

KD11-K

FLOAT PT UN INST EXE
MD-11-DQFPC-B

EP-DQFPC-B-DL-A
COPYRIGHT 1977
FICHE 1 OF 1

JUN 1977
digital
MADE IN USA

[Frame 1: Data Table]	[Frame 2: Data Table]	[Frame 3: Data Table]	[Frame 4: Data Table]	[Frame 5: Data Table]	[Frame 6: Data Table]	[Frame 7: Data Table]	[Frame 8: Data Table]	[Frame 9: Data Table]	[Frame 10: Data Table]
[Frame 11: Data Table]	[Frame 12: Data Table]	[Frame 13: Data Table]	[Frame 14: Data Table]	[Frame 15: Data Table]	[Frame 16: Data Table]	[Frame 17: Data Table]	[Frame 18: Data Table]	[Frame 19: Data Table]	[Frame 20: Data Table]
[Frame 21: Data Table]	[Frame 22: Data Table]	[Frame 23: Data Table]	[Frame 24: Data Table]	[Frame 25: Data Table]	[Frame 26: Data Table]	[Frame 27: Data Table]	[Frame 28: Data Table]	[Frame 29: Data Table]	[Frame 30: Data Table]
[Frame 31: Data Table]	[Frame 32: Data Table]	[Frame 33: Data Table]	[Frame 34: Data Table]	[Frame 35: Data Table]	[Frame 36: Data Table]	[Frame 37: Data Table]	[Frame 38: Data Table]	[Frame 39: Data Table]	[Frame 40: Data Table]
[Frame 41: Data Table]	[Frame 42: Data Table]	[Frame 43: Data Table]	[Frame 44: Data Table]	[Frame 45: Data Table]	[Frame 46: Data Table]	[Frame 47: Data Table]	[Frame 48: Data Table]	[Frame 49: Data Table]	[Frame 50: Data Table]
[Frame 51: Data Table]	[Frame 52: Data Table]	[Frame 53: Data Table]	[Frame 54: Data Table]	[Frame 55: Data Table]	[Frame 56: Data Table]	[Frame 57: Data Table]	[Frame 58: Data Table]	[Frame 59: Data Table]	[Frame 60: Data Table]
[Frame 61: Data Table]	[Frame 62: Data Table]	[Frame 63: Data Table]	[Frame 64: Data Table]	[Frame 65: Data Table]	[Frame 66: Data Table]	[Frame 67: Data Table]	[Frame 68: Data Table]	[Frame 69: Data Table]	[Frame 70: Data Table]
[Frame 71: Data Table]	[Frame 72: Data Table]	[Frame 73: Data Table]	[Frame 74: Data Table]	[Frame 75: Data Table]	[Frame 76: Data Table]	[Frame 77: Data Table]	[Frame 78: Data Table]	[Frame 79: Data Table]	[Frame 80: Data Table]
[Frame 81: Data Table]	[Frame 82: Data Table]	[Frame 83: Data Table]	[Frame 84: Data Table]	[Frame 85: Data Table]	[Frame 86: Data Table]	[Frame 87: Data Table]	[Frame 88: Data Table]	[Frame 89: Data Table]	[Frame 90: Data Table]
[Frame 91: Data Table]	[Frame 92: Data Table]	[Frame 93: Data Table]	[Frame 94: Data Table]	[Frame 95: Data Table]	[Frame 96: Data Table]	[Frame 97: Data Table]	[Frame 98: Data Table]	[Frame 99: Data Table]	[Frame 100: Data Table]

[Small data table in bottom right corner]

B01

EOF10QFPBSEQ
POP10 411

00010000

770608

POP10 411

S HDR10QFPCBSEQ

00010000

770608

I D E N T I F I C A T I O N

PRODUCT CODE: MAINDEC-11-DQFPC-B-D
PRODUCT NAME: PDP-11/6X - FP11-E FLOATING POINT UNIT
 INSTRUCTION EXERCISER
DATE : MAY, 1977
MAINTAINER: DIAGNOSTIC GROUP
AUTHOR: BOB BRAIN / STANLEY HARACKIEWICZ
REVISED BY: DON NORTH

COPYRIGHT (C) 1977
DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS

THIS SOFTWARE 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.

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

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

CONTENTS

1. ABSTRACT
2. REQUIREMENTS
 - 2.1 EQUIPMENT
 - 2.2 STORAGE
 - 2.3 PRELIMINARY PROGRAMS
3. LOADING PROCEDURE
4. STARTING PROCEDURE
 - 4.1 CONTROL SWITCH SETTINGS
 - 4.2 STARTING ADDRESS
 - 4.3 PROGRAM/OPERATOR ACTION
5. OPERATING PROCEDURE
 - 5.1 OPERATIONAL SWITCH SETTINGS
 - 5.2 PROGRAM/OPERATOR ACTION
 - 5.3 HOT (FP11-E) / WARM (PDP-11/6X) SELECTION
6. ERRORS
 - 6.1.1 ERROR MESSAGE FORMAT
 - 6.1.2 FLOATING POINT DATA FORMAT
 - 6.2 RECOVERY
 - 6.3 CAUSES
7. RESTRICTIONS
 - 7.1 STARTING
 - 7.2 OPERATIONAL
8. MISCELLANEOUS
 - 8.1 EXECUTION TIME
 - 8.2 STACK POINTER
 - 8.3 POWER FAIL
9. PROGRAM DESCRIPTION
 - 9.1 ORGANIZATION
 - 9.2 TEST DESCRIPTION
 - 9.3 SUBROUTINE ABSTRACTS
10. ACT/APT/XXDP

1. ABSTRACT

THIS PROGRAM CONTAINS THE PDP-11/6X FLOATING POINT PROCESSOR INSTRUCTION EXERCISER. THIS EXERCISER PROGRAM TESTS THE COMPLETE FLOATING POINT INSTRUCTION SET IN AN EXERCISER ENVIRONMENT, USING VARIOUS COMBINATIONS OF INSTRUCTIONS IN COMMON SOFTWARE CONFIGURATIONS AS THE BASIS FOR THE TESTS. EVERY CONCEIVABLE FLOATING POINT ERROR CONDITION IS DEVELOPED, AND THE RESPONSE CHECKED FOR CORRECTNESS. IN ADDITION, INTERACTION BETWEEN FLOATING POINT INSTRUCTION EXECUTION, INTERRUPTS, AND BASE MACHINE INTERRUPTS (USING THE DL11-W LINE AND/OR KH11-P PROGRAMMABLE CLOCKS) IS ALSO TESTED TO INSURE CORRECT PROCESSING BY BOTH THE BASE MACHINE AND FLOATING POINT MICROCODE. BOTH "HOT" (FP11-E OPTION) AND "WARM" (PDP-11/6X MICROCODE) FLOATING POINT UNITS CAN BE SELECTED FOR TESTING.

2. REQUIREMENTS

2.1 EQUIPMENT

PDP-11/6X STANDARD COMPUTER WITH MINIMUM 16K OF MEMORY. OPTIONAL FP11-E FLOATING POINT UNIT, IF SELECTED. THE DL11-W LINE CLOCK IS USED TO PROVIDE I/O INTERRUPTS DURING EXECUTION. IF PRESENT, THE KH11-P PROGRAMMABLE CLOCK WILL ALSO BE UTILIZED.

2.2 STORAGE

THE PROGRAM USES MEMORY 0-20772(8). THE UPPER 2.0K WORDS ARE RESERVED FOR THE XXDP MONITOR, IF EMPLOYED.

2.3 PRELIMINARY PROGRAMS

THE CPU, CACHE, AND MEMORY TEST PROGRAMS MUST BE RUN FIRST TO VERIFY THE CORRECT OPERATION OF THE BASE MACHINE.

THE PDP-11/6X - FP11-E FLOATING POINT PROCESSOR INSTRUCTION SET TESTS SHOULD THEN BE RUN IN THE FOLLOWING ORDER:

- (1) DQFPA FPU BASIC INSTRUCTION TESTS
- (2) DQFPB FPU ADVANCED INSTRUCTION TESTS
- (3) DQFPC FPU INSTRUCTION EXERCISER
- (4) DQFPD FPU ADD/SUB/MUL/DIV RANDOM EXERCISER

3. LOADING PROCEDURE

USE THE STANDARD PROCEDURE FOR ABSOLUTE TAPES, OR LOAD VIA XXDP MEDIA.

4. STARTING PROCEDURE

4.1 CONTROL SWITCH SETTINGS

SEE SECTION 5.1
SWITCH REGISTER (000000) IS WORST CASE TEST.

4.2 STARTING ADDRESS

THE PROGRAM MUST ALWAYS BE STARTED AT LOCATION 200(8).

4.3 PROGRAM/OPERATOR ACTION

LOADING VIA ABSOLUTE PAPERTAPE:

- (1) LOAD PROGRAM INTO MEMORY USING ABS LOADER.
- (2) LOAD ADDRESS 200 (8).
- (3) SET SWITCHES (SEE SECTION 5.1)
SR=(000000) IS WORST CASE TEST.
- (4) PRESS CONTROL/START TO BEGIN.
- (5) PROGRAM TYPES IDENTIFICATION HEADER (VERIFY THAT THE CORRECT PROGRAM HAS BEEN LOADED!), AND EXECUTION BEGINS.

5. OPERATING PROCEDURE

5.1 OPERATIONAL SWITCH SETTINGS

THE DEFINITION OF THE SPECIFIC BITS IN THE SWITCH REGISTER (EITHER HARDWARE OR SOFTWARE) ARE AS FOLLOWS:

SW15=1	100000	HALT ON ERROR
SW14=1	040000	LOOP ON CURRENTLY EXECUTING TEST
SW13=1	020000	INHIBIT ERROR TYPEOUTS (WHICH IS AN "ERROR MESSAGE" RESULTING FROM AN ERROR DETECTED IN THE HARDWARE)
SW12=1	010000	INHIBIT STATUS TYPEOUTS (WHICH IS A NON-ERROR RELATED INFORMATIVE MESSAGE, SUCH AS "END PASS #XXX")
SW11=1	004000	INHIBIT ITERATIONS PER TEST
SW10	002000	SET=BELL ON ERROR/CLEAR=BELL ON PASS END
SW09=1	001000	LOOP ON ERROR
SW08=1	000400	LOOP ON TEST NUMBER IN "SLPTST" IF SET, THEN THE TEST SPECIFIED BY THE TEST NUMBER CONTAINED IN THE MEMORY WORD "SLPTST" (SEE PROGRAM LISTING) WILL SPECIFY THE DESIRED TEST ON WHICH TO LOOP.
SW01	000002	CLEAR=TEST HOT-FP/WARM-FP ALTERNATELY EACH PASS (IE, PASS#1 HFP, PASS#1 WFP, PASS#2 HFP, PASS#2 WFP, ETC)
SW00	000001	SET=TEST ONLY UNIT SPECIFIED IN SW00 SET=SELECT WARM FP, IF SW01=1

CLEAR=SELECT HOT FP, IF SW01=1

NOTE FOR SW01, SW00 - IF NO HOT FP (FP11-E) IS PRESENT, THEN WARM FP (PDP-11/6X MICROCODE) IS AUTOMATICALLY SELECTED.

5.2 PROGRAM/OPERATOR ACTION

ONCE EXECUTION HAS BEGUN, MINIMAL OPERATOR INTERVENTION IS REQUIRED, UNLESS THE PROGRAM DETECTS AN ERROR IN THE HARDWARE.

IF ALL IS WELL, THE PROGRAM TYPES ITS NAME UPON BEGINNING, AND AT THE START OF EACH PASS, THE CURRENT PASS NUMBER (IN OCTAL) IS ECHOED. NOTE THAT SETTING SW<12>=1 WILL INHIBIT THE TYPEOUT OF THE BEGIN AND END PASS MESSAGES.

IF SW<10>=0, THE CONSOLE BELL WILL BE RUNG AT THE END OF EACH PASS. NOTE THAT ONLY SW<10> AFFECTS THE BELL RINGING AT END OF PASS - SW<12> HAS NO EFFECT ON THIS FUNCTION.

IF AN ERROR OCCURS DURING EXECUTION, MANY VARIATIONS IN ACTION ARE POSSIBLE DEPENDING UPON THE SWITCH SETTINGS.

SW<15>=1 WILL CAUSE THE CPU TO HALT AFTER AN ERROR.

SW<13>=1 WILL ALSO INHIBIT ANY ERROR MESSAGE TYPEOUT THAT WOULD OCCUR AT THIS TIME.

SW<10>=1 WILL CAUSE THE CONSOLE BELL TO BE RUNG ONLY WHEN AN ERROR IS DETECTED (AND NOT AT THE END OF A PASS).

SW<9>=1 CAUSES THE PROGRAM TO LOOP ON THE MOST RECENT ERROR, AS LONG AS IT CONTINUES TO OCCUR.

THERE ARE ALSO SEVERAL OTHER GENERAL USE FUNCTIONS DEFINED BY THE SWITCHES:

SW<11>=1 WILL INHIBIT THE ITERATIONS (=2000(10)) PERFORMED OF EACH TEST ON PASSES 2, 3, 4, ... THRU THE PROGRAM.

SW<14>=1 CAUSES THE PROGRAM TO LOOP INDEFINATELY ON THE CURRENTLY EXECUTING TEST.

SW<8>=1 CAUSES THE PROGRAM TO CONTINUE EXECUTION AS NORMAL, EXCEPT WHEN THE CONTENTS OF MEMORY WORD "SLPTST" MATCHES THE NUMBER OF THE TEST CURRENTLY EXECUTING. AT THIS POINT, THE TEST IS LOOPED ON INDEFINATELY, UNTIL EITHER SW<8>=0 OR "SLPTST" IS CHANGED. NOTE THAT IF "SLPTST" DOES NOT MATCH THE TEST NUMBER OF ANY TEST, THE CONTENTS OF "SLPTST" ARE EFFECTIVELY IGNORED, AND EXECUTION PROCEEDS NORMALLY.

5.3 HOT (FP11-E) / WARM (PDP-11/6X) SELECTION

WHEN THE PROGRAM IS STARTED (AT 200(8)), A MESSAGE IS OPTIONALLY PRINTED INDICATING THE PRESCENCE/ABSCENCE OF AN FP11-E HOT FLOATING POINT UNIT OPTION (BASED UPON WHETHER "WAMI" BIT<04> IS 1/0 RESPECTIVELY).

IF NO FP11-E HOT FP OPTION IS PRESENT, THE MESSAGE IS TYPED,

AND ANY ATTEMPTS TO SELECT IT FOR TESTING VIA SW01 AND SW00 ARE IGNORED. ONLY WARM FP (PDP-11/6X MICROCODE) FLOATING POINT CAN BE TESTED/SELECTED.

IF THE FP11-E IS PRESENT, TEST SELECTION IS AS FOLLOWS:

WHEN SW01=0, THE HOT AND WARM FLOATING POINT UNITS ARE TESTED ALTERNATELY EACH PASS - IN THE ORDER (1) HOT, THEN (2) WARM. NOTE THAT EACH "PASS" NOW CONSISTS OF TWO SEPARATE SUB-PASSES.

WHEN SW01=1, THEN DEDICATED SELECTION OF A PARTICULAR UNIT IS SPECIFIED IN SW00:

SW00=0 --> TEST HFP FP11-E OPTION ONLY
SW00=1 --> TEST WFP PDP-11/6X MICROCODE ONLY

6. ERRORS

6.1 FORMAT OF MESSAGES

6.1.1 ALL ERROR MESSAGES CONSIST OF THREE LINES OF DATA:

THE FIRST LINE IS A BRIEF MESSAGE WHICH EXPLAINS WHAT ERROR WAS DETECTED (EG, THE RESULT OF THE "ABSF" INSTRUCTION WAS BAD).

THE PREFIX "HOT:" OR "WARM:" IS ALSO ATTACHED TO THE MESSAGE TO INDICATE THE SOURCE OF THE ERROR; THE FP11-E UNIT OR THE PDP-11/6X RESPECTIVELY.

THE SECOND LINE CONSISTS OF DATA HEADERS TO IDENTIFY THE VALUES TYPED OUT ON LINE THREE. THESE HEADERS WILL EITHER BE OF THE FORM "EXPECTED" AND "RECEIVED" DATA, OR WILL BE A MNEMONIC NAME OF A WORD LOCATION IN MEMORY OR REGISTERS.

THE THIRD LINE DISPLAYS THE CONTENTS OF THE LOCATIONS SPECIFIED BY LINE TWO AS SIX DIGIT OCTAL NUMBERS. NOTE THAT ALL DATA DISPLAYED IN ANY MESSAGES ARE OCTAL NUMBERS.

AS EXPLAINED IN SECTION 5.2, SETTING SW<13>=1 WILL SUPPRESS THE TYPING OF THESE MESSAGES.

6.1.2 FLOATING POINT UNIT DATA FORMATS:

FLOATING POINT STATUS WORD (FPS):

BIT#	OCTAL	FUNCTION
15	100000	FER - FLOATING ERROR FLAG SET WHEN EITHER FIUV, FIU, FIV, FIC ENABLED AND APPROPRIATE EXCEPTION OCCURRED.
14	040000	FID - FLOATING DISABLE INTERRUPTS NO FP INTERRUPTS TO VECTOR 244(8) IF SET.
13, 12		NOT USED
11	004000	FIUV - FLOATING UNDEFINED VARIABLE INTERRUPT IF SET, (-0) MEMORY DATA IS ERROR
10	002000	FIU - FLOATING INTR UNDERFLOW IF SET AND UNDERFLOW, SET FER, STORE ANSWER, EXPONENT WRONG BY +400(8) IF CLEAR AND UNDERFLOW, ANSWER <-- ZERO
9	001000	FIV - FLOATING OVERFLOW INTERRUPT IF SET AND OVERFLOW, SET FER, STORE ANSWER, EXPONENT WRONG BY -400(8) IF CLEAR AND OVERFLOW, ANSWER <-- ZERO
8	000400	FIC - FLOATING INTEGER CONVERSION INTERRUPT IF SET AND "STCFI" ERROR, ANSWER <-- ZERO, SET ERROR IF CLEAR AND "STCFI" ERROR, ANSWER <-- ZERO
7	000200	FD - FLOATING MODE 1=DOUBLE, 64 BIT OPERANDS (4W) 0=SINGLE, 32 BIT OPERANDS (2W)
6	000100	FL - INTEGER MODE 1=LONG, 32 BIT INTEGERS (2W) 0=SHORT, 16 BIT INTEGERS (1W)
5	000040	FT - ROUND/TRUNCATE MODE 1=TRUNCATE RESULTS 0=ROUND RESULTS
4	000020	FMM - PUT FP11-E ONLY IN MAINTENANCE MODE
3:0	000017	FN-FZ-FV-FC - FLOATING CONDITION CODES

FLOATING EXCEPTION CODES (FEC):

OCTAL	ENABLE	FUNCTION
00	(NONE)	(NOT USED)
02	(NONE)	FP OPCODE ERROR
04	(NONE)	FP DIVIDE-BY-ZERO ERROR
06	W/FIC	FP INTEGER CONVERSION ERROR
10	W/FIV	FP OVERFLOW ERROR
12	W/FIU	FP UNDERFLOW ERROR
14	W/FIUV	FP UNDEFINED-VARIABLE/(-0) ERROR
16	W/FMM	FP MAINTENANCE TRAP

NOTE - IN "FEC" CODE TYPEOUTS IN ERROR MESSAGES ONLY THE LOW ORDER BYTE IS USED - IGNORE THE PROGRAM FLAG BIT IN THE UPPER BYTE.

FLOATING POINT DATA:

IN FLOAT MODE (FD=0), IS 2-16. BIT WORDS, 32. BITS
 IN DOUBLE MODE (FD=1), IS 4-16. BIT WORDS, 64. BITS

FIRST WORD: (BOTH F, D MODES)

B15=SIGN OF NUMBER (1/-, 0/+)
 B14:07=EXPONENT, 8 BITS, FROM -128./+127.
 B06:00=FRACTION, 7 BITS

SECOND WORD: (BOTH F, D MODES)

B15:00=FRACTION, 16 BITS

THIRD, FOURTH WORDS: (ONLY D MODE)

B15:00, B15:00=FRACTION, 32. BITS

IN F MODE, THE COMPOSITE 24. BIT FRACTION
 IS FORMED BY:

.1#[WORD1-BIT<06:00>]#[WORD2-BIT<15:00>]

IN D MODE, THE COMPOSITE 56. BIT FRACTION
 IS FORMED BY:

.1#[WORD1-BIT<06:00>]#[WORD2-BIT<15:00>]
 #[WORD3-BIT<15:00>]#[WORD4-BIT<15:00>]

FOR A MORE DETAILED OPERATION/EXPLANATION OF FLOATING POINT
 DATA FORMATS AND OPERATIONS, SEE THE PDP-11/6X PROCESSOR
 HANDBOOK SECTION ON THE FLOATING POINT INSTRUCTION SET.

6.2 RECOVERY

RECOVERY FROM ERRORS HAS BEEN ATTEMPTED TO BE MADE AS
 AUTOMATIC AND EFFORTLESS AS POSSIBLE. HOWEVER, IN MANY CASES,
 DUE TO THE NATURE OF THE ERROR, THE PROGRAM MAY NOT EVEN BE
 ABLE TO BE RUN (EG. IF THE FLOATING POINT MODULE IS IN A HUNG
 STATE, AND CAN NEVER ENTER THE READY STATE TO ACCEPT A NEW FPP
 INSTRUCTION). AT THIS POINT, SOLVING THE PROBLEM IS A DIRECT
 FUNCTION OF THE OPERATORS' INGENUITY. THIS TEST SERIES HAS
 BEEN DESIGNED TO TEST THE FLOATING POINT PROCESSOR SO THAT
 THESE TYPES OF FAILURES TO RUN WILL BE MINIMAL. THE TESTS
 HAVE BEEN PLACED IN A SPECIFICALLY STRUCTURED SEQUENCE IN THE
 PROGRAM TO IMPLEMENT THIS STRATEGY: TESTING THE MOST BASIC
 ELEMENTS FIRST, PROCEEDING UPWARD IN COMPLEXITY AFTER
 ESTABLISHING THEIR CORRECT OPERATION. THIS IS WHY IT IS
 EXTREMELY IMPORTANT THAT THE FLOATING POINT TEST PROGRAMS BE
 (1) RUN IN THE PRESCRIBED ORDER, AND (2) ONLY BE STARTED AT
 THEIR BEGINNING ADDRESS (USUALLY 200(8)). THE PROGRAM WILL
 DISPLAY, AT AN ERROR, THE MOST PERTINENT INFORMATION RELATING
 TO THE ERROR, AND A BRIEF EXPLANATION OF THE FAILING FUNCTION.

6.3 CAUSES

THESE TEST PROGRAMS ARE NOT HARDWARE ORIENTED, AND AS SUCH IT IS NOT POSSIBLE TO CALL OUT PARTICULAR HARDWARE AREAS AND MODULES RELATING TO A GIVEN FUNCTIONAL FAILURE. HARDWARE DIAGNOSIS FOR A PARTICULAR MACHINE MUST BE DONE USING THE APPROPRIATE ENGINEERING ROM FLOWS AND PRINTS, ALONG WITH THE KNOWN FUNCTIONAL ERRORS (AS DETECTED BY THE PROGRAMS). THIS IS THE INTENT UNDER WHICH THESE INSTRUCTION TESTS WERE DESIGNED AND CODED.

7. RESTRICTIONS

7.1 STARTING

THE PROGRAM MUST BE STARTED AT LOCATION 200(8) ALWAYS.

7.2 OPERATIONAL

THERE ARE NO OPERATIONAL RESTRICTIONS.

8. MISCELLANEOUS

8.1 EXECUTION TIME

MODEL	AVERAGE EXECUTION TIME PER PASS	
	SHORTEST PASS	LONGEST PASS
PDP-11/6X MICROCODE	0:01	1:15
PDP-11/6X W/FP11-E	0:01	0:30

TIMES SPECIFIED AS (MINUTES):(SECONDS)

SHORTEST PASS ::= PASS=1, NO ITERATIONS, USING:
 SMR=(004003) FOR PDP-11/6X MICROCODE
 SMR=(004002) FOR PDP-11/6X W/FP11-E

LONGEST PASS ::= PASS>=2, 2000, ITERATIONS/TEST, USING:
 SMR=(000003) FOR PDP-11/6X MICROCODE
 SMR=(000002) FOR PDP-11/6X W/FP11-E

8.2 STACK POINTER

THE STACK POINTER IS SET TO 1100(8) AT THE START OF EACH PASS. IF ALL IS OPERATING CORRECTLY, IT SHOULD ALSO BE THIS VALUE AT

THE START OF EACH TEST, AND AT THE END OF A PASS.

8.3 POWER FAIL

THE TESTS MAY BE POWER FAILED AT ANY TIME. SPURIOUS ERROR MESSAGES MAY OCCUR IF THE FAILURE OCCURRED WHILE THE F.P.U. WAS EXECUTING A FUNCTION, AS NONE OF ITS REGISTERS (FPS, FEC, FEA, ACCUMULATORS) ARE SAVED IN THE EVENT OF A POWER FAILURE. HOWEVER, THESE MESSAGES SHOULD ONLY OCCUR ONCE (IF AT ALL) IMMEDIATELY AFTER POWER IS RESTORED. WHEN POWER IS RESTORED, "POWER" IS TYPED ON THE CONSOLE AND EXECUTION CONTINUES WHERE IT WAS INTERRUPTED.

NOTE THAT THE "VOLATILE" SWITCH REGISTER CONTENTS ARE SAVED AND RESTORED FROM THE STACK IN A POWER FAIL SEQUENCE; THEREFORE THE SWITCH REGISTER SETTINGS SHOULD NOT BE LOST OVER A POWER FAIL.

9. PROGRAM DESCRIPTION

9.1 ORGANIZATION

THESE PROGRAMS ARE ORGANIZED AS MUCH AS POSSIBLE IN A STRAIGHTFORWARD, LINEAR MANNER. THE MAIN BODY OF CODE IS STRUCTURED AS FOLLOWS:

- (1) INITIALIZATION ROUTINE
 - SETS UP VECTORS, TYPES HEADER, ETC.
- (2) MAIN BODY OF TESTS
 - INLINE TEST CODE, INLINE TEST CALLS
- (3) END OF PASS ROUTINE
 - END OF PASS PROCESSING
- (4) TEST SUBROUTINES
 - SUBROUTINES CONTAINING COMMON TEST CODE
- (5) OVERHEAD ROUTINES
 - SERVICE SUBROUTINES (TYPEOUT, ETC.)

WHEREVER FEASIBLE, COMMON SECTIONS OF CODE FOR WIDELY USED FUNCTIONS ARE CONDENSED INTO SUBROUTINES TO CONSERVE MEMORY. THIS INCLUDES NOT ONLY STANDARD SERVICE ROUTINES (SUCH AS SCOPE, ERROR, AND ASCII TYPEOUT), BUT ALSO TESTING ROUTINES WHICH PERFORM VERY SIMILAR FUNCTIONS. THIS IN MANY CASES (THE "AODF" INSTRUCTION TESTING, FOR EXAMPLE) A SINGLE BODY OF CODE (A SUBROUTINE) IS USED TO PERFORM ALL THE FUNCTIONAL TESTS, WITH A VARIABLE PARAMETER LIST PASSED AT EACH CALL CONTAINING THE DATA OPERANDS AND EXPECTED RESULT FOR EACH INDIVIDUAL TEST. THIS CONSTRUCTION FACILITATES THE ADDITION/DELETION OF TESTS (SHOULD THAT EVER BE NECESSARY), AND ALSO GREATLY CONSERVES MEMORY SPACE REQUIREMENTS WHEN A LARGE NUMBER OF CALLS TO A GIVEN BODY OF CODE ARE REQUIRED.

THE INDIVIDUAL TESTS WITHIN EACH PROGRAM HAVE ALSO BEEN SEQUENCED IN A PARTICULAR ORDER TO FACILITATE THE DETECTION AND RESOLUTION OF ERRORS AS QUICKLY AS POSSIBLE. EACH OF THE TESTS BEGINS AS SIMPLY AS POSSIBLE, FIRST TESTING THE MOST BASIC ELEMENTS. MORE COMPLEX ELEMENTS ARE TESTED AFTERWARDS, EMPLOYING A PHILOSOPHY THAT THE SIMPLER THE TEST, THE BETTER THE RESOLUTION. ALL FUNCTIONS ARE EVENTUALLY TESTED, BUT HOPEFULLY MOST ERRORS WILL BE CAUGHT AND CORRECTED EARLY. A MUCH MORE DETAILED ANALYSIS OF THE SEQUENCE OF TESTS PERFORMED IS PRESENTED IN SECTION 9.2.

9.2 TEST DESCRIPTION

THIS DIAGNOSTIC CONSISTS OF A NUMBER OF TESTS TO EXERCISE COMBINATIONS OF FLOATING POINT INSTRUCTIONS, USING VARIOUS ADDRESS MODES, IN A MORE OR LESS APPLICATION-PROGRAM-TYPE ENVIRONMENT.

THE FIRST SECTION OF TESTS CONTAINS COMMONLY ENCOUNTERED SEQUENCES OF FLOATING POINT CODE. VARIOUS OPERANDS, ADDRESS MODES, AND FLOATING POINT ENVIRONMENTS (I.E., F/D MODES) ARE EMPLOYED. WHEN PRESENT, THE LINE AND/OR PROGRAMMABLE CLOCKS WILL BE EMPLOYED TO GENERATE I/O INTERRUPTS TO CHECK THE FP INSTRUCTION RESTART FEATURE.

THE NEXT SECTION OF CODE CHECKS FLOATING POINT EXCEPTION CONDITIONS. ALL EXCEPTION CONDITIONS ARE GENERATED, AND THE ENABLED/DISABLED AND TRAP/NO-TRAP MODES USED TO VERIFY CORRECT OPERATION.

THE LAST SECTION OF CODE CONSISTS OF A MORE COMPLEX EXERCISER SECTION THAN THAT IN THE FIRST SECTION, TO CHECK A MORE COMPLICATED SERIES OF FLOATING POINT OPERATIONS.

9.3 SUBROUTINE ABSTRACTS

9.3.1 TRAPCATCHER

THE TRAPCATCHER IS A SERIES OF INSTRUCTIONS OCCUPYING THE INTERRUPT VECTOR AREA OF MEMORY. IT CONSISTS OF THE SEQUENCE:

```
.WORD    +2    ;PC AFTER TRAP
.WORD    0     ;PS AFTER TRAP
```

PLACED AT EACH VECTOR ADDRESS IN LOCATIONS 4-776(B) OF MEMORY. THE FIRST WORD OF EACH PAIR ("PC AFTER TRAP") POINTS TO THE SECOND WORD, WHICH SERVES A DUAL PURPOSE AS
(1) THE NEW LOADED PS (ALL ZEROS), AND (2) THE NEXT INSTRUCTION TO EXECUTE (0=HALT).

WHEN THE PROGRAM IS EXECUTING, ANY REQUIRED VECTORS ARE SET UP

IN THE VECTOR AREA WITH APPROPRIATE VALUES; THE OTHERS BEING LEFT IN THE "TRAPCATCHER" STATE. THUS, IF AN UNEXPECTED TRAP EVER OCCURS IN THE MACHINE IT WILL BE CAUGHT, AND THE MACHINE SUBSEQUENTLY HALTED, DISPLAYING THE VECTOR ADDRESS * PLUS FOUR * IN THE ADDRESS LIGHTS.

9.3.2 SCOPE ROUTINE - SSCOPE

THE SCOPE ROUTINE IS ENTERED FROM THE FIRST INSTRUCTION OF EACH TEST IN THE PROGRAM. (NOTE THAT BY DEFINITION, A "TEST" WILL BE DESIGNATED AS THE SECTION OF CODE BETWEEN TWO "SCOPE" STATEMENTS.) THIS ROUTINE PROVIDES THE OVERHEAD CODE NECESSARY TO IMPLEMENT SEVERAL OF THE SWITCH REGISTER CONTROL OPTIONS. UPON ENTRANCE TO A TEST, THE SCOPE STATEMENT AT THE BEGINNING SETS UP CERTAIN LOCATIONS (SEE BELOW) TO SPECIFY THE CURRENT TEST NUMBER AND LOOPING ADDRESS (FOR ITERATIONS). CONTROL IS THEN PASSED TO THE ACTUAL TEST CODE, PERFORMING THE DESIRED TEST. UPON EXIT, THE SCOPE STATEMENT OF THE NEXT TEST IS ENTERED, WHICH DETERMINES WHETHER TO (1) LOOP BACK TO THE PREVIOUS TEST (EG, FOR ITERATIONS) OR (2) INITIALIZE FOR THE NEXT TEST (AS DESCRIBED EARLIER, ABOVE).

ENTRANCE TO THE SCOPE ROUTINE IS VIA AN "IOT" TRAP CALL THROUGH LOCATION 20(8). (FROM THE SCOPE=IOT EQUATE). DEPENDING UPON THE SWITCH SETTINGS (SEE 5.2), CODE IS PRESENT TO: LOAD THE FP11 MICRO BREAK REGISTER, LOOP ON THE CURRENTLY EXECUTING TEST, LOOP ON A SPECIFIC TEST, PERFORM ITERATIONS OF EACH TEST, AND SET UP ADDRESSES FOR POSSIBLE LOOPING ON ERRORS. IMPORTANT VALUES USED IN THIS ROUTINE ARE:

- SIXCNT - MAXIMUM NUMBER OF ITERATIONS PER TEST (GENERALLY WILL BE 2000(10))
- STSTNM - A COUNTER INDICATING THE NUMBER (1-377(8)) OF THE TEST CURRENTLY BEING EXECUTED
- SLPADR - CONTAINS THE ADDRESS TO WHICH THE SCOPE ROUTINE 10370 WILL LOOP, IF THE CURRENT TEST IS BEING LOOPED UPON
- SLPERR - CONTAINS THE ADDRESS TO WHICH THE ERROR ROUTINE (SEE 9.3.3) WILL LOOP, IF AN ERROR OCCURS AND THE LOOPING ON AN ERROR OPTION IS SPECIFIED IN THE SWITCHES. SET UP BY SCOPE, GENERALLY WILL BE THE SAME AS SLPADR, ABOVE.

9.3.3 ERROR ROUTINE - SERROR

THE ERROR ROUTINE IS ENTERED WHEN THE TEST CODE HAS DETERMINED THAT AN ERROR HAS OCCURRED AS PART OF A TEST. THROUGH USE OF THIS ROUTINE, THE TEST HAS A MEANS OF SIGNALING AN ERROR TO THE 10550 OPERATOR/MONITOR; AND IMPLEMENTING THE CONTROL FUNCTIONS FOR HALTING ON ERROR, BELL ON ERROR, AND LOOPING ON ERROR. IN ADDITION, THE ERROR ROUTINE HAS THE PROVISION TO TYPE OUT ON THE OPERATOR'S CONSOLE A MESSAGE BRIEFLY EXPLAINING THE ERROR, AND SOME OF THE MOST PERTINENT

DATA VALUES TO HELP DIAGNOSE THE CAUSE (SEE SECTION 6.2).

