

KD11-K

PDP11/6X FP11E FP INST
MD-11-DQFPC-A

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

APR 1977
digital
MADE IN USA

This microfiche card contains a grid of 100 frames of technical data, arranged in 10 rows and 10 columns. Each frame displays a different set of data, likely related to the PDP11/6X FP11E FP INST MD-11-DQFPC-A system. The data is presented in a structured format, possibly as tables or lists, with varying amounts of text and numerical values. The frames are separated by thin white lines, and the overall layout is consistent across the grid.

11-1

B01

EOFIDOFFPBR580411
DQFPCA.MEM

09-FEB-77 10:26

68010000R EXERC78883 MACY11 27(108891009)FEB-77 10188DRIDGEPCASEQ

00010000

770323

000000

.REPT 0

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DQFPC-A-D
 PRODUCT NAME: PDP-11/6X - FP11-E FLOATING POINT UNIT
 INSTRUCTION EXERCISER
 DATE: MARCH 1977
 MAINTAINER: DIAGNOSTIC GROUP
 AUTHOR: BOB BRAIN / STANLEY HARACKIEWICZ
 REVISED BY: DONALD 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.

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

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 ERRCR 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

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
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103

MAINDEC-11-DQFPC-A

PAGE 3

104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158

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 KW11-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 KW11-P PROGRAMMABLE CLOCK WILL ALSO BE UTILIZED.

2.2 STORAGE

THE PROGRAM USES MEMORY 0-20452(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.

MAINDEC-11-DQFPC-A

PAGE 4

159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214

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)
		SET=TEST ONLY UNIT SPECIFIED IN SW00
SW00	000001	SET=SELECT WARM FP, IF SW01=1

215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270

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 "\$LPTST" MATCHES THE NUMBER OF THE TEST CURRENTLY EXECUTING. AT THIS POINT, THE TEST IS LOOPED ON INDEFINATELY, UNTIL EITHER SW<8>=0 OR "\$LPTST" IS CHANGED. NOTE THAT IF "\$LPTST" DOES NOT MATCH THE TEST NUMBER OF ANY TEST, THE CONTENTS OF "\$LPTST" 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 "WHAMI" BIT<04> IS 1/0 RESPECTIVELY).

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

G71

MAINDEC-11-DQFPC-A

PAGE 6

271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316

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 WFP 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.

317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371

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 FPII-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.

372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426

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

427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482

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

AVERAGE EXECUTION TIME PER PASS

MODEL	SHORTEST PASS	LONGEST PASS
PDP-11/6X	1 SEC	1 MIN:20 SEC
PDP-11/6X W/FP11-E	1 SEC	X MIN:XX SEC

SEC = SECONDS / MIN = MINUTES

SHORTEST PASS ::= NO ITERATIONS, USING SWR=(004000)

LONGEST PASS ::= 2000(10) ITERATIONS/TEST, USING SWR=(000000)

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

483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538

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. THUS IN MANY CASES (THE "ADD" 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

539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594

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(8) 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

595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650

* IN THE ADDRESS LIGHTS.

9.3.2 SCOPE ROUTINE - \$SCOPE

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(B). (FROM THE SCOPE=IOT EQUATE). DEPENDING UPON THE SWITCH SETTINGS (SEE 5.2), CODE IS PRESENT TO: LOAD THE FPU 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:

\$MXCNT - MAXIMUM NUMBER OF ITERATIONS PER TEST (GENERALLY WILL BE 2000(10))
\$STSTM - A COUNTER INDICATING THE NUMBER (1-377(B)) OF THE TEST CURRENTLY BEING EXECUTED
\$LPAOR - CONTAINS THE ADDRESS TO WHICH THE SCOPE ROUTINE 10370 WILL LOOP, IF THE CURRENT TEST IS BEING LOOPED UPON
\$LPERR - 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 \$LPAOR, ABOVE.

9.3.3 ERROR ROUTINE - \$ERROR

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"

651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706

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 F0 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 TYPEOUT 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 TYPEOUT 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 TYPEOUT 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 (\$SPWRDN) SAVES THE GENERAL REGISTERS AND STACK POINTER. THE POWER UP

MAINDEC-11-DQFPC-A

PAGE 14

707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743

ROUTINE (SPWRUP) 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 - SEOP

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.

.ENDR

```

744
745 .TITLE FPU INSTR EXERCISER
746 *COPYRIGHT (C) 1976
747 *DIGITAL EQUIPMENT CORP.
748 *MAYNARD, MASS. 01754
749 *
750 *PROGRAM BY DONALD NORTH
751 *
752 *THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
753 *PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
754 *
755
756 .SBTTL OPERATIONAL SWITCH SETTINGS
757
758 *
759 *
760 * SWITCH OCTAL USE
761 * -----
762 * 15 100000 HALT ON ERROR
763 * 14 040000 LOOP ON CURRENTLY EXECUTING TEST
764 * 13 020000 INHIBIT ERROR TIMEOUTS
765 * 12 010000 INHIBIT STATUS TIMEOUTS
766 * 11 004000 INHIBIT ITERATIONS
767 * 10 000000 0=BEL: ON PASS END
768 * 9 002000 1=BELL ON ERROR
769 * 8 001000 LOOP ON ERROR
770 * 7 000400 LOOP ON TEST NUMBER IN "SLPTST"
771 * 6 000000 0=TEST HFP/WFP ALTERNATELY EACH PASS
772 * 5 000002 1=TEST ONLY UNIT SPECIFIED IN SW<00>
773 * 4 000002 0=SELECT HFP, IF SW<01>=1
774 * 3 000003 1=SELECT WFP, IF SW<01>=1
775
776 .SBTTL BASIC DEFINITIONS
777
778 *INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
779 STACK= 1100
780 .EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL
781 .EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL
782
783 *MISCELLANEOUS DEFINITIONS
784
785 *IT= 11 ;;CODE FOR HORIZONTAL TAB
786 *LF= 12 ;;CODE FOR LINE FEED
787 *CR= 15 ;;CODE FOR CARRIAGE RETURN
788 *CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED
789 *PS= 177776 ;;PROCESSOR STATUS WORD
790 .EQUIV PS,PSW
791 *STKLMT= 177774 ;;STACK LIMIT REGISTER
792 *PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
793 *DSWR= 177570 ;;HARDWARE SWITCH REGISTER
794 *DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
795
796 *GENERAL PURPOSE REGISTER DEFINITIONS
797
798 *R0= %0 ;;GENERAL REGISTER
799 *R1= %1 ;;GENERAL REGISTER
800 *R2= %2 ;;GENERAL REGISTER
801 *R3= %3 ;;GENERAL REGISTER
802 *R4= %4 ;;GENERAL REGISTER
803 *R5= %5 ;;GENERAL REGISTER

