	9499	9811	10100
		10129	10580											
STMP5	001212	565#	2841*	2842	2850*	2872	2878*	2961*	2962	3729*	3749	4444	4512	4573
		4795	4858	4926	5608*	5619	5686	5746*	5754	6198	6333*	6386*	6464*	6644
		6716	6778	6875*	6939*	7002*	7073	7152	7445	7974*	8043*	8062*	8094*	8115*
		8127*	9499											
STMP6	001214	566#	2755*	2853*	2876	2878	3730*	3751	3775	3795	3846	4128	4171	4203
		4435	4575	4698	4797	4860	4928	5621	5688	5747*	5749	5756	6200	6387*
		6465*	6646	6648	6718	6780	6929*	6990*	7008*	7154	7373	7447	10388	
STMP7	001216	567#	2662	3731*	3741	3744	3753	3777	3797	3933	4130	4173	4205	4286
		4437	4695	4700	4930	5623	5690	5987	6202	6651	6711	6720	6782	7156
		7440	7449											
STN	= 000164	18	30	1961	1992#	1993	1994	2017	2036	2075#	2076	2149	2168#	2169
		2224	2233#	2290	2294	2313#	2314	2336	2348#	2355	2358	2389#	2390	2403
		2423	2457#	2458	2478	2488	2509#	2510	2564	2599	2631#	2632	2671	2675
		2682#	2730	2733	2743#	2772	2785	2795#	2819	2833#	2846	2877	2908	2919#
		2920	2942	2946	2954#	2955	2967	2981	2990#	2991	3005	3008	3025#	3026
		3082	3088	3098#	3099	3125	3129	3149#	3150	3171	3179	3190#	3192	3203
		3207	3216#	3217	3223	3254	3263#	3264	3271	3302	3311#	3318	3322	3331#
		3332	3344	3351	3367#	3368	3442	3446	3457#	3458	3468	3472	3481#	3482
		3487	3491	3503#	3504	3531	3535	3543#	3544	3561	3565	3580#	3581	3646
		3659#	3660	3703	3721#	3722	3754	3758	3765#	3805	3809	3819#	3821	3893
		3911#	3912	3963	3967	3984#	3985	4087	4090	4103#	4104	4131	4136	4147#
		4148	4174	4178	4185#	4186	4206	4210	4220#	4221	4256	4261	4271#	4272
		4301	4307	4315#	4316	4345	4351	4359#	4360	4383	4389	4402#	4403	4445
		4449	4464#	4465	4513	4518	4526#	4527	4543	4549	4557#	4558	4576	4585
		4594#	4600	4605	4617#	4618	4669	4676	4684#	4685	4701	4710	4722#	4723
		4737	4742	4750#	4751	4765	4770	4778#	4779	4798	4804	4812#	4813	4827
		4833	4841#	4842	4861	4867	4873#	4874	4880	4886	4903#	4904	4940	4948
		4961#	4962	4988	4996	5004#	5005	5024	5029	5042#	5043	5062	5069	5077#
		5078	5091	5100	5109#	5110	5126	5131	5169#	5170	5265	5273	5286#	5287
		5341	5347	5358#	5359	5376	5383	5399#	5400	5424	5433	5445#	5446	5461
		5469	5477#	5478	5489	5493	5501#	5502	5514	5519	5529#	5530	5553	5557
		5567#	5568	5585	5593	5601#	5602	5632	5636	5645#	5646	5656	5661	5669#
		5670	5699	5704	5712#	5713	5776	5733	5742#	5743	5757	5763	5797#	5798
		6052	6057	6075#	6076	6128	6140	6149#	6150	6161	6170	6180#	6181	6203
		6207	6215#	6216	6224	6229	6241#	6242	6275	6284	6294#	6295	6348	6355
		6380#	6381	6441	6447	6459#	6460	6476	6497	6507#	6508	6527	6531	6544#
		6545	6559	6565	6589#	6590	6665	6670	6685#	6686	6728	6737	6754#	6755
		6788	6795	6806#	6807	6870	6846	6870#	7007	7012	7037#	7038	7098	7103
		7117#	7118	7178	7183	7196#	7197	7207	7275	7231#	7232	7296	7300	7309#
		7310	7320	7325	7334#	7335	7350	7354	7363#	7364	7381	7385	7394#	7395
		7412	7417	7426#	7427	7450	7453	7468#	7469	7490	7496	7507#	7508	7522
		7528	7539#	7540	7548	7555	7564#	7565	7574	7580	7590#	7619	7624	7633#