THE CALLING MECHANISM IS SIMILAR TO THAT EMPLOYED FOR THE SCOPE ROUTINE (VIA A TRAP). EXCEPT IN THIS INSTANCE, THE "EMT" INSTRUCTION IS USED, TRAPPING THROUGH LOCATION 30(8). (NOTE THE EQUATE ERROR N=EMT N). THE LOWER BYTE OF THE EMT INSTRUCTION IS CAPABLE OF TRANSMITTING A NUMBER FROM 0-377(8), WHICH WILL BE TERMED THE "ERROR ITEM NUMBER." THIS NUMBER DETERMINES WHICH ERROR MESSAGE, AND ASSOCIATED DATA VALUES WILL BE TYPED OUT WHEN A PARTICULAR ERROR IS SIGNALLED. IF THIS NUMBER IS ZERO, JUST THE PC OF THE CALLING "ERROR" INSTRUCTION WILL BE TYPED, OTHERWISE, THE NUMBER IS USED AS AN INDEX THROUGH THE ERROR TABLE (SERRTB) TO FIND THE APPROPRIATE VALUES TO TYPE (SEE PROGRAM LISTING FOR FURTHER DETAILS).

IMPORTANT VALUES USED IN THIS ROUTINE ARE:

EREG0 THRU EREG7 - CONTENTS OF GENERAL REGISTERS R0 THRU R7 JUST BEFORE ERROR CALL
 SERTTL - CUMULATIVE NUMBER OF ERRORS ENCOUNTERED TO DATE
 SERRPC - CONTAINS THE PC OF THE "ERROR" INSTRUCTION JUST EXECUTED
 SLPERR - CONTAINS THE ADDRESS WHICH WILL BE LOOPED UPON FOR THE ERROR LOOPING FACILITY

9.3.4 ERROR MESSAGE TIMEOUT ROUTINE - STYPERR

THIS ROUTINE (STYPERR ENTRY POINT) IS CALLED BY THE ERROR PROCESSING ROUTINE DESCRIBED IN 9.3.3 ABOVE. ITS PURPOSE IS TO IMPLEMENT THE ERROR MESSAGE/DATA VALUE ERROR TIMEOUT FACILITY. THE SUBROUTINE WILL, GIVEN THE INDEXING BYTE FROM THE ERROR CALL INSTRUCTION, PICK UP THE CORRECT ERROR MESSAGE VECTOR FROM SERRTB (ERROR TABLE), AND TYPE OUT THE ERROR MESSAGE, DATA HEADER, AND DATA VALUES ON THE CONSOLE.

9.3.5 TYPE ROUTINE - STYPE

THIS ROUTINE IS THE STANDARD SYSTEM TIMEOUT ROUTINE FOR ASCII SINGLE-CHARACTER-PER-BYTE STRINGS. IT IS CALLED THROUGH A TRAP INSTRUCTION WITH THE NEXT WORD CONTAINING THE ADDRESS OF THE FIRST CHARACTER IN THE STRING. TYPING TERMINATES WHEN AN ALL-ZERO BYTE IS FOUND. HORIZONTAL TAB STOPS ARE ALSO AUTOMATICALLY PLACED.

9.3.6 OCTAL NUMBER TYPE ROUTINE - STYPOC

THIS ROUTINE CONVERTS THE TOP NUMBER ON THE STACK TO A 6-DIGIT OCTAL REPRESENTATION, AND TYPES IT ON THE CONSOLE USING THE TYPE ROUTINE STYPE. SEE LISTING FOR OPTIONS AND FURTHER DETAILS.

9.3.7 POWER UP AND DOWN ROUTINES - SPWRUP AND SPWRDN

THESE TWO ROUTINES ARE ENTERED FOR THE POWER UP AND DOWN CONDITIONS, RESPECTIVELY. THE POWER DOWN ROUTINE (SPWPDN) SAVES THE GENERAL REGISTERS AND STACK POINTER. THE POWER UP ROUTINE (SPWUP) CORRESPONDINGLY RESTORES THE REGISTERS, STACK POINTER, AND TYPES THE MESSAGE "POWER" WHEN POWER IS RESTORED. THE VOLATILE INTERNAL SWITCH REGISTER IS ALSO SAVED/RESTORED BY THIS ROUTINE.

9.3.8 END OF PASS ROUTINE - SECP

THE END OF PASS ROUTINE COUNTS THE NUMBER OF PASSES PERFORMED, DINGS THE BELL/TYPES A MESSAGE (IF ENABLED), SETS/CLEARs THE T-BIT (IF ENABLED), AND ALSO INTERFACES TO THE MONITOR, IF PRESENT. IT ALSO OPTIONALLY LOOPS FOR A NUMBER OF SUBPASSES BEFORE SIGNALLING AN END OF PASS CONDITION.

10. ACT/APT/XXDP

10.1 ACT COMPATIBILITY

THIS PROGRAM WILL RUN UNDER THE ACT SYSTEM.

10.2 APT COMPATIBILITY

THIS PROGRAM WILL RUN UNDER THE APT SYSTEM MONITOR. ALL NECESSARY SOFTWARE COMMUNICATION HOOKS ARE PRESENT.

10.3 XXDP COMPATIBILITY

FOR XXDP MEDIA COMPATIBILITY, THE TOP 2K WORDS OF THE 16K WORD MINIMUM MEMORY AREA ARE NOT DISTURBED DURING EXECUTION.

FPU INSTR EXERCISER MACY11 27(1006) 04-MAY-77 19:35
 DQFPCB.P11 04-MAY-77 19:34 TABLE OF CONTENTS

14	OPERATIONAL SWITCH SETTINGS
32	BASIC DEFINITIONS
173	TRAP CATCHER
182	STARTING ADDRESS(ES)
185	ACT11 HOOKS
196	APT PARAMETER BLOCK
219	COMMON TAGS
284	APT MAILBOX-ETABLE
311	ERROR POINTER TABLE
354	PROGRAM DEFINED COMMON TAGS
444	START OF PASS ROUTINE
452	INITIALIZE THE COMMON TAGS
627	T1 TEST OF WRITABILITY OF FPS
667	T2 TEST OF CFCC
695	T3 TEST OF LDD, STD, CMPD OF -1,0,-1,0
749	T4 TEST OF LDD, STD, CMPD OF 0,-1,0,-1
785	T5 TEST OF LDF, STF, CMPF OF -1,0
823	T6 TEST OF LDF, STF, CMPF WITH {=} IN ALL AC'S
875	T7 TEST OF CMPF WITH DATA IN ACO-ACS
899	T10 TEST OF TSTF AND TSTD USING OLD ACO-ACS
941	T11 TEST OF CLRX INSTRUCTIONS
991	T12 TEST OF NEGX
1052	T13 TEST OF ABSX
1108	T14 TEST OF LDEXP & STEXP
1172	T15 TEST OF ADDF & SUBF
1204	T16 TEST OF ADDD AND SUBD
1247	T17 TEST OF MULF AND DIVF
1277	T20 TEST OF MULD AND DIVD
1307	T21 TEST OF LDCFD, LDCDF
1368	T22 TEST OF STCFD, STCDF
1402	T23 TEST OF LDCIF, LDCID, STCFI, STCDI
1470	T24 TEST OF LDCLF, LDCLD, STCFL, STCDL
1540	T25 TEST OF MODF AND MOOD
1599	ERROR CONDITIONS (STST)
1601	T26 LDD OF -0
1642	T27 MULF ERROR - OVERFLOW
1669	T30 DIVF ERROR - UNDERFLOW
1696	T31 STCFI ERROR - CONVERSION(6)
1722	T32 DIVF BY 0 ERROR
1747	T33 LDF -0 ERROR
1772	T34 OPCODE ERROR
1797	T35 ADDF ERROR - OVERFLOW
1824	T36 SUBF ERROR - UNDERFLOW
1852	T37 TEST OF FEC-02, WITH NO TRAP, INTERRUPT DISABLED
1872	T40 TEST OF FEC-04, WITH NO TRAP, INTERRUPT DISABLED
1892	T41 TEST OF FEC-06, WITH NO TRAP, INTERRUPT DISABLED
1912	T42 TEST OF FEC-10, WITH NO TRAP, INTERRUPT DISABLED
1932	T43 TEST OF FEC-12, WITH NO TRAP, INTERRUPT DISABLED
1952	T44 TEST OF FEC-14, WITH NO TRAP, INTERRUPT DISABLED
1972	T45 TEST OF FEC-16, WITH NO TRAP, INTERRUPT DISABLED
2016	T46 TEST OF FEC-02, WITH TRAP, INTERRUPT ENABLED
2036	T47 TEST OF FEC-04, WITH TRAP, INTERRUPT ENABLED
2056	T50 TEST OF FEC-06, WITH NO TRAP, INTERRUPT ENABLED
2076	T51 TEST OF FEC-06, WITH TRAP, INTERRUPT ENABLED
2096	T52 TEST OF FEC-10, WITH NO TRAP, INTERRUPT ENABLED
2116	T53 TEST OF FEC-10, WITH TRAP, INTERRUPT ENABLED

FPU INSTR EXERCISER MACY11 27(1006) 04-MAY-77 19:35
DQFPC8.P11 04-MAY-77 19:34 TABLE OF CONTENTS

SEQ 0002

2136	T54	TEST OF FEC-12, WITH NO TRAP, INTERRUPT ENABLED
2156	T55	TEST OF FEC-12, WITH TRAP, INTERRUPT ENABLED
2176	T56	TEST OF FEC-14, WITH NO TRAP, INTERRUPT ENABLED
2196	T57	TEST OF FEC-14, WITH TRAP, INTERRUPT ENABLED
2216	T60	TEST OF FEC-16, WITH NO TRAP, INTERRUPT ENABLED
2236	T61	TEST OF FEC-16, WITH TRAP, INTERRUPT ENABLED
2270	T62	TEST FOR CONVERSION, ADD AND SUBD
2286	T63	LDD AND STD TEST
2302	T64	MULD AND DIVD TEST
2326	T65	EXERCISER TEST
2370	T66	MODE ONE TEST
2396	T67	ADD EXERCISER
2417	T70	TEST DIVIDE BY 0
2436	T71	EXERCISER FOR ADD, SUBD, MULD AND DIVD
2509	T72	...CLEAR OUT CLOCKS BEFORE SEOP
2535		SUB PASS END CONTROL
2574		END OF PASS ROUTINE (MODIFIED SYSMAC)
2611		TRAP/FEC TESTER SUBR
2752		LINE CLOCK INTERRUPT SERVICE ROUTINES
2784		FPP TRAP CATCHER
2804		SCOPE HANDLER ROUTINE
2868		ERROR HANDLER ROUTINE
2931		ERROR MESSAGE TIMEOUT ROUTINE (MODIFIED SYSMAC)
2997		TYPE ROUTINE
3076		APT COMMUNICATIONS ROUTINE
3133		BINARY TO OCTAL (ASCII) AND TYPE
3210		TRAP DECODER
3233		TRAP TABLE
3247		POWER DOWN AND UP ROUTINES
3294		ERROR MESSAGES, DATA HEADERS, DATA VECTORS, ETC

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
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

```
.TITLE FPU INSTR EXERCISER
.*COPYRIGHT (C) 1976
.*DIGITAL EQUIPMENT CORP.
.*MAYNARD, MASS. 01754
.*
.*PROGRAM BY DONALD NORTH
.*
.*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
.*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
.*
```

.SBTTL OPERATIONAL SWITCH SETTINGS

SWITCH	OCTAL	USE
15	100000	HALT ON ERROR
14	040000	LOOP ON CURRENTLY EXECUTING TEST
13	020000	INHIBIT ERROR TYPEOUTS
12	010000	INHIBIT STATUS TYPEOUTS
11	004000	INHIBIT ITERATIONS
10	000000	0=BELL ON PASS END
	002000	1=BELL ON ERROR
9	001000	LOOP ON ERROR
8	000400	LOOP ON TEST NUMBER IN "SLPTST"
1	000000	0=TEST HFP/WFP ALTERNATELY EACH PASS
	000002	1=TEST ONLY UNIT SPECIFIED IN SW<00>
0	000002	0=SELECT HFP, IF SW<01>=1
	000003	1=SELECT WFP, IF SW<01>=1

.SBTTL BASIC DEFINITIONS

```
.*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100 STACK= 1100
.EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL
.EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL

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

.*GENERAL PURPOSE REGISTER DEFINITIONS
000000 R0= %0 ;;GENERAL REGISTER
000001 R1= %1 ;;GENERAL REGISTER
000002 R2= %2 ;;GENERAL REGISTER
000003 R3= %3 ;;GENERAL REGISTER
000004 R4= %4 ;;GENERAL REGISTER
000005 R5= %5 ;;GENERAL REGISTER
```

57	000006	R6=	%6	:: GENERAL REGISTER
58	000007	R7=	%7	:: GENERAL REGISTER
59	000006	SP=	%6	:: STACK POINTER
60	000007	PC=	%7	:: PROGRAM COUNTER
61		.:#PRIORITY LEVEL DEFINITIONS		
62		PRO=	0	:: PRIORITY LEVEL 0
63	000000	PR1=	40	:: PRIORITY LEVEL 1
64	000040	PR2=	100	:: PRIORITY LEVEL 2
65	000100	PR3=	140	:: PRIORITY LEVEL 3
66	000140	PR4=	200	:: PRIORITY LEVEL 4
67	000200	PR5=	240	:: PRIORITY LEVEL 5
68	000240	PR6=	300	:: PRIORITY LEVEL 6
69	000300	PR7=	340	:: PRIORITY LEVEL 7
70	000340			
71		.:#"SWITCH REGISTER" SWITCH DEFINITIONS		
72		SW15=	100000	
73	100000	SW14=	40000	
74	040000	SW13=	20000	
75	020000	SW12=	10000	
76	010000	SW11=	4000	
77	004000	SW10=	2000	
78	002000	SW09=	1000	
79	001000	SW08=	400	
80	000400	SW07=	200	
81	000200	SW06=	100	
82	000100	SW05=	40	
83	000040	SW04=	20	
84	000020	SW03=	10	
85	000010	SW02=	4	
86	000004	SW01=	2	
87	000002	SW00=	1	
88	000001	.EQUIV	SW09, SW9	
89		.EQUIV	SW08, SW8	
90		.EQUIV	SW07, SW7	
91		.EQUIV	SW06, SW6	
92		.EQUIV	SW05, SW5	
93		.EQUIV	SW04, SW4	
94		.EQUIV	SW03, SW3	
95		.EQUIV	SW02, SW2	
96		.EQUIV	SW01, SW1	
97		.EQUIV	SW00, SW0	
98				
99				
100		.:#DATA BIT DEFINITIONS (BIT00 TO BIT15)		
101	100000	BIT15=	100000	
102	040000	BIT14=	40000	
103	020000	BIT13=	20000	
104	010000	BIT12=	10000	
105	004000	BIT11=	4000	
106	002000	BIT10=	2000	
107	001000	BIT09=	1000	
108	000400	BIT08=	400	
109	000200	BIT07=	200	
110	000100	BIT06=	100	
111	000040	BIT05=	40	
112	000020	BIT04=	20	

```

113      000010      BIT03= 10
114      000004      BIT02= 4
115      000002      BIT01= 2
116      000001      BIT00= 1
117      .EQUIV      BIT09,BIT9
118      .EQUIV      BIT08,BIT8
119      .EQUIV      BIT07,BIT7
120      .EQUIV      BIT06,BIT6
121      .EQUIV      BIT05,BIT5
122      .EQUIV      BIT04,BIT4
123      .EQUIV      BIT03,BIT3
124      .EQUIV      BIT02,BIT2
125      .EQUIV      BIT01,BIT1
126      .EQUIV      BIT00,BIT0
127
128      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
129      000004      ERRVEC= 4      ;: TIME OUT AND OTHER ERRORS
130      000010      RESVEC= 10     ;: RESERVED AND ILLEGAL INSTRUCTIONS
131      000014      TBITVEC=14     ;: "T" BIT
132      000014      TRTVEC= 14     ;: TRACE TRAP
133      000014      BPTVEC= 14     ;: BREAKPOINT TRAP (BPT)
134      000020      IOTVEC= 20     ;: INPUT/OUTPUT TRAP (IOT) **SCOPE**
135      000024      PWRVEC= 24     ;: POWER FAIL
136      000030      EMTVEC= 30     ;: EMULATOR TRAP (EMT) **ERROR**
137      000034      TRAPVEC=34     ;: "TRAP" TRAP
138      000060      TKVEC= 60      ;: TTY KEYBOARD VECTOR
139      000064      TPVEC= 64      ;: TTY PRINTER VECTOR
140      000240      PIRAVEC=240    ;: PROGRAM INTERRUPT REQUEST VECTOR
141
142      ;*MED INSTR EQUATES
143      076600      MED= 076600    ;:OPCODE
144
145      000022      RWHAMI= 022     ;:READ WHAMI
146
147      000144      RFLAG= 144     ;:READ FLAGS
148      000344      WFLAG= 344     ;:WRITE FLAGS
149
150      ;*LINE CLOCK VECTORS, ADDRESSES
151      177546      LKS= 177546    ;:STATUS      KW11-L LINE CLOCK
152      000100      LKV= 100      ;:INTERRUPT VECTOR
153
154      172540      PLKS= 172540    ;:STATUS      KW11-P PROG LINE CLOCK
155      172542      PLKR= 172542    ;:COUNT SET REGISTER
156      172544      PLKD= 172544    ;:READ COUNT REGISTER
157      000104      PLKV= 104      ;:INTERRUPT VECTOR
158
159      ;*FLOATING POINT INTERRUPT VECTOR
160      000244      FPPVEC= 244
161
162      ;*FLOATING POINT REGISTER DEFINITIONS
163      000000      AC0= %0
164      000001      AC1= %1
165      000002      AC2= %2
166      000003      AC3= %3
167      000004      AC4= %4
168      000005      AC5= %5

```

169
170
171
172
173
174
175
176
177
178
179 000174 000000
180 000176 000000
181
182 000200 000137 002400
183
184
185
186
187
188
189
190 000046 014264
191
192 000052 000000
193
194
195
196
197
198
199
200
201
202 000024 000200
203
204 000044 001000
205
206
207
208
209
210 001000
211 001000 000000
212 001002 001244
213 001004 000012
214 001006 000002
215 001010 000000
216 001012 000014
217

.SBTTL TRAP CATCHER

```

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

```

.SBTTL ACT11 HOOKS

```

; *****
; HOOKS REQUIRED BY ACT11
$SVPC=.      ; SAVE PC
.=46         ;; 1) SET LOC.46 TO ADDRESS OF SENDAD IN .SEOP
SENDAD
.=52         ;; 2) SET LOC.52 TO ZERO
.WORD 0      ;; RESTORE PC
.$SVPC
.=1000

```

.SBTTL APT PARAMETER BLOCK

```

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

```

```

$APTHD:
SHIBTS: .WORD 0      ;; TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MADR:  .WORD $MAIL  ;; ADDRESS OF APT MAILBOX (BITS 0-15)
$TSTM:  .WORD 10.    ;; RUN TIM OF LONGEST TEST
$PASTM: .WORD 2.     ;; RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
$UNITM: .WORD 0      ;; ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
.WORD   SETEND-$MAIL/2 ;; LENGTH MAILBOX-ETABLE(WORDS)

```

218
219
220
221
222
223
224
225 001100 001100
226
227 001100 000000
228 001102 000000
229 001104 000000
230 001106 000000
231 001110 000000
232 001112 000000
233 001114 000000
234 001116 000000
235 001120 000001
236 001122 000000
237 001124 000000
238 001126 000000
239 001130 000000
240 001132 000000
241 001134 000000
242 001136 000000
243 001140 000
244 001141 000
245 001142 000000
246
247 001144 177570
248 001146 177570
249 001150 000000
250 001152 177560
251 001154 177562
252 001156 177564
253 001160 177566
254 001162 000
255 001163 002
256 001164 012
257 001165 000
258 001166 000000
259
260 001170 000000
261 001172 000000
262 001174 000000
263 001176 000000
264 001200 000000
265 001202 000000
266 001204 000000
267 001206 000000
268 001210 000000
269 001212 000000
270 001214 000000
271 001216 000000
272 001220 000000
273 001222 000000

.SBTTL COMMON TAGS

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

.SCLTAG: =1100

;;-----START OF CLEAR COMMON TAGS-----

;;-----START OF CLEAR COMMON TAGS-----

STSTNM: .WORD 0
SERFLG: .WORD 0
SICNT: .WORD 0
SLPADR: .WORD 0
SLPERR: .WORD 0
SERTTL: .WORD 0
SITEMB: .WORD 0
SERMAX: .WORD 1
SERPC: .WORD 0
SGDADR: .WORD 0
SBDADR: .WORD 0
SGDDAT: .WORD 0
SBDAT: .WORD 0
SAUTOB: .BYTE 0
SINTAG: .BYTE 0

;; START OF COMMON TAGS
CONTAINS THE TEST NUMBER
CONTAINS ERROR FLAG
CONTAINS SUBTEST ITERATION COUNT
CONTAINS SCOPE LOOP ADDRESS
CONTAINS SCOPE RETURN FOR ERRORS
CONTAINS TOTAL ERRORS DETECTED
CONTAINS ITEM CONTROL BYTE
CONTAINS MAX. ERRORS PER TEST
CONTAINS PC OF LAST ERROR INSTRUCTION
CONTAINS ADDRESS OF 'GOOD' DATA
CONTAINS ADDRESS OF 'BAD' DATA
CONTAINS 'GOOD' DATA
CONTAINS 'BAD' DATA
RESERVED--NOT TO BE USED
AUTOMATIC MODE INDICATOR
INTERRUPT MODE INDICATOR

;;-----END OF CLEAR COMMON TAGS-----

SAR: .WORD DSWR
DISPLA: .WORD DDISP
SLPTST: .WORD 0
STKS: 177560
STKB: 177562
STPS: 177564
STPB: 177566
SNUL: .BYTE 0
SFILLS: .BYTE 2
SFILLC: .BYTE 12
STPFLG: .BYTE 0
SREGAD: .WORD 0

ADDRESS OF SWITCH REGISTER
ADDRESS OF DISPLAY REGISTER
CONTAINS TEST NUMBER TO LOOP UPON
TTY KBD STATUS
TTY KBD BUFFER
TTY PRINTER STATUS REG. ADDRESS
TTY PRINTER BUFFER REG. ADDRESS
CONTAINS NULL CHARACTER FOR FILLS
CONTAINS # OF FILLER CHARACTERS REQUIRED
INSERT FILL CHARS. AFTER A "LINE FEED"
"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
CONTAINS THE ADDRESS FROM WHICH (SREG0) WAS OBTAINED
CONTAINS ((SREGAD)+0)
CONTAINS ((SREGAD)+2)
CONTAINS ((SREGAD)+4)
CONTAINS ((SREGAD)+6)
CONTAINS ((SREGAD)+10)
CONTAINS ((SREGAD)+12)
CONTAINS ((SREGAD)+14)
CONTAINS ((SREGAD)+16)
USER DEFINED
USER DEFINED
USER DEFINED
USER DEFINED
USER DEFINED
USER DEFINED

SREG0: .WORD 0
SREG1: .WORD 0
SREG2: .WORD 0
SREG3: .WORD 0
SREG4: .WORD 0
SREG5: .WORD 0
SREG6: .WORD 0
SREG7: .WORD 0
STMP0: .WORD 0
STMP1: .WORD 0
STMP2: .WORD 0
STMP3: .WORD 0
STMP4: .WORD 0
STMP5: .WORD 0

274 001224 000000
 275 001226 000000
 276 001230 000000
 277 001232 000000
 278 001234 177607 000377
 279 001240 077
 280 001241 015
 281 001242 000012
 282
 283
 284
 285
 286
 287 001244
 288 001244 000000
 289 001246 000000
 290 001250 000000
 291 001252 000000
 292 001254 000000
 293 001256 000000
 294 001260 000000
 295 001262 000000
 296 001264
 297 001264 000
 298 001265 000
 299 001266 000000
 300 001270 000000
 301 001272 000000
 302
 303
 304
 305
 306
 307
 308 001274
 309

```

STMP6: .WORD 0          :: USER DEFINED
STMP7: .WORD 0          :: USER DEFINED
STIMES: 0                :: MAX. NUMBER OF ITERATIONS
SESCAPE: 0               :: ESCAPE ON ERROR ADDRESS
SBELL: .ASCIZ <207><377><377> :: CODE FOR BELL
SQUES: .ASCII /?/       :: QUESTION MARK
SCRLF: .ASCII <15>      :: CARRIAGE RETURN
SLF: .ASCIZ <12>        :: LINE FEED
:*****
.SBTTL APT MAILBOX-ETABLE
:*****
.EVEN
SMAIL: :: APT MAILBOX
SMSGTY: .WORD AMSGTY :: MESSAGE TYPE CODE
SFATAL: .WORD AFATAL :: FATAL ERROR NUMBER
STESTN: .WORD ATESTN :: TEST NUMBER
SPASS: .WORD APASS :: PASS COUNT
SDEVCT: .WORD ADEVCT :: DEVICE COUNT
SUNIT: .WORD AUNIT :: I/O UNIT NUMBER
SMSGAD: .WORD AMSGAD :: MESSAGE ADDRESS
SMSGLG: .WORD AMSGLG :: MESSAGE LENGTH
SETABLE: :: APT ENVIRONMENT TABLE
SENV: .BYTE AENV :: ENVIRONMENT BYTE
SENVH: .BYTE AENVH :: ENVIRONMENT MODE BITS
SSHREG: .WORD ASHREG :: APT SWITCH REGISTER
SUSWR: .WORD AUSWR :: USER SWITCHES
SCPUOP: .WORD ACPUOP :: CPU TYPE, OPTIONS
: *
: * BIT 15-11=CPU TYPE
: * 11/04=01, 11/05=02, 11/20=03, 11/40=04, 11/45=05
: * 11/70=06, PD0=07, 0=10
: * BIT 10=REAL TIME CLOCK
: * BIT 9=FLOATING POINT PROCESSOR
: * BIT 8=MEMORY MANAGEMENT
SETEND:
.MEXIT

```

310
311
312 001274
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328 001274 017431 020374 020726
329 001302 017455 020401 020660
330 001310 017501 020406 020740
331 001316 017525 020413 020664
332 001324 017557 020425 020666
333 001332 017601 020432 020672
334 001340 020001 000000 000000
335 001346 017623 020466 020712
336 001354 017623 020452 020704
337 001362 017623 020444 020700
338 001370 017623 020425 020666
339
340 001376 020001 000000 000000
341 001404 020067 000000 000000
342 001412 020154 020576 020762
343 001420 020223 020576 020770
344 001426 020266 020576 020776
345 001434 020331 020576 021004
346 001442 017675 020536 020732
347 001450 017431 020516 020724
348
349 001456 017725 020612 020744

.SBTTL ERROR POINTER TABLE

SERRTB:

;*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 (SERRPC).
;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

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

;*NOTE: ERROR VECTOR TABLE (SERRTB) HAS BEEN MODIFIED,
ELIMINATING UNUSED VALUE FOR DATA FORMAT POINTER.
ERROR TYPING ROUTINE HAS ALSO BEEN MODIFIED
ACCORDINGLY.

EMV001: .WORD EMA,DHA,DTA ;FPS BAD
EMV002: .WORD EMH,DHA,DTB ;FEC BAD
EMV003: .WORD EMC,DHC,DTC ;FEA BAD
EMV004: .WORD EMD,DHD,DTD ;PS CC-S BAD
EMV005: .WORD EME,DHE,DTE ;BAD ADDR IN RO
EMV006: .WORD EMF,DHF,DTF ;BAD SP
EMV007: .WORD EMP,0,0 ;DIDNT TRAP, SHOULD HAVE
EMV010: .WORD EMH,DHI,DTI ;RESULT BAD - D
EMV011: .WORD EMH,DHH,DTH ;RESULT BAD - F, L
EMV012: .WORD EMH,DHG,DTG ;RESULT BAD - I
EMV013: .WORD EMH,DHE,DTE ;RESULT BAD - RO
;***** VECTORS FOR TRAP/FEC TESTER ROUTINE *****
EMV014: .WORD EMP,0,0 ;NO TRAP, SHOULD HAVE TRAP/FEC TSTR
EMV015: .WORD EMQ,0,0 ;TRAPPED, SHOULDN'T HAVE
EMV016: .WORD EMR,DAL,DTAG ;SP OK
EMV017: .WORD EMS,DHL,DTAH ;STORED PC
EMV020: .WORD EMT,DHL,DTAI ;STORED PS
EMV021: .WORD EMU,DHL,DTAJ ;LOADED PS
EMV022: .WORD EMI,DHK,DTK ;FEC/FEA BAD
EMV023: .WORD EMA,DHJ,DTJ ;FPS BAD
;***** VECTOR FOR UNEXPECTED FPP TRAP *****
EMV024: .WORD EMJ,DHM,DTL ;UNEXPECTED TRAP

350										
351										
352										
353										
354										
355	001464	000000								
356	001466	000000								
357	001470	000000								
358	001472	000000								
359	001474	000000								
360	001476	000000								
361	001500	000000								
362	001502	000000								
363	001504	000000								
364	001506	000000								
365	001510	000000								
366	001512	000000								
367	001514	000000								
368										
369	001516	000000								
370										
371	001520	000000								
372	001522	000000								
373	001524	000000								
374										
375										
376	001526	000000								
377	001530	000000								
378	001532	000000								
379	001534	000000								
380	001536	000000								
381	001540	000000								
382	001542	000000								
383	001544	000000								
384										
385										
386	001546	040200	000000	000000						
387	001554	000000								
388	001556	177777								
389	001560	000000	177777	000000						
390	001566	177777	000000	000000						
391	001574									
392	001574	177777	000001	000000						
393	001602	177777								
394	001604	001560								
395	001606	001566								
396	001610	001726								
397	001612	077777	000000	177777						
398	001620	000000								
399	001622	100000	000000	000000						
400	001630	000000								
401	001632	001170								
402	001634	040252	125252	125252						
403	001642	125252								
404	001644	040325	052525	052525						
405	001652	052525								

```

.SBTTL PROGRAM DEFINED COMMON TAGS
:*VARIABLES
FPS: .WORD 0 ;FPS STORED HERE AFTER STFPS
FEC: .WORD 0 ;FEC STORED HERE AFTER STST
FEA: .WORD 0 ;FEA STORED HERE AFTER STST
FPPGPC: .WORD 0 ;OLD PC SAVED HERE AFTER TRAP
FPPGPS: .WORD 0 ;OLD PS SAVED HERE AFTER TRAP
FPPOSP: .WORD 0 ;SP AFTER TRAP
EXPFEA: .WORD 0 ;EXPECTED FEA
EXPRTS: .WORD 0 ;EXPECTED RETURN ADDR
EXPSP: .WORD 0 ;EXPECTED SP VALUE AFTER TRAP
OPCRCV: .WORD 0 ;PC PUSHED ON STACK AFTER TRAP
OPSRCV: .WORD 0 ;PS PUSHED ON STACK AFTER TRAP
NPSLOD: .WORD 0 ;PS THAT WAS LOADED AFTER TRAP
OSPRCV: .WORD 0 ;SP AFTER FPP TRAP

CLKPRS: .WORD 000000 ;BIT07=1 -> KW11-L PRESENT
;BIT15=1 -> KW11-P PRESENT
KW11LC: .WORD 0 ;KW11-L COUNT OF # OF INTERRUPTS
KW11PC: .WORD 0 ;KW11-P COUNT OF # OF INTERRUPTS
KW11PR: .WORD 0 ;KW11-P COUNT SET

:*REGISTER CONTENTS, AT ERROR, STORED HERE
EREG0: .WORD 0
EREG1: .WORD 0
EREG2: .WORD 0
EREG3: .WORD 0
EREG4: .WORD 0
EREG5: .WORD 0
EREG6: .WORD 0
EREG7: .WORD 0

:*CONSTANTS
FLTONE: .WORD 040200,0,0,0
D1010: .WORD -1
D0101: .WORD 0,-1,0
D1001: .WORD -1,0,0
DSMALL: .WORD -1,1,0,-1
WEIRD: .WORD -1,1,0,-1
AD0101: .WORD D0101
AD1001: .WORD D1001
AD1000: .WORD D1000
DBIG: .WORD 77777,0,-1,0
DMZERO: .WORD 100000,0,0,0
ASREG0: .WORD $REG0
DALTA: .WORD 40252,125252,125252,125252
DALTB: .WORD 40325,52525,52525,52525

```

406	001654	040325	052525	052525	DALTC:	.WORD	40325,52525,52525,52526
407	001662	052526					
408	001664	040000	000000	000000	D40:	.WORD	40000,0,0,0
409	001672	000000					
410	001674	037400	000000	000000	D37:	.WORD	37400,0,0,0
411	001702	000000					
412	001704	040600	000000	000000	D46:	.WORD	40600,0,0,0
413	001712	000000					
414	001714	020000	000000	000000	D20:	.WORD	20000,0,0,0
415	001722	000000					
416	001724	000000			D0100:	.WORD	0
417	001726	177777	000000	000000	D1000:	.WORD	-1,0,0,0
418	001734	000000					
419	001736	000100	000000	000000	D0100X:	.WORD	100,0,0
420	001744	000000	177777	177777	D0111:	.WORD	0,-1,-1,-1
421	001752	177777					
422	001754	000000	054321		D5T01:	.WORD	0,54321
423	001760	001754			AD5T01:	.WORD	D5T01
424	001762	043661	121000	000000	F5T01:	.WORD	43661,121000,0,0
425	001770	000000					