```

800	000006	R6=	%6	:: GENERAL REGISTER
801	000007	R7=	%7	:: GENERAL REGISTER
802	000006	SP=	%6	:: STACK POINTER
803	000007	PC=	%7	:: PROGRAM COUNTER
804				
805		.*PRIORITY LEVEL DEFINITIONS		
806	000000	PR0=	0	:: PRIORITY LEVEL 0
807	000040	PR1=	40	:: PRIORITY LEVEL 1
808	000100	PR2=	100	:: PRIORITY LEVEL 2
809	000140	PR3=	140	:: PRIORITY LEVEL 3
810	000200	PR4=	200	:: PRIORITY LEVEL 4
811	000240	PR5=	240	:: PRIORITY LEVEL 5
812	000300	PR6=	300	:: PRIORITY LEVEL 6
813	000340	PR7=	340	:: PRIORITY LEVEL 7
814				
815		.*"SWITCH REGISTER" SWITCH DEFINITIONS		
816	100000	SW15=	100000	
817	040000	SW14=	40000	
818	020000	SW13=	20000	
819	010000	SW12=	10000	
820	004000	SW11=	4000	
821	002000	SW10=	2000	
822	001000	SW09=	1000	
823	000400	SW08=	400	
824	000200	SW07=	200	
825	000100	SW06=	100	
826	000040	SW05=	40	
827	000020	SW04=	20	
828	000010	SW03=	10	
829	000004	SW02=	4	
830	000002	SW01=	2	
831	000001	SW00=	1	
832		.EQUIV	SW09, SW9	
833		.EQUIV	SW08, SW8	
834		.EQUIV	SW07, SW7	
835		.EQUIV	SW06, SW6	
836		.EQUIV	SW05, SW5	
837		.EQUIV	SW04, SW4	
838		.EQUIV	SW03, SW3	
839		.EQUIV	SW02, SW2	
840		.EQUIV	SW01, SW1	
841		.EQUIV	SW00, SW0	
842				
843		.*DATA BIT DEFINITIONS (BIT00 TO BIT15)		
844	100000	BIT15=	100000	
845	040000	BIT14=	40000	
846	020000	BIT13=	20000	
847	010000	BIT12=	10000	
848	004000	BIT11=	4000	
849	002000	BIT10=	2000	
850	001000	BIT09=	1000	
851	000400	BIT08=	400	
852	000200	BIT07=	200	
853	000100	BIT06=	100	
854	000040	BIT05=	40	
855	000020	BIT04=	20	

```

856      000010      BIT03= 10
857      000004      BIT02= 4
858      000002      BIT01= 2
859      000001      BIT00= 1
860      .EQUIV BIT09,BIT9
861      .EQUIV BIT08,BIT8
862      .EQUIV BIT07,BIT7
863      .EQUIV BIT06,BIT6
864      .EQUIV BIT05,BIT5
865      .EQUIV BIT04,BIT4
866      .EQUIV BIT03,BIT3
867      .EQUIV BIT02,BIT2
868      .EQUIV BIT01,BIT1
869      .EQUIV BIT00,BIT0
870
871      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
872      000004      ERRVEC= 4 ; TIME OUT AND OTHER ERRORS
873      000010      RESVEC= 10 ; RESERVED AND ILLEGAL INSTRUCTIONS
874      000014      TBITVEC=14 ; "T" BIT
875      000014      TRTVEC= 14 ; TRACE TRAP
876      000014      BPTVEC= 14 ; BREAKPOINT TRAP (BPT)
877      000020      IOTVEC= 20 ; INPUT/OUTPUT TRAP (IOT) **SCOPE**
878      000024      PWRVEC= 24 ; POWER FAIL
879      000030      EMTVEC= 30 ; EMULATOR TRAP (EMT) **ERROR**
880      000034      TRAPVEC=34 ; "TRAP" TRAP
881      000060      TKVEC= 60 ; TTY KEYBOARD VECTOR
882      000064      TPVEC= 64 ; TTY PRINTER VECTOR
883      000240      PIRQVEC=240 ; PROGRAM INTERRUPT REQUEST VECTOR
884
885      ;*MED INSTR EQUATES
886      076600      MED= 076600 ; OPCODE
887
888      000022      RWHAMI= 022 ; READ WHAMI
889
890      000144      RFLAG= 144 ; READ FLAGS
891      000344      WFLAG= 344 ; WRITE FLAGS
892
893      ;*LINE CLOCK VECTORS, ADDRESSES
894      177546      LKS= 177546 ; STATUS KW11-L LINE CLOCK
895      000100      LKV= 100 ; INTERRUPT VECTOR
896
897      172540      PLKS= 172540 ; STATUS KW11-P PROG LINE CLOCK
898      172542      PLKR= 172542 ; COUNT SET REGISTER
899      172544      PLKD= 172544 ; READ COUNT REGISTER
900      000104      PLKV= 104 ; INTERRUPT VECTOR
901
902      ;*FLOATING POINT INTERRUPT VECTOR
903      000244      FPPVEC= 244
904
905      ;*FLOATING POINT REGISTER DEFINITIONS
906      000000      AC0= %0
907      000001      AC1= %1
908      000002      AC2= %2
909      000003      AC3= %3
910      000004      AC4= %4
911      000005      AC5= %5

```

```

912
913
914
915      .SBTTL TRAP CATCHER
916
917      000000      .=0
918      ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
919      ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
920      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
921      000174      .=174
922 000174 000000  DISPREG: .WORD 0      ;; SOFTWARE DISPLAY REGISTER
923 000176 000000  SWREG:   .WORD 0      ;; SOFTWARE SWITCH REGISTER
924
925 000200 000137 002100 .SBTTL STARTING ADDRESS(ES)
926      JMP      @#START ;; JUMP TO STARTING ADDRESS OF PROGRAM
927
928      .SBTTL ACT11 HOOKS
929
930      ;*****
931      ;HOOKS REQUIRED BY ACT11
932      $SVPC=.      ;SAVE PC
933 000046 000046      .=46      ;; 1)SET LOC.46 TO ADDRESS OF SENDAD IN .SEOP
934      SENDAD      ;
935 000052 000052      .=52      ;; 2)SET LOC.52 TO ZERO
936      .WORD 0      ;
937      .=$SVPC      ;; RESTORE PC
938      .=1000
939      .SBTTL APT PARAMETER BLOCK
940
941      ;*****
942      ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
943      ;*****
944      .SX=.      ;SAVE CURRENT LOCATION
945 000024 000024      .=24      ;SET POWER FAIL TO POINT TO START OF PROGRAM
946      200      ;FOR APT START UP
947 000044 000044      .=44      ;POINT TO APT INDIRECT ADDRESS PNTR.
948      $APTHDR    ;POINT TO APT HEADER BLOCK
949      .=.SX      ;RESET LOCATION COUNTER
950      ;*****
951      ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-POPI1 DIAGNOSTIC
952      ;INTERFACE SPEC.
953 001000
954 001000 000000  $APTHD:
955 001002 001244  $HIBTS: .WORD 0      ;; TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
956 001004 000001  $MBAOR: .WORD $MAIL    ;; ADDRESS OF APT MAILBOX (BITS 0-15)
957 001006 000001  $STMT:  .WORD 1      ;; RUN TIM OF LONGEST TEST
958 001010 000000  $PASTM: .WORD 1      ;; RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
959 001012 000014  $UNITM: .WORD 0      ;; ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
960      .WORD $ETEND-$MAIL/2 ;; LENGTH MAILBOX-ETABLE(WORDS)

```


H02

```
1017 001230 000000 $TIMES: 0 ;:MAX. NUMBER OF ITERATIONS
1018 001232 000000 $ESCAPE:0 ;:ESCAPE ON ERROR ADDRESS
1019 001234 177607 000377 $BELL: .ASCIZ <207><377><377> ;:CODE FOR BELL
1020 001240 077 $QUES: .ASCII /?/ ;:QUESTION MARK
1021 001241 015 $CRLF: .ASCII <15> ;:CARRIAGE RETURN
1022 001242 000012 $LF: .ASCIZ <12> ;:LINE FEED
1023 ;:*****
1024 .SBTTL APT MAILBOX-ETABLE
1025 ;:*****
1026 ;:*****
1027 .EVEN
1028 001244 $MAIL: ;:APT MAILBOX
1029 001244 000000 $MSGTY: .WORD AMSGTY ;:MESSAGE TYPE CODE
1030 001246 000000 $FATAL: .WORD AFATAL ;:FATAL ERROR NUMBER
1031 001250 000000 $TESTN: .WORD ATESTN ;:TEST NUMBER
1032 001252 000000 $PASS: .WORD APASS ;:PASS COUNT
1033 001254 000000 $DEVCT: .WORD ADEVCT ;:DEVICE COUNT
1034 001256 000000 $UNIT: .WORD AUNIT ;:I/O UNIT NUMBER
1035 001260 000000 $MSGAD: .WORD AMSGAD ;:MESSAGE ADDRESS
1036 001262 000000 $MSGLG: .WORD AMSGLG ;:MESSAGE LENGTH
1037 001264 $ETABLE: ;:APT ENVIRONMENT TABLE
1038 001264 000 $ENV: .BYTE AENV ;:ENVIRONMENT BYTE
1039 001265 000 $ENVM: .BYTE AENVM ;:ENVIRONMENT MODE BITS
1040 001266 000000 $SWREG: .WORD ASWREG ;:APT SWITCH REGISTER
1041 001270 000000 $USWR: .WORD AUSWR ;:USER SWITCHES
1042 001272 000000 $CPUOP: .WORD ACPUOP ;:CPU TYPE, OPTIONS
1043 ;:BITS 15-11=CPU TYPE
1044 ;: 11/04=01, 11/05=02, 11/20=03, 11/40=04, 11/45=05
1045 ;: 11/70=06, PDQ=07, Q=10
1046 ;:BIT 10=REAL TIME CLOCK
1047 ;:BIT 9=FLOATING POINT PROCESSOR
1048 ;:BIT 8=MEMORY MANAGEMENT
1049 001274 $ETEND:
1050 .MEXIT
```

1051 .SBTTL ERROR POINTER TABLE

1052
1053 001274

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.