	7634	7655	7664	7673#	7674	7691	7695	7704#	7705	7719	7724	7738#	7739
	7778	7783	7794#	7795	7854	7858	7865#	7866	7874	7880	7890#	8011	8016
	8023#												
\$TPB 001150	545#	8611*	8622										
\$TPPLG 001155	549#	8560	8622										
\$TPS 001146	544#	8609	8622										
\$TRAP 047646	1910	9036#											
\$TRP = 000011	9044#	9054#	9055#	9056#	9057#	9058#	9059#	9060#	9061#	9062#			
\$TRPAD 047670	9041	9052#											
\$TSTM 004624	1894#												
\$TSTM 001102	524#	8161*	8204	8214	8244	8266*	8267	8272	8276	8292	8304	8339	8693
\$TYPB= ***** U	9058												
\$TYPDS 047140	8895#	9057											
\$TYPE 045544	8560#	8654	9044	9053									
\$TYPEC 045756	8590	8597	8604	8609#	8610								
\$TYPEX 046024	8615	8617	8620#										
\$TYPOC 046736	8835#	9054											
\$TYPON 046752	8834	8837#	9056										
\$TYPOS 046712	8830#	9055											
\$UNIT 001246	586#												
\$UNITM 004630	1896#												
\$USWR 001260	593#												
\$XTSTR 044116	8231#												
\$SACO 001434	636#												
\$SAC1 001430	634#												
\$SAC2 001424	632#												
\$SAC3 001420	630#	3848											
\$SAC4 001414	628#												
\$SAC5 001410	626#	3820	3869	3885									
\$SGET4= 000000	8189#												
\$STSTW 001266	604#	8693*	8775	9144	9221	9284	9351	9479	9499	10280	10388	10562	10677
	10692	10706	10720										
\$OFILL 047135	8831*	8835*	8845	8880#									
\$40CAT= ***** U	8228	8314											
.	493#	497#	508	509#	511#	513#	521#	574	610#	611#	1881	1882#	1884#
	1886#	1903	1918	1919	2082	2111	2132	2185	2204	2236	2252	2264	2277
	2321	2393	2461	2511	2525	2554	2568	2639	2687	2717	2804	2812	2847
	2882	2973	2936	3050	3077	3086#	3120	3227	3252#	3275	3300#	3372	3410
	3427	3525	3555	3597	3607	3616	3630	3674	3687	3771	3791	3845	3996
	4047	4114	4158	4195	4228	4278	4422	4473	4503	4652	4967	5174	5201
	5244	5257	5308	5371	5374	5363	5403	5419	5456	5801	5878	5951	6023
	6054#	6101	6137#	6155	6187	6260	6306	6331	6354#	6388	6466	6511	6548
	6623	6689	6759	6999	7049	7083	7126	7168	7242	7258	7274	7290	7401
	7436	7475	7512	7544	7571	7597	7638	7708	7745	7767	7811	7837	7869
	7899	7931	8032	8064	8096	8174#	8196	8197#	8275	8276	8339	8482#	8622
	8675#	8803#	8949#	8989#	9089	9122	9498#	9810#	10128#	10705#			
.\$ASTA= ***** U	8627	8630											
.\$X = 004620	1881#	1886											

CONNEN	1#	447#
CONN1	1960#	1963
CONN1A	2036#	2038
CONN10	2487#	2490
CONN11	2595#	2601
CONN12	2674#	2677
CONN13	2732#	2735
CONN14	2784#	2787
CONN15	2818#	2821
CONN16	2907#	2910
CONN17	2945#	2948
CONN2	2148#	2151
CONN20	2980#	2983
CONN21	3007#	3010
CONN22	3087#	3090
CONN23	3128#	3131
CONN24	3178#	3181
CONN25	3206#	3209
CONN26	3253#	3256
CONN27	3301#	3304
CONN3	2293#	2296
CONN30	3321#	3324
CONN31	3350#	3353
CONN32	3445#	3448
CONN33	3471#	3474
CONN34	3490#	3493
CONN35	3534#	3537
CONN36	3564#	3567
CONN37	3645#	3648
CONN4	2223#	2226
CONN40	3702#	3705
CONN41	3757#	3760
CONN42	3808#	3811
CONN43	4209#	4212
CONN44	4260#	4263
CONN45	4306#	4309
CONN46	4350#	4353
CONN47	4448#	4451
CONN5	2337#	2340
CONN50	4517#	4520
CONN51	4548#	4551
CONN52	4584#	4587
CONN53	4604#	4607
CONN54	4675#	4678
CONN55	4709#	4712
CONN56	4741#	4744
CONN57	4769#	4772
CONN6	2358#	2360
CONN60	4803#	4806
CONN61	4832#	4835
CONN62	4866#	4869
CONN63	4947#	4950
CONN64	4995#	4998
CONN65	5028#	5031
CONN66	5068#	5071
CONN67	5099#	5102