426							
427							
428							
429	001772	005015	005012	042115	: #MESSAGES FOR BEGIN PROGRAM/START OF PASS		
430	002000	030455	026461	050504	BGNMES:	.ASCII	<CR><LF><LF><LF>"MD-11-DQFPC-"
431	002006	050106	026503				
432	002012	027102			.ASCII	"B."	
433	002014	027056			.ASCII	"."	
434	002016	042120	026520	030461	.ASCIIZ	"POP-11/6X F.P.U. INSTRUCTION EXERCISER"<CR><LF>	
435	002024	033057	020130	027106			
436	002032	027120	027125	044440			
437	002040	051516	051124	041525			
438	002046	044524	047117	042440			
439	002054	042530	041522	051511			
440	002062	051105	005015	000			
441	002067	015	050012	051501	NWPAS1:	.ASCIIZ	<CR><LF>"PASS #"
442	002074	020123	000043				

```

443
444
445
446
447
448
449
450 002400
451
452
453 002400 012706 001100
454 002404 005026
455 002406 022706 001144
456 002412 001374
457 002414 012706 001100
458
459 002420 012737 015244 000020
460 002426 012737 000340 000022
461 002434 012737 015522 000030
462 002442 012737 000340 000032
463 002450 012737 017156 000034
464 002456 012737 000340 000036
465 002464 012737 017224 000024
466 002472 012737 000340 000026
467 002500 013737 014234 014226
468 002506 005037 001230
469 002512 005037 001232
470 002516 012737 000001 001120
471 002524 012737 002524 001110
472 002532 012737 002532 001112
473
474
475 002540 013746 000004
476 002544 012737 002600 000004
477 002552 012737 177570 001144
478 002560 012737 177570 001146
479 002566 022777 177777 176350
480 002574 001012
481
482 002576 000403
483 002600 012716 002606 64S:
484 002604 000002
485 002606 012737 000176 001144 65S:
486 002614 012737 000174 001146
487 002622 012637 000004 66S:
488
489 002626 005037 001252
490 002632 132737 000200 001265
491 002640 001403
492 002642 012737 001266 001144
493 002650
494
495
496 002650 012737 015160 000244
497 002656 012737 000300 000246
498

```

```

.SBTTL START OF PASS ROUTINE

;;*****
.ENABL AMA ;ASSEMBLE ALL RELATIVE REFERENCES AS ABSOLUTE
;;*****

START:
.SBTTL INITIALIZE THE COMMON TAGS
;;CLEAR THE COMMON TAGS (SCMTAG) AREA
MOV #SCMTAG,R6 ;:FIRST LOCATION TO BE CLEARED
CLR (R6)+ ;:CLEAR MEMORY LOCATION
CMP #SMR,R6 ;:DONE?
BNE -6 ;:LOOP BACK IF NO
MOV #STACK,SP ;:SETUP THE STACK POINTER
;;INITIALIZE A FEW VECTORS
MOV #SSCOPE,#IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE
MOV #340,#IOTVEC+2 ;:LEVEL 7
MOV #SEERROR,#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,#PMRVEC ;:POWER FAILURE VECTOR
MOV #340,#PMRVEC+2 ;:LEVEL 7
MOV #SENDCI,#SEOPCT ;:SETUP END-OF-PROGRAM COUNTER
CLR STIMES ;:INITIALIZE NUMBER OF ITERATIONS
CLR #ESCAPE ;:CLEAR THE ESCAPE ON ERROR ADDRESS
MOV #1,#SERMAX ;:ALLOW ONE ERROR PER TEST
MOV #.#,#SLPADR ;:INITIALIZE THE LOOP ADDRESS FOR SCOPE
MOV #.#,#SLPERR ;: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 #64S,#ERRVEC ;:SET UP ERROR VECTOR
MOV #DSMR,#SMR ;:SETUP FOR A HARDWARE SWICH REGISTER
MOV #DDISP,#DISPLAY ;:AND A HARDWARE DISPLAY REGISTER
CMP #-1,#SMR ;:TRY TO REFERENCE HARDWARE SMR
BNE 66S ;:BRANCH IF NO TIMEOUT TRAP OCCURRED
;AND THE HARDWARE SMR IS NOT = -1
BR 65S ;:BRANCH IF NO TIMEOUT
MOV #65S,(SP) ;:SET UP FOR TRAP RETURN
RTI
MOV #SWREG,#SMR ;:POINT TO SOFTWARE SMR
MOV #DISPREG,#DISPLAY ;:RESTORE ERROR VECTOR
MOV (SP)+,#ERRVEC
CLR #PASS ;:CLEAR PASS COUNT
BITB #APTSIZE,#ENVH ;:TEST USER SIZE UNDER APT
BEQ 67S ;:YES,USE NON-APT SWITCH
MOV #SSWREG,#SMR ;:NO,USE APT SWITCH REGISTER
67S:
;SET UP FPP UNEXPECTED TRAP CATCHER - - - - -
MOV #FPPILT,#FPPVEC ;:NEW PC AT FPP TRAP
MOV #SPR6,#FPPVEC+2 ;:NEW PS AT FPP TRAP

```

```

499          :FIND, INITIALIZE ANY LINE CLOCKS PRESENT - - - - -
500 002664 005037 001516 CLR CLKPRS ;CLEAR CLOCK PRESENT FLAG
501 002670 013746 000004 MOV @ERRVEC, -(SP) ;SAVE OLD TIMEOUT VECTOR
502 002674 013746 000006 MOV @ERRVEC+2, -(SP) ;
503 002700 012737 002750 000004 MOV @NKW11L, @ERRVEC ;SET NEW TIMEOUT VECTOR
504 002706 012737 000340 000006 MOV @PR7, @ERRVEC+2 ;
505
506 002714 005037 177546 CLR @PLKS ;KW11-L THERE ? (CLEAR STATUS IF YES)
507 002720 052737 000200 001516 BIS @BIT7, CLKPRS ;YES, IT WAS
508 002726 005037 001520 CLR KW11LC ;CLEAR COUNT
509 002732 012737 015022 000100 MOV @IKW11L, @PLKV ;KW11-L SERVICE PC, PS
510 002740 012737 000300 000102 MOV @PR6, @PLKV+2 ;
511 002746 000402 BR TKW11P ;
512 002750 062706 000004 NKW11L: ADD @4, SP ;NO KW11-L (TIMEOUT); POP JUNK OFF STK
513
514 002754 012737 003032 000004 TKW11P: MOV @NKW11P, @ERRVEC ;SET NEW TIMEOUT VECTOR (SAME PS)
515
516 002762 005037 172540 CLR @PLKS ;KW11-P THERE ? (CLEAR STATUS IF YES)
517 002766 052737 100000 001516 BIS @BIT15, CLKPRS ;YES, IT WAS
518 002774 005037 001522 CLR KW11PC ;CLEAR COUNT
519 003000 012737 001777 001524 MOV @1777, KW11PR ;INITIAL COUNT REGISTER
520 003006 012737 015050 000104 MOV @IKW11P, @PLKV ;KW11-P SERVICE PC, PS
521 003014 012737 000300 000106 MOV @PR6, @PLKV+2 ;
522 003022 013737 001524 172542 MOV KW11PR, @PLKR ;SET COUNT REGISTER
523 003030 000402 BR CLKDON ;
524 003032 062706 000004 NKW11P: ADD @4, SP ;NO KW11-P (TIMEOUT); POP JUNK OFF STK
525
526 003036 012637 000006 CLKDON: MOV (SP)+, @ERRVEC+2 ;RESTORE OLD TIMEOUT VECTOR
527 003042 012637 000004 MOV (SP)+, @ERRVEC ;
528
529 003046 104401 001772 TYPE ,BGNMES ;ID MESSAGE AT START
530
531 ;////////////////////////////////////
532 ; MESSAGE ON WHETHER OR NOT HFP UNIT IS PRESENT
533 ;
534 003052 076600 000022 MED ,RWHAMI ;RWHAMI INTO RD
535 003056 032700 000020 BIT @BIT04, RD ;IS THERE A HFP UNIT ?
536 003062 001403 BEQ 66$ ;NO, BR
537 003064 104401 003100 TYPE ,64$ ;INDICATE FP11-E PRESENT
538 003070 000453 BR NEWPAS ;GO FOR SUBPASS INIT
539 003072 104401 003140 66$: TYPE ,65$ ;INDICATE NO FP11-E
540 003076 000450 BR NEWPAS ;GO FOR SUBPASS INIT
541
542 003100 005015 020052 050106 64$: .ASCIZ <15><12>*" FP11-E HFP UNIT PRESENT *"<15><12>
543 003106 030461 042455 044040
544 003114 050106 052440 044516
545 003122 020124 051120 051505
546 003130 047105 020124 006452
547 003136 000012
548 003140 005015 020052 047516 65$: .ASCIZ <15><12>*" NO FP11-E HFP UNIT - ALL TESTS WFP ONLY *"<15><12>
549 003146 043040 030520 026461
550 003154 020105 043110 020120
551 003162 047125 052111 026440
552 003170 040440 046114 052040
553 003176 051505 051524 053440
554 003204 050106 047440 046116

```

```

555 003212 020131 006452 000012
556
557
558
559
560
561
562
563
564
565 003220 012706 001100 NEWPAS: MOV #STACK,SP ;RESET STACK PTR
566
567 003224 032777 010000 175712 BIT #BIT12,2SWR ;INHIBIT STATUS TYPEOUTS ?
568 003232 001011 BNE SUBPAS ;BR IF YES
569
570 003234 104401 002067 TYPE NWPAS1 ;"PASS #"
571 003240 013746 001252 MOV $PASS,-(SP) ;PASS COUNT INTO ...
572 003244 005216 INC (SP) ; 1-N RANGE
573 003246 104403 TYPOS ;TYPE OCTAL
574 003250 006 000 .BYTE 6,0 ; 6 DIGITS, NO LEADING ZEROS
575 003252 104401 001241 TYPE ,$CRLF ;END THE LINE
576
577
578
579
580
581
582 003256 012706 001100 SUBPAS: MOV #STACK,SP ;RESET SP FOR INSURANCE
583
584 003262 076600 000022 MED ,RWHAMI ;GET WHAMI INTO RO
585 003266 032700 000020 BIT #BIT04,RO ;1=HFP PRESENT, 0=NO
586 003272 001430 BEQ 20$ ;IF NO HFP, TEST WARM ONLY
587
588 003274 076600 000144 MED ,RFLAG ;GET FLAGS INTO RO
589
590 003300 032777 000002 175636 BIT #SW01,2SWR ;SW01: 1=HFP OR WFP TEST ONLY
591 003306 001413 BEQ 1$ ; 0=ALTERNATE HFP/WFP PER PASS
592
593 003310 032777 000001 175626 BIT #SW00,2SWR ;SW00: 1=HFP ONLY
594 003316 001403 BEQ 2$ ; 0=HFP ONLY
595 003320 042700 010000 BIC #BIT12,RO ;CLEAR HFP ENABLE FLAG<5> FOR WFP
596 003324 000402 BR 3$ ;
597 003326 052700 010000 2$: BIS #BIT12,RO ;SET HFP ENABLE FLAG<5> FOR HFP
598 003332 076600 000344 3$: MED ,WFLAG ;REWRITE FLAGS
599
600 003336 032700 010000 1$: BIT #BIT12,RO ;TEST WHO'S ENABLED: HOT, WARM
601 003342 001404 BEQ 20$ ;SET APPROPRIATE HEADER:
602
603 003344 012737 017414 015766 19$: MOV #ASCOT,HOTWRM ;"HOT: "
604 003352 000403 BR 21$ ;
605 003354 012737 017422 015766 20$: MOV #ASCWRM,HOTWRM ;"WARM: "
606 003362
607
608
609 003362 105737 001516 ;*START KW11-L LINE CLOCK IF PRESENT
610 003366 100003 †STB CLKPRS ;KW11-L PRESENT ?
BPL NKWLI ;NO

```

```

611 003370 052737 000100 177546      BIS      #BIT6,2#LKS      ;YES, START IT
612
613                                     :#START KW11-P PROGRAMMABLE LINE CLOCK, IF PRESENT
614 003376 005737 001516      NKWL1:  IST      CLKPRS      :KW11-P PRESENT ?
615 003402 100003                                     :NO
616 003404 052737 000101 172540      BIS      #BIT6+BIT0,2#PLKS      ;YES, START IT
617
618 003412 005037 177776      NKWP1:  CLR      2#PSW      ;SETUP FOR INTERRUPT ENABLE, PRO
619
620

```

```

621
622 ;*****
623 ;.DSABL AMA ;ASSEMBLE ALL REFERENCES AS THEY ARE PRINTED
624 ;*****
625
626 ;*****
627 ;#TEST 1 TEST OF WRITABILITY OF FPS
628 ;*****
629 TST1: SCOPE
630 LDFPS #147757 ;TEST FPS WITH ALL ONES
631 STFPS FPS ;GET STATUS
632 CMP #147757,FPS ;CHECK IT
633 BEQ +4
634 ERROR i ;FPS NOT 147777
635
636 LDFPS #40000 ;TEST FPS WITH 40000
637 STFPS FPS ;STORE FLOATING POINT STATUS
638 CMP #40000,FPS ;CHECK FLOATING POINT STATUS
639 BEQ +4 ;BRANCH IF OK
640 ERROR i ;FPS NOT EQUAL TO 40000
641
642 SETD ;TEST OF DOUBLE BIT ON A 1
643 STFPS FPS ;STORE FLOATING POINT STATUS
644 CMP #40200,FPS ;CHECK FLOATING POINT STATUS
645 BEQ +4 ;BRANCH IF OK
646 ERROR i ;FPS NOT EQUAL TO 40200
647
648 SETF ;TEST OF DOUBLE BIT ON A 0
649 STFPS FPS ;STORE FLOATING POINT STATUS
650 CMP #40000,FPS ;CHECK FLOATING POINT STATUS
651 BEQ +4 ;BRANCH IF OK
652 ERROR i ;FPS NOT EQUAL TO 40000
653
654 SETL ;TEST OF LONG BIT ON A 1
655 STFPS FPS ;STORE FLOATING POINT STATUS
656 CMP #40100,FPS ;CHECK FLOATING POINT STATUS
657 BEQ +4 ;BRANCH IF OK
658 ERROR i ;FPS NOT EQUAL TO 40100
659
660 SETI ;TEST OF LONG BIT ON A 0
661 STFPS FPS ;STORE FLOATING POINT STATUS
662 CMP #40000,FPS ;CHECK FLOATING POINT STATUS
663 BEQ +4 ;BRANCH IF OK
664 ERROR i ;FPS NOT EQUAL TO 40000

```

```

665
666
667
668 003564 000004
669 003566 170127 040017
670 003572 170000
671 003574 013700 177776
672 003600 042700 177760
673 003604 022700 000017
674 003610 001401
675 003612 104004
676
677 003614 170127 040012
678 003620 170000
679 003622 013700 177776
680 003626 042700 177760
681 003632 022700 000012
682 003636 001401
683 003640 104004
684
685 003642 170127 040005
686 003646 170000
687 003650 013700 177776
688 003654 042700 177760
689 003660 022700 000005
690 003664 001401
691 003666 104004

```

```

*****
:TEST 2 TEST OF CFCC
*****
TST2: SCOPE
LDFPS #40017 ;LOAD ALL STATUS BITS TO 1'S
CFCC ;GET THEM INTO PS
MOV @#PS,RO ;GET THEM FOR TYPING
BIC #177760,RO ;CLEAR JUNK
CMP #17,RO ;ALL SET?
BEQ .+4
ERROR 4 ;PS NOT 17

LDFPS #40012 ;LOAD FPS WITH 12
CFCC ;GET INTO PS
MOV @#PS,RO ;GET FOR TYPING
BIC #177760,RO ;CLEAR JUNK
CMP #12,RO ;SAME AS LD?
BEQ .+4
ERROR 4 ;PS NOT 12

LDFPS #40005 ;LOAD FPS WITH 5
CFCC ;GET BITS
MOV @#PS,RO ;GET FOR TYPING
BIC #177760,RO ;CLEAR JUNK
CMP #5,RO ;SAME?
BEQ .+4
ERROR 4 ;PS NOT 5

```


G03

FPU INSTR EXERCISER
DGFPCB.P11 04-MAY-77

MACY11 27(1006)
19:34

04-MAY-77 19:35 PAGE 17
T3 TEST OF LDD, STD, CMPD OF -1,0,-1,0

SEQ 0018

```

692
693
694
695 003670 000004
696 003672 170127 047600
697 003676 012700 001556
698 003702 172420
699 003704 022700 001566
700 003710 001401
701 003712 104005
702
703 003714 010601
704 003716 162701 000010
705 003722 174046
706 003724 010600
707 003726 020106
708 003730 001401
709 003732 104006
710
711 003734 170267 175524
712 003740 022767 047610 175516
713 003746 001401
714 003750 104001
715
716 003752 010602
717 003754 012703 001170
718 003760 012223
719 003762 012223
720 003764 012223
721 003766 012223
722 003770 021667 175562
723 003774 001401
724 003776 104010
725 004000 026667 000002 175552
726 004006 001401
727 004010 104010
728 004012 026667 000004 175542
729 004020 001401
730 004022 104010
731 004024 026667 000006 175532
732 004032 001401
733 004034 104010
734
735 004036 062701 000010
736 004042 173426
737 004044 010600
738 004046 020106
739 004050 001401
740 004052 104006
741 004054 170267 175404
742 004060 170000
743 004062 001401
744 004064 104010

```

```

*****
:TEST 3 TEST OF LDD, STD, CMPD OF -1,0,-1,0
*****
TST3: SCOPE
LDFPS #47600
MOV #D1010,R0 ;GET ADDRESS OF DATA WORD
LDD (R0)+,AC0 ;LOAD INTO ACO
CMP #D1010+10,R0 ;IS THE NEW ADDRESS RIGHT?
BEQ .+4
ERROR 5 ;RO NOT D1010+10

MOV SP,R1
SUB #10,R1
STD ACO,-(SP) ;GET THE DATA BACK
MOV SP,R0 ;SAVE THE SP FOR TYPING
CMP R1,SP ;SP DECREMENTED PROPERLY?
BEQ .+4
ERROR 6 ;SP NOT SP-10

STFPS FPS ;STORE FLOATING POINT STATUS
CMP #47610,FPS ;CHECK FLOATING POINT STATUS
BEQ .+4 ;BRANCH IF OK
ERROR 1 ;FPS NOT EQUAL TO 47610

MOV SP,R2 ;MOVE ANSWER TO DISPLAY
MOV #SREG0,R3
MOV (R2)+,(R3)+
MOV (R2)+,(R3)+
MOV (R2)+,(R3)+
MOV (R2)+,(R3)+
CMP (SP),D1010 ;CHECK FIRST PIECE OF DATA
BEQ .+4
ERROR 10 ;DATA IN (SP) NOT D1010
CMP 2(SP),D1010+2 ;CHECK SECOND
BEQ .+4
ERROR 10 ;DATA IN 2(SP) NOT D1010+2
CMP 4(SP),D1010+4 ;CHECK THIRD
BEQ .+4
ERROR 10 ;DATA IN 4(SP) NOT D1010+4
CMP 6(SP),D1010+6 ;CHECK FOURTH
BEQ .+4
ERROR 10 ;DATA IN 6(SP) NOT D1010+6

ADD #10,R1
CMPD (SP)+,AC0 ;RECHECK DATA AND SP
MOV SP,R0 ;SAVE SP FOR TYPING
CMP R1,SP ;CHECK ADDRESS IN SP
BEQ .+4
ERROR 6 ;SP NOT RESTORED
STFPS FPS ;GET STATUS
CFCC ;NOW GET THE FP CONDITION CODES
BEQ .+4 ;IF IT HALTS HERE IT MUST BE THE
ERROR 10 ;CMPD BECAUSE CFCC IS ALREADY CONFIRMED

```

H03

FPU INSTR EXERCISER MACY11 27(1006) 04-MAY-77 19:35 PAGE 18
 DGFPCB.P11 04-MAY-77 19:34 T4 TEST OF LDD, STD, CMPD OF 0,-1,0,-1

SEQ 0019

```

745
746
747
748 004066 000004
749 004070 170127 047600
750 004074 172437 001560
751 004100 170267 175360
752 004104 022767 047604 175352
753 004112 001401
754 004114 104001
755
756 004116 012700 001170
757 004122 174010
758 004124 026767 175430 175036
759 004132 001401
760 004134 104010
761 004136 026767 175420 175026
762 004144 001401
763 004146 104010
764 004150 026767 175410 175016
765 004156 001401
766 004160 104010
767 004162 026767 175400 175006
768 004170 001401
769 004172 104010
770
771 004174 012704 001602
772 004200 173474 000002
773
774
775
776 004204 170267 175254
777 004210 170000
778 004212 001401
779 004214 104010

;*****
;#TEST 4    TEST OF LDD, STD, CMPD OF 0,-1,0,-1
;*****
↑ST4:    SCOPE
         LDFPS    #47600
         LDD    2#00101,ACO    ;LOAD 0,-1,0,-1 INTO ACO #PIC*
         STFPS    FPS    ;STORE FLOATING POINT STATUS
         CMP    #47604,FPS    ;CHECK FLOATING POINT STATUS
         BEQ    .+4    ;BRANCH IF OK
         ERROR    1    ;FPS NOT EQUAL TO 47604

         MOV    #SREG0,R0    ;ADDRESS TO BE STORED INTO
         STD    ACO,(R0)    ;STORE IT INTO SREG0-3 #PIC*
         CMP    D0101,SREG0    ;FIRST WORD OK?
         BEQ    .+4
         ERROR    10    ;SREG0 NOT D0101
         CMP    D0101+2,SREG1    ;SECOND
         BEQ    .+4
         ERROR    10    ;SREG1 NOT D0101+2
         CMP    D0101+4,SREG2    ;THIRD
         BEQ    .+4
         ERROR    10    ;SREG2 NOT D0101+4
         CMP    D0101+6,SREG3    ;FOURTH
         BEQ    .+4
         ERROR    10    ;SREG3 NOT D0101+6

         MOV    #ADD101-2,R4    ;ADDRESS-2 OF DATA
         CMPD    2(4),ACO    ;CHECK DATA IN ACO #PIC*
         ;NOTE: CMPD SEES A ZERO EXP IN ACO, AND THUS
         ;    AFTER ITS EXECUTION ACO ← (0,0,0,0),
         ;    A TRUE FLTPT ZERO
         STFPS    FPS    ;GET STATUS
         CFCC    ;GET CONDITION CODES
         BEQ    .+4
         ERROR    10    ;CMPD FAILED
  
```

```

780
781
782
783 004216 000004
784
785 004220 170127 047400
786 004224 172467 175326
787 004230 012700 001170
788 004234 174020
789 004236 022700 001174
790 004242 001401
791 004244 104005
792
793 004246 026767 175304 174714
794 004254 001401
795 004256 104011
796 004260 026767 175274 174704
797 004266 001401
798 004270 104011
799
800 004272 170011
801 004274 012700 001200
802 004300 174040
803 004302 022700 001170
804 004306 001401
805 004310 104005
806
807 004312 012700 001610
808 004316 173430
809 004320 022700 001612
810 004324 001401
811 004326 104005
812
813 004330 170267 175130
814 004334 170000
815 004336 001401
816 004340 104011

```

```

*****
;TEST 5 TEST OF LDF, STF, CMPF OF -1,0
*****
TST5: SCOPE
LDFPS #47400 ;NOTE: ACO = (0,0,0,0) FROM PREV TEST
LDF D1010,ACO ;SET FLOATING MODE
MOV #SREG0,RO ;LOAD -1,0 INTO ACO
STF ACO,(RO)+ ;POINTED TO ANSWER AREA *PIC*
CMP #SREG2,RO ;STORE RESULT
BEQ .+4 ;INCREMENTED PROPERLY
ERROR 5 ;RO NOT SREG0+4

CMP D1010,SREG0 ;CHECK FIRST WORD
BEQ .+4
ERROR 11 ;SREG0 NOT D1010
CMP D1010+2,SREG1 ;SECOND
BEQ .+4
ERROR 11 ;SREG1 NOT D1010+2

SETD ;GO TO DOUBLE MODE
MOV #SREG4,RO ;ADDRESS OF DATA+10
STD ACO,-(RO) ;GET DATA
CMP #SREG0,RO ;CHECK FOR PROPER DECREMENTATION
BEQ .+4
ERROR 5 ;RO NOT SREG0

MOV #AD1000,RO ;LOAD ADDRESS OF ADDRESS OF DATA
CMPD 2(RO)+,ACO ;CHECK THE DATA
CMP #AD1000+2,RO ;RO GETS INCREMENTED BY 2
BEQ .+4
ERROR 5 ;RO NOT AD1001+2

STFPS FPS ;GET STATUS
CFCC ;COPY CONDITION CODES
BEQ .+4 ;EITHER CMPD FAILED OR THE
ERROR 11 ;LDF MODIFIED RIGHT HALF

```

```

817
818
819
820 004342 000004
821 004344 170127 042400
822 004350 172427 140640
823 004354 172527 140200
824 004360 172627 000000
825 004364 172727 040640
826 004370 174004
827 004372 174305
828
829 004374 174067 174570
830 004400 173427 140640
831 004404 170267 175054
832 004410 170000
833 004412 001401
834 004414 104011
835
836 004416 174137 001170
837 004422 173567 174542
838 004426 170267 175032
839 004432 170000
840 004434 001401
841 004436 104011
842 004440 012704 001170
843 004444 174214
844 004446 173667 174516
845 004452 170267 175006
846 004456 170000
847 004460 001401
848 004462 104011
849
850 004464 174377 175142
851 004470 173767 174474
852 004474 170267 174764
853 004500 170000
854 004502 001401
855 004504 104011
856
857 004506 173404
858 004510 170267 174750
859 004514 170000
860 004516 001401
861 004520 104000
862
863 004522 173705
864 004524 170267 174734
865 004530 170000
866 004532 001401
867 004534 104000
    
```

```

*****
:TEST 6 TEST OF LDF, STF, CMPF WITH (<=>) IN ALL AC'S
*****
TST6: SCOPE
LDFPS 847400 ;LOAD FLOATING MODE
LDF 8-5,AC0 ;LOAD AC0 WITH -5
LDF 8-1,AC1 ;LOAD AC1 WITH -1
LDF 80,AC2 ;LOAD AC2 WITH 0
LDF 85,AC3 ;LOAD AC3 WITH 5
STF AC0,AC4 ;LOAD AC4 WITH -5
STF AC3,AC5 ;LOAD AC5 WITH 5

STF AC0,SREG0 ;GET AC0
CMPF 8-5,AC0 ;CHECK IT
STFPS FPS ;GET STATUS
CFCC ;GET CC
BEQ +4
ERROR 11 ;AC0 NOT -5

STF AC1,2#SREG0 ;GET AC1
CMPF SREG0,AC1 ;CHECK IT
STFPS FPS ;GET STATUS
CFCC ;GET CC
BEQ +4
ERROR 11 ;AC1 NOT -1
MOV 8#SREG0,R4 ;POINTER TO ANSWER AREA
STF AC2,(4) ;PUT DATA INTO SREG0
CMPF SREG0,AC2 ;CHECK DATA
STFPS FPS ;GET STATUS
CFCC ;GET CC
BEQ +4
ERROR 11 ;AC2 NOT 0

STF AC3,2#SREG0 ;PUT DATA INTO SREG0
CMPF SREG0,AC3 ;CHECK DATA
STFPS FPS ;GET STATUS
CFCC ;GET CC
BEQ +4
ERROR 11 ;AC3 NOT 5

CMPF AC4,AC0 ;CHECK AC4 FOR -5
STFPS FPS ;GET STATUS
CFCC ;GET CC
BEQ +4
ERROR 0 ;AC0 NOT AC4

CMPF AC5,AC3 ;CHECK AC5 FOR 5
STFPS FPS ;GET STATUS
CFCC ;GET CC
BEQ +4
ERROR 0 ;AC3 NOT AC5
    
```

K03

FPU INSTR EXERCISER MACY11 27(1006) 04-MAY-77 19:35 PAGE 21
DQFPCB.P11 04-MAY-77 19:34 T7 TEST OF CMPF WITH DATA IN ACO-ACS

SEQ 0022

```

868 .....
869 .....
870 .....
871 004536 000004 .....
872 004540 170127 041000 .....
873 004544 173405 .....
874 004546 170267 174712 174704 .....
875 004552 022767 041000 .....
876 004560 001401 .....
877 004562 104001 .....
878 .....
879 004564 173704 .....
880 004566 170267 174672 .....
881 004572 022767 041010 174664 .....
882 004600 001401 .....
883 004602 104001 .....
884 .....
885 004604 173767 175002 .....
886 004610 170267 174650 .....
887 004614 022767 041000 174642 .....
888 004622 001401 .....
889 004624 104001 .....
890 .....

```

```

*****
*TEST 7 TEST OF CMPF WITH DATA IN ACO-ACS
*****
TST7: SCOPE
LDFPS 41000 ;LOAD STATUS WITH 0
CMPF ACS,ACO ;CMP 5 TO -5
STFPS FPS ;STORE FLOATING POINT STATUS
CMP 41000,FPS ;CHECK FLOATING POINT STATUS
BEQ +4 ;BRANCH IF OK
ERROR i ;FPS NOT EQUAL TO 41000

CMPF AC4,AC3 ;CMP -5 TO 5
STFPS FPS ;STORE FLOATING POINT STATUS
CMP 41010,FPS ;CHECK FLOATING POINT STATUS
BEQ +4 ;BRANCH IF OK
ERROR i ;FPS NOT EQUAL TO 41010

CMPF DBIG,AC3 ;MAKE IT OVERFLOW
STFPS FPS ;GET STATUS
CMP 41000,FPS ;CHECK FLOATING POINT STATUS
BEQ +4
ERROR i ;FPS NOT 141002

```

```

891 .....
892 .....
893 .....
894 004626 000004 .....
895 004630 170127 040000 .....
896 004634 170501 .....
897 004636 170267 174622 .....
898 004642 022767 040010 174614 .....
899 004650 001401 .....
900 004652 104001 .....
901 .....
902 004654 170505 .....
903 004656 170267 174602 .....
904 004662 022767 040000 174574 .....
905 004670 001401 .....
906 004672 104001 .....
907 .....
908 004674 170502 .....
909 004676 170267 174562 .....
910 004702 022767 040004 174554 .....
911 004710 001401 .....
912 004712 104001 .....
913 .....
914 004714 170527 177777 .....
915 004720 000401 .....
916 004722 104001 .....
917 004724 170267 174534 .....
918 004730 022767 040010 174526 .....
919 004736 001401 .....
920 004740 104001 .....
921 .....
922 004742 170011 .....
923 004744 170527 000000 .....
924 004750 000403 .....
925 004752 104001 .....
926 004754 104001 .....
927 004756 104001 .....
928 004760 170267 174500 .....
929 004764 022767 040204 174472 .....
930 004772 001401 .....
931 004774 104001 .....

TST10: SCOPE
LDFPS #40000
TSTF AC1 ;TEST AC1 = -1
STFPS FPS ;GET STATUS
CMP #40010,FPS ;CHECK STATUS
BEQ .+4
ERROR 1 ;N BIT NOT SET

TSTF AC5 ;TEST AC5 = 5
STFPS FPS ;GET STATUS
CMP #40000,FPS ;CHECK STATUS
BEQ .+4
ERROR 1 ;NOT 0

TSTF AC2 ;TEST AC2 = 0
STFPS FPS ;GET STATUS
CMP #40004,FPS ;CHECK STATUS
BEQ .+4
ERROR 1 ;Z BIT NOT SET

TSTF #-1 ;TEST FOR THE N BIT IN LINE
BR .+4 ;SHOULD GO HERE
ERROR 1 ;INCREMENTED BY 4 NOT 2
STFPS FPS ;GET STATUS
CMP #40010,FPS ;CHECK THE N BIT
BEQ .+4
ERROR 1 ;N BIT NOT SET

SETD ;SET DOUBLE MODE
TSTD #0 ;TEST FOR Z BIT IN LINE
BR .+10 ;SHOULD GO HERE
ERROR 1 ;NOT HERE
ERROR 1 ;OR HERE
ERROR 1 ;OR HERE
STFPS FPS ;GET STATUS
CMP #40204,FPS ;CHECK STATUS
BEQ .+4
ERROR 1 ;Z BIT NOT SET

```

```

932
933
934
935 004776 000004
936 005000 170127 040200
937 005004 172467 174546
938 005010 174067 174154
939 005014 170001
940 005016 170467 174146
941 005022 170267 174436
942 005026 022767 040004 174430
943 005034 001401
944 005036 104001
945
946 005040 170567 174124
947 005044 170000
948 005046 001401
949 005050 104011
950
951 005052 026767 174504 174114
952 005060 001401
953 005062 104010
954 005064 026767 174474 174104
955 005072 001401
956 005074 104010
957
958 005076 170011
959 005100 170400
960 005102 170267 174356
961 005106 022767 040204 174350
962 005114 001401
963 005116 104001
964 005120 174067 174044
965 005124 170500
966 005126 170000
967 005130 001401
968 005132 104010
969
970 005134 172467 174462
971 005140 170400
972 005142 170267 174316
973 005146 022767 040204 174310
974 005154 001401
975 005156 104001
976 005160 174067 174004
977 005164 170500
978 005166 170000
979 005170 001401
980 005172 104010

```

```

*****
: *TEST 11 TEST OF CLRX INSTRUCTIONS
*****
TST11: SCOPE
LDFFS #40200 ;DOUBLE MODE
LDO D1010,AC0 ;LOAD A -1
STD ACO,SREG0 ;PUT INTO SREG0
SETF ;SET FLOATING
CLRF SREG0 ;CLEAR IT OUT
STFPS FPS ;STORE FLOATING POINT STATUS
CMP #40004,FPS ;CHECK FLOATING POINT STATUS
BEQ .+4 ;BRANCH IF OK
ERROR 1 ;FPS NOT EQUAL TO 40004

TSTF SREG0 ;TEST FOR ZERO
CFCC ;GET CC
BEQ .+4
ERROR 11 ;ACO NOT ZERO

CMP D1010+4,SREG2 ;CHECK THIRD WORD
BEQ .+4
ERROR 10 ;SREG2 NOT -1
CMP D1010+6,SREG3 ;CHECK FOURTH
BEQ .+4
ERROR 10 ;SREG3 NOT 0

SETD
CLRD ACO ;CLEAR THE REST OF ACO
STFPS FPS ;STORE FLOATING POINT STATUS
CMP #40204,FPS ;CHECK FLOATING POINT STATUS
BEQ .+4 ;BRANCH IF OK
ERROR 1 ;FPS NOT EQUAL TO 40204
STD ACO,SREG0 ;STORE RESULT
TSTD ACO ;DID IT CLEAR
CFCC ;GET STATUS
BEQ .+4
ERROR 10 ;DID NOT CLEAR

LDO DMZERO,AC0 ;LOAD A MINUS ZERO
CLRD ACO ;CLEAR IT OUT
STFPS FPS ;STORE FLOATING POINT STATUS
CMP #40204,FPS ;CHECK FLOATING POINT STATUS
BEQ .+4 ;BRANCH IF OK
ERROR 1 ;FPS NOT EQUAL TO 40204
STD ACO,SREG0 ;STORE RESULT
TSTD ACO ;CHECK IT
CFCC ;GET CC
BEQ .+4
ERROR 10 ;DID NOT CLEAR