1069	001274	017071	020034	020366	EMV001:	.WORD	EMA,DHA,DTA	:FPS BAD
1070	001302	017115	020041	020320	EMV002:	.WORD	EMB,DHB,DTB	:FEC BAD
1071	001310	017141	020046	020400	EMV003:	.WORD	EMC,DHC,DTC	:FEA BAD
1072	001316	017165	020053	020324	EMV004:	.WORD	EMD,DHD,DTD	:PS CC-S BAD
1073	001324	017217	020065	020326	EMV005:	.WORD	EHE,DHE,DTE	:BAD ADDR IN RO
1074	001332	017241	020072	020332	EMV006:	.WORD	EMF,DHF,DTF	:BAD SP
1075	001340	017441	000000	000000	EMV007:	.WORD	EMP,0,0	:DIDNT TRAP, SHOULD HAVE
1076	001346	017263	020126	020352	EMV010:	.WORD	EMH,DHI,DTI	:RESULT BAD - D
1077	001354	017263	020112	020344	EMV011:	.WORD	EMH,DHI,DTI	:RESULT BAD - F, L
1078	001362	017263	020104	020340	EMV012:	.WORD	EMH,DHG,DTG	:RESULT BAD - I
1079	001370	017263	020065	020326	EMV013:	.WORD	EMH,DHE,DTE	:RESULT BAD - RO
1080								
1081	001376	017441	000000	000000	EMV014:	.WORD	EMP,0,0	:NO TRAP, SHOULD HAVE TRAP/FEC TSTR
1082	001404	017527	000000	000000	EMV015:	.WORD	EMQ,0,0	:TRAPPED, SHOULDN'T HAVE
1083	001412	017614	020236	020422	EMV016:	.WORD	EMR,DHL,DTAG	:SP OK
1084	001420	017663	020236	020430	EMV017:	.WORD	EMS,DHL,DTAH	:STORED PC
1085	001426	017726	020236	020436	EMV020:	.WORD	EMT,DHL,DTAI	:STORED PS
1086	001434	017771	020236	020444	EMV021:	.WORD	EMU,DHL,DTAJ	:LOADED PS
1087	001442	017335	020176	020372	EMV022:	.WORD	EMI,DHK,DTK	:FEC/FEA BAD
1088	001450	017071	020156	020364	EMV023:	.WORD	EMA,DHJ,DTJ	:FPS BAD
1089								
1090	001456	017365	020252	020404	EMV024:	.WORD	EMJ,DHM,DTL	:UNEXPECTED TRAP

```

1091
1092
1093
1094      .SBTTL PROGRAM DEFINED COMMON TAGS
1095      :#VARIABLES
1096 001464 000000 FPS: .WORD 0 ;FPS STORED HERE AFTER STFPS
1097 001466 000000 FEC: .WORD 0 ;FEC STORED HERE AFTER STST
1098 001470 000000 FEA: .WORD 0 ;FEA STORED HERE AFTER STST
1099 001472 000000 FPPOPC: .WORD 0 ;OLD PC SAVED HERE AFTER TRAP
1100 001474 000000 FPPOPS: .WORD 0 ;OLD PS SAVED HERE AFTER TRAP
1101 001476 000000 FPPOSP: .WORD 0 ;SP AFTER TRAP
1102 001500 000000 EXPFEA: .WORD 0 ;EXPECTED FEA
1103 001502 000000 EXPRTS: .WORD 0 ;EXPECTED RETURN ADDR
1104 001504 000000 EXPSP: .WORD 0 ;EXPECTED SP VALUE AFTER TRAP
1105 001506 000000 OPCRCV: .WORD 0 ;PC PUSHED ON STACK AFTER TRAP
1106 001510 000000 OPSRCV: .WORD 0 ;PS PUSHED ON STACK AFTER TRAP
1107 001512 000000 NPSLOD: .WORD 0 ;PS THAT WAS LOADED AFTER TRAP
1108 001514 000000 OSPRCV: .WORD 0 ;SP AFTER FPP TRAP
1109
1110 001516 000000 CLKPRS: .WORD 000000 ;BIT07=1 -> KW11-L PRESENT
1111 ;BIT15=1 -> KW11-P PRESENT
1112 001520 000000 KW11LC: .WORD 0 ;KW11-L COUNT OF # OF INTERRUPTS
1113 001522 000000 KW11PC: .WORD 0 ;KW11-P COUNT OF # OF INTERRUPTS
1114 001524 000000 KW11PR: .WORD 0 ;KW11-P COUNT SET
1115
1116      :#REGISTER CONTENTS, AT ERROR, STORED HERE
1117 001526 000000 EREG0: .WORD 0
1118 001530 000000 EREG1: .WORD 0
1119 001532 000000 EREG2: .WORD 0
1120 001534 000000 EREG3: .WORD 0
1121 001536 000000 EREG4: .WORD 0
1122 001540 000000 EREG5: .WORD 0
1123 001542 000000 EREG6: .WORD 0
1124 001544 000000 EREG7: .WORD 0
1125
1126      :#CONSTANTS
1127 001546 040200 000000 000000 FLTONE: .WORD 040200,0,0,0
1128 001554 000000
1129 001556 177777 D1010: .WORD -1
1130 001560 000000 177777 000000 D0101: .WORD 0,-1,0
1131 001566 177777 000000 000000 D1001: .WORD -1,0,0
1132 001574 DSMALL:
1133 001574 177777 000001 000000 WEIRD: .WORD -1,1,0,-1
1134 001602 177777
1135 001604 001560 AD0101: .WORD D0101
1136 001606 001566 AD1001: .WORD D1001
1137 001610 001726 AD1000: .WORD D1000
1138 001612 077777 000000 177777 DBIG: .WORD 77777,0,-1,0
1139 001620 000000
1140 001622 100000 000000 000000 DMZERO: .WORD 100000,0,0,0
1141 001630 000000
1142 001632 001170 ASREG0: .WORD $REG0
1143 001634 040252 125252 125252 DALTA: .WORD 40252,125252,125252,125252
1144 001642 125252
1145 001644 040325 052525 052525 DALTB: .WORD 40325,52525,52525,52525
1146 001652 052525
  
```

K02

FPU INSTR EXERCISER MACY11 27(1006) 09-FEB-77 10:28 PAGE 25
DQFPCA.P11 09-FEB-77 10:26 PROGRAM DEFINED COMMON TAGS

1147	001654	040325	052525	052525	DALTC:	.WORD	40325,52525,52525,52526
1148	001662	052526					
1149	001664	040000	000000	000000	D40:	.WORD	40000,0,0,0
1150	001672	000000					
1151	001674	037400	000000	000000	D37:	.WORD	37400,0,0,0
1152	001702	000000					
1153	001704	040600	000000	000000	D46:	.WORD	40600,0,0,0
1154	001712	000000					
1155	001714	020000	000000	000000	D20:	.WORD	20000,0,0,0
1156	001722	000000					
1157	001724	000000			D0100:	.WORD	0
1158	001726	177777	000000	000000	D1000:	.WORD	-1,0,0,0
1159	001734	000000					
1160	001736	000100	000000	000000	D0100X:	.WORD	100,0,0
1161	001744	000000	177777	177777	D0111:	.WORD	0,-1,-1,-1
1162	001752	177777					
1163	001754	000000	054321		DST01:	.WORD	0,54321
1164	001760	001754			ADST01:	.WORD	DST01
1165	001762	043661	121000	000000	FST01:	.WORD	43661,121000,0,0
1166	001770	000000					
1167							
1168							
1169							