CONM7	2423#	2425
CONM70	5130#	5133
CONM71	5272#	5275
CONM72	5346#	5349
CONM73	5382#	5385
CONM74	5432#	5435
CONM75	5468#	5471
CONM76	5492#	5495
CONM77	5518#	5521
CON100	5556#	5559
CON101	4885#	4888
CON102	5592#	5595
CON103	5635#	5638
CON104	5660#	5663
CON105	5703#	5706
CON106	3893#	3895
CON107	5762#	5765
CON110	6056#	6059
CON111	6139#	6142
CON112	6206#	6209
CON113	6228#	6231
CON114	6283#	6286
CON115	6355#	6357
CON116	6169#	6172
CON117	5732#	5735
CON120	6446#	6449
CON121	6496#	6499
CON122	6530#	6533
CON123	6564#	6567
CON124	6669#	6672
CON125	6736#	6739
CON126	6794#	6797
CON127	6845#	6848
CON130	7012#	7014
CON131	7102#	7105
CON132	7182#	7185
CON133	7299#	7302
CON134	7324#	7327
CON135	7353#	7356
CON136	7384#	7387
CON137	7416#	7419
CON140	7453#	7455
CON141	7495#	7498
CON142	7577#	7530
CON143	7554#	7557
CON144	7579#	7582
CON145	7623#	7626
CON146	7663#	7666
CON147	7694#	7697
CON150	7723#	7726
CON151	7782#	7785
CON152	7879#	7882
CON153	8015#	8018
CON43A	3967#	3969
CON45A	4089#	4092
CON45B	4135#	4138