```

```

*****
;#TEST 12      TEST OF NEGX
*****
†ST12:
981          000004          040000          ;LDFPS #40000
982          170127          140640          ;LDF #5,ACD ;LOAD ACD WITH -5
983          172427          174250          ;NEGF ACD ;MAKE IT 5
984 005174    000004          040000          ;STFPS FPS ;STORE FLOATING POINT STATUS
985 005176    170127          174250          ;CMP #40000,FPS ;CHECK FLOATING POINT STATUS
986 005202    172427          040000          174242 ;BEQ .+4 ;BRANCH IF OK
987 005206    170700          104001          ;ERROR 1 ;FPS NOT EQUAL TO 40000
988 005210    170267          173736          ;STF ACD,SREGD ;GET THE RESULT
989 005214    022767          040640          ;CMPF #5,ACD ;CHECK THE RESULT
990 005222    001401          170000          ;CFCC ;GET CC
991 005224    104001          001401          ;BEQ .+4
992 005226    104011          174067          ;ERROR 11 ;RESULT NOT 5
993 005232    173427          170700          ;NEGF SREGD ;MAKE IT -5
994 005236    170000          174210          ;STFPS FPS ;STORE FLOATING POINT STATUS
995 005240    001401          040010          174202 ;CMP #40010,FPS ;CHECK FLOATING POINT STATUS
996 005242    104011          001401          ;BEQ .+4 ;BRANCH IF OK
997 005244    170767          104001          ;ERROR 1 ;FPS NOT EQUAL TO 40010
1000 005250    170267          140640          173674 ;CMP #140640,SREGD ;CHECK THE RESULT
1001 005254    022767          001401          ;BEQ .+4
1002 005262    001401          104011          ;ERROR 11 ;RESULT NOT -5
1003 005264    104001          005767          173666 ;TST SREG1 ;REST 0?
1004 005266    022767          001401          ;BEQ .+4 ;SKIP IF OK
1005 005274    001401          104011          ;ERROR 11
1006 005276    104011          170127          047400 ;LDFPS #47400 ;TURN ON INTERRUPTS
1007 005300    005767          170400          ;CLR ACD ;CLEAR ACD
1008 005304    001401          170700          ;NEGF ACD ;NEGATE IT
1009 005306    104011          170267          174140 ;STFPS FPS ;STORE FLOATING POINT STATUS
1010 005310    170127          047404          174132 ;CMP #47404,FPS ;CHECK FLOATING POINT STATUS
1011 005314    170400          001401          ;BEQ .+4 ;BRANCH IF OK
1012 005316    170700          104001          ;ERROR 1 ;FPS NOT EQUAL TO 47404
1013 005320    170267          174067          173626 ;STF ACD,SREGD ;GET RESULT
1014 005324    022767          170500          ;TSTF ACD ;CHECK IT
1015 005326    001401          170000          ;CFCC ;GET CC
1016 005332    001401          001401          ;BEQ .+4
1017 005334    104001          104011          ;ERROR 11 ;RESULT NOT 0
1018 005336    104011          170011          ;SETD ;SET DOUBLE MODE
1019 005342    170500          170400          ;CLR ACD ;CLEAR ACD
1020 005344    170000          170700          ;NEGF ACD ;NEGATE ACD
1021 005346    001401          170267          174100 ;STFPS FPS ;STORE FLOATING POINT STATUS
1022 005350    104011          047604          174072 ;CMP #47604,FPS ;CHECK FLOATING POINT STATUS
1023 005352    170011          001401          ;BEQ .+4 ;BRANCH IF OK
1024 005354    170400          104001          ;ERROR 1 ;FPS NOT EQUAL TO 47604
1025 005356    170700          174067          173566 ;STF ACD,SREGD ;GET RESULT
1026 005360    170267          170500          ;TSTF ACD ;TEST RESULT
1027 005364    022767          174100          ;SETD
1028 005372    001401          174072          ;CLR ACD
1029 005374    104001          173566          ;NEGF ACD
1030 005376    174067          170500          ;STFPS FPS
1031 005378    170500          174100          ;CMP #47604,FPS
1032 005380    022767          174072          ;BEQ .+4
1033 005382    001401          173566          ;ERROR 1 ;FPS NOT EQUAL TO 47604
1034 005384    104001          170500          ;STF ACD,SREGD
1035 005386    174067          170500          ;TSTF ACD
1036 005402    170500          170500

```


B04

FPU INSTR EXERCISER MACY11 27(1006) 04-MAY-77 19:35 PAGE 25
DQFPCB.P11 04-MAY-77 19:34 T12 TEST OF NEGX

SEQ 0026

1037 005404 170000
1038 005406 001401
1039 005410 104011
1040

CFCC
BEQ i+4
ERROR 11

;GET CC
;RESULT NOT 0

```

1041
1042
1043
1044 005412 000004
1045 005414 170127 040000
1046 005420 172427 140200
1047 005424 174067 173540
1048 005430 170667 173534
1049 005434 170267 174024
1050 005440 022767 040000 174016
1051 005446 001401
1052 005450 104001
1053
1054 005452 022767 040200 173510
1055 005460 001401
1056 005462 104011
1057
1058 005464 005767 173502
1059 005470 001401
1060 005472 104011
1061
1062 005474 170600
1063 005476 170267 173762
1064 005502 174067 173462
1065 005506 173427 040200
1066 005512 170000
1067 005514 001401
1068 005516 104011
1069
1070 005520 172467 174076
1071 005524 170600
1072 005526 170267 173732
1073 005532 022767 040004 173724
1074 005540 001401
1075 005542 104001
1076
1077 005544 174067 173420
1078 005550 170500
1079 005552 170000
1080 005554 001401
1081 005556 104011
1082
1083 005560 170011
1084 005562 172467 173770
1085 005566 170600
1086 005570 170267 173670
1087 005574 022767 040200 173662
1088 005602 001401
1089 005604 104001
1090 005606 174067 173356
1091 005612 173467 173774
1092 005616 170000
1093 005620 001401
1094 005622 104010
1095

*****
:TEST 13 TEST OF ABSX
*****
TST13: SCOPE
LDFPS #40000
LDF # -1, ACO ;LOAD A -1
STF ACO, SREG0 ;GET IT
ABSF SREG0 ;MAKE IT A 1
STFPS FPS ;STORE FLOATING POINT STATUS
CMP #40000, FPS ;CHECK FLOATING POINT STATUS
BEQ +4 ;BRANCH IF OK
ERROR 1 ;FPS NOT EQUAL TO 40000

CMP #40200, SREG0 ;CHECK FOR A 1
BEQ +4
ERROR 11 ;RESULT NOT 1

TST SREG1 ;REST 0?
BEQ +4 ;SKIP IF OK
ERROR 11

ABSF ACO ;ABS IT AGAIN
STFPS FPS ;GET STATUS
STF ACO, SREG0 ;STORE ANSWER
CMPF #1, ACO ;CHECK FOR A 1
CFCC ;GET CC
BEQ +4
ERROR 11 ;RESULT NOT 1

LDF DMZERO, ACO ;LOAD A MINUS ZERO
ABSF ACO ;ABS IT
STFPS FPS ;STORE FLOATING POINT STATUS
CMP #40004, FPS ;CHECK FLOATING POINT STATUS
BEQ +4 ;BRANCH IF OK
ERROR 1 ;FPS NOT EQUAL TO 40004

STF ACO, SREG0 ;GET RESULT
TSTF ACO ;TEST FOR 0
CFCC ;GET CC
BEQ +4
ERROR 11 ;RESULT NOT 0

SETD ;SET DOUBLE MODE
LDD D1010, ACO ;LOAD -1,0,-1,0
ABSD ACO ;ABS IT
STFPS FPS ;STORE FLOATING POINT STATUS
CMP #40200, FPS ;CHECK FLOATING POINT STATUS
BEQ +4 ;BRANCH IF OK
ERROR 1 ;FPS NOT EQUAL TO 40200
STD ACO, SREG0 ;GET RESULT
CMPD DBIG, ACO ;CHECK THE RESULT
CFCC ;GET CC
BEQ +4
ERROR 10 ;RESULT NOT 7777,0,-1,0

```

```

1096
1097
1098
1099 005624 000004
1100 005626 170127 040200
1101 005632 172467 173720
1102 005636 176427 177600
1103 005642 170267 173616
1104 005646 022767 040204 173610
1105 005654 001401
1106 005656 104001
1107
1108 005660 174067 173304
1109 005664 170500
1110 005666 170000
1111 005670 001401
1112 005672 104010
1113
1114 005674 175067 173270
1115 005700 013700 177776
1116 005704 042700 177760
1117 005710 022700 000010
1118 005714 001401
1119 005716 104004
1120
1121 005720 022767 177600 173242
1122 005726 001401
1123 005730 104012
1124
1125 005732 170001
1126 005734 172467 173620
1127 005740 176427 000200
1128 005744 170267 173514
1129 005750 022767 040006 173506
1130 005756 001401
1131 005760 104001
1132
1133 005762 174067 173202
1134 005766 005767 173176
1135 005772 001401
1136 005774 104011
1137
1138 005776 005767 173170
1139 006002 001401
1140 006004 104011
1141
1142 006006 175067 173156
1143 006012 022767 177600 173150
1144 006020 001401
1145 006022 104012
1146
1147 006024 176527 000052
1148 006030 175100
1149 006032 022700 000052
1150 006036 001401
1151 006040 104013

```

```

*****
;#TEST 14 TEST OF LDEXP & STEXP
*****
TST14: SCOPE
LDFPS #40200 ;SET DOUBLE MODE
LDD D1010,ACO ;LOAD A -1,0,-1,0
LDEXP #-200,ACO ;CLEAR THE EXPONENT
STFPS FPS ;STORE FLOATING POINT STATUS
CMP #40204,FPS ;CHECK FLOATING POINT STATUS
BEQ .+4 ;BRANCH IF OK
ERROR 1 ;FPS NOT EQUAL TO 40204

STD ACO,SREGO ;GET THE RESULT
TSTD ACO ;IS IT 0
CFCC
BEQ .+4
ERROR 10 ;ACO NOT 0

STEXP ACO,SREGO ;GET THE RESULT
MOV #PS,RO ;GET PS BITS
BIC #177760,RO ;CLEAR JUNK
CMP #10,RO ;IS IT OK?
BEQ .+4 ;SKIP IF OK
ERROR 4 ;PS IS WRONG

CMP #-200,SREGO ;CHECK IT
BEQ .+4
ERROR 12 ;EXPONENT NOT 0

SETF ;SET FLOATING MODE
LDF D0101,ACO ;LOAD A 0,-1
LDEXP #200,ACO ;SET EXPONENT TO -1
STFPS FPS ;STORE FLOATING POINT STATUS
CMP #40006,FPS ;CHECK FLOATING POINT STATUS
BEQ .+4 ;BRANCH IF OK
ERROR 1 ;FPS NOT EQUAL TO 40006

STF ACO,SREGO ;SAVE RESULT
TST SREGO ;CHECK FIRST WORD
BEQ .+4
ERROR 11 ;SREGO NOT 0

TST SREG1 ;CHECK SECOND WORD
BEQ .+4
ERROR 11 ;SREG1 NOT ZERO

STEXP ACO,SREGO ;GET THE EXPONENT BACK
CMP #-200,SREGO ;CHECK IT
BEQ .+4
ERROR 12 ;EXPONENT NOT -200

LDEXP #52,AC1 ;LOAD ALT 1'S
STEXP AC1,RO ;GET THEM BACK
CMP #52,RO ;OK?
BEQ .+4
ERROR 13 ;EXP NOT 252

```

E04

1152						
1153	006042	176627	000025	LDEXP	#25,AC2	;LOAD OTHER ALT 1'S
1154	006046	175200		STEXP	AC2,RO	;GET IT BACK
1155	006050	022700	000025	CMP	#25,RO	;CHECK IT
1156	006054	001401		BEQ	.+4	
1157	006056	104013		ERROR	13	;EXP NOT 125
1158						

```

1159
1160
1161
1162 006060 000004
1163 006062 170127 047400
1164 006066 172427 040600
1165 006072 172027 040400
1166 006076 170267 173362
1167 006102 022767 047400 173354
1168 006110 001401
1169 006112 104001
1170
1171 006114 174067 173050
1172 006120 173427 040700
1173 006124 170000
1174 006126 001401
1175 006130 104011
1176
1177 006132 173027 041020
1178 006136 170267 173322
1179 006142 022767 047410 173314
1180 006150 001401
1181 006152 104001
1182
1183 006154 174067 173010
1184 006160 173427 140500
1185 006164 170000
1186 006166 001401
1187 006170 104011
1188
1189

```

```

*****
;TEST 15 TEST OF ADDF & SUBF
*****
†ST15: SCOPE
LDFPS #47400 ;LOAD FLOATING MODE
LDF #4,ACO ;LOAD A 4
ADDF #2,ACO ;ADD A 2
STFPS FPS ;STORE FLOATING POINT STATUS
CMP #47400,FPS ;CHECK FLOATING POINT STATUS
BEQ .+4 ;BRANCH IF OK
ERROR 1 ;FPS NOT EQUAL TO 47400

STF ACO,$REGO ;STORE RESULT
CMPF #6,ACO ;CHECK RESULT
CFCC ;GET CC
BEQ .+4
ERROR 11 ;RESULT NOT 6

SUBF #9,ACO ;SUBTRACT 9 FROM 6
STFPS FPS ;STORE FLOATING POINT STATUS
CMP #47410,FPS ;CHECK FLOATING POINT STATUS
BEQ .+4 ;BRANCH IF OK
ERROR 1 ;FPS NOT EQUAL TO 47410

STF ACO,$REGO ;STORE RESULT IN $REGO
CMPF #-3,ACO ;CHECK RESULT
CFCC ;GET CC
BEQ .+4
ERROR 11 ;6 - 9 NOT -3?

```

```

1190
1191
1192
1193 006172 000004
1194 006174 170127 047600
1195 006200 172527 141400
1196 006204 172127 040640
1197 006210 170267 173250
1198 006214 022767 047610 173242
1199 006222 001401
1200 006224 104001
1201
1202 006226 174177 173400
1203 006232 173527 141330
1204 006236 170000
1205 006240 001401
1206 006242 104010
1207
1208 006244 172667 173364
1209 006250 172267 173370
1210 006254 170267 173204
1211 006260 022767 047600 173176
1212 006266 001401
1213 006270 104001
1214
1215 006272 174277 173334
1216 006276 173627 040500
1217 006302 170000
1218 006304 001401
1219 006306 104010
1220
1221 006310 173267 173320
1222 006314 170267 173144
1223 006320 022767 047600 173136
1224 006326 001401
1225 006330 104001
1226
1227 006332 174267 172632
1228 006336 173667 173312
1229 006342 170000
1230 006344 001401
1231 006346 104010

```

```

*****
;TEST 16 TEST OF ADDO AND SUBD
*****
TST16: SCOPE
LDFPS #47600 ;SET DOUBLE MODE
LDD #-32, AC1 ;LOAD A -32.
ADD #5, AC1 ;ADD 5 TO -32.
STFPS FPS ;STORE FLOATING POINT STATUS
CMP #47610, FPS ;CHECK FLOATING POINT STATUS
BEQ +4 ;BRANCH IF OK
ERROR 1 ;FPS NOT EQUAL TO 47610

STD AC1, SASREGO ;GET RESULT
CMPD #-27., AC1 ;CHECK RESULT
CFCC ;GET CC
BEQ +4
ERROR 10 ;-32.+5 NOT -27

LDD DALTA, AC2 ;LOAD ALT 1'S
ADD DALTB, AC2 ;ADD OTHER 1'S
STFPS FPS ;STORE FLOATING POINT STATUS
CMP #47600, FPS ;CHECK FLOATING POINT STATUS
BEQ +4 ;BRANCH IF OK
ERROR 1 ;FPS NOT EQUAL TO 47600

STD AC2, SASREGO ;GET RESULT
CMPD #3, AC2 ;CHECK RESULT
CFCC ;GET CC
BEQ +4
ERROR 10 ;SREGO NOT DALTA

SUBD DALTA, AC2 ;SUBTRACT IT BACK
STFPS FPS ;STORE FLOATING POINT STATUS
CMP #47600, FPS ;CHECK FLOATING POINT STATUS
BEQ +4 ;BRANCH IF OK
ERROR 1 ;FPS NOT EQUAL TO 47600

STD AC2, SREGO ;GET THE RESULT
CMPD DALTC, AC2 ;CHECK IT
CFCC ;GET CC
BEQ +4
ERROR 10 ;SREGO NOT DALTA

```

H04

```

1232
1233
1234
1235 006350 000004
1236 006352 170127 047400
1237 006356 172727 040740
1238 006362 171327 140500
1239 006366 170267 173072
1240 006372 022767 047410 173064
1241 006400 001401
1242 006402 104001
1243
1244 006404 174367 172560
1245 006410 173727 141250
1246 006414 170000
1247 006416 001401
1248 006420 104011
1249
1250 006422 174727 140740
1251 006426 170267 173032
1252 006432 022767 047400 173024
1253 006440 001401
1254 006442 104001
1255
1256 006444 174367 172520
1257 006450 173727 040500
1258 006454 170000
1259 006456 001401
1260 006460 104011

```

```

*****
:TEST 17 TEST OF MULF AND DIVF
*****
TST17: SCOPE
LDFPS #47400 ;LOAD FLOATING MODE
LDF #7,AC3 ;LOAD A7
MULF #-3,AC3 ;X -3
STFPS FPS ;STORE FLOATING POINT STATUS
CMP #47410,FPS ;CHECK FLOATING POINT STATUS
BEQ +4 ;BRANCH IF OK
ERROR 1 ;FPS NOT EQUAL TO 47410

STF AC3,SREG0 ;GET RESULT
CMPF #-21.,AC3 ;CHECK RESULT
CFCC ;GET CC
BEQ +4
ERROR 11 ;X -3 NOT -21.

DIVF #-7,AC3 ;DIVIDE BY -3
STFPS FPS ;STORE FLOATING POINT STATUS
CMP #47400,FPS ;CHECK FLOATING POINT STATUS
BEQ +4 ;BRANCH IF OK
ERROR 1 ;FPS NOT EQUAL TO 47400

STF AC3,SREG0 ;GET RESULT
CMPF #3,AC3 ;CHECK RESULT
CFCC ;GET CC
BEQ +4
ERROR 11 ;-21. / -7 NOT 3

```

```

1261
1262
1263
1264 006462 000004
1265 006464 170127 047600
1266 006470 172427 140640
1267 006474 171027 140500
1268 006500 170267 172760
1269 006504 022767 047600 172752
1270 006512 001401
1271 006514 104001
1272
1273 006516 174067 172446
1274 006522 173427 041160
1275 006526 170000
1276 006530 001401
1277 006532 104010
1278
1279 006534 174427 140400
1280 006540 170267 172720
1281 006544 022767 047610 172712
1282 006552 001401
1283 006554 104001
1284
1285 006556 174067 172406
1286 006562 173427 140760
1287 006566 170000
1288 006570 001401
1289 006572 104010

```

```

*****
:TEST 20 TEST OF MULD AND DIVD
*****
†ST20: SCOPE
LDFPS #47600 ;LOAD DOUBLE MODE
LDD #-5,ACO ;LOAD A -5
MULD #-3,ACO ;MUL BY -3
STFPS FPS ;STORE FLOATING POINT STATUS
CMP #47600,FPS ;CHECK FLOATING POINT STATUS
BEQ .+4 ;BRANCH IF OK
ERROR 1 ;FPS NOT EQUAL TO 47600

STD ACO,$REGD ;GET RESULT
CMPD #15.,ACO ;CHECK RESULT
CFCC ;GET CC
BEQ .+4
ERROR 10 ;-5 X -3 NOT +15.

DIVD #-2,ACO ;15. / -2
STFPS FPS ;STORE FLOATING POINT STATUS
CMP #47610,FPS ;CHECK FLOATING POINT STATUS
BEQ .+4 ;BRANCH IF OK
ERROR 1 ;FPS NOT EQUAL TO 47610

STD ACO,$REGD ;STORE RESULT
CMPD #-7.5,ACO ;CHECK RESULT
CFCC ;GET CC
BEQ .+4
ERROR 10 ;15. / -2 NOT -7.5

```



```

1290
1291
1292
1293 006574 000004
1294 006576 170127 047600
1295 006602 172477 172776
1296 006606 012700 001556
1297 006612 177420
1298 006614 022700 001562
1299 006620 001401
1300 006622 104005
1301
1302 006624 170267 172634
1303 006630 022767 047610 172626
1304 006636 001401
1305 006640 104001
1306
1307 006642 174067 172322
1308 006646 173467 173054
1309 006652 170000
1310 006654 001401
1311 006656 104010
1312
1313 006660 172567 172674
1314 006664 170001
1315 006666 012700 001566
1316 006672 177540
1317 006674 022700 001556
1318 006700 001401
1319 006702 104005
1320
1321 006704 170267 172554
1322 006710 022767 047410 172546
1323 006716 001401
1324 006720 104001
1325
1326 006722 170011
1327 006724 174167 172240
1328 006730 173567 172640
1329 006734 170000
1330 006736 001401
1331 006740 104010
1332
1333 006742 170127 047440
1334 006746 177567 172604
1335 006752 174167 172212
1336 006756 173567 172604
1337 006762 170267 172476
1338 006766 170000
1339 006770 001401
1340 006772 104010
1341
1342 006774 170127 047400
1343 007000 177567 172552
1344 007004 174167 172160
1345 007010 170267 172450

*****
*TEST 21 TEST OF LDCFD,LDCDF
*****
†ST21: SCOPE
LDFPS #47600 ;SET DOUBLE MODE
LDD #D0101,ACO ;LOAD A 0,-1,0,-1
MOV #D1010,RO ;GET ADDRESS OF DATA
LDCFD (RO)+,ACO ;LOAD A -1,0,0,0
CMP #D1010+4,RO ;INC BY 4?
BEQ .+4
ERROR 5 ;RO NOT #D1010+4

STFPS FPS ;STORE FLOATING POINT STATUS
CMP #47610,FPS ;CHECK FLOATING POINT STATUS
BEQ .+4 ;BRANCH IF OK
ERROR 1 ;FPS NOT EQUAL TO 47610

STD ACO,SREGO ;GET ANSWER
CMPD D1000,ACO ;CHECK RESULT
CFCC ;GET CC
BEQ .+4
ERROR 10 ;ACO NOT -1,0,0,0

LDD D0101,AC1 ;LOAD A 0,-1,0,-1
SETF ;SET FLOATING MODE
MOV #D1010+10,RO ;GET ADDRESS OF DATA +10
LDCDF -(RO),AC1 ;LOAD A -1,0
CMP #D1010,RO ;ADDRESS DECREMENT BY 10?
BEQ .+4
ERROR 5 ;RO NOT #D1010

STFPS FPS ;STORE FLOATING POINT STATUS
CMP #47410,FPS ;CHECK FLOATING POINT STATUS
BEQ .+4 ;BRANCH IF OK
ERROR 1 ;FPS NOT EQUAL TO 47410

SETD AC1,SREGO ;GET RESULT
STD WEIRD,AC1 ;CHECK RESULT
CFCC
BEQ .+4
ERROR 10 ;RESULT NOT -1,1,0,-1

LDFPS #47440 ;SET DOUBLE AND TRUNCATE MODES
LDCDF D1010,AC1 ;LOAD IT
STD AC1,SREGO ;GET RESULT
CMPD D1001,AC1 ;CHECK RESULT
STFPS FPS ;GET STATUS
CFCC ;GET CC
BEQ .+4
ERROR 10 ;AC1 NOT -1,0,0,1

LDFPS #47400 ;SET ROUND AND FLOATING MODES
LDCDF D1010,AC1 ;LOAD A -1,0
STD AC1,SREGO ;GET THE RESULT
STFPS FPS ;GET STATUS

```

K04

FPU INSTR EXERCISER MACY11 27(1006) 04-MAY-77 19:35 PAGE 34
DQFPCB.P11 04-MAY-77 19:34 T21 TEST OF LDCFD,LDCDF

SEQ 0035

1346 007014 022767 000001 172150
1347 007022 001401
1348 007024 104010
1349

CMP #1,SREG1
BEQ +4
ERROR i0

;CHECK WORD 2 FOR A 1
;LDCDF DID NOT ROUND PROPERLY

```

1350
1351
1352
1353 007026 000004
1354 007030 170127 040200
1355 007034 172667 172516
1356 007040 170001
1357 007042 176202
1358 007044 170011
1359 007046 174267 172116
1360 007052 173667 172650
1361 007056 170000
1362 007060 001401
1363 007062 104010
1364
1365 007064 172767 172654
1366 007070 176303
1367 007072 174367 172072
1368 007076 173767 172634
1369 007102 170000
1370 007104 001401
1371 007106 104010
1372
1373 007110 172467 172630
1374 007114 170127 000040
1375 007120 176000
1376 007122 170011
1377 007124 174067 172040
1378 007130 173467 172570
1379 007134 170000
1380 007136 001401
1381 007140 104010
1382

```

```

*****
:TEST 22 TEST OF STCFD,STCDF
*****
↑ST22: SCOPE
LDFPS #40200 ;SET DOUBLE MODE
LDD D1010,AC2 ;LOAD A -1,0,-1,0
SETF ;SET FLOATING MODE
STCDF AC2,AC2 ;CLEAR RIGHT HALF
SETD ;SET DOUBLE MODE
STD AC2,$REG0 ;GET RESULT
CMPD D1000,AC2 ;IS IT -1,0,0,0
CFCC ;GET CC
BEQ .+4
ERROR 10 ;AC1 NOT -1,0,0,0

LDD D0111,AC3 ;LOAD A 0,-1,0,-1
STCFD AC3,AC3 ;CLEAR OUT RIGHT HALF!?
STD AC3,$REG0 ;GET RESULT
CMPD D0100X,AC3 ;CHECK RESULT
CFCC ;GET CC
BEQ .+4
ERROR 10 ;SREG2 NOT 100,0,0 (ROUND)

LDD D0111,AC0 ;LOAD 0,-1,0,-1
LDFPS #40 ;FLOATING AND TRUNCATE MODES
STCFD AC0,AC0 ;CLEAR RIGHT HALF
SETD ;SET DOUBLE
STD AC0,$REG0 ;GET RESULT
CMPD D0100,AC0 ;CHECK IT
CFCC ;GET CC
BEQ .+4
ERROR 10 ;AC0 NOT 0,-1,0,0 (TRUNCATE)

```

M04

```

1383
1384
1385
1386 007142 000004
1387 007144 170127 047400
1388 007150 177027 000005
1389 007154 170267 172304
1390 007160 022767 047400 172276
1391 007166 001401
1392 007170 104001
1393
1394 007172 174067 171772
1395 007176 173427 040640
1396 007202 170000
1397 007204 001401
1398 007206 104011
1399
1400 007210 175400
1401 007212 022700 000005
1402 007216 001401
1403 007220 104013
1404
1405 007222 170011
1406 007224 172567 172330
1407 007230 012700 001556
1408 007234 177120
1409 007236 022700 001560
1410 007242 001401
1411 007244 104005
1412
1413 007246 170267 172212
1414 007252 022767 047610 172204
1415 007260 001401
1416 007262 104001
1417
1418 007264 174167 171700
1419 007270 173527 140200
1420 007274 170000
1421 007276 001401
1422 007300 104010
1423
1424 007302 175500
1425 007304 022700 177777
1426 007310 001401
1427 007312 104013
1428
1429 007314 170001
1430 007316 177227 054321
1431 007322 174267 171642
1432 007326 170267 172132
1433 007332 173667 172424
1434 007336 170000
1435 007340 001401
1436 007342 104011
1437
1438 007344 052737 000017 177776

```

```

*****
:TEST 23 TEST OF LDCIF,LDCID,STCFI,STCDI
*****
TST23: SCOPE
LDFPS #47400 ;FLOATING MODE
LDCIF #5,AC0 ;STORE A 5
STFPS FPS ;STORE FLOATING POINT STATUS
CMP #47400,FPS ;CHECK FLOATING POINT STATUS
BEQ +4 ;BRANCH IF OK
ERROR 1 ;FPS NOT EQUAL TO 47400

STF ACO,$REGO ;GET THE RESULT
CMPF #5,AC0 ;CHECK IT
CFCC ;GET CC
BEQ +4
ERROR 11 ;ACO NOT 5.0

STCFI ACO,RO ;CONVERT IT BACK
CMP #5,RO ;CHECK RESULT
BEQ +4
ERROR 13 ;RO NOT 5

SETD ;SET DOUBLE MODE
LDD D0101,AC1 ;LOAD JUNK
MOV #D1010,RO ;LOAD ADDRESS OF DATA
LDCID (RO)+,AC1 ;CONVERT TO -1.0
CMP #D1010+2,RO ;CHECK ADDRESS
BEQ +4
ERROR 5 ;RO NOT #D1010+2

STFPS FPS ;STORE FLOATING POINT STATUS
CMP #47610,FPS ;CHECK FLOATING POINT STATUS
BEQ +4 ;BRANCH IF OK
ERROR 1 ;FPS NOT EQUAL TO 47610

STD AC1,$REGO ;GET RESULT
CMPD #-1,AC1 ;CHECK RESULT
CFCC ;GET CC
BEQ +4
ERROR 10 ;AC1 NOT -1.0

STCDI AC1,RO ;CONVERT IT BACK
CMP #-1,RO ;CHECK RESULT
BEQ +4
ERROR 13 ;RO NOT -1

SETF ;SET FLOATING MODE
LDCIF #54321,AC2 ;LOAD 54321
STF AC2,$REGO ;GET RESULT
STFPS FPS ;GET STATUS
CMPF F5T01,AC2 ;CHECK IT
CFCC ;CHECK CC
BEQ +4
ERROR 11 ;AC2 NOT 54321.

BIS #17,0#PS ;SET PS

```

N04

FPU INSTR EXERCISER MACY11 27(1006) 04-MAY-77 19:35 PAGE 37
DGFPCB.P11 04-MAY-77 19:34 T23 TEST OF LDCIF,LDCID,STCFI,STCDI

SEQ 0038

1439	007352	175667	171612	STCFI	AC2,SREGO	; CONVERT IT BACK
1440	007356	013700	177776	MOV	38PS,RO	; GET PS
1441	007362	042700	177760	BIC	#177760,RO	; CLEAR JUNK
1442	007366	001401		BEQ	+4	; SKIP IF OK
1443	007370	104004		ERROR	4	; PS NOT 0
1444	007372	170267	172066	STFPS	FPS	; GET STATUS
1445						
1446	007376	022767	054321 171564	CMP	#54321,SREGO	; CHECK RESULT
1447	007404	001401		BEQ	+4	
1448	007406	104012		ERROR	12	; SREGO NOT 54321
1449						

B05

FPU INSTR EXERCISER MACY11 27(1006) 04-MAY-77 19:35 PAGE 38
 D9FPC8.P11 04-MAY-77 19:34 T24 TEST OF LDCLF, LDCLD, STCF, STCDL

SEQ 0039

```

1450
1451
1452
1453 007410 000004
1454 007412 170127 047500
1455 007416 177327 177773
1456 007422 000401
1457 007424 104000
1458
1459 007426 170267 172032
1460 007432 022767 047510 172024
1461 007440 001401
1462 007442 104001
1463
1464 007444 174367 171520
1465 007450 173727
1466 007452 144640
1467 007454 170000
1468 007456 001401
1469 007460 104011
1470
1471 007462 012700 001632
1472 007466 175730
1473 007470 022700 001634
1474 007474 001401
1475 007476 104005
1476
1477 007500 170267 171760
1478 007504 022767 177773 171456
1479 007512 001401
1480 007514 104011
1481
1482 007516 005767 171450
1483 007522 001401
1484 007524 104011
1485
1486 007526 170011
1487 007530 177067 172220
1488 007534 170267 171724
1489 007540 022767 047700 171716
1490 007546 001401
1491 007550 104001
1492
1493 007552 174067 171412
1494 007556 173467 172200
1495 007562 170000
1496 007564 001401
1497 007566 104010
1498
1499 007570 012700 001174
1500 007574 175440
1501 007576 022700 001170
1502 007602 001401
1503 007604 104005
1504
1505 007606 170267 171652
  
```

```

*****
:TEST 24    TEST OF LDCLF, LDCLD, STCF, STCDL
*****
TST24:    SCOPE
          LDFPS    #47500    ;FLOATING AND LONG MODES
          LDCLF    # -5, AC3    ;LOAD A -5
          BR        .+4
          ERROR    0            ;LDCLF INCREMENTED BY 4 NOT 2
          STFPS    FPS           ;STORE FLOATING POINT STATUS
          CMP      #47510, FPS    ;CHECK FLOATING POINT STATUS
          BEQ      .+4            ;BRANCH IF OK
          ERROR    1            ;FPS NOT EQUAL TO 47510
          STF      AC3, $REG0    ;GET THE RESULT
          CMPF     (7)+, AC3    ;CHECK IT
                  144640
          CFCC     ;GET CC
          BEQ      .+4
          ERROR    11            ;AC3 NOT -5
          MOV      #ASREG0, RO    ;SET UP ADDRESS OF ADDRESS
          STCF     AC3, 2(RO)+    ;STORE IN $REG0
          CMP      #ASREG0+2, RO ;CHECK ADDRESS
          BEQ      .+4
          ERROR    5            ;ADDRESS IN FPU NOT ASREG0+2
          STFPS    FPS           ;GET STATUS
          CMP      # -5, $REG0    ;CHECK LEFT HALF
          BEQ      .+4
          ERROR    11            ;LEFT NOT -5
          TST      $REG1        ;CHECK RIGHT HALF OF RESULT
          BEQ      .+4
          ERROR    11            ;$REG1 NOT 0
          SETD     ;SET DOUBLE MODE
          LDCLD    DST01, ACO    ;LOAD WEIRD NUMBER
          STFPS    FPS           ;STORE FLOATING POINT STATUS
          CMP      #47700, FPS    ;CHECK FLOATING POINT STATUS
          BEQ      .+4            ;BRANCH IF OK
          ERROR    1            ;FPS NOT EQUAL TO 47700
          STD      ACO, $REG0    ;GET RESULT
          CMPD     FST01, ACO    ;CHECK IT
          CFCC     ;GET CC
          BEQ      .+4
          ERROR    10            ;ACO NOT FST01
          MOV      # $REG2, RO    ;GET IT BACK
          STCDL    ACO, -(RO)    ;CHECK ADDRESS
          CMP      # $REG0, RO    ;RO NOT # $REG0
          BEQ      .+4
          ERROR    5
          STFPS    FPS           ;STORE FLOATING POINT STATUS
  
```

FPU INSTR EXERCISER MACY11 27(1006) 04-MAY-77 19:35 PAGE 39
 DQFPCB.P11 04-MAY-77 19:34 T24 TEST OF LDCLF,LDCLD,STCFL,STCDL