1170	001772	005015	005012	042115	: #MESSAGES FOR BEGIN PROGRAM/START OF PASS		
1171	002000	030455	026461	050504	BGNMES:	.ASCII	<15><12><12><12>"MD-11-DQFPC-A..."
1172	002006	050106	026503	027101			
1173	002014	027056					
1174	002016	042120	026520	030461	.ASCIZ	"PDP-11/6X F.P.U. INSTRUCTION EXERCISER"	<15><12>
1175	002024	033057	020130	027106			
1176	002032	027120	027125	044440			
1177	002040	051516	051124	041525			
1178	002046	044524	047117	042440			
1179	002054	042530	041522	051511			
1180	002062	051105	005015	000			
1181	002067	015	050012	051501	NWPAS1:	.ASCIZ	<15><12>"PASS #"
1182	002074	020123	000043				

```

1183 .SBTTL START OF PASS ROUTINE
1184
1185 .EVEN ;START ON AN EVEN BOUNDARY
1186
1187 ;*****
1188 .ENABL AMA ;ASSEMBLE ALL RELATIVE REFERENCES AS ABSOLUTE
1189 ;*****
1190
1191 002100 START:
1192 .SBTTL INITIALIZE THE COMMON TAGS
1193 ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
1194 002100 012706 001100 MOV #CMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
1195 002104 005026 CLR (R6)+ ;;CLEAR MEMORY LOCATION
1196 002106 022706 001146 CMP #SWR,R6 ;;DONE?
1197 002112 001374 BNE -6 ;;LOOP BACK IF NO
1198 002114 012706 001100 MOV #STACK,SP ;;SETUP THE STACK POINTER
1199 ;;INITIALIZE A FEW VECTORS
1200 002120 012737 014704 000020 MOV #SCOPE,@IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
1201 002126 012737 000340 000022 MOV #340,@IOTVEC+2 ;;LEVEL 7
1202 002134 012737 015162 000030 MOV #SERIOR,@EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
1203 002142 012737 000340 000032 MOV #340,@EMTVEC+2 ;;LEVEL 7
1204 002150 012737 016616 000034 MOV #STRAP,@TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
1205 002156 012737 000340 000036 MOV #340,@TRAPVEC+2 ;;LEVEL 7
1206 002164 012737 016664 000024 MOV #SPWRON,@PWRVEC ;;POWER FAILURE VECTOR
1207 002172 012737 000340 000026 MOV #340,@PWRVEC+2 ;;LEVEL 7
1208 002200 013737 013700 013672 MOV SENDCT,SEOPCT ;;SETUP END-OF-PROGRAM COUNTER
1209 002206 005037 001230 CLR STIMES ;;INITIALIZE NUMBER OF ITERATIONS
1210 002212 005037 001232 CLR SESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
1211 002216 012737 000001 001122 MOV #1,SERMAX ;;ALLOW ONE ERROR PER TEST
1212 002224 012737 002224 001110 MOV #,$LPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
1213 002232 012737 002232 001114 MOV #,$LPERR ;;SETUP THE ERROR LOOP ADDRESS
1214 ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
1215 ;;EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
1216 002240 013746 000004 MOV @ERRVEC-(SP) ;;SAVE ERROR VECTOR
1217 002244 012737 002300 000004 MOV #64$,@ERRVEC ;;SET UP ERROR VECTOR
1218 002252 012737 177570 001146 MOV #DSWR,SWR ;;SETUP FOR A HARDWARE SWITCH REGISTER
1219 002260 012737 177570 001150 MOV #DISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
1220 002266 022777 177777 176652 CMP #-1,@SWR ;;TRY TO REFERENCE HARDWARE SWR
1221 002274 001012 BNE 66$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
1222 ;;AND THE HARDWARE SWR IS NOT = -1
1223 002276 000403 BR 65$ ;;BRANCH IF NO TIMEOUT
1224 002300 012716 002306 64$: MOV #65$, (SP) ;;SET UP FOR TRAP RETURN
1225 002304 000002 RTI
1226 002306 012737 000176 001146 65$: MOV #SWREG,SWR ;;POINT TO SOFTWARE SWR
1227 002314 012737 000174 001150 MOV #DISPREG,DISPLAY
1228 002322 012637 000004 66$: MOV (SP)+,@ERRVEC ;;RESTORE ERROR VECTOR
1229
1230 002326 005037 001252 CLR $PASS ;;CLEAR PASS COUNT
1231 002332 132737 000200 001265 BITB #APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
1232 002340 001403 BEQ 67$ ;;YES,USE NON-APT SWITCH
1233 002342 012737 001266 001146 67$: MOV #SSWREG,SWR ;;NO,USE APT SWITCH REGISTER
1234 002350
1235
1236 ;SET UP FPP UNEXPECTED TRAP CATCHER - - - - -
1237 002350 012737 014620 000244 MOV #FPPILT,@FPPVEC ;;NEW PC AT FPP TRAP
1238 002356 012737 000300 000246 MOV #PR6,@FPPVEC+2 ;;NEW PS AT FPP TRAP
  
```

```

1239
1240
1241 002364 005037 001516      .FIND, INITIALIZE ANY LINE CLOCKS PRESENT - - - - -
1242 002370 013746 000004      CLR      CLKPRS      ;CLEAR CLOCK PRESENT FLAG
1243 002374 013746 000006      MOV      @#ERRVEC, -(SP) ;SAVE OLD TIMEOUT VECTOR
1244 002400 012737 002450 000004      MOV      @#ERRVEC+2, -(SP) ;
1245 002406 012737 000340 000006      MOV      @NKW11L, @#ERRVEC ;SET NEW TIMEOUT VECTOR
1246
1247 002414 005037 177546      CLR      @#LKS      ;KW11-L THERE ? (CLEAR STATUS IF YES)
1248 002420 052737 000200 001516      BIS      @BIT7, CLKPRS ;YES, IT WAS
1249 002426 005037 001520      CLR      KW11LC      ;CLEAR COUNT
1250 002432 012737 014462 000100      MOV      @IKW11L, @#LKV ;KW11-L SERVICE PC, PS
1251 002440 012737 000300 000102      MOV      @PR6, @#LKV+2 ;
1252 002446 000402      BR      TKW11P      ;
1253 002450 062706 000004      NKW11L: ADD      @4, SP ;NO KW11-L (TIMEOUT); POP JUNK OFF STK
1254
1255 002454 012737 002532 000004      TKW11P: MOV      @NKW11P, @#ERRVEC ;SET NEW TIMEOUT VECTOR (SAME PS)
1256
1257 002462 005037 172540      CLR      @#PLKS      ;KW11-P THERE ? (CLEAR STATUS IF YES)
1258 002466 052737 100000 001516      BIS      @BIT15, CLKPRS ;YES, IT WAS
1259 002474 005037 001522      CLR      KW11PC      ;CLEAR COUNT
1260 002500 012737 001777 001524      MOV      @1777, KW11PR ;INITIAL COUNT REGISTER
1261 002506 012737 014510 000104      MOV      @IKW11P, @#PLKV ;KW11-P SERVICE PC, PS
1262 002514 012737 000300 000106      MOV      @PR6, @#PLKV+2 ;
1263 002522 013737 001524 172542      MOV      KW11PR, @#PLKR ;SET COUNT REGISTER
1264 002530 000402      BR      CLKDON      ;
1265 002532 062706 000004      NKW11P: ADD      @4, SP ;NO KW11-P (TIMEOUT); POP JUNK OFF STK
1266
1267 002536 012637 000006      CLKDON: MOV      (SP)+, @#ERRVEC+2 ;RESTORE OLD TIMEOUT VECTOR
1268 002542 012637 000004      MO.      (SP)+, @#ERRVEC ;
1269
1270 002546 104401 001772      TYPE      ,BGNMES      ;ID MESSAGE AT START
1271
1272 ;//////////////////////////////////////
1273 ; MESSAGE ON WHETHER OR NOT HFP UNIT IS PRESENT
1274 ;
1275 002552 076600 000022      MED      ,RWHAMI      ;WHAMI INTO RO
1276 002556 032700 000020      BIT      @BIT04, RO ;IS THERE A HFP UNIT ?
1277 002562 001403      BEQ      66$      ;NO, BR
1278 002564 104401 002600      TYPE      ,64$      ;INDICATE FP11-E PRESENT
1279 002570 000453      BR      NEWPAS      ;GO FOR SUBPASS INIT
1280 002572 104401 002640      66$: TYPE      ,65$      ;INDICATE NO FP11-E
1281 002576 000450      BR      NEWPAS      ;GO FOR SUBPASS INIT
1282
1283 002600 005015 020052 050106      64$: .ASCIZ <15><12>*" FP11-E HFP UNIT PRESENT *"<15><12>
1284 002606 030461 042455 044040
1285 002614 050106 052440 044516
1286 002622 020124 051120 051505
1287 002630 047105 020124 006452
1288 002636 000012
1289 002640 005015 020052 047516      65$: .ASCIZ <15><12>*" NO FP11-E HFP UNIT - ALL TESTS WFP ONLY *"<15><12>
1290 002646 043040 030520 026461
1291 002654 020105 043110 020120
1292 002662 047125 052111 026440
1293 002670 040440 046114 052040
1294 002676 051505 051524 053440

```

```

1295 002704 050106 047440 046116
1296 002712 020131 006452 000012
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306 002720 012706 001100      NEWPAS: MOV      #STACK,SP          ;RESET STACK PTR
1307
1308 002724 032777 010000 176214      BIT      #BIT12,@SWR          ;INHIBIT STATUS TYPEOUTS ?
1309 002732 001011                BNE      SUBPAS              ;BR IF YES
1310
1311 002734 104401 002067                TYPE     NEWPAS1             ;"PASS #"
1312 002740 013746 001252                MOV      $PASS,-(SP)         ;PASS COUNT INTO ...
1313 002744 005216                INC      (SP)                ; 1-N RANGE
1314 002746 104403                TYP0S   TYPE                ;TYPE OCTAL
1315 002750      006      000                .BYTE   6,0                 ; 6 DIGITS, NO LEADING ZEROS
1316 002752 104401 001241                TYPE     , $CRLF            ;END THE LINE
1317
1318
1319
1320
1321
1322
1323 002756 076600 000022      SUBPAS: MED     ,RWHAMI        ;GET WHAMI INTO RO
1324 002762 032700 000020                BIT     #BIT04,RO           ;1=HFP PRESENT, 0=NO
1325 002766 001430                BEQ     20$                 ;IF NO HFP, TEST WARM ONLY
1326
1327 002770 076600 000144                MED     ,RFLAG             ;GET FLAGS INTO RO
1328
1329 002774 032777 000002 176144      BIT     #SW01,@SWR          ;SW01: 1=HFP OR WFP TEST ONLY
1330 003002 001413                BEQ     1$                 ; 0=ALTERNATE HFP/WFP PER PASS
1331
1332 003004 032777 000001 176134      BIT     #SW00,@SWR          ;SW00: 1=WFP ONLY
1333 003012 001403                BEQ     2$                 ; 0=HFP ONLY
1334 003014 042700 010000                BIC     #BIT12,RO           ;CLEAR HFP ENABLE FLAG<5> FOR WFP
1335 003020 000402                BR      3$                 ;
1336 003022 052700 010000      2$:   BIS     #BIT12,RO       ;SET HFP ENABLE FLAG<5> FOR HFP
1337 003026 076600 000344      3$:   MED     ,WFLAG          ;REWRITE FLAGS
1338
1339 003032 032700 010000      1$:   BIT     #BIT12,RO       ;TEST WHO'S ENABLED: HOT, WARM
1340 003036 001404                BEQ     20$                ;SET APPROPRIATE HEADER:
1341
1342 003040 012737 017054 015426      19$:  MOV     #ASCHOT,HOTWARM    ;"HOT: "
1343 003046 000403                BR      21$                ;
1344 003050 012737 017062 015426      20$:  MOV     #ASCWARM,HOTWARM   ;"WARM: "
1345 003056 005037 001102      21$:  CLR     $STNM            ;ALL DONE, RESET TEST NUMBER COUNTER
1346
1347
1348 003062 105737 001516                .*START KW11-L LINE CLOCK IF PRESENT
1349 003066 100003                TSTB   CLKPRS              ;KW11-L PRESENT ?
1350 003070 052737 000100 177546      BPL     NKWL1              ;NO
1350 003070 052737 000100 177546      BIS     #BIT6,@LKS         ;YES, START IT