COM45C	4177#	4180													
COM51A	4388#	4391													
COL32A	7224#	7227													
COL54A	7857#	7860													
ENDCOM	1#	447#													
ERRINS	8276#	8292													
ERROR	66#	2011	2015	2018	2020	2024	2026	2028	2096	2097	2102	2103	2126	2178	2192
	2212	2246	2248	2261	2273	2287	2291	2328	2329	2333	2357	2398	2399	2400	2407
	2409	2415	2418	2421	2473	2476	2481	2486	2542	2545	2551	2585	2588	2594	2646
	2653	2656	2666	2667	2672	2712	2731	2783	2802	2809	2814	2816	2844	2866	2869
	2875	2879	2899	2902	2904	2931	2944	2965	2975	2978	2979	3000	3003	3006	3040
	3041	3046	3067	3070	3073	3084	3110	3112	3117	3127	3166	3168	3173	3177	3201
	3205	3244	3247	3250	3292	3295	3298	3320	3342	3346	3384	3392	3393	3397	3403
	3444	3466	3470	3489	3515	3516	3521	3533	3552	3563	3592	3593	3602	3612	3621
	3641	3668	3679	3698	3738	3745	3755	3785	3806	3891	3935	3950	3965	4034	4088
	4132	4175	4207	4233	4235	4236	4237	4253	4258	4298	4304	4342	4348	4388	4386
	4439	4446	4486	4489	4495	4499	4515	4534	4536	4537	4538	4546	4569	4579	4602
	4634	4635	4641	4644	4648	4659	4666	4667	4672	4674	4697	4707	4730	4734	4739
	4758	4762	4767	4790	4801	4818	4820	4871	4822	4830	4853	4864	4879	4881	4882
	4883	4932	4937	4942	4945	4981	4985	4989	4993	5008	5014	5015	5016	5022	5026
	5052	5060	5064	5066	5089	5095	5096	5116	5117	5118	5124	5128	5189	5192	5197
	5218	5221	5225	5232	5236	5253	5267	5296	5300	5317	5330	5343	5369	5371	5372
	5373	5378	5381	5413	5426	5428	5463	5465	5486	5491	5511	5516	5543	5546	5550
	5554	5580	5583	5588	5590	5629	5634	5650	5652	5653	5654	5659	5696	5701	5720
	5721	5722	5728	5731	5751	5758	5761	5819	5822	5921	5933	5996	6006	6095	6118
	6121	6174	6130	6133	6164	6167	6194	6204	6226	6252	6256	6273	6277	6288	6300
	6301	6302	6324	6327	6345	6350	6405	6406	6412	6415	6419	6424	6428	6429	6435
	6437	6444	6472	6473	6474	6483	6487	6490	6493	6495	6520	6524	6529	6555	6562
	6620	6621	6633	6634	6635	6638	6650	6653	6660	6662	6668	6709	6713	6722	6724
	6731	6734	6785	6791	6793	6819	6830	6831	6838	6843	6930	6991	7010	7062	7063
	7078	7080	7091	7100	7141	7143	7162	7164	7180	7217	7220	7221	7250	7266	7282
	7298	7316	7323	7348	7352	7375	7382	7383	7408	7415	7442	7451	7452	7483	7488
	7493	7519	7526	7553	7578	7606	7611	7617	7622	7643	7644	7645	7652	7658	7662
	7683	7686	7687	7693	7716	7717	7722	7759	7780	7803	7828	7855	7877	7878	7922
	7947	7988	8013	8055	8080	8122	8147	8999	9001	9014	9016	9025			
ESCAPE	1#	447#	1994	2076	2169	2314	2390	2458	2510	2632	2845	2920	2955	2991	3026
	3099	3150	3192	3217	3264	3332	3368	3458	3482	3504	3544	3581	3660	3722	3912
	3985	4104	4148	4186	4221	4272	4316	4360	4403	4465	4527	4558	4618	4685	4723
	4751	4779	4813	4842	4874	4904	4962	5005	5043	5078	5110	5170	5287	5359	5400
	5446	5478	5502	5530	5568	5602	5646	5670	5713	5743	5798	6076	6150	6181	6216
	6242	6295	6381	6460	6508	6545	6590	6686	6755	6807	7038	7118	7197	7232	7310
	7335	7364	7395	7427	7469	7508	7540	7565	7590	7634	7674	7705	7866		
FLOTST	5#	2524	2567	2881	3049	3226	3274	5200	5800	6100	6304				
FPDWH	9062#	9075													
FPUP	9062#	9098													
GETPRI	1#	447#													
MULT	1#	447#													
MYTAGS	5#	603													
NEWTST	1#	447#	7580												
POP	1#	447#	8535	8669	8670	8936	9108								
PUSH	1#	447#	8515	8630	8632	8653	8895	9069							
REPORT	1#	447#													
SCOINS	8198#	8214													
SCOPE	67#	1991	2074	2167	2232	2312	2347	2388	2456	2508	2630	2681	2742	2794	2832
	2918	2953	2989	3024	3097	3148	3189	3215	3262	3310	3330	3366	3456	3480	3502
	3542	3579	3658	3720	3764	3818	3910	3983	4102	4146	4184	4219	4270	4314	4358