SEQ 0040

1506	007612	022767	047700	171644	CMP	#47700,FPS	;CHECK FLOATING POINT STATUS
1507	007620	001401			BEQ	.+4	;BRANCH IF OK
1508	007622	104001			ERROR	i	;FPS NOT EQUAL TO 47700
1509							
1510	007624	170267	171634		STFPS	FPS	;GET STATUS
1511	007630	026767	172120	171332	CMP	DST01,\$REG0	;CHECK LEFT
1512	007636	001401			BEQ	.+4	
1513	007640	104011			ERROR	ii	;\$REG0 NOT
1514							
1515	007642	026767	172110	171322	CMP	DST01+2,\$REG1	;CHECK RIGHT
1516	007650	001401			BEQ	.+4	
1517	007652	104011			ERROR	ii	;\$REG1 NOT
1518							

```

1519
1520
1521
1522 007654 0000
1523 007656 170127 047400
1524 007662 172467 171760
1525 007666 171427 040200
1526 007672 170267 047400
1527 007676 022767 171560
1528 007704 001401
1529 007706 104001
1530
1531 007710 174067 171254
1532 007714 170500
1533 007716 170000
1534 007720 001401
1535 007722 104011
1536
1537 007724 174167 171240
1538 007730 173567 171712
1539 007734 170000
1540 007736 001401
1541 007740 104011
1542
1543 007742 172627 040200
1544 007746 171667 171742
1545 007752 170267 171506
1546 007756 022767 047400 171500
1547 007764 001401
1548 007766 104001
1549
1550 007770 174267 171174
1551 007774 173667 171714
1552 010000 170000
1553 010002 001401
1554 010004 104011
1555
1556 010006 174367 171156
1557 010012 170503
1558 010014 170000
1559 010016 001401
1560 010020 104011
1561
1562 010022 170011
1563 010024 172467 171654
1564 010030 171467 171640
1565 010034 174067 171130
1566 010040 173467 171620
1567 010044 170000
1568 010046 001401
1569 010050 104010
1570
1571 010052 174367 171112
1572 010056 170503
1573 010060 170000
1574 010062 001401

```

```

*****
#TEST 25 TEST OF MODF AND MODD
*****
TST25: SCOPE
LOFPS #47400 ;SET FLOATING MODE
LDF DALTB+2,ACO ;LOAD 52525,52525 INTO ACO
MODF #1,ACO ;MOD BY A 1 #MGT#
STFPS FPS ;STORE FLOATING POINT STATUS
CMP #47404,FPS ;CHECK FLOATING POINT STATUS
BEQ +4 ;BRANCH IF OK
ERROR 1 ;FPS NOT EQUAL TO 47404

STF ACO,$REGO ;GET RESULT
TSTF ACO ;CHECK FRACTION
CFCC
BEQ +4
ERROR 11 ;FRACTION NOT 0

STF AC1,$REGO ;GET IT
CMPF DALTB+2,AC1 ;CHECK INTEGER
CFCC
BEQ +4
ERROR 11 ;INTEGER IS NOT 52525,52525

LDF #1,AC2 ;LOAD A 1
MODF D20,AC2 ;MOD BY 20000,0
STFPS FPS ;STORE FLOATING POINT STATUS
CMP #47400,FPS ;CHECK FLOATING POINT STATUS
BEQ +4 ;BRANCH IF OK
ERROR 1 ;FPS NOT EQUAL TO 47400

STF AC2,$REGO ;GET IT
CMPF D20,AC2 ;CHECK FRACT
CFCC
BEQ +4
ERROR 11 ;RESULT NOT 20000,0

STF AC3,$REGO ;GET INT
TSTF AC3 ;CHECK FOR 0
CFCC
BEQ +4
ERROR 11 ;RESULT NOT 0

SETD ;SET DOUBLE MODE
LDD D46,ACO ;LOAD A 40600,0,0,0
MODD D37,ACO ;MOD BY 37400,0,0,0
STD ACO,$REGO ;STORE ANSWER
CMPD D40,ACO ;CHECK FOR 40000,0,0,0
CFCC
BEQ +4
ERROR 10 ;RESULT NOT 40000,0,0,0

STD AC3,$REGO ;GET THE RESULT
TSTD AC3 ;CHECK FOR 0
CFCC
BEQ +4

```


E05

FPU INSTR EXERCISER MACY11 27(1006) 04-MAY-77 19:35 PAGE 41
DQFPCB.P11 04-MAY-77 19:34 T25 TEST OF MODF AND MODD

SEQ 0042

1575 010064 104010
1576

ERROR 10

;RESULT NOT 0

.SBTTL ERROR CONDITIONS (STST)

1577										
1578										
1579										
1580										
1581										
1582	010066	000004								
1583	010070	170127	047600							
1584	010074	172467	171564							
1585	010100	172467	171516							
1586	010104	170267	171354							
1587	010110	170367	171352							
1588	010114	022767	147614	171342						
1589	010122	001401								
1590	010124	104001								
1591	010126	022767	000014	171332						
1592	010134	001401								
1593	010136	104001								
1594	010140	022767	010100	171322						
1595	010146	001401								
1596	010150	104001								
1597										
1598	010152	174067	171012							
1599	010156	173467	171502							
1600	010162	170267	171276							
1601	010166	170000								
1602	010170	001401								
1603	010172	104010								
1604										
1605	010174	170127	040200							
1606	010200	172467	171416							
1607	010204	170267	171254							
1608	010210	022767	040214	171246						
1609	010216	001401								
1610	010220	104001								
1611										
1612	010222	174067	170742							
1613	010226	173467	171370							
1614	010232	170267	171226							
1615	010236	170000								
1616	010240	001401								
1617	010242	104010								
1618										
1619										
1620										
1621										
1622	010244	000004								
1623	010246	170127	001000							
1624	010252	005001								
1625	010254	172427	076101							
1626	010260	171027	076101							
1627	010264	174067	170700							
1628	010270	170267	171170							
1629	010274	005701								
1630	010276	001001								
1631	010300	104007								
1632	010302									

```

*****
:TEST 26      LDD OF -0
*****

```

```

1S:
TST26:  SCOPE      #47600      ;DOUBLE MODE
        LDFPS     #47600      ;LOAD DUMMY DATA
        LDD      D40,ACD      ;LOAD A -0
        LDD      DMZERO,ACD   ;STORE FLOATING POINT STATUS
        STFPS    FPS          ;STORE EXCEPTION CODES
        STST     FEC          ;CHECK FLOATING POINT STATUS
        CMP      #147614,FPS   ;BRANCH IF OK
        BEQ      .+4          ;FPS NOT EQUAL TO 147614
        ERROR    1            ;CHECK FLOATING EXCEPTION CODE
        CMP      #14,FEC       ;BRANCH IF OK
        BEQ      .+4          ;FEC NOT EQUAL TO 14
        ERROR    1            ;CHECK FLOATING EXCEPTION ADDRESS
        CMP      #10100,FEA    ;BRANCH IF OK
        BEQ      .+4          ;FEA NOT EQUAL TO 10100
        ERROR    1

        STD      ACD,$REGO     ;GET RESULT FOR TYPING
        CMPD     D40,ACD      ;DID IT CHANGE ACD?
        STFPS    FPS          ;GET STATUS
        CFCC     .+4          ;GET CC
        BEQ      .+4          ;SKIP IF OK
        ERROR    10          ;RESULT IS WRONG

        LDFPS     #40200      ;DOUBLE MODE
        LDF      DMZERO,ACD   ;LOAD A -0
        STFPS    FPS          ;STORE FLOATING POINT STATUS
        CMP      #40214,FPS   ;CHECK FLOATING POINT STATUS
        BEQ      .+4          ;BRANCH IF OK
        ERROR    1            ;FPS NOT EQUAL TO 40214

        STD      ACD,$REGO     ;GET RESULT
        CMPD     DMZERO,ACD   ;CMP TO -0
        STFPS    FPS          ;GET STATUS
        CFCC     .+4          ;GET CC
        BEQ      .+4          ;SKIP IF OK
        ERROR    10          ;RESULT NOT -0

```

```

*****
:TEST 27      MULF ERROR - OVERFLOW
*****

```

```

3S:
TST27:  SCOPE      #1000      ;FLOATING/OVERFLOW
        LDFPS     #1000      ;CLEAR FLAG WORD
        CLR      R1          ;LOAD A LARGE NUMBER INTO ACD
        LDF      #1E36,ACD   ;MULTIPLY BY A LARGE NUMBER
        MULF     #1E36,ACD   ;GET FOR TYPING
        STF      ACD,$REGO   ;GET STATUS
        STFPS    FPS          ;DID IT TRAP?
        TST      R1          ;YES
        BNE     3S          ;DID NOT TRAP ON OVERFLOW
        ERROR    7

```

1633	010302	170267	171156		STFPS	FPS		: STORE FLOATING POINT STATUS
1634	010306	170367	171154		STST	FEC		: STORE EXCEPTION CODES
1635	010312	022767	101002	171144	CMP	#101002,FPS		: CHECK FLOATING POINT STATUS
1636	010320	001401			BEQ	.+4		: BRANCH IF OK
1637	010322	104001			ERROR	1		: FPS NOT EQUAL TO 101002
1638	010324	022767	000010	171134	CMP	#10,FEC		: CHECK FLOATING EXCEPTION CODE
1639	010332	001401			BEQ	.+4		: BRANCH IF OK
1640	010334	104001			ERROR	1		: FEC NOT EQUAL TO 10
1641	010336	022767	010260	171124	CMP	#10260,FEA		: CHECK FLOATING EXCEPTION ADDRESS
1642	010344	001401			BEQ	.+4		: BRANCH IF OK
1643	010346	104001			ERROR	1		: FEA NOT EQUAL TO 10260

: *TEST 30 DIVF ERROR - UNDERFLOW

1647					TST30:	SCOPE		
1648	010350	000004			LDFPS	#2000		: FLOATING/UNDERFLOW
1649	010352	170127	002000		CLR	R1		: CLEAR FLAG WORD
1650	010356	005001			LDF	#1E-36,AC0		: LOAD A SMALL NUMBER
1651	010360	172427	002252		DIVF	#1E36,AC0		: DIVIDE BY A LARGE NUMBER
1652	010364	174427	076101	1S:	STF	AC0,SREG0		: GET FOR TYPING
1653	010370	174067	170574		STFPS	FPS		: GET STATUS
1654	010374	170267	171064		TST	R1		: DID IT TRAP?
1655	010400	005701			BNE	3S		: SKIP IF SET
1656	010402	001001			ERROR	7		: DID NOT TRAP ON UNDERFLOW

1658	010406			3S:	STFPS	FPS		: STORE FLOATING POINT STATUS
1659	010406	170267	171052		STST	FEC		: STORE EXCEPTION CODES
1660	010412	170367	171050		CMP	#102000,FPS		: CHECK FLOATING POINT STATUS
1661	010416	022767	102000	171040	BEQ	.+4		: BRANCH IF OK
1662	010424	001401			ERROR	1		: FPS NOT EQUAL TO 102000
1663	010426	104001			CMP	#12,FEC		: CHECK FLOATING EXCEPTION CODE
1664	010430	022767	000012	171030	BEQ	.+4		: BRANCH IF OK
1665	010436	001401			ERROR	1		: FEC NOT EQUAL TO 12
1666	010440	104001			CMP	#10364,FEA		: CHECK FLOATING EXCEPTION ADDRESS
1667	010442	022767	010364	171020	BEQ	.+4		: BRANCH IF OK
1668	010450	001401			ERROR	1		: FEA NOT EQUAL TO 10364

: *TEST 31 STCFI ERROR - CONVERSION(6)

1670					TST31:	SCOPE		
1671					LDFPS	#400		: FLOATING/INTEGER/CONVERSION
1672					CLR	R1		: CLEAR FLAG WORD
1673					LDF	#1E36,AC1		: LOAD LARGE NUMBER
1674	010454	000004			STCFI	AC1,SREG0		: TRY TO STUFF INTO 16 BITS
1675	010456	170127	000400		STFPS	FPS		: GET STATUS
1676	010462	005001			TST	R1		: TRAP FLAG SET?
1677	010464	172527	076101	1S:	BNE	3S		: SKIP IF SET
1678	010470	175567	170474		ERROR	7		: DID NOT TRAP ON CONVERT
1679	010474	170267	170764					

1680	010500	005701		3S:	STFPS	FPS		: STORE FLOATING POINT STATUS
1681	010502	001001			STST	FEC		: STORE EXCEPTION CODES
1682	010504	104007			CMP	#100405,FPS		: CHECK FLOATING POINT STATUS
1683	010506				BEQ	.+4		: BRANCH IF OK
1684	010506	170267	170752		ERROR	1		: FPS NOT EQUAL TO 100405
1685	010512	170367	170750					
1686	010516	022767	100405	170740				
1687	010524	001401						
1688	010526	104001						

```

1689 010530 022767 000006 170730      CMP      #6,FEC      ;CHECK FLOATING EXCEPTION CODE
1690 010536 001401                      BEQ      .+4         ;BRANCH IF OK
1691 010540 104001                      ERROR    1           ;FEC NOT EQUAL TO 6
1692 010542 022767 010470 170720      CMP      #10470,FEA ;CHECK FLOATING EXCEPTION ADDRESS
1693 010550 001401                      BEQ      .+4         ;BRANCH IF OK
1694 010552 104001                      ERROR    1           ;FEA NOT EQUAL TO 10470
1695
1696
1697
1698
1699 010554 000004                      ;*****
1700 010556 170127 000000                      ;*TEST 32      DIVF BY 0 ERROR
1701 010562 005001                      ;*****
1702 010564 174527 000000      1S:    DIVF      #0,AC1 ;DIVIDE BY 0
1703 010570 170267 170670      STFPS    FPS        ;GET STATUS
1704 010574 005701                      TST      R1         ;CHECK FLAG
1705 010576 001001                      BNE      3S        ;SKIP IF SET
1706 010600 104007                      ERROR    7         ;DIVIDE BY 0 DID NOT TRAP
1707 010602
1708 010602 170267 170656      3S:    STFPS    FPS        ;STORE FLOATING POINT STATUS
1709 010606 170367 170654      STST     FEC        ;STORE EXCEPTION CODES
1710 010612 022767 100000 170644      CMP      #100000,FPS ;CHECK FLOATING POINT STATUS
1711 010620 001401                      BEQ      .+4         ;BRANCH IF OK
1712 010622 104001                      ERROR    1           ;FPS NOT EQUAL TO 100000
1713 010624 022767 000004 170634      CMP      #4,FEC     ;CHECK FLOATING EXCEPTION CODE
1714 010632 001401                      BEQ      .+4         ;BRANCH IF OK
1715 010634 104001                      ERROR    1           ;FEC NOT EQUAL TO 4
1716 010636 022767 010564 170624      CMP      #10564,FEA ;CHECK FLOATING EXCEPTION ADDRESS
1717 010644 001401                      BEQ      .+4         ;BRANCH IF OK
1718 010646 104001                      ERROR    1           ;FEA NOT EQUAL TO 10564
1719
1720
1721
1722
1723 010650 000004                      ;*****
1724 010652 170127 004000                      ;*TEST 33      LDF -0 ERROR
1725 010656 005001                      ;*****
1726 010660 172667 170736      1S:    LDF      DMZERO,AC2 ;LOAD AN UNDEFINED VARIABLE
1727 010664 170267 170574      STFPS    FPS        ;GET STATUS
1728 010670 005701                      TST      R1         ;CHECK FLAG
1729 010672 001001                      BNE      3S        ;SKIP IF SET
1730 010674 104007                      ERROR    7         ;LOAD OF -0 DID NOT TRAP
1731 010676
1732 010676 170267 170562      3S:    STFPS    FPS        ;STORE FLOATING POINT STATUS
1733 010702 170367 170560      STST     FEC        ;STORE EXCEPTION CODES
1734 010706 022767 104014 170550      CMP      #104014,FPS ;CHECK FLOATING POINT STATUS
1735 010714 001401                      BEQ      .+4         ;BRANCH IF OK
1736 010716 104001                      ERROR    1           ;FPS NOT EQUAL TO 104014
1737 010720 022767 000014 170540      CMP      #14,FEC    ;CHECK FLOATING EXCEPTION CODE
1738 010726 001401                      BEQ      .+4         ;BRANCH IF OK
1739 010730 104001                      ERROR    1           ;FEC NOT EQUAL TO 14
1740 010732 022767 010660 170530      CMP      #10660,FEA ;CHECK FLOATING EXCEPTION ADDRESS
1741 010740 001401                      BEQ      .+4         ;BRANCH IF OK
1742 010742 104001                      ERROR    1           ;FEA NOT EQUAL TO 10660
1743
1744
;*****

```

```

1745
1746
1747 010744 000004
1748 010746 170127 000000
1749 010752 005001
1750 010754 177707
1751 010756 170267 170502
1752 010762 005701
1753 010764 001001
1754 010766 104007
1755 010770
1756 010770 170267 170470
1757 010774 170367 170466
1758 011000 022767 100000 170456
1759 011006 001401
1760 011010 104001
1761 011012 022767 000002 170446
1762 011020 001401
1763 011022 104001
1764 011024 022767 010754 170436
1765 011032 001401
1766 011034 104001
1767
1768
1769
1770
1771 011036 000004
1772 011040 170127 001000
1773 011044 005001
1774 011046 172767 170540
1775 011052 172367 170534
1776 011056 170267 170402
1777 011062 174367 170102
1778 011066 005701
1779 011070 001001
1780 011072 104007
1781 011074
1782 011074 170267 170364
1783 011100 170367 170362
1784 011104 022767 101006 170352
1785 011112 001401
1786 011114 104001
1787 011116 022767 000010 170342
1788 011124 001401
1789 011126 104001
1790 011130 022767 011052 170332
1791 011136 001401
1792 011140 104001
1793
1794
1795
1796
1797 011142 000004
1798 011144 170127 002000
1799 011150 005001
1800 011152 172427 000430

```

```

; *TEST 34 OPCODE ERROR
; *****
TST34: SCOPE OPCODE ERROR
LDFPS #0 ; FLOATING
CLR R1 ; CLEAR FLAG
1S: 177707 ; ILLEGAL OPCODE
STFPS FPS ; GET STATUS
TST R1 ; CHECK FLAG
BNE 3S ; SKIP IF SET
ERROR 7 ; NOT AN ILLEGAL OPCODE

3S: STFPS FPS ; STORE FLOATING POINT STATUS
STST FEC ; STORE EXCEPTION CODES
CMP #100000,FPS ; CHECK FLOATING POINT STATUS
BEQ .+4 ; BRANCH IF OK
ERROR 1 ; FPS NOT EQUAL TO 100000
CMP #2,FEC ; CHECK FLOATING EXCEPTION CODE
BEQ .+4 ; BRANCH IF OK
ERROR 1 ; FEC NOT EQUAL TO 2
CMP #10754,FEA ; CHECK FLOATING EXCEPTION ADDRESS
BEQ .+4 ; BRANCH IF OK
ERROR 1 ; FEA NOT EQUAL TO 10754

; *****
; *TEST 35 ADDF ERROR - OVERFLOW
; *****
TST35: SCOPE ADDF ERROR - OVERFLOW
LDFPS #1000 ; FLOATING/OVERFLOW
CLR R1 ; CLEAR FLAG
1S: LDF DBIG,AC3 ; LOAD A BIG NUMBER
ADDF DBIG,AC3 ; MAKE OVERFLOW
STFPS FPS ; GET STATUS
STF AC3,SREG0 ; GET RESULT
TST R1 ; FLAG SET?
BNE 3S ; SKIP IF SET
ERROR 7 ; DID NOT TRAP ON OVERFLOW

3S: STFPS FPS ; STORE FLOATING POINT STATUS
STST FEC ; STORE EXCEPTION CODES
CMP #101006,FPS ; CHECK FLOATING POINT STATUS
BEQ .+4 ; BRANCH IF OK
ERROR 1 ; FPS NOT EQUAL TO 101006
CMP #10,FEC ; CHECK FLOATING EXCEPTION CODE
BEQ .+4 ; BRANCH IF OK
ERROR 1 ; FEC NOT EQUAL TO 10
CMP #11052,FEA ; CHECK FLOATING EXCEPTION ADDRESS
BEQ .+4 ; BRANCH IF OK
ERROR 1 ; FEA NOT EQUAL TO 11052

; *****
; *TEST 36 SUBF ERROR - UNDERFLOW
; *****
TST36: SCOPE SUBF ERROR - UNDERFLOW
LDFPS #2000 ; FLOATING/UNDERFLOW
CLR R1 ; CLEAR FLAG
LDF #.07E-37,AC0 ; LOAD SMALL NUMBER

```

1801	011156	173027	000504			1S:	SUBF	#,D9E-37,ACO	:	SUBF SMALL NUMBER
1802	011162	174067	170002				STF	ACO,SREGO	:	GET RESULT
1803	011166	170267	170272				STFPS	FPS	:	GET STATUS
1804	011172	005701					TST	R1	:	FLAG SET?
1805	011174	001001					BNE	3S	:	SKIP IF SET
1806	011176	104007					ERROR	7	:	NO TRAP ON UNDERFLOW
1807	011200					3S:			:	
1808	011200	170267	170260				STFPS	FPS	:	STORE FLOATING POINT STATUS
1809	011204	170367	170256				STST	FEC	:	STORE EXCEPTION CODES
1810	011210	022767	102014	170246			CMP	#102014,FPS	:	CHECK FLOATING POINT STATUS
1811	011216	001401					BEQ	.+4	:	BRANCH IF OK
1812	011220	104001					ERROR	1	:	FPS NOT EQUAL TO 102014
1813	011222	022767	000012	170236			CMP	#12,FEC	:	CHECK FLOATING EXCEPTION CODE
1814	011230	001401					BEQ	.+4	:	BRANCH IF OK
1815	011232	104001					ERROR	1	:	FEC NOT EQUAL TO 12
1816	011234	022767	011156	170226			CMP	#11156,FEA	:	CHECK FLOATING EXCEPTION ADDRESS
1817	011242	001401					BEQ	.+4	:	BRANCH IF OK
1818	011244	104001					ERROR	1	:	FEA NOT EQUAL TO 11156

 ;*TEST 37 TEST OF FEC-02, WITH NO TRAP, INTERRUPT DISABLED
 ;*TEST 37 TEST OF FEC-02, WITH NO TRAP, INTERRUPT DISABLED

1824	011246	000004				TST37:	SCOPE			
1825	011250	012704	014640				MOV	#FEC02,R4	:	PTR TO FPP FEC CODE GENERATOR
1826	011254	012705	011302				MOV	#FPPN1,R5	:	PTR TO TEST DATA SET
1827	011260	012767	014640	170212			MOV	#FII02,EXPFEA	:	ADDR(FPP BAD INSTR) EXPECTED
1828	011266	012767	014642	170206			MOV	#NXT02,EXPTS	:	ADDR(NEXT INSTR) EXPECTED
1829										
1830	011274	004737	014300				JSR	PC,#TRPTST	:	GO TEST
1831										
1832	011300					64S:				
1833	011300	000406					BR	TST40	::	
1834										
1835	011302					FPPN1:				
1836	011302	000157	000040	000157			.WORD	000157,000040,000157	:	PSW'S: BEFORE, LOADED, AFTER
1837	011310	047417	147417	100002			.WORD	047417,147417,100002	:	OLDFPS/NEWFPS/FECCODE

 ;*TEST 40 TEST OF FEC-04, WITH NO TRAP, INTERRUPT DISABLED
 ;*TEST 40 TEST OF FEC-04, WITH NO TRAP, INTERRUPT DISABLED

1843	011316	000004				TST40:	SCOPE			
1844	011320	012704	014650				MOV	#FEC04,R4	:	PTR TO FPP FEC CODE GENERATOR
1845	011324	012705	011352				MOV	#FPPN2,R5	:	PTR TO TEST DATA SET
1846	011330	012767	014654	170142			MOV	#FII04,EXPFEA	:	ADDR(FPP BAD INSTR) EXPECTED
1847	011336	012767	014660	170136			MOV	#NXT04,EXPTS	:	ADDR(NEXT INSTR) EXPECTED
1848										
1849	011344	004737	014300				JSR	PC,#TRPTST	:	GO TEST
1850										
1851	011350					64S:				
1852	011350	000406					BR	TST41	::	
1853										
1854	011352					FPPN2:				
1855	011352	000157	000040	000157			.WORD	000157,000040,000157	:	PSW'S: BEFORE, LOADED, AFTER
1856	011360	047417	147400	100004			.WORD	047417,147400,100004	:	OLDFPS/NEWFPS/FECCODE

K05

FPU INSTR EXERCISER
DQFPCB.P11

MACY11 27(1006)
04-MAY-77 19:34

04-MAY-77 19:35 PAGE 47
T40 TEST OF FEC-04, WITH NO TRAP, INTERRUPT DISABLED

SEQ 0048

1857
1858
1859
1860
1861
1862 011366 000004
1863 011370 012704 014666
1864 011374 012705 011422
1865 011400 012767 014672 170072
1866 011406 012767 014676 170066
1867
1868 011414 004737 014300
1869
1870 011420
1871 011420 000406
1872
1873 011422
1874 011422 000155 000042 000142
1875 011430 047412 147405 100006
1876
1877
1878
1879
1880
1881 011436 000004
1882 011440 012704 014704
1883 011444 012705 011472
1884 011450 012767 014714 170022
1885 011456 012767 014716 170016
1886
1887 011464 004737 014300
1888
1889 011470
1890 011470 000406
1891
1892 011472
1893 011472 000151 000046 000151
1894 011500 047411 147406 100010
1895
1896
1897
1898
1899
1900 011506 000004
1901 011510 012704 014724
1902 011514 012705 011542
1903 011520 012767 014730 167752
1904 011526 012767 014734 167746
1905
1906 011534 004737 014300
1907
1908 011540
1909 011540 000406
1910
1911 011542
1912 011542 000157 000040 000157

:TEST 41 TEST OF FEC-06, WITH NO TRAP, INTERRUPT DISABLED

TST41: SCOPE
MOV #FEC06,R4 ; PTR TO FPP FEC CODE GENERATOR
MOV #FPPN3,RS ; PTR TO TEST DATA SET
MOV #FII06,EXPFEA ; ADDR(FPP BAD INSTR) EXPECTED
MOV #NXT06,EXPRTS ; ADDR(NEXT INSTR) EXPECTED
JSR PC,@TRPTST ; GO TEST

645: BR TST42 ;;

FPPN3: ; TEST DATA SET FPPN-3:
.WORD 000155,000042,000142 ; PSW'S: BEFORE, LOADED, AFTER
.WORD 047412,147405,100006 ; OLDFPS/NEWFPS/FECCODE

:TEST 42 TEST OF FEC-10, WITH NO TRAP, INTERRUPT DISABLED

TST42: SCOPE
MOV #FEC10,R4 ; PTR TO FPP FEC CODE GENERATOR
MOV #FPPN4,RS ; PTR TO TEST DATA SET
MOV #FII10,EXPFEA ; ADDR(FPP BAD INSTR) EXPECTED
MOV #NXT10,EXPRTS ; ADDR(NEXT INSTR) EXPECTED
JSR PC,@TRPTST ; GO TEST

645: BR TST43 ;;

FPPN4: ; TEST DATA SET FPPN-4:
.WORD 000151,000046,000151 ; PSW'S: BEFORE, LOADED, AFTER
.WORD 047411,147406,100010 ; OLDFPS/NEWFPS/FECCODE

:TEST 43 TEST OF FEC-12, WITH NO TRAP, INTERRUPT DISABLED

TST43: SCOPE
MOV #FEC12,R4 ; PTR TO FPP FEC CODE GENERATOR
MOV #FPPN5,RS ; PTR TO TEST DATA SET
MOV #FII12,EXPFEA ; ADDR(FPP BAD INSTR) EXPECTED
MOV #NXT12,EXPRTS ; ADDR(NEXT INSTR) EXPECTED
JSR PC,@TRPTST ; GO TEST

645: BR TST44 ;;

FPPN5: ; TEST DATA SET FPPN-5:
.WORD 000157,000040,000157 ; PSW'S: BEFORE, LOADED, AFTER

1913 011550 047417 147400 100012

.WORD 047417,147400,100012 ; OLDFPS/NEWFPS/FECCODE

1914
1915

1916
1917
1918
:*****
:TEST 44 TEST OF FEC-14, WITH NO TRAP, INTERRUPT DISABLED
:*****

1919 011556 000004
1920 011560 012704 014742
1921 011564 012705 011612
1922 011570 012767 014742 167702
1923 011576 012767 014746 167676

TST44: SCOPE
MOV #FEC14,R4 ; PTR TO FPP FEC CODE GENERATOR
MOV #FPPN6,R5 ; PTR TO TEST DATA SET
MOV #FII14,EXPFEA ; ADDR(FPP BAD INSTR) EXPECTED
MOV #NXT14,EXPRTS ; ADDR(NEXT INSTR) EXPECTED

1924
1925 011604 004737 014300

JSR PC,@TRPTST ; GO TEST

1926
1927 011610
1928 011610 000406

645: BR TST45 ;

1929
1930 011612
1931 011612 000143 000054 000143
1932 011620 047403 147414 100014

FPPN6: ; TEST DATA SET FPPN-6:
.WORD 000143,000054,000143 ; PSW'S: BEFORE, LOADED, AFTER
.WORD 047403,147414,100014 ; OLDFPS/NEWFPS/FECCODE

1933
1934

1935
1936
1937
:*****
:TEST 45 TEST OF FEC-16, WITH NO TRAP, INTERRUPT DISABLED
:*****

1938 011626 000004

TST45: SCOPE

1939
1940

;///
;THIS TEST IS EXECUTED FOR HFP ONLY, NOT IN WFP MODE

1941
1942
1943 011630 076600 000022
1944 011634 032700 000020
1945 011640 001421

MED #RWHAMI ;GET WHAMI INTO RO
BIT #BIT04,RO ;HFP IN SYSTEM ?
BEQ 645 ;BR IF NONE

1946
1947 011642 076600 000144
1948 011646 032700 010000
1949 011652 001414

MED #RFLAG ;GET FLAGS INTO RO
BIT #BIT12,RO ;HFP ENABLED ?
BEQ 645 ;BR IF NOT

1950
1951

;///

1952
1953 011654 012704 014754
1954 011660 012705 011706
1955 011664 012767 014762 167606
1956 011672 012767 014764 167602

MOV #FEC16,R4 ; PTR TO FPP FEC CODE GENERATOR
MOV #FPPN7,R5 ; PTR TO TEST DATA SET
MOV #FII16,EXPFEA ; ADDR(FPP BAD INSTR) EXPECTED
MOV #NXT16,EXPRTS ; ADDR(NEXT INSTR) EXPECTED

1957
1958 011700 004737 014300

JSR PC,@TRPTST ; GO TEST

1959
1960 011704
1961 011704 000406

645: BR TST46 ;

1962
1963 011706
1964 011706 000157 000052 000140
1965 011714 047420 147420 100016

FPPN7: ; TEST DATA SET FPPN-7:
.WORD 000157,000052,000140 ; PSW'S: BEFORE, LOADED, AFTER
.WORD 047420,147420,100016 ; OLDFPS/NEWFPS/FECCODE

1966
1967
1968

M05

FPU INSTR EXERCISER MACY11 27(1006) 04-MAY-77 19:35 PAGE 49
DQFPCB.P11 04-MAY-77 19:34

T45 TEST OF FEC-16, WITH NO TRAP, INTERRUPT DISABLED

SEQ 0050

1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024

;/;;/;
: NOTE: IN THE FOLLOWING GROUP OF TESTS, THE LOADED PS
: MUST CONTAIN PROCESSOR PRIORITY OF (6) OR GREATER
: TO LOCK OUT ANY CLOCK INTERRUPTS (AT BR6). THEY
: ARE SETUP HERE WITH PRIORITY (7).
;/;;/;

: #TEST 46 TEST OF FEC-02, WITH TRAP, INTERRUPT ENABLED
: *****

TST46: SCOPE
MOV #FEC02,R4 ; PTR TO FPP FEC CODE GENERATOR
MOV #FPPT10,R5 ; PTR TO TEST DATA SET
MOV #FII02,EXPFEA ; ADDR(FPP BAD INSTR) EXPECTED
MOV #NXT02,EXPTS ; ADDR(NEXT INSTR) EXPECTED
JSR PC,@TRPTST ; GO TEST

645: BR TST47 ;;

FPPT10: ; TEST DATA SET FPPT-10:
.WORD 000353,000104,000353 ; PSW'S: BEFORE, LOADED, AFTER
.WORD 000017,100017,140002 ; OLDFPS/NEWFPS/FECCODE

: #TEST 47 TEST OF FEC-04, WITH TRAP, INTERRUPT ENABLED
: *****

TST47: SCOPE
MOV #FEC04,R4 ; PTR TO FPP FEC CODE GENERATOR
MOV #FPPT11,R5 ; PTR TO TEST DATA SET
MOV #FII04,EXPFEA ; ADDR(FPP BAD INSTR) EXPECTED
MOV #NXT04,EXPTS ; ADDR(NEXT INSTR) EXPECTED
JSR PC,@TRPTST ; GO TEST

645: BR TST50 ;;

FPPT11: ; TEST DATA SET FPPT-11:
.WORD 000350,000107,000350 ; PSW'S: BEFORE, LOADED, AFTER
.WORD 000017,100000,140004 ; OLDFPS/NEWFPS/FECCODE

: #TEST 50 TEST OF FEC-06, WITH NO TRAP, INTERRUPT ENABLED
: *****

TST50: SCOPE
MOV #FEC06,R4 ; PTR TO FPP FEC CODE GENERATOR
MOV #FPPN12,R5 ; PTR TO TEST DATA SET
MOV #FII06,EXPFEA ; ADDR(FPP BAD INSTR) EXPECTED
MOV #NXT06,EXPTS ; ADDR(NEXT INSTR) EXPECTED

N05

FPU INSTR EXERCISER MACY11 27(1006) 04-MAY-77 19:35 PAGE 50
DQFPCB.P11 04-MAY-77 19:34

T50 TEST OF FEC-06, WITH NO TRAP, INTERRUPT ENABLED

SEQ 0051

2025 012070 004737 014300

JSR PC,3#TRPTST ; GO TEST

2026 012074

645:

2027 012074 000406

BR TST51 ;;

2028 012076

FPPN12: ; TEST DATA SET FPPN-12:

2029 012076 000353 000104 000345

.WORD 000353,000104,000345 ; PSW'S: BEFORE, LOADED, AFTER

2030 012104 007012 007005 000000

.WORD 007012,007005,000000 ; OLDFPS/NEWFPS/FECCODE

2031 012104

2032 012104

2033 012104

2034 012104

2035 012104

2036 012104

2037 012104

; TEST 51 TEST OF FEC-06, WITH TRAP, INTERRUPT ENABLED

2038 012112 000004

TST51: SCOPE

2039 012114 012704 014666

MOV #FEC06,R4 ; PTR TO FPP FEC CODE GENERATOR

2040 012120 012705 012146

MOV #FPPT13,R5 ; PTR TO TEST DATA SET

2041 012124 012767 014672 167346

MOV #FII06,EXPFEA ; ADDR(FPP BAD INSTR) EXPECTED

2042 012132 012767 014676 167342

MOV #NXT06,EXPTS ; ADDR(NEXT INSTR) EXPECTED

2043 012140 004737 014300

JSR PC,3#TRPTST ; GO TEST

2044 012144

2045 012144

2046 012144 000406

645:

BR TST52 ;;

2047 012146

FPPT13: ; TEST DATA SET FPPT-13:

2048 012146 000353 000104 000345

.WORD 000353,000104,000345 ; PSW'S: BEFORE, LOADED, AFTER

2049 012154 000412 100405 140006

.WORD 000412,100405,140006 ; OLDFPS/NEWFPS/FECCODE

2050 012154

2051 012154

2052 012154

2053 012154

2054 012154

2055 012154

2056 012154

2057 012162 000004

; TEST 52 TEST OF FEC-10, WITH NO TRAP, INTERRUPT ENABLED

2058 012164 012704 014704

TST52: SCOPE

2059 012170 012705 012216

MOV #FEC10,R4 ; PTR TO FPP FEC CODE GENERATOR

2060 012174 012767 014714 167276

MOV #FPPN14,R5 ; PTR TO TEST DATA SET

2061 012202 012767 014716 167272

MOV #FII10,EXPFEA ; ADDR(FPP BAD INSTR) EXPECTED

2062 012210 004737 014300

MOV #NXT10,EXPTS ; ADDR(NEXT INSTR) EXPECTED

2063 012214

JSR PC,3#TRPTST ; GO TEST

2064 012214 000406

645:

BR TST53 ;;

2065 012216

FPPN14: ; TEST DATA SET FPPN-14:

2066 012216 000350 000107 000350

.WORD 000350,000107,000350 ; PSW'S: BEFORE, LOADED, AFTER

2067 012224 006411 006406 000000

.WORD 006411,006406,000000 ; OLDFPS/NEWFPS/FECCODE

2068 012232 000004

2069 012234 012704 014704

; TEST 53 TEST OF FEC-10, WITH TRAP, INTERRUPT ENABLED

2070 012240 012705 012266

TST53: SCOPE

2071 012244 012767 014714 167226

MOV #FEC10,R4 ; PTR TO FPP FEC CODE GENERATOR

2072 012252 012767 014716 167222

MOV #FPPT15,R5 ; PTR TO TEST DATA SET

MOV #FII10,EXPFEA ; ADDR(FPP BAD INSTR) EXPECTED

MOV #NXT10,EXPTS ; ADDR(NEXT INSTR) EXPECTED

```

2081 012260 004737 014300 JSR PC, @TRPTST ; GO TEST
2082
2083
2084 012264 645: BR TST54 ;;
2085 012264 000406
2086
2087 012266 FPPT15: ; TEST DATA SET FPPT-15:
2088 012266 000350 000107 000350 .WORD 000350,000107,000350 ; PSW'S: BEFORE, LOADED, AFTER
2089 012274 001011 101006 140010 .WORD 001011,101006,140010 ; OLDFPS/NEWFPS/FECCODE
2090
2091

```

```

*****
; TEST 54 TEST OF FEC-12, WITH NO TRAP, INTERRUPT ENABLED
*****

```

```

2092
2093
2094
2095 012302 000004 TST54: SCOPE
2096 012304 012704 014724 MOV #FEC12,R4 ; PTR TO FPP FEC CODE GENERATOR
2097 012310 012705 012336 MOV #FPPN16,R5 ; PTR TO TEST DATA SET
2098 012314 012767 014730 167156 MOV #FII12,EXPFEA ; ADDR(FPP BAD INSTR) EXPECTED
2099 012322 012767 014734 167152 MOV #NXT12,EXPRTS ; ADDR(NEXT INSTR) EXPECTED
2100

```

```

2101 012330 004737 014300 JSR PC, @TRPTST ; GO TEST

```

```

2102
2103 012334 645: BR TST55 ;;
2104 012334 000406
2105

```

```

2106 012336 FPPN16: ; TEST DATA SET FPPN-16:
2107 012336 000350 000107 000350 .WORD 000350,000107,000350 ; PSW'S: BEFORE, LOADED, AFTER
2108 012344 005413 005404 000000 .WORD 005413,005404,000000 ; OLDFPS/NEWFPS/FECCODE
2109
2110

```

```

*****
; TEST 55 TEST OF FEC-12, WITH TRAP, INTERRUPT ENABLED
*****

```

```

2111
2112
2113
2114 012352 000004 TST55: SCOPE
2115 012354 012704 014724 MOV #FEC12,R4 ; PTR TO FPP FEC CODE GENERATOR
2116 012360 012705 012406 MOV #FPPT17,R5 ; PTR TO TEST DATA SET
2117 012364 012767 014730 167106 MOV #FII12,EXPFEA ; ADDR(FPP BAD INSTR) EXPECTED
2118 012372 012767 014734 167102 MOV #NXT12,EXPRTS ; ADDR(NEXT INSTR) EXPECTED
2119

```

```

2120 012400 004737 014300 JSR PC, @TRPTST ; GO TEST

```

```

2121
2122 012404 645: BR TST56 ;;
2123 012404 000406
2124

```

```

2125 012406 FPPT17: ; TEST DATA SET FPPT-17:
2126 012406 000350 000107 000350 .WORD 000350,000107,000350 ; PSW'S: BEFORE, LOADED, AFTER
2127 012414 002017 102000 140012 .WORD 002017,102000,140012 ; OLDFPS/NEWFPS/FECCODE
2128
2129

```

```

*****
; TEST 56 TEST OF FEC-14, WITH NO TRAP, INTERRUPT ENABLED
*****

```

```

2130
2131
2132
2133 012422 000004 TST56: SCOPE
2134 012424 012704 014742 MOV #FEC14,R4 ; PTR TO FPP FEC CODE GENERATOR
2135 012430 012705 012456 MOV #FPPN20,R5 ; PTR TO TEST DATA SET
2136 012434 012767 014742 167036 MOV #FII14,EXPFEA ; ADDR(FPP BAD INSTR) EXPECTED

```

```

2137 012442 012767 014746 167032      MOV    #NXT14,EXPRTS ; ADDR(NEXT INSTR) EXPECTED
2138
2139 012450 004737 014300                JSR    PC,@TRPTST   ; GO TEST
2140
2141 012454                                645:  BR    TST57           ;;
2142 012454 000406
2143
2144 012456                                FPPN20: ; TEST DATA SET FPPN-20:
2145 012456 000343 000114 000343          .WORD 000343,000114,000343 ; PSW'S: BEFORE, LOADED, AFTER
2146 012464 003403 003414 000000          .WORD 003403,003414,000000 ; OLDFPS/NEWFPS/FECODE
2147

```

```

2148
2149
2150
2151
2152 012472 000004                                ;*****
2153 012474 012704 014742                    ;#TEST 57      TEST OF FEC-14, WITH TRAP, INTERRUPT ENABLED
2154 012500 012705 012526                    ;*****
2155 012504 012767 014742 166766          TST57: SCOPE
2156 012512 012767 014746 166762          MOV    #FEC14,R4      ; PTR TO FPP FEC CODE GENERATOR
2157
2158 012520 004737 014300                MOV    #FPPT21,RS     ; PTR TO TEST DATA SET
2159
2160 012524                                MOV    #FII14,EXPFEA  ; ADDR(FPP BAD INSTR) EXPECTED
2161 012524 000406                                MOV    #NXT14,EXPRTS  ; ADDR(NEXT INSTR) EXPECTED
2162
2163 012526                                JSR    PC,@TRPTST     ; GO TEST
2164 012526 000343 000114 000343
2165 012534 004003 104014 140014
2166
2167

```

```

2168
2169
2170
2171 012542 000004                                ;*****
2172 012544 012704 014754                    ;#TEST 60      TEST OF FEC-16, WITH NO TRAP, INTERRUPT ENABLED
2173 012550 012705 012576                    ;*****
2174 012554 012767 014762 166716          TST60: SCOPE
2175 012562 012767 014764 166712          MOV    #FEC16,R4      ; PTR TO FPP FEC CODE GENERATOR
2176
2177 012570 004737 014300                MOV    #FPPN22,RS     ; PTR TO TEST DATA SET
2178
2179 012574                                MOV    #FII16,EXPFEA  ; ADDR(FPP BAD INSTR) EXPECTED
2180 012574 000406                                MOV    #NXT16,EXPRTS  ; ADDR(NEXT INSTR) EXPECTED
2181
2182 012576                                JSR    PC,@TRPTST     ; GO TEST
2183 012576 000357 000145 000340
2184 012604 007400 007400 000000
2185
2186

```

```

2187
2188
2189
2190 012612 000004                                ;*****
2191
2192

```

```

;#TEST 61      TEST OF FEC-16, WITH TRAP, INTERRUPT ENABLED
;*****
TST61: SCOPE
;////////////////////////////////////

```

;THIS TEST IS EXECUTED FOR HFP ONLY, NOT IN WFP MODE

2193
2194
2195 012614 076600 000022
2196 012620 032700 000020
2197 012624 001421
2198
2199 012626 076600 000144
2200 012632 032700 010000
2201 012636 001414
2202
2203

MED ,RWHAMI ;GET WHAMI INTO RO
BIT #BIT04,RO ;HFP IN SYSTEM ?
BEQ 64\$;BR IF NONE

MED ,RFLAG ;GET FLAGS INTO RO
BIT #BIT12,RO ;HFP ENABLED ?
BEQ 64\$;BR IF NOT

////////////////////////////////////

2204
2205 012640 012704 014754
2206 012644 012705 012672
2207 012650 012767 014762 166622
2208 012656 012767 014764 166616
2209
2210 012664 004737 014300
2211
2212 012670
2213 012670 000406
2214
2215 012672
2216 012672 000357 000152 000341
2217 012700 000020 100020 140016
2218
2219

MOV #FEC16,R4 ; PTR TO FPP FEC CODE GENERATOR
MOV #FPPT23,R5 ; PTR TO TEST DATA SET
MOV #FII16,EXPFEA ; ADDR(FPP BAD INSTR) EXPECTED
MOV #NXT16,EXPTS ; ADDR(NEXT INSTR) EXPECTED

JSR PC,@TRPTST ; GO TEST

64\$: BR TST62 ;;

FPPT23: ; TEST DATA SET FPPT-23:
.WORD 000357,000152,000341 ; PSW'S: BEFORE, LOADED, AFTER
.WORD 000020,100020,140016 ; OLDFPS/NEWFPS/FECODE

E06

FPU INSTR EXERCISER MACY11 27(1006) 04-MAY-77 19:35 PAGE 54
 DQFPCB.P11 04-MAY-77 19:34 T62 TEST FOR CONVERSION, ADD AND SUBD

SEQ 0055

```

2220
2221
2222
2223 012706 000004
2224 012710 170127 047600
2225 012714 177327 000004
2226 012720 177227 000002
2227 012724 172302
2228 012726 173302
2229 012730 175737 001170
2230 012734 022737 000004 001170
2231 012742 170267 166516
2232 012746 001401
2233 012750 104012
2234
2235
2236
2237
2238 012752 000004
2239 012754 177027 000005
2240 012760 174005
2241 012762 172605
2242 012764 172200
2243 012766 173205
2244 012770 175637 001170
2245 012774 022737 000005 001170
2246 013002 170267 166456
2247 013006 001401
2248 013010 104012
2249
  
```

```

*****
;TEST 62    TEST FOR CONVERSION, ADD AND SUBD
*****
†ST62:    SCOPE
          LDFPS    #47600
          LDCID    #4,AC3           ;LOAD AC3 WITH A FLOATING 4
          LDCID    #2,AC2           ;LOAD AC2 WITH A FLOATING 2
          ADD      AC2,AC3           ;ADD 2+4 = 6 IN AC3
          SUBD     AC2,AC3           ;SUB 2 FROM 6
          STCDI    AC3,2#SREGO       ;2#SREGO SHOULD =4
          CMP      #4,2#SREGO       ;DOES 2#SREGO =4
          STFPS    FPS               ;GET STATUS
          BEQ      +4                ;YES
          ERROR    12                ;2#SREGO SHOULD = 4
*****
;TEST 63    LDD AND STD TEST
*****
†ST63:    SCOPE
          LDCID    #5,AC0           ;LOAD AC0 WITH A FLOATING 5
          STD      AC0,AC5           ;NOW PUT IT INTO AC5
          LDD      AC5,AC2           ;NOW PUT IT INTO AC2
          ADD      AC0,AC2           ;ADD 5 TO 5
          SUBD     AC5,AC2           ;SUB 5 FROM 10
          STCDI    AC2,2#SREGO       ;PUT ANS INTO 2#SREGO
          CMP      #5,2#SREGO       ;WERE THE TWO AC'S EQUAL
          STFPS    FPS               ;GET STATUS
          BEQ      +4                ;YES
          ERROR    12                ;SREGO SHOULD = 5
  
```

```

2250
2251
2252
2253 013012 000004
2254 013014 012702 001200
2255 013020 177227 000052
2256 013024 174267 166150
2257 013030 177327 000025
2258 013034 171322
2259 013036 000240
2260 013040 174742
2261 013042 171322
2262 013044 000240
2263 013046 172342
2264 013050 012704 000025
2265 013054 173367 166120
2266 013060 005304
2267 013062 001374
2268 013064 175737 001170
2269 013070 170267 166370
2270 013074 022737 000052 001170
2271 013102 001401
2272 013104 104012
  
```

```

*****
:TEST 64 MUL0 AND DIVD TEST
*****
†ST64: SCOPE
MOV #SREG4,R2 ;ADDR(DATA)
LDCID #52,AC2 ;LOAD AC2 WITH FLOATING DOUBLE 52
STD AC2,SREG4 ;PUT IT INTO SREG4
LDCID #25,AC3 ;LOAD AC3 WITH FL DB 25
MULD (R2)+,AC3 ;MUL 52X25 RESULT IN AC3
NOP ;DELAY SOME
DIVD -(R2),AC3 ;DIV 52 INTO RESULT IN AC3
MULD (R2)+,AC3 ;MUL 52X25
NOP ;DELAY SOME
ADD -(R2),AC3 ;ADD AC4 TO AC3 TO MAKE 53 TIMES
MOV #25,R4 ;SET UP COUNTER
SUBD SREG4,AC3 ;SUB 52 FROM AC3 25 TIMES
DEC R4
BNE SUBT ;SUB 53 TIMES
STCDI AC3,@#SREG0 ;ANS SHOULD BE 52
STFPS FPS ;GET STATUS
CMP #52,@#SREG0 ;IS ANS CORRECT?
BEQ +4 ;YES
ERROR i2 ;SREG0 SHOULD BE 52
  
```

SUBT:

```

2273
2274
2275
2276 013106 000004
2277 013110 177027 025252
2278 013114 177127 000025
2279 013120 171100
2280 013122 174137 001170
2281 013126 012702 000024
2282 013132 174005
2283 013134 172005
2284 013136 005302
2285 013140 001375
2286 013142 174037 001200
2287 013146 170267 166312
2288 013152 173437 001170
2289 013156 170000
2290 013160 001401
2291 013162 104010
2292 013164 177327 000002
2293 013170 012702 000013
2294 013174 174304
2295 013176 171304
2296 013200 005302
2297 013202 001375
2298 013204 175703
2299 013206 010337 001170
2300 013212 022703 010000
2301 013216 170267 166242
2302 013222 001401
2303 013224 104012
2304 013226 177227 010000
2305 013232 177127 000002
2306 013236 012702 000013
2307 013242 174601
2308 013244 005302
2309 013246 001375
2310 013250 175603
2311 013252 010237 001170
2312 013256 022703 000002
2313 013262 170267 166176
2314 013266 001401
2315 013270 104012

```

```

*****
:TEST 65 EXERCISER TEST
*****
TST65: SCOPE
LDCID #25252,AC0 ;LOAD 25252 INTO AC0
LDCID #25,AC1 ;LOAD AC1 WITH 25
MULD AC0,AC1 ;MUL 25252X25 ANS IN AC1
STD AC1,@#SREG0 ;SAV ANS IN @#SREG0
MOV #24,R2 ;SET UP COUNT
STD AC0,AC5 ;PUT 25252 INTO AC5
AAD: ADD AC5,AC0 ;ADD 25252 TO 25252
DEC R2 ;DO 25 TIMES
BNE AAD ;DONE?
STD AC0,@#SREG4 ;LOAD TO PRINT
STFPS FPS ;GET STATUS
CMPD @#SREG0,AC0 ;IS ANS CORRECT?
CFCC
BEQ .+4 ;YES
ERROR 10 ;EITHER THE ADD OR THE MUL DID NOT WORK
LDCID #2,AC3 ;LOAD AC3 WITH A 2
MOV #13,R2 ;SET UP COUNTER
STD AC3,AC4 ;LOAD AC4 WITH A 2
MMUL: MULD AC4,AC3 ;MUL 2 X 2
DEC R2 ;DO 16 TIMES
BNE MMUL
STCDI AC3,R3 ;PUT ANS IN R3
MOV R3,@#SREG0 ;NOW PUT IT INTO @#SREG0
CMP #10000,R3 ;ANS SHOULD BE 10000
STFPS FPS ;GET STATUS
BEQ .+4 ;CONT. IF ANS IS CORRECT
ERROR 12 ;ANS SHOULD BE 10000
LDCID #10000,AC2 ;LOAD AC2 WITH 10000
LDCID #2,AC1 ;LOAD AC1 WITH A 2
MOV #13,R2 ;SET UP COUNTER
DDIV: DIVD AC1,AC2 ;DIVD 2 INTO 65536 16 TIMES
DEC R2 ;COUNT NO OF TIMES
BNE DDIV ;ARE WE DONE?
STCDI AC2,R3 ;STOR ANS INTO R3
MOV R2,@#SREG0 ;PUT IT INTO @#SREG0 FOR TYPING
CMP #2,R3 ;IS ANS CORRECT
STFPS FPS ;GET STATUS
BEQ .+4 ;ANS IS CORRECT
ERROR 12 ;SREG0 SHOULD EQUAL =2

```



```

2316
2317
2318
2319 013272 000004
2320 013274 170127 047400
2321 013300 177327 000005
2322 013304 012702 001170
2323 013310 177227 000007
2324 013314 174322
2325 013316 173737 001170
2326 013322 170267 166136
2327 013326 170000
2328 013330 001401
2329 013332 104011
2330 013334 171242
2331 013336 012703 000006
2332 013342 173212
2333 013344 005303
2334 013346 001375
2335 013350 175637 001170
2336 013354 022737 000005 001170
2337 013362 170267 166076
2338 013366 001401
2339 013370 104012
2340
2341
2342
2343
2344 013372 000004
2345 013374 170127 047600
2346 013400 177027 000252
2347 013404 177227 052525
2348 013410 174204
2349 013412 012702 000251
2350 013416 171002
2351 013420 000240
2352 013422 172204
2353 013424 005302
2354 013426 001375
2355 013430 174237 001170
2356 013434 170267 166024
2357 013440 173402
2358 013442 170000
2359 013444 001401
2360 013446 104010

```

```

*****
*TEST 66 MODE ONE TEST
*****
†ST66: SCOPE
LDFPS #47400
LDCIF #5,AC3 ;AC3=5
MOV #SREG0,R2 ;R2=SREG0
LDCIF #7,AC2 ;LOAD 7 INTO AC2
STF AC3,(R2)+ ;SREG0 SHOULD =5=R2
CMPF #SREG0,AC3 ;DOES #SREG0 = 5
STFPS FPS ;GET STATUS
CFCC
BEQ .+4 ;YES BRANCH
ERROR i1 ;SREG0 SHOULD CONTAIN 5
MULF -(R2),AC2 ;MUL 5 X 7,AC2 = 35
MOV #6,R3 ;SET UP COUNTER
SUBFM: SUBF (R2),AC2 ;SUB 5 FROM 35
DEC R3 ;DO 7 TIMES
BNE SUBFM
STCFI AC2,#SREG0 ;#SREG0 SHOULD =5
CMP #5,#SREG0 ;DOES #SREG0 =5
STFPS FPS ;GET STATUS
BEQ .+4 ;BRANCH IF YES
ERROR i2 ;ANS SHOULD =5

```

```

*****
*TEST 67 ADD EXERCISER
*****
†ST67: SCOPE
LDFPS #47600
LDCID #252,AC0 ;LOAD AC0 WITH 252
LDCID #52525,AC2 ;LOAD AC2 WITH 52525
STD AC2,AC4 ;AC4=52525
MOV #251,R2 ;SET UP COUNTER
MULD AC2,AC0 ;AC0=52525 X 252
NOP ;DELAY SOME
AADDM: ADD AC4,AC2 ;ADD 52525 TO 252
DEC R2 ;DO 252 TIMES
BNE AADDM ;DONE?
STD AC2,#SREG0 ;GET FOR PRINTING
STFPS FPS ;GET STATUS
CMPD AC2,AC0 ;DOES AC2 = AC0?
CFCC
BEQ .+4 ;BRANCH IF EQUAL
ERROR i0 ;ANS SHOULD BE.....

```

```

2361
2362
2363
2364 013450 000004
2365 013452 170127 047600
2366 013456 177027 077777
2367 013462 177127 000000
2368 013466 174401
2369 013470 175437 001170
2370 013474 170267 165764
2371 013500 170337 001466
2372 013504 022737 077777 001170
2373 013512 001401
2374 013514 104012
2375 013516 022737 000004 001466
2376 013524 001401
2377 013526 104002
2378
2379
2380
2381
2382 013530 000004
2383 013532 170127 047400
2384 013536 170011
2385 013540 177027 077777
2386 013544 177127 002525
2387 013550 012702 000012
2388 013554 174401
2389 013556 171001
2390 013560 172001
2391 013562 173001
2392 013564 005302
2393 013566 001372
2394 013570 175437 001170
2395 013574 170267 165664
2396 013600 022737 077777 001170
2397 013606 001401
2398 013610 104012
2399 013612 172437 001556
2400 013616 012702 001170
2401 013622 174022
2402 013624 172537 001556
2403 013630 174122
2404 013632 172737 001200
2405 013636 170267 165622
2406 013642 173737 001170
2407 013646 170000
2408 013650 001401
2409 013652 104010
2410 013654 172537 001560
2411 013660 174142
2412 013662 172012
2413 013664 173042
2414 013666 173001
2415 013670 175437 001170
2416 013674 022737 000000 001170

```

```

*****
*TEST 70 TEST DIVIDE BY 0
*****
TST70: SCOPE
LDFPS #47600
LDCID #77777,AC0
LDCID #0,AC1
DIVD AC1,AC0 ;DIVIDE 0 INTO 77777
STCDI AC0,#SREG0 ;LOAD #SREG0 WITH 77777
STFPS FPS ;GET STATUS
STST #FEC ;GET #FEC STATUS
CMP #77777,#SREG0
BEQ .+4
ERROR i2 ;ANS SHOULD =77777
CMP #4,#FEC ;DID WE TRY TO DIV BY 0?
BEQ .+4 ;YES
ERROR 2 ;FEC SHOULD =4

*****
*TEST 71 EXERCISER FOR ADD, SUBD, MULD AND DIVD
*****
TST71: SCOPE
LDFPS #47400
SETD ;SET DOUBLE MODE
LDCID #77777,AC0 ;LOAD AC0 WITH 77777
LDCID #2525,AC1 ;LOAD AC1 WITH 2525
MOV #12,R2 ;SET UP COUNTER
EXLOP: DIVD AC1,AC0 ;DIVIDE 2525 INTO 77777
MULD AC1,AC0 ;MUL 2525 X ANS
ADD AC1,AC0 ;ADD 2525 TO ANS
SUBD AC1,AC0 ;SUB 2525 FROM ANS
DEC R2 ;DO 12 TIMES
BNE EXLOP ;DONE?
STCDI AC0,#SREG0 ;LOAD ANSWER INTO #SREG0
STFPS FPS ;GET STATUS
CMP #77777,#SREG0 ;IS #SREG0 CORRECT
BEQ .+4 ;BRANCH IF CORRECT
ERROR i2
EX4: LDD #01010,AC0 ;GET DATA
MOV #SREG0,R2
STD AC0,(R2)+
LDD #01010,AC1
STD AC1,(R2)+
LDD #SREG4,AC3 ;GET STATUS
STFPS FPS
CMPD #SREG0,AC3
CFCC
BEQ .+4
ERROR i0 ;SREG0 AND SREG4 SHOULD =1010
LDD #00101,AC1 ;STORED AC1 IN SREG4-7
STD AC1,-(R2)
ADD (R2),AC0
SUBD -(R2),AC0
SUBD AC1,AC0
STCDI AC0,#SREG0
CMP #0,#SREG0

```

2417	013702	170267	165556		STFPS	FPS	;GET STATUS
2418	013706	001401			BEG	.+4	
2419	013710	104012			ERROR	12	
2420	013712	170001		MORE:	SETF		;SET FLOATING MODE
2421	013714	177027	000525		LDCIF	#525,AC0	;LOAD ACP WITH 525
2422	013720	177127	000252		LDCIF	#252,AC1	;LOAD AC1 WITH 252
2423	013724	174104			STF	AC1,AC4	;LOAD AC4 WITH 252
2424	013726	172104			ADDF	AC4,AC1	;ADD 252 TO 252
2425	013730	172701			LDF	AC1,AC3	;PUT ANS IN AC3=524
2426	013732	173003			SUBF	AC3,AC0	;SUB 524 FROM 525
2427	013734	175037	001170		STEXP	AC0,@#SREG0	;1 IN SREG0
2428	013740	170267	165520		STFPS	FPS	;GET STATUS
2429	013744	022737	000001	001170	CMP	#1,@#SREG0	;CORRECT ANS SHOULD BE 1
2430	013752	001401			BEG	.+4	
2431	013754	104012			ERROR	12	;ANS SHOULD BE 5
2432	013756	177027	000021		LDCIF	#21,AC0	;LOAD AC0 WITH 21
2433	013762	171000			MULF	AC0,AC0	;21 TIMES 21 AC0 = 441
2434	013764	174427	040400		DIVF	#2,AC0	;DIV BY 2
2435	013770	012701	001552		MOV	#FLTONE+4,R1	;ADDR(DATA)+4
2436	013774	171441			MODF	-(R1),AC0	;MUL 1 * 441.
2437	013776	170267	165462		STFPS	FPS	;GET STATUS
2438	014002	175537	001170		STCFI	AC1,@#SREG0	;PUT SREG0 IN SREG0
2439	014006	022737	000220	001170	CMP	#220,@#SREG0	;IS IT EQUAL?
2440	014014	001401			BEG	.+4	;YES
2441	014016	104012			ERROR	12	;SHOULD = 220
2442	014020	171427	041040		MODF	#10,AC0	;GET FRACTION
2443	014024	175537	001170		STCFI	AC1,@#SREG0	;PUT AC0 INTO SREG0
2444	014030	022737	000005	001170	CMP	#5,@#SREG0	;SREG0 SHOULD = 5
2445	014036	001401			BEG	.+4	
2446	014040	104012			ERROR	12	;SREG0 SHOULD = 5

```

*****
*****
;TEST 72 ...CLEAR OUT CLOCKS BEFORE SEOP
*****

```

```

†ST72: SCOPE
      CLR STIMES ;NO ITER. OF THIS TEST
      CLR SERFLG ;NO ERRORS EITHER

```

```

; *DONE WITH BASIC INSTRUCTION EXERCISER
; *CLEAR ALL LINE CLOCKS ENABLED IN SYSTEM

```

2461	014054	105767	165436		TSTB	CLKPRS	;KW11-L PRESENT ?
2462	014060	100002			BPL	NKWL2	;NO
2463	014062	005037	177546		CLR	@#LKS	;YES, CLEAR IT
2464							
2465	014066	005767	165424	NKWL2:	TST	CLKPRS	;KW11-P PRESENT ?
2466	014072	100002			BPL	NKWP2	;NO
2467	014074	005037	172540		CLR	@#PLKS	;YES, CLEAR IT
2468	014100			NKWP2:			
2469							

2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525

```

;*****
;.ENABL AMA ;ASSEMBEL ALL PC RELATIVE REFERENCES AS ABSOLUTE
;*****
;*****
.SBTTL SUB PASS END CONTROL

SCOPE ;CHECK FOR TEST ITERATIONS HERE
CLR STIMES ;DONT ITERATE THIS "TEST"
CLR SERFLG ;NO ERRORS HERE
CLR STSTNM ;ZAP TEST ## WHEN DONE WITH A PASS

;IF TEST ONLY EITHER HFP OR WFP, ENTER "EOP" ROUTINE DIRECTLY
; IF IN ALTERNATE HFP/WFP MODE,
; COMPLEMENT FLAG<5>, HFP ENABLE BIT,
; ENTER EOP ROUTINE ONLY IF ABOUT TO TEST HFP NEXT,
; TESTING SEQUENCE IS: PASS#1 HFP SUB-PASS
; PASS#1 WFP SUB-PASS
; PASS#2 HFP SUB-PASS
; ...

014116 000004 001230 MED ,WHAMI ;GET WHAMI INTO RO
014102 005037 001104 BIT #BIT04,RO ;I=HFP PRESENT, 0=NONE
014106 005037 001104 BEQ SEOP ;EXIT IF NONE

014112 005037 001102 BIT #SW01,SWR ;I=HFP OR WFP TEST ONLY
014116 076600 000022 BNE SEOP ;0=ALTERNATE HFP AND WFP TESTS

014140 012701 010000 MOV #BIT12,R1 ;HFP PRESENT, AND IN ALTERNATE MODE;
014144 076600 000144 MED RFLAG ;SO READ FLAGS
014150 030100 BIT R1,RO ;COMPLEMENT FLAG<5>=BIT12=HFP ENABLE FLAG
014152 001402 BEQ IS ;
014154 040100 BIC R1,RO ;CLEAR BIT 12
014156 000401 BR 2S ;
014160 050100 1S: BIS R1,RO ;SET BIT 12
014162 076600 000344 2S: MED ,WFLAG ;REWRITE FLAGS

014166 030100 BIT R1,RO ;HFP OR WFP NEXT ?
014170 001002 BNE SEOP ;IF HFP AGAIN, START NEW PASS
014172 000137 003256 JMP #SUBPAS ;IF WFP, NEXT SUBPASS

;*****
.SBTTL END OF PASS ROUTINE (MODIFIED SYSMAC)

;*INCREMENT THE PASS NUMBER ($PASS)
;*IF SW<10>=0, DING BELL ON PASS END
;*IF THERE'S A MONITOR, GO TO IT
;* ELSE JUMP TO NEWPAS

SEOP:
CLR SERFLG ;ZERO ERROR COUNT
CLR STSTNM ;ZERO TEST NUMBER

```

2526	014206	005037	001230		CLR	\$TIMES	;ZERO NUMBER OF ITERATIONS
2527	014212	005237	001252		INC	\$PASS	;INCREMENT PASS COUNT,
2528	014216	042737	100000	001252	BIC	#100000,\$PASS	;BUT NEVER LET IN GO NEGATIVE
2529	014224	005327			DEC	(PC)+	;PASS LOOP ?
2530	014226	000001			SEOPCT:	.WORD	
2531	014230	003021			BGT	\$DOAGN	;YES
2532	014232	012737			MOV	(PC)+,\$(PC)+	;RESTORE COUNTER
2533	014234	000001			SENDCT:	.WORD	
2534	014236	014226			SEOPCT		
2535	014240	032777	002000	164676	BIT	#SW10,\$SWR	;BELL ON PASS END ?
2536	014246	001002			BNE	\$GET42	;NO
2537	014250	104401	001234		TYPE	,\$BELL	;YES
2538							
2539	014254	013700	000042		\$GET42:	MOV	;\$42,R0
2540	014260	001405			BEQ	\$DOAGN	;NO MONITOR
2541	014262	000005			RESET		;CLEAR WORLD
2542							
2543	014264	004710			SENDAD:	JSR	PC,(R0)
2544	014266	000240			NOP		;GO TO MONITOR
2545	014270	000240			NOP		;RESERVED FOR ACT11
2546	014272	000240			NOP		;RESERVED FOR ACT11
2547							
2548	014274	000137	003220		\$DOAGN:	JMP	;\$NEWPAS
2549							;RETURN
2550							
2551							

.SBTTL TRAP/FEC TESTER SUBR

```

2553 014300 013746 000244 TRPTST: MOV 3#FPPVEC, -(SP) ; SAVE OLD FPPVEC PC
2554 014304 013746 000246 MOV 3#FPPVEC+2, -(SP) ; AND PS
2555 014310 012737 014402 000244 MOV #FPPTRP, 3#FPPVEC ; NEW VECTOR PC FOR TEST
2556 014316 016537 000002 000246 MOV 2(R5), 3#FPPVEC+2 ; NEW VECTOR PS FOR TEST
2557 014324 012700 000006 MOV #6, R0 ; LOAD STMPO-5
2558 014330 010501 MOV #5, R1 ; WITH TEST DATA SETS
2559 014332 012702 001210 MOV #STMPO, R2 ; FOR DISPLAY LATER
2560 014336 012122 MOV (R1)+, (R2)+
2561 014340 077002 SOB R0, -2
2562 014342 012737 014352 001112 MOV #FLPERR, $LPERR ; ERROR LOOP TO HERE
2563 014350 010602 MOV SP, R2 ; SAVE GOOD SP
2564
2565
2566 014352 010206 FLPERR: MOV R2, SP ; RESET TO GOOD SP
2567 014354 170165 LDFPS 6(R5) ; SET UP INITIAL FPS
2568 014360 011537 177776 MOV (R5), 3#PSW ; SET UP INITIAL PSW
2569
2570
2571
2572
2573
2574
2575 014364 000114 JMP (R4) ; GO TEST W/O USING STACK
2576
2577
2578 014366 032765 040000 000012 FPPNTP: BIT #BIT14, 12(R5) ; NO TRAP EXPECTED ?
2579 014374 001455 BEQ FCONT1 ; BR IF YES
2580 014376 104014 ERROR 14 ; DIDNT TRAP, SHOULD HAVE
2581
2582 014400 000453 BR FCONT1 ; CONTINUE WITH TEST
2583
2584
2585 014402 013737 177776 001512 FPPTRP: TRAP RETURNS HERE
2586 014410 011637 001506 MOV 3#PSW, NPSLOD ; SAVE LOADED PSW
2587 014414 016637 000002 001510 MOV (SP), OPCRCV ; GET PUSHED PC
2588 014422 010637 001514 MOV 2(SP), OPSRCV ; GET PUSHED PS
2589 014426 012716 014434 MOV SP, OSPRCV ; GET SP AFTER TRAP
2590 014432 000002 RTI ; FUDGE RETURN
2591
2592
2593 014434 032765 040000 000012 2$: BIT #BIT14, 12(R5) ; TRAP EXPECTED ?
2594 014442 001001 BNE 15 ; BR IF YES
2595 014444 104015 ERROR 15 ; TRAPPED, SHOULDNT HAVE
2596
2597 014446 010237 001504 1$: MOV R2, EXPSP ; GENERATE EXPECTED SP CONTENTS
2598 014452 162737 000004 001504 SUB #4, EXPSP ; AFTER TRAP
2599 014460 023737 001514 001504 CMP OSPRCV, EXPSP ; IS SP WHERE IT SHOULD BE?
2600 014466 001401 BEQ 64$ ;
2601 014470 104016 ERROR 16 ; NOT EQUAL, SIGNAL ERROR
2602
2603 014472 023737 001502 001506 CMP EXPRTS, OPCRCV ; CHECK STORED PC IS OK
2604 014500 001401 BEQ 65$ ;
2605 014502 104017 ERROR 17 ; NOT EQUAL, SIGNAL ERROR
2606 014504 65$:
2607 014504 026537 000004 001510 CMP 4(R5), OPSRCV ; CHECK STORED PS IS OK