```

B03

FPU INSTR EXERCISER MACY11 27(1006) 09-FEB-77 10:28 PAGE 29
DOFPCA.P11 09-FEB-77 10:26 INITIALIZE THE COMMON TAGS

```
1351
1352
1353 003076 005737 001516 NKWL1: ;*START KW11-P PROGRAMMABLE LINE CLOCK, IF PRESENT
1354 003102 100003          BPL      CLKPRS          ;KW11-P PRESENT ?
1355 003104 052737 000101 172540  BPL      NKWP1          ;NO
1356          BIS      #BIT6+BIT0,2#PLKS      ;YES, START IT
1357 003112 005037 177776  NKWP1: CLR      2#PSW          ;SETUP FOR INTERRUPT ENABLE, PRO
1358
1359
```

```

1360
1361 ;*****
1362 ;.DSABL AMA ;ASSEMBLE ALL REFERENCES AS THEY ARE PRINTED
1363 ;*****
1364
1365 ;*****
1366 ;#TEST 1 TEST OF WRITABILITY OF FPS
1367 ;*****
1368 003116 000004
1369 003120 170127 147757
1370 003124 170267 176334
1371 003130 022767 147757 176326
1372 003136 001401
1373 003140 104001
1374
1375 003142 170127 040000 LDFPS #147757 ;TEST FPS WITH ALL ONES
1376 003146 170267 176312 STFPS FPS ;GET STATUS
1377 003152 022767 040000 176304 CMP #147757,FPS ;CHECK IT
1378 003160 001401 BEQ .+4
1379 003162 104001 ERROR 1 ;FPS NOT 147777
1380
1381 003164 170011 SETD ;TEST OF DOUBLE BIT ON A 1
1382 003166 170267 176272 STFPS FPS ;STORE FLOATING POINT STATUS
1383 003172 022767 040200 176264 CMP #40200,FPS ;CHECK FLOATING POINT STATUS
1384 003200 001401 BEQ .+4 ;BRANCH IF OK
1385 003202 104001 ERROR 1 ;FPS NOT EQUAL TO 40200
1386
1387 003204 170001 SETF ;TEST OF DOUBLE BIT ON A 0
1388 003206 170267 176252 STFPS FPS ;STORE FLOATING POINT STATUS
1389 003212 022767 040000 176244 CMP #40000,FPS ;CHECK FLOATING POINT STATUS
1390 003220 001401 BEQ .+4 ;BRANCH IF OK
1391 003222 104001 ERROR 1 ;FPS NOT EQUAL TO 40000
1392
1393 003224 170012 SETL ;TEST OF LONG BIT ON A 1
1394 003226 170267 176232 STFPS FPS ;STORE FLOATING POINT STATUS
1395 003232 022767 040100 176224 CMP #40100,FPS ;CHECK FLOATING POINT STATUS
1396 003240 001401 BEQ .+4 ;BRANCH IF OK
1397 003242 104001 ERROR 1 ;FPS NOT EQUAL TO 40100
1398
1399 003244 170002 SETI ;TEST OF LONG BIT ON A 0
1400 003246 170267 176212 STFPS FPS ;STORE FLOATING POINT STATUS
1401 003252 022767 040000 176204 CMP #40000,FPS ;CHECK FLOATING POINT STATUS
1402 003260 001401 BEQ .+4 ;BRANCH IF OK
1403 003262 104001 ERROR 1 ;FPS NOT EQUAL TO 40000

```

```

1404
1405
1406
1407 003264 000004
1408 003266 170127 040017
1409 003272 170000
1410 003274 013700 177776
1411 003300 042700 177760
1412 003304 022700 000017
1413 003310 001401
1414 003312 104004
1415
1416 003314 170127 040012
1417 003320 170000
1418 003322 013700 177776
1419 003326 042700 177760
1420 003332 022700 000012
1421 003336 001401
1422 003340 104004
1423
1424 003342 170127 040005
1425 003346 170000
1426 003350 013700 177776
1427 003354 042700 177760
1428 003360 022700 000005
1429 003364 001401
1430 003366 104004

```

```

*****
;TEST 2 TEST OF CFCC
*****
†ST2: 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

```

E03

FPU INSTR EXERCISER MACY11 27(1006) 09-FEB-77 10:28 PAGE 32
 DQFPCA.P11 09-FEB-77 10:26 T3 TEST OF LDD, STD, CMPD OF -1,0,-1,0

```

1431
1432
1433
1434 003370 000004
1435 003372 170127 047600
1436 003376 012700 001556
1437 003402 172420
1438 003404 022700 001566
1439 003410 001401
1440 003412 104005
1441
1442 003414 010601
1443 003416 162701 000010
1444 003422 174046
1445 003424 010600
1446 003426 020106
1447 003430 001401
1448 003432 104006
1449
1450 003434 170267 176024
1451 003440 022767 047610 176016
1452 003446 001401
1453 003450 104001
1454
1455 003452 010602
1456 003454 012703 001170
1457 003460 012223
1458 003462 012223
1459 003464 012223
1460 003466 012223
1461 003470 021667 176062
1462 003474 001401
1463 003476 104010
1464 003500 026667 000002 176052
1465 003506 001401
1466 003510 104010
1467 003512 026667 000004 176042
1468 003520 001401
1469 003522 104010
1470 003524 026667 000006 176032
1471 003532 001401
1472 003534 104010
1473
1474 003536 062701 000010
1475 003542 173426
1476 003544 010600
1477 003546 020106
1478 003550 001401
1479 003552 104006
1480 003554 170267 175704
1481 003560 170000
1482 003562 001401
1483 003564 104010

```

```

*****
:TEST 3            TEST OF LDD, STD, CMPD OF -1,0,-1,0
*****
↑ST3:    SCOPE
         LDFPS        #47600
         MOV         #D1010,RO        ;GET ADDRESS OF DATA WORD
         LDD         (RO)+,ACO        ;LOAD INTO ACO
         CMP         #D1010+10,RO     ;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,RO            ;SAVE THE SP FOR TYPING
         CMP         R1,SP            ;SP DECREMENTED PROPERLY?
         BEQ         .+4
         ERROR       6                ;SP NOT SP-10

         STFPS       FPS
         CMP         #47610,FPS       ;STORE FLOATING POINT STATUS
         BEQ         .+4              ;CHECK FLOATING POINT STATUS
         ERROR       1                ;BRANCH IF OK
                     ;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)+,ACO        ;RECHECK DATA AND SP
         MOV         SP,RO            ;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