K5

	4401	4463	4525	4556	4593	4616	4683	4721	4749	4777	4811	4840	4872	4902	4960
	5003	5041	5076	5108	5168	5285	5357	5398	5444	5476	5500	5528	5566	5600	5644
	5668	5711	5741	5796	6074	6148	6179	6214	6240	6293	6379	6458	6506	6543	6588
	6684	6753	6805	6869	7036	7116	7195	7230	7308	7333	7362	7393	7425	7467	7506
SETLOO	5#	2082	2111	2132	2185	2203	2236	2251	2263	2276	2321	2393	2461	2511	2553
	2639	2687	2717	2803	2811	2847	2923	2936	3076	3119	3372	3410	3427	3524	3554
	3597	3607	3616	3630	3674	3687	3771	3791	3845	3996	4047	4114	4158	4195	4228
	4278	4422	4473	4502	4651	4967	5174	5244	5257	5308	5321	5334	5363	5403	5419
	5456	5878	5951	6023	6155	6187	6259	6331	6388	6466	6511	6548	6622	6689	6759
	6998	7049	7082	7126	7167	7242	7258	7274	7290	7401	7436	7475	7512	7544	7571
	7597	7638	7708	7745	7767	7811	7837	7869	7899	7931	8032	8064	8096		
SETPRT	1#	447#													
SETTRA	9044#	9054	9055	9056	9057	9058	9059	9060	9061						
SETUP	1#	447#	1898												
SKIP	1#	447#	2017	2290	2355	2403	2478	2564	2671	2730	2772	2877	2942	2967	3005
	3082	3125	3171	3203	3223	3271	3318	3344	3442	3468	3487	3531	3561	3754	3805
	3881	3963	4087	4131	4174	4206	4256	4301	4345	4383	4445	4513	4543	4576	4600
	4669	4701	4737	4765	4798	4827	4861	4880	4940	4988	5024	5062	5091	5126	5265
	5341	5376	5424	5461	5489	5514	5553	5585	5632	5656	5699	5726	5757	6052	6128
	6161	6203	6224	6275	6348	6441	6476	6527	6559	6665	6728	6788	6840	7007	7098
	7178	7207	7296	7320	7350	7381	7412	7450	7490	7522	7548	7574	7619	7655	7691
	7719	7778	7854	7874	8011	8145									
SLASH	1#	447#													
SPACE	447#														
STARS	1#	447#	506	517	574	578	1878	1880	1887	1961	1990	2036	2073	2149	2166
	2224	2231	2294	2311	2338	2346	2358	2387	2473	2455	2488	2507	2522	2552	2565
	2599	2629	2675	2680	2733	2741	2785	2793	2819	2831	2870	2908	2917	2946	2952
	2981	2988	3008	3023	3047	3074	3088	3096	3129	3147	3179	3188	3207	3214	3224
	3254	3261	3272	3302	3309	3322	3329	3351	3365	3404	3422	3446	3455	3472	3479
	3491	3501	3535	3541	3565	3578	3594	3603	3613	3622	3646	3657	3671	3680	3703
	3719	3758	3763	3786	3809	3817	3893	3909	3967	3982	4035	4090	4101	4136	4145
	4178	4183	4210	4218	4261	4269	4307	4313	4351	4357	4389	4400	4449	4462	4500
	4518	4524	4549	4555	4585	4592	4605	4615	4619	4649	4676	4682	4710	4720	4742
	4748	4770	4776	4804	4810	4833	4839	4867	4871	4886	4901	4948	4959	4996	5002
	5029	5040	5069	5075	5100	5107	5131	5167	5193	5226	5233	5237	5254	5273	5284
	5301	5318	5331	5347	5356	5383	5397	5433	5443	5469	5475	5493	5499	5519	5527
	5557	5565	5593	5599	5636	5643	5661	5667	5704	5710	5733	5740	5763	5795	5824
	5948	6057	6073	6096	6125	6140	6147	6170	6178	6207	6213	6229	6239	6257	6284
	6292	6329	6355	6378	6447	6457	6497	6505	6531	6542	6565	6587	6613	6670	6683
	6737	6752	6795	6804	6846	6868	6936	6996	7012	7035	7065	7103	7115	7144	7183
	7194	7225	7229	7251	7267	7263	7300	7307	7325	7332	7354	7361	7385	7392	7417
	7474	7453	7466	7496	7505	7528	7537	7555	7562	7580	7588	7624	7631	7664	7671
	7695	7702	7724	7736	7761	7783	7792	7830	7858	7863	7880	7888	7924	7949	7990
	8016	8021	8057	8082	8124	8152	8201	8279	8339	8355	8412	8430	8465	8470	8499
	8545	8625	8680	8807	8885	8953	8990	9004	9019	9030	9065	9091			
SWRSU	1#	447#	1920#												
TRMTRP	9044#														
TYPBIN	1#	447#													
TYPDEC	1#	447#	8175	8182											
TYPNAM	1#	447#	1940												
TYPNUM	1#	447#													
TYPOCS	1#	447#													
TYPOCT	1#	447#	8700	8747											
TYPTXT	1#	447#	8171	8178	8479										
SMSN	5#	4429	5811	5901	5977	6045	6926	6987							