```

NOTE: THE "LOADED PS" IS SETUP WITH PROCESSOR PRIORITY = (7), SO THAT INTERRUPTS FROM ANY CLOCKS ENABLED ARE EFFECTIVELY LOCKED OUT DURING THIS TESTING ROUTINE.

```

2608 014512 001401          BEQ      66$          ;
2609 014514 104020          ERROR    20          ; NOT EQUAL, SIGNAL ERROR
2610 014516          66$:
2611
2612 014516 026537 000002 001512      CMP      2(R5),NPSLOD ; CHECK LOADED PS IS OK
2613 014524 001401          BEQ      67$          ;
2614 014526 104021          ERROR    21          ; NOT EQUAL, SIGNAL ERROR
2615 014530          67$:
2616
2617          ;----- CONTINUE -----
2618 014530 170237 001464      FCONT1: STFPS   FPS          ; STORE FPS AFTER
2619 014534 170337 001466      STST    FEC          ; STORE FEC/FEA AFTER
2620
2621 014540 023765 001464 000010      CMP      FPS,10(R5)   ; CHECK FPS
2622 014546 001401          BEQ      65$          ; FPS IS OK
2623 014550 104023          ERROR    23          ; FPS BAD
2624 014552 005765 000012      65$:  TST      12(R5)     ; DOES FEC/FEA APPLY?
2625 014556 100014          BPL      66$          ; NO - SKIP TEST
2626 014560 013737 001500 001500      MOV      EXPFEA,EXPFEA ; GET EXPECTED FEA
2627 014566 123765 001466 000012      CMPB    FEC,12(R5)   ; COMPARE FEC-S
2628 014574 001004          BNE      64$          ; NOT EQUAL
2629 014576 023737 001470 001500      CMP      FEA,EXPFEA  ; COMPARE FEA-S
2630 014604 001401          BEQ      66$          ; FEC, FEA OK
2631 014606 104022          64$:  ERROR    22          ; FEC OR FEA ARE BAD
2632 014610          66$:
2633
2634 014610 010206          MOV      R2,SP       ; RESTORE GOOD SP AFTER TEST, IN CASE
2635
2636 014612 012637 000246      MOV      (SP)+,2#FPPVEC+2 ; RESET STANDARD FPP VECTOR
2637 014616 012637 000244      MOV      (SP)+,2#FPPVEC  ;
2638
2639 014622 005037 177776      CLR      2#PSW       ; SETUP FOR INTERRUPT ENABLE, PRO
2640 014626 005003          CLR      R3         ; ZEROES FOR LDUB
2641 014630 170003          LDUB    HFP,UBRK <- R3, ZEROES
2642 014632 170127 040000      LDFPS   #040000     ; INTR DISABLE, FMM=0
2643 014636 000207          RTS      PC         ; RETURN TO TEST CALLER
2644
2645          ;----- LOCAL FEC-CODE GENERATING SUBR -----
2646
2647 014640          FEC02:
2648 014640 170406          FII02: 170406       ; ILLEGAL FPP INSTR
2649 014642 174000          NXT02: STF      ACO,ACO ; FORCE HFP/WFP SERVICE, EVEN AT PR7
2650 014644 000137 014366      JMP      FPPNTP     ;
2651
2652 014650 172437 015002      FEC04: LDF      FLT001,ACO ;
2653 014654 174437 014772      FII04: DIVF    FLTZER,ACO ; X/0.0 = ?
2654 014660 174000          NXT04: STF      ACO,ACO ; FORCE HFP/WFP SERVICE, EVEN AT PR7
2655 014662 000137 014366      JMP      FPPNTP     ;
2656
2657 014666 172537 015002      FEC06: LDF      FLT001,AC1 ;
2658 014672 175537 001170      FII06: STCFI   AC1,$REGO ; FLT001 > LGST I
2659 014676 174000          NXT06: STF      ACO,ACO ; FORCE HFP/WFP SERVICE, EVEN AT PR7
2660 014700 000137 014366      JMP      FPPNTP     ;
2661
2662 014704 172637 015002      FEC10: LDF      FLT001,AC2 ;
2663 014710 172737 015006      LDF      FLT002,AC3 ;

```

2664	014714	171203		FII10:	MULF	AC3,AC2	:	LG * LG = OVFLW
2665	014716	174000		NXT10:	STF	AC0,AC0	:	FORCE HFP/WFP SERVICE, EVEN AT PR7
2666	014720	000137	014366		JMP	FPPNTP	:	
2667							:	
2668	014724	172637	015012	FEC12:	LDF	FLT003,AC2	:	
2669	014730	171237	015016	FII12:	MULF	FLT004,AC2	:	SM * SM = UNDFLW
2670	014734	174000		NXT12:	STF	AC0,AC0	:	FORCE HFP/WFP SERVICE, EVEN AT PR7
2671	014736	000137	014366		JMP	FPPNTP	:	
2672							:	
2673	014742			FEC14:			:	
2674	014742	172737	014776	FII14:	LDF	FLTMZR,AC3	:	-0.0 -> AC3
2675	014746	174000		NXT14:	STF	AC0,AC0	:	FORCE HFP/WFP SERVICE, EVEN AT PR7
2676	014750	000137	014366		JMP	FPPNTP	:	
2677							:	
2678	014754	012703	000342	FEC16:	MOV	#342,R3	:	HFP U-ADDR FOR "LDF/D" INSTR
2679	014760	170003			LDUB		:	
2680	014762	172400		FII16:	LDF	AC0,AC0	:	U-BREAK TRAP
2681	014764	174000		NXT16:	STF	AC0,AC0	:	FORCE HFP/WFP SERVICE, EVEN AT PR7
2682	014766	000137	014366		JMP	FPPNTP	:	
2683							:	
2684							:	
2685	014772	000000	000000	:----- CONSTANTS -----				
2686	014776	100000	000000	FLTZER:	.WORD	000000,000000	:	+0.0
2687	015002	044000	000000	FLTMZR:	.WORD	100000,000000	:	-0.0
2688	015006	074377	177777	FLT001:	.WORD	044000,000000	:	.1E+20 = 65536
2689				FLT002:	.WORD	074377,177777	:	.111 ... 1E+161
2690	015012	030000	000000				:	; NOTE FLT001*FLT002 = .111 ... 1E+200 (OVERFLOW)
2691	015016	010000	000000	FLT003:	.WORD	030000,000000	:	.1E-40
2692				FLT004:	.WORD	010000,000000	:	.1E-140
							:	; NOTE FLT003*FLT004 = .1E-201 (UNDERFLOW)


```

2693
2694
2695 015022 005037 177546          IKW11L: CLR      2#LKS      ;CLEAR STATUS
2696 015026 005237 001520          ;INC      KW11LC      ;BUMP COUNTER
2697 015032 042737 100000 001520  ;BIC      #BIT15,KW11LC ;NEVER OVERFLOW
2698 015040 052737 000100 177546  ;BIS      #BIT6,2#LKS  ;RESTART CLOCK
2699 015046 000002          ;RTI          ;AND RETURN
2700
2701 015050 005037 172540          IKW11P: CLR      2#PLKS   ;CLEAR STATUS
2702 015054 005237 001522          ;INC      KW11PC      ;BUMP COUNTER
2703 015060 042737 100000 001522  ;BIC      #BIT15,KW11PC ;NEVER OVERFLOW
2704
2705 ;THE NEXT 13 INSTRUCTIONS GENERATE A 10 BIT RANDOM
2706 ;INTEGER FROM 0001-1777 INCLUSIVE TO FEED INTO THE
2707 ;KW11-P COUNT SET REGISTER.
2708 015066 005046          CLR      -(SP)      ;TEMP STORAGE FOR SHIFTED OUT BIT
2709 015070 006237 001524          ASR      KW11PR     ;SFT OLD CONTENTS RITE 1; LOB INTO C
2710 015074 006116          ROL      (SP)      ;PUT OLD LOB ON STACK
2711 015076 032737 000004 001524  ;BIT      #BIT2,KW11PR ;WAS OTHER BIT ON ?
2712 015104 001403          BEQ     1$        ;NO
2713 015106 005116          COM     (SP)      ;YES, COMPLEMENT OLD LOB
2714 015110 042716 177776          BIC     #177776,(SP) ;ELIMINATE JUNK
2715 015114 000316          1$: SWAB  (SP)     ;FAST SHIFT LOB TO BIT9 POSITION
2716 015116 006316          ASL     (SP)
2717 015120 052637 001524          BIS     (SP)+,KW11PR ;PUT NEW HOB INTO COUNTER
2718 015124 042737 176000 001524  ;BIC     #176000,KW11PR ;NO HIGH ORDER JUNK
2719 015132 001003          BNE     2$        ;ZERO IS AN ILLEGAL STATE,
2720 015134 012737 001777 001524  ;MOV     #1777,KW11PR ;SO RESET TO START
2721
2722 015142 013737 001524 172542  2$: MOV     KW11PR,2#PLKR ;SET NEW COUNT
2723 015150 052737 000101 172540  ;BIS     #BIT6+BIT0,2#PLKS ;RESTART CLOCK
2724 015156 000002          ;RTI          ;AND RETURN

```

```

2725          .SBTTL  FPP TRAP CATCHER
2726
2727 015160 010637 001514      FPPILT: MOV      SP,OSPRCV      ; SP AFTER TRAP
2728 015164 012637 001472      MOV      (SP)+,FPPOPC      ; POP OLD PC FOR DISPLAY
2729 015170 012637 001474      MOV      (SP)+,FPPOPS      ; POP OLD PS FOR DISPLAY
2730 015174 170237 001464      STFPS   FPS                ; GET FPS
2731 015200 170337 001466      STST    FEC                ; GET FEC/FEA
2732 015204 005737 001464      TST     FPS                ; TEST ERROR BIT
2733 015210 100007                BPL     1$                 ; OFF - NO ERROR BIT SET, BUT TRAPPED
2734
2735 015212 032737 040000 001464      BIT     #040000,FPS        ; TEST INTERRUPT ENABLE BIT
2736 015220 001003                BNE     1$                 ; ON - INTR DISABLED, BUT TRAPPED
2737
2738 015222 012701 177777                MOV     #-1,R1            ; OFF - ABLE TO INTR, SO IGNORE IT,
2739 015226 000401                BR      2$                 ; BUT FLAG THAT IT OCCURRED
2740 015230 104024                1$:   ERROR 24            ; AND SKIP THE ERROR
2741 015232 013746 001474                2$:   MOV     FPPOPS,-(SP) ; SIGNAL UNEXPECTED FPP TRAP
2742 015236 013746 001472                MOV     FPOPC,-(SP)      ; PUSH PSM
2743 015242 000002                RTI                      ; PUSH PC
2744

```

;CONTINUE, RECOVER AT LAST TRAP ONLY

2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800

015244
015244
015244 032777 040000 163672
015252 001114
015254 000416
015256 013746 000004
015262 012737 015302 000004
015270 005737 177060
015274 012637 000004
015300 000463
015302 022626
015304 012637 000004
015310 000423
015312
015312 032777 000400 163624
015320 001404
015322 023737 001150 001102
015330 001465
015332 005737 001104
015336 001421
015340 023737 001120 001104
015346 101015
015350 032777 001000 163566
015356 001404
015360 013737 001112 001110
015366 000446
015370 005037 001104
015374 005037 001230
015400 000415
015402 032777 004000 163534
015410 001011
015412 005737 001252
015416 001406
015420 005237 001106
015424 023737 001230 001106
015432 002024
015434 012737 000001 001106
015442 013737 015520 001230
015450 005237 001102
015454 013737 001102 001250
015462 011637 001110

```
.SBTTL SCOPE HANDLER ROUTINE
;*****
;THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
;AND LOAD THE TEST NUMBER($STNM) INTO THE DISPLAY REG.(DISPLAY<15:0>)
;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
;SW14=1 LOOP ON TEST
;SW11=1 INHIBIT ITERATIONS
;SM09=1 LOOP ON ERROR
;SM08=1 LOOP ON TEST IN "SLPTST"
;CALL
;* SCOPE ;;SCOPE=IOT

$SCOPE:
64S:
1S: BIT #BIT14,$SWR ;;LOOP ON PRESENT TEST?
BNE $OVER ;;YES IF SW14=1
;*****START OF CODE FOR THE XOR TESTER*****
$XTSTR: BR 6S ;;IF RUNNING ON THE "XOR" TESTER CHANGE
THIS INSTRUCTION TO A "NOP" (NOP=240)
2765 MOV $ERRVEC,-(SP) ;;SAVE THE CONTENTS OF THE ERROR VECTOR
2766 MOV #55,$ERRVEC ;;SET FOR TIMEOUT
2767 TST $#177060 ;;TIME OUT ON XOR?
2768 MOV (SP)+,$ERRVEC ;;RESTORE THE ERROR VECTOR
2769 BR $SVLAD ;;GO TO THE NEXT TEST
2770 5S: CMP (SP)+,(SP)+ ;;CLEAR THE STACK AFTER A TIME OUT
2771 MOV (SP)+,$ERRVEC ;;RESTORE THE ERROR VECTOR
2772 BR 7S ;;LOOP ON THE PRESENT TEST
2773 6S:;*****END OF CODE FOR THE XOR TESTER*****
2774 BIT #BIT08,$SWR ;;LOOP ON SPEC. TEST?
2775 BEQ 2S ;;BR IF NO
2776 CMP $SLPTST,$STNM ;;ON THE RIGHT TEST?
2777 BEQ $OVER ;;BR IF YES
2778 2S: TST $SERFLG ;;HAS AN ERROR OCCURRED?
2779 BEQ 3S ;;BR IF NO
2780 CMP $SERMAX,$SERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
2781 BHI 3S ;;BR IF NO
2782 BIT #BIT09,$SWR ;;LOOP ON ERROR?
2783 BEQ 4S ;;BR IF NO
2784 7S: MOV $SLPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
2785 BR $OVER
2786 4S: CLR $SERFLG ;;ZERO THE ERROR FLAG
2787 CLR $TIMES ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
2788 BR 1S ;;ESCAPE TO THE NEXT TEST
2789 3S: BIT #BIT11,$SWR ;;INHIBIT ITERATIONS?
2790 BNE 1S ;;BR IF YES
2791 TST $PASS ;;IF FIRST PASS OF PROGRAM
2792 BEQ 1S ;;INHIBIT ITERATIONS
2793 INC $ICNT ;;INCREMENT ITERATION COUNT
2794 CMP $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
2795 BGE $OVER ;;BR IF MORE ITERATION REQUIRED
2796 1S: MOV #1,$ICNT ;;REINITIALIZE THE ITERATION COUNTER
2797 MOV $SMXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
2798 $SVLAD: INC $STNM ;;COUNT TEST NUMBERS
2799 MOV $STNM,$TESTN ;;SET TEST NUMBER IN APT MAILBOX
2800 MOV (SP),$LPADR ;;SAVE SCOPE LOOP ADDRESS
```

F07

FPU INSTR EXERCISER MACY11 27(1006) 04-MAY-77 19:35 PAGE 68
DQFPCB.P11 04-MAY-77 19:34 SCOPE HANDLER ROUTINE

SEQ 0069

2801	015466	011637	001112		MOV	(SP), SLPERR	:: SAVE ERROR LOOP ADDRESS
2802	015472	005037	001232		CLR	SESCAPE	:: CLEAR THE ESCAPE FROM ERROR ADDRESS
2803	015476	012737	000001	001120	MOV	#1, SERMAX	:: ONLY ALLOW ONE(1) ERROR ON NEXT TEST
2804	015504	013777	001102	163434	SOVER: MOV	\$TSTNM, @DISPLAY	:: DISPLAY TEST NUMBER
2805	015512	013716	001110		MOV	SLPADR, (SP)	:: FUDGE RETURN ADDRESS
2806	015516	000002			RTI		:: FIXES PS
2807	015520	003720			SMXCNT: 2000.		:: MAX. NUMBER OF ITERATIONS

```

2808 .SBTTL ERROR HANDLER ROUTINE
2809
2810 ::*****
2811 ::THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
2812 ::SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
2813 ::AND GO TO STYPERR ON ERROR
2814 ::THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
2815 ::SW15=1 HALT ON ERROR
2816 ::SW13=1 INHIBIT ERROR TYPEOUTS
2817 ::SW10=1 BELL ON ERROR
2818 ::SW09=1 LOOP ON ERROR
2819 ::CALL
2820 ::* ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER
2821
2822 SERROR:
2823 015522 010037 001526 MOV R0, EREG0 ; DISPLAY R0
2824 015526 010137 001530 MOV R1, EREG1 ; R1
2825 015532 010237 001532 MOV R2, EREG2 ; R2
2826 015536 010337 001534 MOV R3, EREG3 ; R3
2827 015542 010437 001536 MOV R4, EREG4 ; R4
2828 015546 010537 001540 MOV R5, EREG5 ; R5
2829 015552 010637 001542 MOV R6, EREG6 ; GET R6(SP) BEFORE TRAP
2830 015556 062737 000004 ADD #4, EREG6
2831 015564 011637 001544 MOV (SP), EREG7 ; PC -> ERROR CALL INSTR
2832 015570 005237 001104 7$: INC $ERFLG ; SET THE ERROR FLAG
2833 015574 001775 BEQ 7$ ; DON'T LET THE FLAG GO TO ZERO
2834 015576 013777 001102 163342 MOV $STNM, $DISPLAY ; DISPLAY TEST NUMBER
2835 015604 032777 002000 163332 BIT #BIT10, $SWR ; BELL ON ERROR?
2836 015612 001402 BEQ 1$ ; NO - SKIP
2837 015614 104401 001234 TYPE , $BELL ; RING BELL
2838 015620 005237 001114 1$: INC $ERTTL ; COUNT THE NUMBER OF ERRORS
2839 015624 011637 001122 MOV (SP), $ERRPC ; GET ADDRESS OF ERROR INSTRUCTION
2840 015630 162737 000002 001122 SUB #2, $ERRPC
2841 015636 117737 163260 001116 MOVB 2$ERRPC, $ITEMB ; STRIP AND SAVE THE ERROR ITEM CODE
2842 015644 032777 020000 163272 BIT #BIT13, $SWR ; SKIP TYPEOUT IF SET
2843 015652 001004 BNE 20$ ; SKIP TYPEOUTS
2844 015654 004737 015764 JSR PC, $TYPERR ; GO TO USER ERROR ROUTINE
2845 015660 104401 001241 TYPE , $CRLF
2846 015664
2847 015664 122737 000001 001264 20$: CMPB #APTENV, $ENV ; RUNNING IN APT MODE
2848 015672 001007 BNE 2$ ; NO SKIP APT ERROR REPORT
2849 015674 113737 001116 015706 MOVB $ITEMB, 21$ ; SET ITEM NUMBER AS ERROR NUMBER
2850 015702 004737 016500 JSR PC, $ATY4 ; REPORT FATAL ERROR TO APT
2851 015706 000 .BYTE 0
2852 015707 000 .BYTE 0
2853 015710 000777 22$: BR 22$ ; APT ERROR LOOP
2854 015712 005777 163226 2$: TST $SWR ; HALT ON ERROR
2855 015716 100001 BPL 3$ ; SKIP IF CONTINUE
2856 015720 000000 HALT ; HALT ON ERROR!
2857 015722 032777 001000 163214 3$: BIT #BIT09, $SWR ; LOOP ON ERROR SWITCH SET?
2858 015730 001402 BEQ 4$ ; BR IF NO
2859 015732 013716 001112 MOV $LPERR, (SP) ; FUDGE RETURN FOR LOOPING
2860 015736 005737 001232 4$: TST $ESCAPE ; CHECK FOR AN ESCAPE ADDRESS
2861 015742 001402 BEQ 5$ ; BR IF NONE
2862 015744 013716 001232 5$: MOV $ESCAPE, (SP) ; FUDGE RETURN ADDRESS FOR ESCAPE
2863 015750

```

H07

FPU INSTR EXERCISER MACY11 27(1006) 04-MAY-77 19:35 PAGE 70
DGFPCB.P11 04-MAY-77 19:34 ERROR HANDLER ROUTINE

SEQ 0071

2864	015750	022737	014264	000042	CMP	#SENDAD,2#42	::ACT-11 AUTO-ACCEPT?
2865	015756	001001			BNE	65	:::BRANCH IF NO
2866	015760	000000			HALT		:::YES
2867	015762			65:			
2868	015762	000002		645:	RTI		;RETURN

```

2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883 015764
2884 015764 104401
2885 015766 001241
2886 015770 010046
2887 015772 010146
2888 015774 005000
2889 015776 153700 001116
2890 016002 001004
2891
2892 016004 013746 001122
2893 016010 104402
2894 016012 000452
2895 016014 005300
2896 016016 006300
2897 016020 010001
2898 016022 006300
2899 016024 060100
2900 016026 062700 001274
2901 016032 012037 016042
2902 016036 001404
2903 016040 104401
2904 016042 000000
2905 016044 104401 001241
2906 016050 104401 016160
2907 016054 012037 016064
2908 016060 001402
2909 016062 104401
2910 016064 000000
2911 016066 104401 001241
2912 016072 017746 000054
2913 016076 104402
2914 016100 104401 016156
2915 016104 017746 000044
2916 016110 104402
2917 016112 104401 016156
2918 016116 011000
2919 016120 001407
2920 016122 013046
2921 016124 104402
2922 016126 005710
2923 016130 001403
2924 016132 104401 016156

```

;;*****

.SBTTL ERROR MESSAGE TYPEOUT ROUTINE (MODIFIED SYSMAC)

```

: *THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
: *ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE",
: *(SERRTB) THE ERROR MESSAGE, DATA HEADER, AND DATA VALUES TO PRINT.
: *THIS ROUTINE IS IDENTICAL TO THE SYSMAC ROUTINE SERRTYP, EXCEPT THIS
: *ROUTINE PUTS A <HT> BETWEEN OCTAL TYPED DATA VALUES, SO THAT EACH
: *VALUE STARTS AT A HORIZONTAL TAB STOP. ALSO, THE DATA FORMAT
: *POINTER HAS BEEN ELIMINATED FROM THE ERROR VECTOR. THIS ROUTINE
: *ALSO ALWAYS PRINTS $TESTN AND $ERRPC AS THE FIRST TWO DATA ELEMENTS
: *(WITH APPROPRIATE HEADERS).

```

```

STYPERR:
HOTWRM: .WORD $CRLF
        MOV RO,-(SP)
        MOV R1,-(SP)
        CLR RO
        BISB @($ITEMB,RO)
        BNE 1$
        MOV $ERRPC,-(SP)
        TYPOC
        BR 7$
1$: DEC RO
    ASL RO
    MOV RO,R1
    ASL RO
    ADD R1,RO
    ADD @($SERRTB,RO)
    MOV (RO)+,2$
    BEQ 3$
2$: .WORD 0
    TYPE , $CRLF
3$: TYPE 11$
    MOV (RO)+,4$
    BEQ 5$
4$: .WORD 0
5$: TYPE , $CRLF
    MOV @8$,-(SP)
    TYPOC
    TYPE ,10$
    MOV @9$,-(SP)
    TYPOC
    TYPE ,10$
    MOV (RO),RO
    BEQ 7$
6$: MOV @($RO)+,-(SP)
    TYPOC
    TST (RO)
    BEQ 7$
    TYPE ,10$
        TYPE "HOT" OR "WARM"
        PTR TO MESSAGE
        SAVE RO
        SAVE R1
        PICKUP ITEM INDEX
        IF ITEM NUMBER FROM ERROR 0,
        JUST TYPE PC OF ERROR
        GET ERROR PC FOR TYPEOUT
        TYPE OCTAL, ALL DIGITS
        EXIT
        ADJUST ERROR # FOR TABLE INDEX
        OF 6 BYTES/ENTRY
        FORM TABLE PTR
        PICKUP "ERROR MESSAGE" PTR
        SKIP TYPEOUT IF NULL
        TYPE "ERROR MESSAGE"
        "ERROR MESSAGE" PTR HERE
        CR & LF
        "TEST # ERR PC" HEADER
        PICKUP "DATA HEADER" PTR
        SKIP TYPEOUT IF NULL
        TYPE "DATA HEADER"
        "DATA HEADER" PTR HERE
        CR & LF
        ($TESTN)
        OCTAL W/ LEADING ZEROS
        <HT>
        ($ERRPC)
        OCTAL W/ LEADING ZEROS
        <HT>
        PICKUP "DATA TABLE" PTR
        EXIT IF NULL
        SAVE ... FOR TYPEOUT
        TYPE OCTAL, ALL DIGITS
        ANOTHER NUMBER ?
        NO - EXIT
        TAB BETWEEN ELEMENTS

```

2925	016136	000771				BR	6\$:	LOOP ON DATA TABLE VECTOR
2926	016140	012601			7\$:	MOV	(SP)+,R1	:	RESTORE R1
2927	016142	012600				MOV	(SP)+,R0	:	RESTORE R0
2928	016144	104401	001241			TYPE	SCRLF	:	CR & LF
2929	016150	000207				RTS	PC	:	RETURN
2930	016152	001250			8\$:	.WORD	\$TESTN	:	
2931	016154	001122			9\$:	.WORD	\$ERRPC	:	
2932	016156	000011			10\$:	.ASCIZ	<11>	:	<HT>
2933	016160	042524	052123	021440	11\$:	.ASCIZ	"TEST # ERR PC	:	
2934	016166	042411	051122	050040				:	
2935	016174	004503	000					:	
2936		016200				.EVEN		:	

.SBTTL TYPE ROUTINE

:ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
:THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
:NOTE1: \$NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
:NOTE2: \$FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
:NOTE3: \$FILLC CONTAINS THE CHARACTER TO FILL AFTER.

*CALL:
:1) USING A TRAP INSTRUCTION
: TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING

*OR
: TYPE
: MESADR

2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992

016200 105737 001165
016204 100002
016206 000000
016210 000430
016212 010046
016214 017600 000002
016220 122737 000001 001264
016226 001011
016230 132737 000100 001265
016236 001405
016240 010037 016250
016244 004737 016470
016250 000000
016252 132737 000040 001265
016260 001003
016262 112046
016264 001005
016266 005726
016270 012600
016272 062716 000002
016276 000002
016300 122716 000011
016304 001430
016306 122716 000200
016312 001006
016314 005726
016316 104401
016320 001241
016322 105037 016456
016326 000755
016330 004737 016412
016334 123726 001164
016340 001350
016342 013746 001162
016346 105366 000001
016352 002770
016354 004737 016412
016360 105337 016456

STYPE: TSTB \$TPFLG
BPL 1\$
HALT
BR 3\$
1\$: MOV R0, -(SP)
MOV #2(SP), R0
CMPB #APTENV, SENV
BNE 62\$
BITB #APTPOOL, SENVM
BEQ 62\$
MOV R0, 61\$
JSR PC, SATY3
61\$: .WORD 0
62\$: BITB #APTCSUP, SENVM
BNE 60\$
2\$: MOV# (R0)+, -(SP)
BNE 4\$
60\$: MOV (SP)+, R0
3\$: ADD #2, (SP)
RTI
4\$: CMPB #HT, (SP)
BEQ 8\$
CMPB #CRLF, (SP) ;;BRANCH IF NOT <CRLF>
BNE 5\$
TST (SP)+ ;;POP <CR><LF> EQUIV
TYPE ;;TYPE A CR AND LF
\$CRLF
CLRB \$CHARCNT
BR 2\$
5\$: JSR PC, STYPEC
6\$: CMPB \$FILLC, (SP)+
BNE 2\$
MOV \$NULL, -(SP)
7\$: DECB 1(SP)
BLT 6\$
JSR PC, STYPEC
DECB \$CHARCNT

IS THERE A TERMINAL?
BR IF YES
HALT HERE IF NO TERMINAL
LEAVE
SAVE R0
GET ADDRESS OF ASCIZ STRING
RUNNING IN APT MODE
NO, GO CHECK FOR APT CONSOLE
SPOOL MESSAGE TO APT
NO, GO CHECK FOR CONSOLE
SETUP MESSAGE ADDRESS FOR APT
SPOOL MESSAGE TO APT
MESSAGE ADDRESS
APT CONSOLE SUPPRESSED
YES, SKIP TYPE OUT
PUSH CHARACTER TO BE TYPED ONTO STACK
BR IF IT ISN'T THE TERMINATOR
IF TERMINATOR POP IT OFF THE STACK
RESTORE R0
ADJUST RETURN PC
RETURN
BRANCH IF <HT>
BRANCH IF NOT <CRLF>
POP <CR><LF> EQUIV
TYPE A CR AND LF
CLEAR CHARACTER COUNT
GET NEXT CHARACTER
GO TYPE THIS CHARACTER
IS IT TIME FOR FILLER CHARS.?
IF NO GO GET NEXT CHAR.
GET # OF FILLER CHARS. NEEDED
AND THE NULL CHAR.
DOES A NULL NEED TO BE TYPED?
BR IF NO--GO POP THE NULL OFF OF STACK
GO TYPE A NULL
DO NOT COUNT AS A COUNT

```

2993 016364 000770          BR      75          ;;LOOP
2994
2995          ;HORIZONTAL TAB PROCESSOR
2996
2997 016366 112716 000040      85:    MOVB    0' (SP)          ;; REPLACE TAB WITH SPACE
2998 016372 004737 016412      95:    JSR     PC,$TYPEC          ;; TYPE A SPACE
2999 016376 132737 000007 016456  BITB    87,$CHARCNT          ;; BRANCH IF NOT AT
3000 016404 001372          BNE     95          ;; TAB STOP
3001 016406 005726          TST     (SP)+          ;; POP SPACE OFF STACK
3002 016410 000724          BR      25          ;; GET NEXT CHARACTER
3003 016412 105777 162540      $TYPEC: TSTB   25TPS          ;; WAIT UNTIL PRINTER IS READY
3004 016416 100375          BPL     $TYPEC
3005 016420 116677 000002 162532      MOVB   2(SP),25TPB          ;; LOAD CHAR TO BE TYPED INTO DATA REG.
3006 016426 122766 000015 000002      CMPB   8CR,2(SP)          ;; IS CHARACTER A CARRIAGE RETURN?
3007 016434 001003          BNE     15          ;; BRANCH IF NO
3008 016436 105037 016456          CLRB   $CHARCNT          ;; YES--CLEAR CHARACTER COUNT
3009 016442 000406          BR      $TYPEX          ;; EXIT
3010 016444 122766 000012 000002  15:    CMPB   8LF,2(SP)          ;; IS CHARACTER A LINE FEED?
3011 016452 001402          BEQ    $TYPEX          ;; BRANCH IF YES
3012 016454 105227          INCB   (PC)+          ;; COUNT THE CHARACTER
3013 016456 000000      $CHARCNT: .WORD 0          ;; CHARACTER COUNT STORAGE
3014 016460 000207      $TYPEX: RTS    PC
3015

```

```

3016 .SBTTL APT COMMUNICATIONS ROUTINE
3017
3018 ::*****
3019 016462 112737 000001 016726 SATY1: MOVB #1,SFFLG ;;TO REPORT FATAL ERROR
3020 016470 112737 000001 016724 SATY3: MOVB #1,SMFLG ;;TO TYPE A MESSAGE
3021 016476 000403 BR SATYC
3022 016500 112737 000001 016726 SATY4: MOVB #1,SFFLG ;;TO ONLY REPORT FATAL ERROR
3023 016506 SATYC:
3024 016506 010046 MOV RO,-(SP) ;;PUSH RO ON STACK
3025 016510 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
3026 016512 105737 016724 TSTB SMFLG ;;SHOULD TYPE A MESSAGE?
3027 016516 001450 BEQ 5S ;;IF NOT: BR
3028 016520 122737 000001 001264 CMPB #APTENV,SENV ;;OPERATING UNDER APT?
3029 016526 001031 BNE 3S ;;IF NOT: BR
3030 016530 132737 000100 001265 BITB #APTPOOL,SENVH ;;SHOULD SPOOL MESSAGES?
3031 016536 001425 BEQ 3S ;;IF NOT: BR
3032 016540 017600 000004 MOV #4(SP),RO ;;GET MESSAGE ADDR.
3033 016544 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
3034 016552 005737 001244 1S: TST $MSGTYPE ;;SEE IF DONE W/ LAST XMISSION?
3035 016556 001375 BNE 1S ;;IF NOT: WAIT
3036 016560 010037 001260 MOV RO,$MSGAD ;;PUT ADDR IN MAILBOX
3037 016564 105720 2S: TSTB (RO)+ ;;FIND END OF MESSAGE
3038 016566 001376 BNE 2S
3039 016570 163700 001260 SUB $MSGAD,RO ;;SUB START OF MESSAGE
3040 016574 006200 ASR RO ;;GET MESSAGE LNTH IN WORDS
3041 016576 010037 001262 MOV RO,$MSGLGT ;;PUT LENGTH IN MAILBOX
3042 016602 012737 000004 001244 MOV #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
3043 016610 000413 BR 5S
3044 016612 017637 000004 016636 3S: MOV #4(SP),4S ;;PUT MSG ADDR IN JSR LINKAGE
3045 016620 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDRESS
3046 016626 013746 177776 MOV 177776,-(SP) ;;PUSH 177776 ON STACK
3047 016632 004737 016200 JSR PC,$TYPE ;;CALL TYPE MACRO
3048 016636 000000 4S: .WORD 0
3049 016640 5S:
3050 016640 105737 016726 10S: TSTB $FFLG ;;SHOULD REPORT FATAL ERROR?
3051 016644 001416 BEQ 12S ;;IF NOT: BR
3052 016646 005737 001264 TST $SENV ;;RUNNING UNDER APT?
3053 016652 001413 BEQ 12S ;;IF NOT: BR
3054 016654 005737 001244 11S: TST $MSGTYPE ;;FINISHED LAST MESSAGE?
3055 016660 001375 BNE 11S ;;IF NOT: WAIT
3056 016662 017637 000004 001246 MOV #4(SP),$FATAL ;;GET ERROR #
3057 016670 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
3058 016676 005237 001244 INC $MSGTYPE ;;TELL APT TO TAKE ERROR
3059 016702 105037 016726 12S: CLRB $FFLG ;;CLEAR FATAL FLAG
3060 016706 105037 016725 CLRB $LFLG ;;CLEAR LOG FLAG
3061 016712 105037 016724 CLRB $MFLG ;;CLEAR MESSAGE FLAG
3062 016716 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
3063 016720 012600 MOV (SP)+,RO ;;POP STACK INTO RO
3064 016722 000207 RTS PC ;;RETURN
3065 016724 000 SMFLG: .BYTE 0 ;;MESSG. FLAG
3066 016725 000 SLFLG: .BYTE 0 ;;LOG FLAG
3067 016726 000 SFFLG: .BYTE 0 ;;FATAL FLAG
3068 016730 .EVEN
3069 000200 APTSIZE=200
3070 000001 APTENV=001
3071 000100 APTPOOL=100