```

F03

FPU INSTR EXERCISER MACY11 27(1006) 09-FEB-77 10:28 PAGE 33
 DQFPCA.P11 09-FEB-77 10:26 T4 TEST OF LOD, STD, CMPD OF 0,-1,0,-1

```

1484
1485
1486
1487 003566 000004
1488 003570 170127 047600
1489 003574 172437 001560
1490 003600 170267 175660
1491 003604 022767 047604 175652
1492 003612 001401
1493 003614 104001
1494
1495 003616 012700 001170
1496 003622 174010
1497 003624 026767 175730 175336
1498 003632 001401
1499 003634 104010
1500 003636 026767 175720 175326
1501 003644 001401
1502 003646 104010
1503 003650 026767 175710 175316
1504 003656 001401
1505 003660 104010
1506 003662 026767 175700 175306
1507 003670 001401
1508 003672 104010
1509
1510 003674 012704 001602
1511 003700 173474 000002
1512
1513
1514
1515 003704 170267 175554
1516 003710 170000
1517 003712 001401
1518 003714 104010

;*****
;TEST 4    TEST OF LOD, STD, CMPD OF 0,-1,0,-1
;*****
†ST4:    SCOPE
LDFPS    #47600
LOD      @#00101,AC0    ;LOAD 0,-1,0,-1 INTO AC0 *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      AC0,(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      #AD0101-2,R4   ;ADDRESS-2 OF DATA
CMPD     @2(4),AC0      ;CHECK DATA IN AC0 *PIC*
;NOTE:    CMPD SEES A ZERO EXP IN AC0, AND THUS
;         AFTER ITS EXECUTION AC0 ← (0,0,0,0),
;         A TRUE FLTPT ZERO
STFPS    FPS           ;GET STATUS
CFCC                   ;GET CONDITION CODES
BEQ      .+4
ERROR    10            ;CMPD FAILED
  
```

G03

FPU INSTR EXERCISER MACY11 27(1006) 09-FEB-77 10:28 PAGE 34
 DQFPCA.P11 09-FEB-77 10:26 TS TEST OF LDF, STF, CMPF OF -1,0

```

1519
1520
1521
1522 003716 000004
1523
1524 003720 170127 047400
1525 003724 172467 175626
1526 003730 012700 001170
1527 003734 174020
1528 003736 022700 001174
1529 003742 001401
1530 003744 104005
1531
1532 003746 026767 175604 175214
1533 003754 001401
1534 003756 104011
1535 003760 026767 175574 175204
1536 003766 001401
1537 003770 104011
1538
1539 003772 170011
1540 003774 012700 001200
1541 004000 174040
1542 004002 022700 001170
1543 004006 001401
1544 004010 104005
1545
1546 004012 012700 001610
1547 004016 173430
1548 004020 022700 001612
1549 004024 001401
1550 004026 104005
1551
1552 004030 170267 175430
1553 004034 170000
1554 004036 001401
1555 004040 104011

;*****
;TEST 5 TEST OF LDF, STF, CMPF OF -1,0
;*****
↑ST5: SCOPE
LDFPS #47400
LDF D1010,AC0
MOV #SREG0,RO
STF AC0,(RO)+
CMP #SREG2,RO
BEQ .+4
ERROR 5
;NOTE: AC0 = (0,0,0,0) FROM PREV TEST
;SET FLOATING MODE
;LOAD -1,0 INTO AC0
;POINTED TO ANSWER AREA *PIC*
;STORE RESULT
;INCREMENTED PROPERLY
;RO NOT SREG0+4

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

SETD
MOV #SREG4,RO
STD AC0,-(RO)
CMP #SREG0,RO
BEQ .+4
ERROR 5
;GO TO DOUBLE MODE
;ADDRESS OF DATA+10
;GET DATA
;CHECK FOR PROPER DECREMENTATION
;RO NOT SREG0

MOV #AD1000,RO
CMPD 2(RO)+,AC0
CMP #AD1000+2,RO
BEQ .+4
ERROR 5
;LOAD ADDRESS OF ADDRESS OF DATA
;CHECK THE DATA
;RO GETS INCREMENTED BY 2
;RO NOT AD1001+2

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

H03

FPU INSTR EXERCISER MACY11 27(1006) 09-FEB-77 10:28 PAGE 35
DQFPCA.P11 09-FEB-77 10:26 T6 TEST OF LDF, STF, CMPF WITH (<=>) IN ALL AC'S

```
1556
1557
1558
1559 004042 000004
1560 004044 170127 047400
1561 004050 172427 140640
1562 004054 172527 140200
1563 004060 172627 000000
1564 004064 172727 040640
1565 004070 174004
1566 004072 174305
1567
1568 004074 174067 175070
1569 004100 173427 140640
1570 004104 170267 175354
1571 004110 170000
1572 004112 001401
1573 004114 104011
1574
1575 004116 174137 001170
1576 004122 173567 175042
1577 004126 170267 175332
1578 004132 170000
1579 004134 001401
1580 004136 104011
1581 004140 012704 001170
1582 004144 174214
1583 004146 173667 175016
1584 004152 170267 175306
1585 004156 170000
1586 004160 001401
1587 004162 104011
1588
1589 004164 174377 175442
1590 004170 173767 174774
1591 004174 170267 175264
1592 004200 170000
1593 004202 001401
1594 004204 104011
1595
1596 004206 173404
1597 004210 170267 175250
1598 004214 170000
1599 004216 001401
1600 004220 104000
1601
1602 004222 173705
1603 004224 170267 175234
1604 004230 170000
1605 004232 001401
1606 004234 104000

;*****
;TEST 6 TEST OF LDF, STF, CMPF WITH (<=>) IN ALL AC'S
;*****
†ST6: SCOPE
LDFPS #47400 ;LOAD FLOATING MODE
LDF #-5,AC0 ;LOAD AC0 WITH -5
LDF #-1,AC1 ;LOAD AC1 WITH -1
LDF #0,AC2 ;LOAD AC2 WITH 0
LDF #5,AC3 ;LOAD AC3 WITH 5
STF AC0,AC4 ;LOAD AC4 WITH -5
STF AC3,AC5 ;LOAD AC5 WITH 5

STF AC0,$REGO ;GET AC0
CMPF #-5,AC0 ;CHECK IT
STFPS FPS ;GET STATUS
CFCC ;GET CC
BEQ .+4
ERROR 11 ;AC0 NOT -5

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

STF AC3,3,$REGO ;PUT DATA INTO $REGO
CMPF $REGO,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
```

```

1607
1608
1609
1610 004236 000004
1611 004240 170127 041000
1612 004244 173405
1613 004246 170267 175212
1614 004252 022767 041000 175204
1615 004260 001401
1616 004262 104001
1617
1618 004264 173704
1619 004266 170267 175172
1620 004272 022767 041010 175164
1621 004300 001401
1622 004302 104001
1623
1624 004304 173767 175302
1625 004310 170267 175150
1626 004314 022767 041000 175142
1627 004322 001401
1628 004324 104001
1629

```

```

*****
;TEST 7 TEST OF CMPF WITH DATA IN ACO-ACS
*****
↑ST7: SCOPE
LDFPS #41000 ;LOAD STATUS WITH 0
CMPF ACS,AC0 ;CMP 5 TO -5
STFPS FPS ;STORE FLOATING POINT STATUS
CMP #41000,FPS ;CHECK FLOATING POINT STATUS
BEQ .+4 ;BRANCH IF OK
ERROR 1 ;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 1 ;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 1 ;FPS NOT 141002

```

J03

FPU INSTR EXERCISER MACY11 27(1006) 09-FEB-77 10:28 PAGE 37
 DDFPCA.P11 09-FEB-77 10:26 T10 TEST OF TSTF AND TSTD USING OLD ACO-ACS

```

1630
1631
1632
1633 004326 000004
1634 004330 170127 040000
1635 004334 170501
1636 004336 170267 175122
1637 004342 022767 040010 175114
1638 004350 001401
1639 004352 104001
1640
1641 004354 170505
1642 004356 170267 175102
1643 004362 022767 040000 175074
1644 004370 001401
1645 004372 104001
1646
1647 004374 170502
1648 004376 170267 175062
1649 004402 022767 040004 175054
1650 004410 001401
1651 004412 104001
1652
1653 004414 170527 177777
1654 004420 000401
1655 004422 104001
1656 004424 170267 175034
1657 004430 022767 040010 175026
1658 004436 001401
1659 004440 104001
1660
1661 004442 170011
1662 004444 170527 000000
1663 004450 000403
1664 004452 104001
1665 004454 104001
1666 004456 104001
1667 004460 170267 175000
1668 004464 022767 040204 174772
1669 004472 001401
1670 004474 104001

:*****
:TEST 10    TEST OF TSTF AND TSTD USING OLD ACO-ACS
:*****
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    #0            ;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
  
```

K03

FPU INSTR EXERCISER MACY11 27(1006) 09-FEB-77 10:28 PAGE 38
 DDFPCA.P11 09-FEB-77 10:26 T11 TEST OF CLRX INSTRUCTIONS

```

1671
1672
1673
1674 004476 000004
1675 004500 170127 040200
1676 004504 172467 175046
1677 004510 174067 174454
1678 004514 170001
1679 004516 170467 174446
1680 004522 170267 174736
1681 004526 022767 040004 174730
1682 004534 001401
1683 004536 104001
1684
1685 004540 170567 174424
1686 004544 170000
1687 004546 001401
1688 004550 104011
1689
1690 004552 026767 175004 174414
1691 004560 001401
1692 004562 104010
1693 004564 026767 174774 174404
1694 004572 001401
1695 004574 104010
1696
1697 004576 170011
1698 004600 170400
1699 004602 170267 174656
1700 004606 022767 040204 174650
1701 004614 001401
1702 004616 104001
1703 004620 174067 174344
1704 004624 170500
1705 004626 170000
1706 004630 001401
1707 004632 104010
1708
1709 004634 172467 174762
1710 004640 170400
1711 004642 170267 174616
1712 004646 022767 040204 174610
1713 004654 001401
1714 004656 104001
1715 004660 174067 174304
1716 004664 170500
1717 004666 170000
1718 004670 001401
1719 004672 104010

```