<MENTS	2#	1961	2036	2149	2274	2294	2338	2358	2423	2488	2599	2675	2733	2785	2819
	2908	2946	2981	3008	3088	3129	3179	3207	3254	3302	3322	3351	3446	3472	3491
	3535	3565	3646	3703	3758	3809	3893	3967	4090	4136	4178	4210	4261	4307	4351
	4389	4449	4519	4549	4585	4605	4676	4710	4742	4770	4804	4833	4867	4886	4948
	4996	5029	5069	5100	5131	5273	5347	5383	5433	5469	5493	5519	5557	5593	5636
	5661	5704	5733	5763	6057	6140	6170	6207	6229	6284	6355	6447	6497	6531	6565
	6670	6737	6795	6846	7012	7103	7183	7225	7300	7325	7354	7385	7417	7453	7496
	7528	7555	7624	7664	7695	7724	7783	7858	7880	8016					
\$\$CMRE	514#	552	553	554	555	556	557	558	559						
\$\$CMTM	514#	560	561	562	563	564	565	566	567						
\$\$ESCA	1#	447#	1994	2076	2169	2314	2390	2458	2510	2632	2846	2920	2955	2991	3026
	3099	3150	3192	3217	3264	3332	3368	3458	3482	3504	3544	3581	3660	3722	3912
	3985	4104	4148	4186	4221	4272	4316	4360	4403	4465	4527	4558	4618	4685	4723
	4751	4779	4813	4842	4874	4904	4962	5005	5043	5078	5110	5170	5287	5359	5400
	5446	5478	5502	5530	5568	5602	5646	5670	5713	5743	5798	6076	6150	6181	6216
	6242	6295	6381	6460	6508	6545	6590	6686	6755	6807	7038	7118	7197	7232	7310
	7335	7364	7395	7427	7469	7508	7540	7565	7590	7634	7674	7705	7866		
\$\$NEWT	1#	4#	447#	1961	2036	2149	2224	2294	2338	2358	2423	2488	2599	2675	2733
	2785	2819	2908	2946	2981	3008	3088	3129	3179	3207	3254	3302	3322	3351	3446
	3472	3491	3535	3565	3646	3703	3758	3809	3893	3967	4090	4136	4178	4210	4261
	4307	4351	4389	4449	4518	4549	4585	4605	4676	4710	4742	4770	4804	4833	4867
	4886	4948	4996	5029	5069	5100	5131	5273	5347	5383	5433	5469	5493	5519	5557
	5593	5636	5661	5704	5733	5763	6057	6140	6170	6207	6229	6284	6355	6447	6497
	6531	6565	6670	6737	6795	6846	7012	7103	7183	7225	7300	7325	7354	7385	7417
	7453	7496	7528	7555	7580	7624	7664	7695	7724	7783	7858	7880	8016		
\$\$SET	9044#	9054	9055	9056	9057	9058	9059	9060	9061						
\$\$SFIP	1#	447#	2017	2290	2355	2403	2478	2564	2671	2730	2772	2877	2942	2967	3005
	3082	3125	3171	3203	3223	3271	3318	3344	3442	3468	3487	3531	3561	3754	3805
	3881	3963	4087	4131	4174	4206	4256	4301	4345	4383	4445	4513	4543	4576	4600
	4669	4701	4737	4765	4798	4827	4861	4880	4940	4988	5024	5062	5091	5126	5265
	5341	5376	5424	5461	5489	5514	5553	5585	5632	5656	5699	5726	5757	6052	6128
	6161	6203	6224	6275	6348	6441	6476	6577	6559	6665	6728	6788	6840	7007	7098
	7178	7207	7296	7320	7350	7381	7412	7450	7490	7522	7548	7574	7619	7655	7691
	7719	7778	7854	7874	8011										
•EQUAT	1#														
•FP11C	5#	447													
•HEADE	1#	20													
•KT11	1#														
•SETUP	1#	699													
•SWRHI	1#	45													
•SWRLO	57#														
•SACT1	1#	503													
•SAPT8	1#	575#													
•SAPTH	1#	1875													
•SAPTY	1#	8622													
•SASTA	1#														
•SCATC	1#	490													
•SCMTA	1#	514													
•SDB2D	1#														
•SDB2O	1#	8950													
•SDIV	1#														
•SEOP	1#	8149													
•SERRO	1#	8276													
•SERRT	1#	5#	8680												
•SHULT	1#														
•SPOWE	1#	2#	9062												

MS

.SRAND	1#	
.SRDDE	1#	
.SRDOC	1#	
.SREAD	1#	
.SR2A7	1#	
.SSAVE	1#	8496
.\$SR2D	1#	
.\$SB20	1#	
.\$SCOP	1#	8198
.\$SIZE	1#	
.\$SUPR	1#	
.STRAP	1#	9027
.STYPR	1#	
.STYPD	1#	8882
.STYPE	1#	8542
.STYPO	1#	8804
.\$40CA	1#	
.1170	1#	58

. ABS. 071212 000

ERRORS DETECTED: 0

,DEFPAA.LST/CRF/SOL=DEFPAA.SML,DEFPAA.P11
RUN-TIME: 28 39 3 SECONDS
RUN-TIME RATIO: 810/71=11.3
CORE USED: 45K (89 PAGES)