```

NO7

FPU INSTR EXERCISER MACY11 27(1006) 04-MAY-77 19:35 PAGE 76
DQFPCB.P11 04-MAY-77 19:34 APT COMMUNICATIONS ROUTINE

SEQ 0077

3072

000040

APTCSUP=040

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098 016730 017646 000000
3099 016734 116637 000001 017153
3100 016742 112637 017155
3101 016746 062716 000002
3102 016752 000406
3103 016754 112737 000001 017153
3104 016762 112737 000006 017155
3105 016770 112737 000005 017152
3106 016776 010346
3107 017000 010446
3108 017002 010546
3109 017004 113704 017155
3110 017010 005404
3111 017012 062704 000006
3112 017016 110437 017154
3113 017022 113704 017153
3114 017026 016605 000012
3115 017032 005003
3116 017034 006105 1S:
3117 017036 000404 BR 3S:
3118 017040 006105 2S:
3119 017042 006105
3120 017044 006105
3121 017046 010503
3122 017050 006103 3S:
3123 017052 105337 017154
3124 017056 100016
3125 017060 042703 177770
3126 017064 001002
3127 017066 005704
3128 017070 001403

```

```

*****
: THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
: OCTAL (ASCII) NUMBER AND TYPE IT.
: $TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
: $CALL:
: *      MOV      NUM,-(SP)      ;; NUMBER TO BE TYPED
: *      TYPOS    ;; CALL FOR TYPEOUT
: *      .BYTE   N                ;; N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
: *      .BYTE   M                ;; M=1 OR 0
: *                                     ;; 1=TYPE LEADING ZEROS
: *                                     ;; 0=SUPPRESS LEADING ZEROS
: $STYON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
: $TYPOS OR $TYPOC
: $CALL:
: *      MOV      NUM,-(SP)      ;; NUMBER TO BE TYPED
: *      TYPON    ;; CALL FOR TYPEOUT
: $STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
: $CALL:
: *      MOV      NUM,-(SP)      ;; NUMBER TO BE TYPED
: *      TYPOC    ;; CALL FOR TYPEOUT
: $TYPOS: MOV      2(SP),-(SP)    ;; PICKUP THE MODE
:          MOVVB   1(SP),SOFILL  ;; LOAD ZERO FILL SWITCH
:          MOVVB   (SP)+,SOMODE+1 ;; NUMBER OF DIGITS TO TYPE
:          ADD     #2,(SP)        ;; ADJUST RETURN ADDRESS
:          BR      $TYPON
: $TYPOC: MOVVB   #1,SOFILL      ;; SET THE ZERO FILL SWITCH
:          MOVVB   #6,SOMODE+1   ;; SET FOR SIX(6) DIGITS
: $TYPON: MOVVB   #5,SOCNT      ;; SET THE ITERATION COUNT
:          MOV     R3,-(SP)      ;; SAVE R3
:          MOV     R4,-(SP)      ;; SAVE R4
:          MOV     R5,-(SP)      ;; SAVE R5
:          MOVVB   SOMODE+1,R4   ;; GET THE NUMBER OF DIGITS TO TYPE
:          NEG     R4
:          ADD     #6,R4         ;; SUBTRACT IT FOR MAX. ALLOWED
:          MOVVB   R4,SOMODE     ;; SAVE IT FOR USE
:          MOVVB   SOFILL,R4     ;; GET THE ZERO FILL SWITCH
:          MOV     12(SP),R5     ;; PICKUP THE INPUT NUMBER
:          CLR     R3           ;; CLEAR THE OUTPUT WORD
:          ROL    R5           ;; ROTATE MSB INTO "C"
:          BR     3S          ;; GO DO MSB
:          ROL    R5           ;; FORM THIS DIGIT
:          ROL    R5
:          ROL    R5
:          MOV    R5,R3
:          ROL    R3           ;; GET LSB OF THIS DIGIT
:          DECB   SOMODE       ;; TYPE THIS DIGIT?
:          BPL    7S          ;; BR IF NO
:          BIC    #177770,R3   ;; GET RID OF JUNK
:          BNE    4S          ;; TEST FOR 0
:          TST   R4           ;; SUPPRESS THIS 0?
:          BEQ   5S          ;; BR IF YES

```

3129	017072	005204		4S:	INC	R4	:: DON'T SUPPRESS ANYMORE 0'S
3130	017074	052703	000060		BIS	0'0,R3	:: MAKE THIS DIGIT ASCII
3131	017100	052703	000040	5S:	BIS	0' R3	:: MAKE ASCII IF NOT ALREADY
3132	017104	110337	017150		MOVB	R3,8S	:: SAVE FOR TYPING
3133	017110	104401	017150		TYPE	8S	:: GO TYPE THIS DIGIT
3134	017114	105337	017152	7S:	DECB	\$OCNT	:: COUNT BY 1
3135	017120	003347			BGT	2S	:: BR IF MORE TO DO
3136	017122	002402			BLT	6S	:: BR IF DONE
3137	017124	005204			INC	R4	:: INSURE LAST DIGIT ISN'T A BLANK
3138	017126	000744			BR	2S	:: GO DO THE LAST DIGIT
3139	017130	012605		6S:	MOV	(SP)+,R5	:: RESTORE R5
3140	017132	012604			MOV	(SP)+,R4	:: RESTORE R4
3141	017134	012603			MOV	(SP)+,R3	:: RESTORE R3
3142	017136	016666	000002 000004		MOV	2(SP),4(SP)	:: SET THE STACK FOR RETURNING
3143	017144	012616			MOV	(SP)+,(SP)	
3144	017146	000002			RTI		:: RETURN
3145	017150	000		8S:	.BYTE	0	:: STORAGE FOR ASCII DIGIT
3146	017151	000			.BYTE	0	:: TERMINATOR FOR TYPE ROUTINE
3147	017152	000		\$OCNT:	.BYTE	0	:: OCTAL DIGIT COUNTER
3148	017153	000		\$OFILL:	.BYTE	0	:: ZERO FILL SWITCH
3149	017154	0C0000		\$OMODE:	.WORD	0	:: NUMBER OF DIGITS TO TYPE

```

3150
3151
3152
3153
3154
3155
3156
3157
3158 017156 010046
3159 017160 016600 000002
3160 017164 005740
3161 017166 111000
3162 017170 006300
3163 017172 016000 017212
3164 017176 000200
3165
3166
3167
3168
3169 017200 011646
3170 017202 016666 000004 000002
3171 017210 000002
3172
3173
3174
3175
3176
3177
3178
3179
3180 017212 017200
3181 017214 016200
3182 017216 016754
3183 017220 016730
3184 017222 016770
3185
3186

```

.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.

```

```

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

```

;; THIS IS USE TO HANDLE THE "GETPRI" MACRO

```

$TRAP2: MOV    (SP), -(SP)       ;; MOVE THE PC DOWN
        MOV    4(SP), 2(SP)     ;; MOVE THE PSW DOWN
        RTI                    ;; RESTORE THE PSW

```

.SBTTL TRAP TABLE

```

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

```

```

; ROUTINE
;-----
$TRPAD: .WORD  $TRAP2           TRAP+1(104401)  TTY TYPEOUT ROUTINE
        .TYPE  ;;CALL=TYPE
        .TYPOC ;;CALL=TYPOC   TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
        .TYPOS ;;CALL=TYPOS   TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
        .TYPON ;;CALL=TYPON   TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)

```

.SBTTL POWER DOWN AND UP ROUTINES

```

3187
3188
3189
3190
3191 017224 012737 017376 000024
3192 017232 012737 000340 000026
3193 017240 010046
3194 017242 010146
3195 017244 010246
3196 017246 010346
3197 017250 010446
3198 017252 010546
3199 017254 017746 161664
3200 017260 010637 017402
3201 017264 012737 017276 000024
3202 017272 000000
3203 017274 000776
3204
3205
3206
3207 017276 012737 017376 000024
3208 017304 013706 017402
3209 017310 005037 017402
3210 017314 005237 017402
3211 017320 001375
3212 017322 011600
3213 017324 076600 000226
3214 017330 012677 161610
3215 017334 012605
3216 017336 012604
3217 017340 012603
3218 017342 012602
3219 017344 012601
3220 017346 012600
3221 017350 012737 017224 000024
3222 017356 012737 000340 000026
3223 017364 104401
3224 017366 017404
3225 017370 012716
3226 017372 002400
3227 017374 000002
3228 017376 000000
3229 017400 000776
3230 017402 000000
3231 017404 005015 047520 042527
3232 017412 000122
3233

```

POWER DOWN ROUTINE

```

$PWRDN: MOV $SILLUP,2#$PWRVEC ;; SET FOR FAST UP
MOV $340,2#$PWRVEC+2 ;; PRIO:7
MOV RO,-(SP) ;; PUSH RO ON STACK
MOV R1,-(SP) ;; PUSH R1 ON STACK
MOV R2,-(SP) ;; PUSH R2 ON STACK
MOV R3,-(SP) ;; PUSH R3 ON STACK
MOV R4,-(SP) ;; PUSH R4 ON STACK
MOV R5,-(SP) ;; PUSH R5 ON STACK
MOV $SMR,-(SP) ;; PUSH $SMR ON STACK
MOV SP,$SAVR6 ;; SAVE SP
MOV $PWRUP,2#$PWRVEC ;; SET UP VECTOR
HALT
BR .-2 ;; HANG UP

```

POWER UP ROUTINE

```

$PWRUP: MOV $SILLUP,2#$PWRVEC ;; SET FOR FAST DOWN
MOV $SAVR6,SP ;; GET SP
CLR $SAVR6 ;; WAIT LOOP FOR THE TTY
1$: INC $SAVR6 ;; WAIT FOR THE INC
BNE 1$ ;; OF WORD
MOV (SP),RO ;; GET SAVED SMR OFF STACK
MED 226 ;; RESTORE SMR CONTENTS
MOV (SP)+,$SMR ;; POP STACK INTO $SMR
MOV (SP)+,R5 ;; POP STACK INTO R5
MOV (SP)+,R4 ;; POP STACK INTO R4
MOV (SP)+,R3 ;; POP STACK INTO R3
MOV (SP)+,R2 ;; POP STACK INTO R2
MOV (SP)+,R1 ;; POP STACK INTO R1
MOV (SP)+,RO ;; POP STACK INTO RO
MOV $PWRDN,2#$PWRVEC ;; SET UP THE POWER DOWN VECTOR
MOV $340,2#$PWRVEC+2 ;; PRIO:7
TYPE $POWER ;; REPORT THE POWER FAILURE
$PWRMG: .WORD (PC)+,(SP) ;; POWER FAIL MESSAGE POINTER
$PWRAD: .WORD START ;; RESTART AT START
RTI ;; RESTART ADDRESS
$SILLUP: HALT ;; THE POWER UP SEQUENCE WAS STARTED
BR .-2 ;; BEFORE THE POWER DOWN WAS COMPLETE
$SAVR6: 0 ;; PUT THE SP HERE
$POWER: .ASCIZ <15><12>"POWER"
.EVEN

```


FPU INSTR EXERCISER
DQFPCB.P11 04-MAY-77MACY11 27(1006)
19:3404-MAY-77 19:35 PAGE 81
ERROR MESSAGES, DATA HEADERS, DATA VECTORS, ETC

SEQ 0082

```

3234
3235
3236
3237 017414 047510 035124 000040 ASCHOT: .ASCIZ "HOT: "
3238 017422 040527 046522 020072 ASCWRM: .ASCIZ "WARM: "
3239 017430 000
3240
3241
3242 017431 122 041505 044505 EMA: .ASCIZ "RECEIVED FPS IS BAD"
3243 017436 042526 020104 050106
3244 017444 020123 051511 041040
3245 017452 042101 000
3246 017455 122 041505 044505 EMB: .ASCIZ "RECEIVED FEC IS BAD"
3247 017462 042526 020104 042506
3248 017470 020103 051511 041040
3249 017476 042101 000
3250 017501 122 041505 044505 EMC: .ASCIZ "RECEIVED FEA IS BAD"
3251 017506 042526 020104 042506
3252 017514 020101 051511 041040
3253 017522 042101 000
3254 017525 122 041505 044505 EMD: .ASCIZ "RECEIVED PSW CC-S ARE BAD"
3255 017532 042526 020104 051520
3256 017540 020127 041503 051455
3257 017546 040440 042522 041040
3258 017554 042101 000
3259 017557 102 042101 040440 EME: .ASCIZ "BAD ADDRESS IN RO"
3260 017564 042104 042522 051523
3261 017572 044440 020116 030122
3262 017600 000
3263 017601 102 042101 051440 EMF: .ASCIZ "BAD STACK POINTER"
3264 017606 040524 045503 050040
3265 017614 044517 052116 051105
3266 017622 000
3267 017623 122 051505 046125 EMH: .ASCIZ "RESULT OF FPP ARITHMETIC OPERATION IS BAD"
3268 017630 020124 043117 043040
3269 017636 050120 040440 044522
3270 017644 044124 042515 044524
3271 017652 020103 050117 051105
3272 017660 052101 047511 020116
3273 017666 051511 041040 042101
3274 017674 000
3275 017675 122 041505 044505 EMI: .ASCIZ "RECEIVED FEC/FEA IS BAD"
3276 017702 042526 020104 042506
3277 017710 027503 042506 020101
3278 017716 051511 041040 042101
3279 017724 000
3280 017725 125 042516 050130 EMJ: .ASCIZ "UNEXPECTED FPP TRAP, IGNORED AND CONTINUING"
3281 017732 041505 042524 020104
3282 017740 050106 020120 051124
3283 017746 050101 020054 043511
3284 017754 047516 042522 020104
3285 017762 047101 020104 047503
3286 017770 052116 047111 044525
3287 017776 043516 000
3288 020001 103 052520 042040 EMP: .ASCIZ "CPU DID NOT TRAP ON FPP EXCEPTION, BUT IT SHOULD HAVE"
3289 020006 042111 047040 052117

```

3290	020014	052040	040522	020120
3291	020022	047117	043040	050120
3292	020030	042440	041530	050105
3293	020036	044524	047117	020054
3294	020044	052502	020124	052111
3295	020052	051440	047510	046125
3296	020060	020104	040510	042526
3297	020066	000		
3298	020067	103	052520	052040
3299	020074	040522	050120	042105
3300	020102	047440	020116	050106
3301	020110	020120	054105	042503
3302	020116	052120	047511	026116
3303	020124	041040	052125	044440
3304	020132	020124	044123	052517
3305	020140	042114	047040	052117
3306	020146	044040	053101	000105
3307	020154	043101	042524	020122
3308	020162	051124	050101	020054
3309	020170	052123	041501	020113
3310	020176	047520	047111	042524
3311	020204	020122	051511	044440
3312	020212	041516	051117	042522
3313	020220	052103	000	
3314	020223	101	052106	051105
3315	020230	052040	040522	026120
3316	020236	051440	047524	042522
3317	020244	020104	041520	044440
3318	020252	020123	047111	047503
3319	020260	051122	041505	000124
3320	020266	043101	042524	020122
3321	020274	051124	050101	020054
3322	020302	052123	051117	042105
3323	020310	050040	020123	051511
3324	020316	044440	041516	051117
3325	020324	042522	052103	000
3326	020331	101	052106	051105
3327	020336	052040	040522	026120
3328	020344	046040	040517	042504
3329	020352	020104	051520	044440
3330	020360	020123	047111	047503
3331	020366	051122	041505	000124
3332				
3333				
3334				
3335				
3336				
3337	020374	043040	051520	000
3338	020401	040	042506	000103
3339	020406	043040	040505	000
3340	020413	040	050106	004523
3341	020420	020040	030122	000
3342	020425	040	051040	000060
3343	020432	020040	030522	020011
3344	020440	051440	000120	
3345	020444	051044	043505	000060

EMQ: .ASCIZ "CPU TRAPPED ON FPP EXCEPTION, BUT IT SHOULD NOT HAVE"

EMR: .ASCIZ "AFTER TRAP, STACK POINTER IS INCORRECT"

EMS: .ASCIZ "AFTER TRAP, STORED PC IS INCORRECT"

EMT: .ASCIZ "AFTER TRAP, STORED PS IS INCORRECT"

EMU: .ASCIZ "AFTER TRAP, LOADED PS IS INCORRECT"

;DATA HEADERS HERE

DHA: .ASCIZ "FPS"
 DHB: .ASCIZ "FEC"
 DHC: .ASCIZ "FEA"
 DHD: .ASCIZ "FPS RO"
 DHE: .ASCIZ "RO"
 DHF: .ASCIZ "R1 SP"
 DHG: .ASCIZ "\$REGO"

```

3346 020452 051044 043505 004460 DHM: .ASCIZ "$REGO $REG1"
3347 020460 051044 043505 000061
3348 020466 051044 043505 004460 DHI: .ASCIZ "$REGO $REG1 $REG2 $REG3"
3349 020474 051044 043505 004461
3350 020502 051044 043505 004462
3351 020510 051044 043505 000063
3352 020516 054105 023520 026504 DHJ: .ASCIZ "EXP'D-FPS-RCV'D"
3353 020524 050106 026523 041522
3354 020532 023526 000104
3355 020536 054105 023520 026504 DHK: .ASCIZ "EXP'D-FEC-RCV'D EXP'D-FEA-RCV'D"
3356 020544 042506 026503 041522
3357 020552 023526 004504 054105
3358 020560 023520 026504 042506
3359 020566 026501 041522 023526
3360 020574 000104
3361 020576 054105 023520 004504 DHL: .ASCIZ "EXP'D RCV'D"
3362 020604 041522 023526 000104
3363 020612 043040 051520 020011 DHM: .ASCIZ "FPS FEC FEA OLD PS OLD PC SP AFTER"
3364 020620 042506 004503 043040
3365 020626 040505 047411 042114
3366 020634 050040 004523 046117
3367 020642 020104 041520 051411
3368 020650 020120 043101 042524
3369 020656 000122

```

```

3370
3371
3372 ;DATA VECTORS HERE
3373 .EVEN
3374 020660 001466 000000 DTB: .WORD FEC,0
3375 020664 001464 DTD: .WORD FPS,0
3376 020666 001526 000000 DTE: .WORD EREGO,0
3377 020672 001530 001542 000000 DTF: .WORD EREG1,EREG6,0
3378 020700 001170 000000 DTG: .WORD $REGO,0
3379 020704 001170 001172 000000 DTH: .WORD $REGO,$REG1,0
3380 020712 001170 001172 001174 DTI: .WORD $REGO,$REG1,$REG2,$REG3,0
3381 020720 001176 000000
3382 020724 001220 DTJ: .WORD $TMP4
3383 020726 001464 000000 DTA: .WORD FPS,0
3384 020732 001222 001466 001500 DTK: .WORD $TMP5,FEC,EXPFEA
3385 020740 001470 000000 DTC: .WORD FEA,0
3386 020744 001464 001466 001470 DTL: .WORD FPS,FEC,FEA,FPPOPS,FPPOPC,OSPRCV,0
3387 020752 001474 001472 001514
3388 020760 000000
3389 020762 001504 001514 000000 DTAG: .WORD EXPSP,OSPRCV,0
3390 020770 001502 001506 000000 DTAH: .WORD EXPRTS,OPCRCV,0
3391 020776 001214 001510 000000 DTAI: .WORD $TMP2,OPSRCV,0
3392 021004 001212 001512 000000 DTAJ: .WORD $TMP1,NPSLOD,0
3393
3394
3395 000001 ;THE END
.END

```

RAD	013134	2283#	2285		
RADDN	013422	2352#	2354		
RBASE =	000000	286			
ACDW1 =	000000	286			
ACDW2 =	000000	286			
ACPUOP =	000000	286	301		
ADDW0 =	000000	286			
ADDW1 =	000000	286			
ADDW10 =	000000	286			
ADDW11 =	000000	286			
ADDW12 =	000000	286			
ADDW13 =	000000	286			
ADDW14 =	000000	286			
ADDW15 =	000000	286			
ADDW2 =	000000	286			
ADDW3 =	000000	286			
ADDW4 =	000000	286			
ADDW5 =	000000	286			
ADDW6 =	000000	286			
ADDW7 =	000000	286			
ADDW8 =	000000	286			
ADDW9 =	000000	286			
ADEVCT =	000000	286	292		
ADEVN =	000000	286			
AD101	001604	394#	771	1295	
AD1000	001610	396#	807	809	
AD1001	001606	395#			
ADST01	001760	423#			
AENV =	000000	286	297		
AENVN =	000000	286	298		
AFATAL =	000000	286	289		
AMADR1 =	000000	286			
AMADR2 =	000000	286			
AMADR3 =	000000	286			
AMADR4 =	000000	286			
AMANS1 =	000000	286			
AMANS2 =	000000	286			
AMANS3 =	000000	286			
AMANS4 =	000000	286			
AMSGAD =	000000	286	294		
AMSGLC =	000000	286	295		
AMSGTY =	000000	286	288		
AMTYP1 =	000000	286			
AMTYP2 =	000000	286			
AMTYP3 =	000000	286			
AMTYP4 =	000000	286			
APASS =	000000	286	291		
APRIOR =	000000	286			
APTCSU =	000040	2967	3072#		
APTENV =	000001	2847	2960	3028	3070#
APTSIZ =	000200	490	3069#		
APTSPO =	000100	2962	3030	3071#	
ASCHOT	017414	603	3237#		
ASCHARM	017422	605	3238#		
ASWREG =	000000	286	299		
ATESTN =	000000	286	290		

AUNIT = 000000	286	293							
AUSMR = 000000	286	300							
AVECT1 = 000000	286								
AVECT2 = 000000	286								
ASREG0 001632	401#	850*	1202*	1215*	1471	1473			
BGNMES 001772	429#	529							
BIT0 = 000001	126#	616	2723						
BIT00 = 000001	116#	126							
BIT01 = 000002	115#	125							
BIT02 = 000004	114#	124							
BIT03 = 000010	113#	123							
BIT04 = 000020	112#	122	535	585	1944	2196	2494		
BIT05 = 000040	111#	121							
BIT06 = 000100	110#	120							
BIT07 = 000200	109#	119							
BIT08 = 000400	108#	118	2774						
BIT09 = 001000	107#	117	2782	2857					
BIT1 = 000002	125#								
BIT10 = 002000	106#	2835							
BIT11 = 004000	105#	2789							
BIT12 = 010000	104#	567	595	597	600	1948	2200	2500	
BIT13 = 020000	103#	2842							
BIT14 = 040000	102#	2578	2592	2760					
BIT15 = 100000	101#	517	2697	2703					
BIT2 = 000004	124#	2711							
BIT3 = 000010	123#								
BIT4 = 000020	122#								
BIT5 = 000040	121#								
BIT6 = 000100	120#	611	616	2698	2723				
BIT7 = 000200	119#	507							
BIT8 = 000400	118#								
BIT9 = 001000	117#								
BPTVEC = 000014	133#								
CLKDON 003036	523	526#							
CLKPRS 001516	369#	500*	507*	517*	609	614	2461	2465	
CR = 000015	41#	429	434	441	3006	3016			
CRLF = 000200	42#	2977	3016						
DALTA 001634	402#	1208	1221						
DALTB 001644	404#	1209	1524	1538					
DALTC 001654	406#	1228							
DBIG 001612	397#	885	1091	1774	1775				
DDISP = 177570	48#	248	478						
DDIV 013242	2307#	2309							
DHA 020374	328	3337#							
DHB 020401	329	3338#							
DHC 020406	330	3339#							
DHD 020413	331	3340#							
DHE 020425	332	338	3342#						
DHF 020432	333	3343#							
DHG 020444	337	3345#							
DHH 020452	336	3346#							
DHI 020466	335	3348#							
DHJ 020516	347	3352#							
DHK 020536	346	3355#							
DHL 020576	342	343	344	345	3361#				
DHM 020612	349	3363#							

SPWRMG	017366	3224#																	
SPWRUP	017276	3201	3207#																
\$QUES	001240	279#	2869	3016															
\$ROCHR=	*****	3187																	
\$RODEC=	*****	3187																	
\$ROLIN=	*****	3187																	
\$RODOCT=	*****	3187																	
\$REGAD	001166	258#																	
\$REGO	001170	260#	401	717	756	758	787	793	803	829*	836*	837	842	844					
		851	938#	940*	946	964*	976*	993*	999*	1005	1021*	1035*	1047*	1048*					
		1054	1064*	1077*	1090*	1108*	1114*	1121	1133*	1134	1142*	1143	1171*	1183*					
		1227*	1244*	1256*	1273*	1285*	1307*	1327*	1335*	1344*	1359*	1367*	1377*	1394*					
		1418*	1431*	1439*	1446	1464*	1478	1493*	1501	1511	1531*	1537*	1550*	1556*					
		1565*	1571*	1598*	1612*	1627*	1653*	1678*	1777*	1802*	2229*	2230	2244*	2245					
		2268*	2270	2280*	2288	2299*	2311*	2322	2325	2335*	2336	2355*	2369*	2372					
		2394*	2396	2400	2406	2415*	2416	2427*	2429	2438*	2439	2443*	2444	2658*					
		3378	3379	3380															
\$REG1	001172	261#	761	796	1009	1058	1138	1346	1482	1515	3379	3380							
\$REG2	001174	262#	764	789	951	1499	3380												
\$REG3	001176	263#	767	954	3380														
\$REG4	001200	264#	801	2254	2256*	2265	2286*	2404											
\$REG5	001202	265#																	
\$REG6	001204	266#																	
\$REG7	001206	267#																	
\$R2A =	*****	3187																	
\$SAVRE=	*****	3187																	
\$SAVR6	017402	3200*	3208	3209*	3210*	3230*													
\$SCOPE	015244	459	2758#																
\$SETUP=	000037	444#	458	459	461	463	465	467	468	469	471	2759	2823	2857					
		2864																	
\$STUP =	177777	444#																	
\$SVLAD	015450	2769	2798#																
\$SVPC =	000204	188#	193																
\$SWR =	167400	1#	12	276	277	278	468	469	471	472	630	669	696	749					
		784	821	872	895	936	985	1045	1100	1163	1194	1236	1265	1294					
		1354	1387	1454	1523	1583	1623	1649	1675	1700	1724	1748	1772	1798					
		1825	1844	1863	1882	1901	1920	1939	1982	2001	2020	2039	2058	2077					
		2096	2115	2134	2153	2172	2191	2224	2239	2254	2277	2320	2345	2365					
		2383	2455	2750	2751	2752	2753	2754	2760	2772	2774	2775	2778	2779					
		2780	2787	2788	2789	2801	2804	2807	2814	2815	2816	2817	2818	2835					
		2842	2854	2857	2869	3227													
		299#	492																
\$SWREG	001266	2754	2776																
\$SWRMK=	000000	290#	2799*	2930															
\$TESTN	001250	276#	468*	2455*	2479*	2526*	2787*	2794	2797*	2807									
\$TIMES	001230	251#																	
\$TKB	001154	250#																	
\$TKS	001152	268#	2560																
\$TMP0	001210	269#	3392																
\$TMP1	001212	270#	3391																
\$TMP2	001214	271#																	
\$TMP3	001216	272#	3382																
\$TMP4	001220	273#	3384																
\$TMP5	001222	274#																	
\$TMP6	001224	275#																	
\$TMP7	001226	1#																	
\$TN =	000073	1#	12	626	630*	665	669*	692	696*	745	749*	780	784*	817					

821#	868	872#	891	895#	932	936#	981	985#	1041	1045#	1096	1100#
1159	1163#	1190	1194#	1232	1236#	1261	1265#	1290	1294#	1350	1354#	1383
1387#	1450	1454#	1519	1523#	1579	1583#	1619	1623#	1645	1649#	1671	1675#
1696	1700#	1720	1724#	1744	1748#	1768	1772#	1794	1798#	1821	1825#	1833
1840	1844#	1852	1859	1863#	1871	1878	1882#	1890	1897	1901#	1909	1916
1920#	1928	1935	1939#	1961	1978	1982#	1990	1997	2001#	2009	2016	2020#
2028	2035	2039#	2047	2054	2058#	2066	2073	2077#	2085	2092	2096#	2104
2111	2115#	2123	2130	2134#	2142	2149	2153#	2161	2168	2172#	2180	2187
2191#	2213	2220	2224#	2235	2239#	2250	2254#	2273	2277#	2316	2320#	2341
2345#	2361	2365#	2379	2383#	2451	2455#						
253#	3005#	3016										
257#	2954	3016										
252#	3003	3016										
463	3158#											
3169#	3180											
3173#	3182#	3183#	3184#	3185#								
3163	3180#											
213#												
228#	2481#	2525#	2776	2798#	2799	2804	2808	2834	2869			
3185												
3185												
2954#	3047	3173	3181									
2984	2991	2998	3003#	3004								
2844	2883#											
3009	3011	3014#										
3103#	3182											
3102	3105#	3184										
3098#	3183											
293#												
215#												
300#												
2763#												
3099#	3103#	3113	3148#									
2760	2844											
174#	178#	188	189#	191#	193#	194#	200	201#	203#	205#	224#	282
445#	456	471	472	633	639	645	651	657	663	674	682	690
700	708	713	723	726	729	732	739	743	753	759	762	765
768	778	790	794	797	804	810	815	833	840	847	854	860
866	876	882	888	899	905	911	915	919	924	930	943	948
952	955	962	967	974	979	990	996	1002	1006	1010	1018	1024
1032	1038	1051	1055	1059	1067	1074	1080	1088	1093	1105	1111	1118
1122	1130	1135	1139	1144	1150	1156	1168	1174	1180	1186	1199	1205
1212	1218	1224	1230	1241	1247	1253	1259	1270	1276	1282	1288	1299
1304	1310	1318	1323	1330	1339	1347	1362	1370	1380	1391	1397	1402
1410	1415	1421	1426	1435	1442	1447	1456	1461	1468	1474	1479	1483
1490	1496	1502	1507	1512	1516	1528	1534	1540	1547	1553	1559	1568
1574	1589	1592	1595	1602	1609	1616	1636	1639	1642	1662	1665	1668
1687	1690	1693	1711	1714	1717	1735	1738	1741	1759	1762	1765	1785
1788	1791	1811	1814	1817	2232	2247	2271	2290	2302	2314	2328	2338
2359	2373	2376	2397	2408	2418	2430	2440	2445	2562	2807	2808	2869
2936#	3016	3068#	3203	3229								
3020	3023											
200#	205											

STPB 001160
 STPFLG 001165
 STPS 001156
 STRAP 017156
 STRAP2 017200
 STRP = 000005
 STRPAD 017212
 STSTM 001004
 STSTM 001102
 STYPB= ***** U
 STYPD= ***** U
 STYPE 016200
 STYPEC 016412
 STYPER 015764
 STYPEX 016460
 STYPOC 016754
 STYPOH 016770
 STYPOS 016730
 SUNIT 001256
 SUNITH 001010
 SUSWR 001270
 SXTSTR 015254
 SOFILL 017153
 S4OCAT= ***** U
 = 021012

.SASTA= ***** U
 .SX = 001000

CMPFLT	18														
COMMEN	1418														
COMM00	18														
COMM01	18														
COMM02	18														
COMM03	18														
COMM04	18														
COMM05	18														
COMM06	18														
COMM07	18														
COMM1	18														
COMM10	18														
COMM11	18														
COMM12	18														
COMM13	18														
COMM14	18														
COMM15	18														
COMM16	18														
COMM17	18														
COMM2	18														
COMM20	18														
COMM21	18														
COMM22	18														
COMM23	18														
COMM24	18														
COMM25	18														
COMM26	18														
COMM27	18														
COMM3	18														
COMM30	18														
COMM31	18														
COMM32	18														
COMM33	18														
COMM34	18														
COMM35	18														
COMM36	18														
COMM37	18														
COMM4	18														
COMM40	18														
COMM41	18														
COMM42	18														
COMM43	18														
COMM44	18														
COMM45	18														
COMM46	18														
COMM47	18														
ENDCOM	1418														
ERRCMP	18	2598	2603	2607	2612										
ERRLUR	18	2868													
ERROR	358	634	640	646	652	658	664	675	683	691	701	709	714	724	727
	730	733	740	744	754	760	763	766	769	779	791	795	798	805	811
	816	834	841	848	855	861	867	877	883	889	900	906	912	916	920
	925	926	927	931	944	949	953	956	963	968	975	980	991	997	1003
	1007	1011	1019	1025	1033	1039	1052	1056	1060	1068	1075	1081	1089	1094	1106
	1112	1119	1123	1131	1136	1140	1145	1151	1157	1169	1175	1181	1187	1200	1206
	1213	1219	1225	1231	1242	1248	1254	1260	1271	1277	1283	1289	1300	1305	1311

FPU INSTR EXERCISER MACY11 27(1006) 04-MAY-77 19:35 PAGE 100
DQFPCB.P11 04-MAY-77 19:34 CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0099

.SAPTH	18	195
.SAPTY	18	3016
.SCATC	18	172
.SCHTA	18	218
.SEOP	18	2513
.SERRO	18	2808
.SPOWE	18	3187
.SSCOP	18	2745
.STRAP	18	3150
.STYER	18	2869
.STYPE	18	2937
.STYPO	18	3073

. ABS. 021012 000

ERRORS DETECTED: 0
DEFAULT GLOBALS GENERATED: 0

DQFPCB.BIN,DQFPCB.LST/CRF/SOL/P/DOC/CPU:70/EX/EN:WRP/NL:TTM=DQFPCB.MAC,DQFPCB.P11
RUN-TIME: 19 14 1 SECONDS
RUN-TIME RATIO: 68/35=1.9
CORE USED: 24K (48 PAGES)

DOCUMENT PAGES: 99
WRAP-AROUND: 0%

USER SYMBOLS: 540
MACRO NAMES: 143
UNDF SYMBOLS: 14
DISK BLOCKS READ: 984
DISK BLKS WRITTEN: 428
KILO CORE SECONDS: 1248