```

*****
: *TEST 11    TEST OF CLRX INSTRUCTIONS
*****
↑ST11:    SCOPE
          LDFPS    #40200            ; DOUBLE MODE
          LDD     D1010,ACD        ; LOAD A -1
          STD     ACD,$REGO        ; PUT INTO $REGO
          SETF                    ; SET FLOATING
          CLRF     $REGO           ; 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     $REGO           ; TEST FOR ZERO
          CFCC                    ; GET CC
          BEQ     .+4
          ERROR    11            ; ACD NOT ZERO

          CMP     D1010+4,$REG2    ; CHECK THIRD WORD
          BEQ     .+4
          ERROR    10            ; $REG2 NOT -1
          CMP     D1010+6,$REG3    ; CHECK FOURTH
          BEQ     .+4
          ERROR    10            ; $REG3 NOT 0

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

          LDD     DMZERO,ACD      ; LOAD A MINUS ZERO
          CLRD     ACD            ; 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     ACD,$REGO       ; STORE RESULT
          TSTD    ACD            ; CHECK IT
          CFCC                    ; GET CC
          BEQ     .+4
          ERROR    10            ; DID NOT CLEAR

```

```

1720
1721
1722
1723 004674 000004
1724 004676 170127 040000
1725 004702 172427 140640
1726 004706 170700
1727 004710 170267 174550
1728 004714 022767 040000 174542
1729 004722 001401
1730 004724 104001
1731
1732 004726 174067 174236
1733 004732 173427 040640
1734 004736 170000
1735 004740 001401
1736 004742 104011
1737
1738 004744 170767 174220
1739 004750 170267 174510
1740 004754 022767 040010 174502
1741 004762 001401
1742 004764 104001
1743
1744 004766 022767 140640 174174
1745 004774 001401
1746 004776 104011
1747
1748 005000 005767 174166
1749 005004 001401
1750 005006 104011
1751
1752 005010 170127 047400
1753 005014 170400
1754 005016 170700
1755 005020 170267 174440
1756 005024 022767 047404 174432
1757 005032 001401
1758 005034 104001
1759
1760 005036 174067 174126
1761 005042 170500
1762 005044 170000
1763 005046 001401
1764 005050 104011
1765
1766 005052 170011
1767 005054 170400
1768 005056 170700
1769 005060 170267 174400
1770 005064 022767 047604 174372
1771 005072 001401
1772 005074 104001
1773
1774 005076 174067 174066
1775 005102 170500

```

```

*****
*TEST 12 TEST OF NEGX
*****
†ST12: SCOPE
LDFPS #40000
LDF #-5,ACD ;LOAD ACD WITH -5
NEGF ACD ;MAKE IT 5
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

STF ACD,$REGO ;GET THE RESULT
CMPF #5,ACD ;CHECK THE RESULT
CFCC ;GET CC
BEQ .+4
ERROR 11 ;RESULT NOT 5

NEGF $REGO ;MAKE IT -5
STFPS FPS ;STORE FLOATING POINT STATUS
CMP #40010,FPS ;CHECK FLOATING POINT STATUS
BEQ .+4 ;BRANCH IF OK
ERROR 1 ;FPS NOT EQUAL TO 40010

CMP #140640,$REGO ;CHECK THE RESULT
BEQ .+4
ERROR 11 ;RESULT NOT -5

TST $REG1 ;REST 0?
BEQ .+4 ;SKIP IF OK
ERROR 11

LDFPS #47400 ;TURN ON INTERUPTS
CLRF ACD ;CLEAR ACD
NEGF ACD ;NEGATE IT
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 ACD,$REGO ;GET RESULT
TSTF ACD ;CHECK IT
CFCC ;GET CC
BEQ .+4
ERROR 11 ;RESULT NOT 0

SETD ;SET DOUBLE MODE
CLRD ACD ;CLEAR ACD
NEGD ACD ;NEGATE ACD
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

STF ACD,$REGO ;GET RESULT
TSTF ACD ;TEST RESULT

```

M03

FPU INSTR EXERCISER MACY11 27(1006) 09-FEB-77 10:28 PAGE 40
DQFPCA.P11 09-FEB-77 10:26 T12 TEST OF NEGX

1776 005104 170000
1777 005106 001401
1778 005110 104011
1779

CFCC
BEQ i+4
ERROR ii

;GET CC
;RESULT NOT 0

```

1780
1781
1782
1783 005112 000004
1784 005114 170127 040000
1785 005120 172427 140200
1786 005124 174067 174040
1787 005130 170667 174034
1788 005134 170267 174324
1789 005140 022767 040000 174316
1790 005146 001401
1791 005150 104001
1792
1793 005152 022767 040200 174010
1794 005160 001401
1795 005162 104011
1796
1797 005164 005767 174002
1798 005170 001401
1799 005172 104011
1800
1801 005174 170600
1802 005176 170267 174262
1803 005202 174067 173762
1804 005206 173427 040200
1805 005212 170000
1806 005214 001401
1807 005216 104011
1808
1809 005220 172467 174376
1810 005224 170600
1811 005226 170267 174232
1812 005232 022767 040004 174224
1813 005240 001401
1814 005242 104001
1815
1816 005244 174067 173720
1817 005250 170500
1818 005252 170000
1819 005254 001401
1820 005256 104011
1821
1822 005260 170011
1823 005262 172467 174270
1824 005266 170600
1825 005270 170267 174170
1826 005274 022767 040200 174162
1827 005302 001401
1828 005304 104001
1829 005306 174067 173656
1830 005312 173467 174274
1831 005316 170000
1832 005320 001401
1833 005322 104010
1834

```

```

*****
:TEST 13 TEST OF ABSX
*****
T13: SCOPE
LDFPS #40000
LDF #-1,ACD ;LOAD A -1
STF ACD,$REGO ;GET IT
ABSF $REGO ;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,$REGO ;CHECK FOR A 1
BEQ .+4
ERROR 11 ;RESULT NOT 1

TST $REG1 ;REST 0?
BEQ .+4 ;SKIP IF OK
ERROR 11

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

LDF DMZERO,ACD ;LOAD A MINUS ZERO
ABSF ACD ;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 ACD,$REGO ;GET RESULT
TSTF ACD ;TEST FOR 0
CFCC ;GET CC
BEQ .+4
ERROR 11 ;RESULT NOT 0

SETD ;SET DOUBLE MODE
LDD D1010,ACD ;LOAD -1,0,-1,0
ABSD ACD ;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 ACD,$REGO ;GET RESULT
CMPD DBIG,ACD ;CHECK THE RESULT
CFCC ;GET CC
BEQ .+4
ERROR 10 ;RESULT NOT 77777,0,-1,0

```

```

1835
1836
1837
1838 005324 000004
1839 005326 170127 040200
1840 005332 172467 174220
1841 005336 176427 177600
1842 005342 170267 174116
1843 005346 022767 040204 174110
1844 005354 001401
1845 005356 104001
1846
1847 005360 174067 173604
1848 005364 170500
1849 005366 170000
1850 005370 001401
1851 005372 104010
1852
1853 005374 175067 173570
1854 005400 013700 177776
1855 005404 042700 177760
1856 005410 022700 000010
1857 005414 001401
1858 005416 104004
1859
1860 005420 022767 177600 173542
1861 005426 001401
1862 005430 104012
1863
1864 005432 170001
1865 005434 172467 174120
1866 005440 176427 000200
1867 005444 170267 174014
1868 005450 022767 040006 174006
1869 005456 001401
1870 005460 104001
1871
1872 005462 174067 173502
1873 005466 005767 173476
1874 005472 001401
1875 005474 104011
1876
1877 005476 005767 173470
1878 005502 001401
1879 005504 104011
1880
1881 005506 175067 173456
1882 005512 022767 177600 173450
1883 005520 001401
1884 005522 104012
1885
1886 005524 176527 000052
1887 005530 175100
1888 005532 022700 000052
1889 005536 001401
1890 005540 104013

```

```

*****
;TEST 14 TEST OF LDEXP & STEXP
*****
†ST14: 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,r,PS ;CHECK FLOATING POINT STATUS
BEQ .+4 ;BRANCH IF OK
ERROR 1 ;FPS NOT EQUAL TO 40204

STD ACO,$REG0 ;GET THE RESULT
TSTD ACO ;IS IT 0
CFCC
BEQ .+4
ERROR 10 ;ACO NOT 0

STEXP ACO,$REG0 ;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,$REG0 ;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,$REG0 ;SAVE RESULT
TST $REG0 ;CHECK FIRST WORD
BEQ .+4
ERROR 11 ;$REG0 NOT 0

TST $REG1 ;CHECK SECOND WORD
BEQ .+4
ERROR 11 ;$REG1 NOT ZERO

STEXP ACO,$REG0 ;GET THE EXPONENT BACK
CMP #-200,$REG0 ;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

```

C04

FPU INSTR EXERCISER MACY11 27(1006) 09-FEB-77 10:28 PAGE 43
DQFPCA.P11 09-FEB-77 10:26 T14 TEST OF LDEXP & STEXP

1891						
1892	005542	176627	000025	LDEXP	#25,AC2	;LOAD OTHER ALT 1'S
1893	005546	175200		STEXP	AC2,RO	;GET IT BACK
1894	005550	022700	000025	CMP	#25,RO	;CHECK IT
1895	005554	001401		BEQ	+4	
1896	005556	104013		ERROR	13	;EXP NOT 125
1897						

```

1898
1899
1900
1901 005560 000004
1902 005562 170127 047400
1903 005566 172427 040600
1904 005572 172027 040400
1905 005576 170267 173662
1906 005602 022767 047400 173654
1907 005610 001401
1908 005612 104001
1909
1910 005614 174067 173350
1911 005620 173427 040700
1912 005624 170000
1913 005626 001401
1914 005630 104011
1915
1916 005632 173027 041020
1917 005636 170267 173622
1918 005642 022767 047410 173614
1919 005650 001401
1920 005652 104001
1921
1922 005654 174067 173310
1923 005660 173427 140500
1924 005664 170000
1925 005666 001401
1926 005670 104011
1927
1928

```

```

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

```

```

1929
1930
1931
1932 005672 000004
1933 005674 170127 047600
1934 005700 172527 141400
1935 005704 172127 040640
1936 005710 170267 173550
1937 005714 022767 047610 173542
1938 005722 001401
1939 005724 104001
1940
1941 005726 174177 173700
1942 005732 173527 141330
1943 005736 170000
1944 005740 001401
1945 005742 104010
1946
1947 005744 172667 173664
1948 005750 172267 173670
1949 005754 170267 173504
1950 005760 022767 047600 173476
1951 005766 001401
1952 005770 104001
1953
1954 005772 174277 173634
1955 005776 173627 040500
1956 006002 170000
1957 006004 001401
1958 006006 104010
1959
1960 006010 173267 173620
1961 006014 170267 173444
1962 006020 022767 047600 173436
1963 006026 001401
1964 006030 104001
1965
1966 006032 174267 173132
1967 006036 173667 173612
1968 006042 170000
1969 006044 001401
1970 006046 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,$SREGO ;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,$SREGO ;GET RESULT
CMPD #3,AC2 ;CHECK RESULT
CFCC ;GET CC
BEQ .+4
ERROR 10 ;$SREGO NOT DALTAN

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

```

```

1971
1972
1973
1974 006050 000004
1975 006052 170127 047400
1976 006056 172727 040740
1977 006062 171327 140500
1978 006066 170267 173372
1979 006072 022767 047410 173364
1980 006100 001401
1981 006102 104001
1982
1983 006104 174367 173060
1984 006110 173727 141250
1985 006114 170000
1986 006116 001401
1987 006120 104011
1988
1989 006122 174727 140740
1990 006126 170267 173332
1991 006132 022767 047400 173324
1992 006140 001401
1993 006142 104001
1994
1995 006144 174367 173020
1996 006150 173727 040500
1997 006154 170000
1998 006156 001401
1999 006160 104011

```

```

*****
;TEST 17 TEST OF MULF AND DIVF
*****
↑ST17: 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,$REGD ;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,$REGD ;GET RESULT
CMPF #3,AC3 ;CHECK RESULT
CFCC ;GET CC
BEQ .+4
ERROR 11 ;-21. / -7 NOT 3

```

G04

```

2000
2001
2002
2003 006162 000004
2004 006164 170127 047600
2005 006170 172427 140640
2006 006174 171027 140500
2007 006200 170267 173260
2008 006204 022767 047600 173252
2009 006212 001401
2010 006214 104001
2011
2012 006216 174067 172746
2013 006222 173427 041160
2014 006226 170000
2015 006230 001401
2016 006232 104010
2017
2018 006234 174427 140400
2019 006240 170267 173220
2020 006244 022767 047610 173212
2021 006252 001401
2022 006254 104001
2023
2024 006256 174067 172706
2025 006262 173427 140760
2026 006266 170000
2027 006270 001401
2028 006272 104010

```

```

*****
;TEST 20 TEST OF MULD AND DIVD
*****
↑ST20: SCOPE
LDFPS #47600 ;LOAD DOUBLE MODE
LDD #-5,AC0 ;LOAD A -5
MULD #-3,AC0 ;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,$REG0 ;GET RESULT
CMPD #15.,AC0 ;CHECK RESULT
CFCC ;GET CC
BEQ .+4
ERROR 10 ;-5 X -3 NOT +15.

DIVD #-2,AC0 ;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,$REG0 ;STORE RESULT
CMPD #-7.5,AC0 ;CHECK RESULT
CFCC ;GET CC
BEQ .+4
ERROR 10 ;15. / -2 NOT -7.5

```

H04

FPU INSTR EXERCISER MACY11 27(1006) 09-FEB-77 10:28 PAGE 48
DOFPCA.P11 09-FEB-77 10:26

T21 TEST OF LDCFD,LDCDF

```

2029 .....
2030 :*TEST 21 TEST OF LDCFD,LDCDF
2031 :*TEST 21 TEST OF LDCFD,LDCDF
2032 006274 000004
2033 006276 170127 047600
2034 006302 172477 173276
2035 006306 012700 001556
2036 006312 177420
2037 006314 022700 001562
2038 006320 001401
2039 006322 104005
2040
2041 006324 170267 173134 STFPS FPS ;STORE FLOATING POINT STATUS
2042 006330 022767 047610 173126 CMP #47610,FPS ;CHECK FLOATING POINT STATUS
2043 006336 001401 BEQ .+4 ;BRANCH IF OK
2044 006340 104001 ERROR 1 ;FPS NOT EQUAL TO 47610
2045
2046 006342 174067 172622 STD ACO,$REGO ;GET ANSWER
2047 006346 173467 173354 CMPD D1000,ACO ;CHECK RESULT
2048 006352 170000 CFCC ;GET CC
2049 006354 001401 BEQ .+4
2050 006356 104010 ERROR 10 ;ACO NOT -1,0,0,0
2051
2052 006360 172567 173174 LDD D0101,AC1 ;LOAD A 0,-1,0,-1
2053 006364 170001 SETF ;SET FLOATING MODE
2054 006366 012700 001566 MOV #D1010+10,RO ;GET ADDRESS OF DATA +10
2055 006372 177540 LDCDF -(RO),AC1 ;LOAD A -1,0
2056 006374 022700 001556 CMP #D1010,RO ;ADDRESS DECREMENT BY 10?
2057 006400 001401 BEQ .+4
2058 006402 104005 ERROR 5 ;RO NOT #D1010
2059
2060 006404 170267 173054 STFPS FPS ;STORE FLOATING POINT STATUS
2061 006410 022767 047410 173046 CMP #47410,FPS ;CHECK FLOATING POINT STATUS
2062 006416 001401 BEQ .+4 ;BRANCH IF OK
2063 006420 104001 ERROR 1 ;FPS NOT EQUAL TO 47410
2064
2065 006422 170011 SETD
2066 006424 174167 172540 STD AC1,$REGO ;GET RESULT
2067 006430 173567 173140 CMPD WEIRD,AC1 ;CHECK RESULT
2068 006434 170000 CFCC
2069 006436 001401 BEQ .+4
2070 006440 104010 ERROR 10 ;RESULT NOT -1,1,0,-1
2071
2072 006442 170127 047440 LDFPS #47440 ;SET DOUBLE AND TRUNCATE MODES
2073 006446 177567 173104 LDCDF D1010,AC1 ;LOAD IT
2074 006452 174167 172512 STD AC1,$REGO ;GET RESULT
2075 006456 173567 173104 CMPD D1001,AC1 ;CHECK RESULT
2076 006462 170267 172776 STFPS FPS ;GET STATUS
2077 006466 170000 CFCC ;GET CC
2078 006470 001401 BEQ .+4
2079 006472 104010 ERROR 10 ;AC1 NOT -1,0,0,1
2080
2081 006474 170127 047400 LDFPS #47400 ;SET ROUND AND FLOATING MODES
2082 006500 177567 173052 LDCDF D1010,AC1 ;LOAD A -1,0
2083 006504 174167 172460 STD AC1,$REGO ;GET THE RESULT
2084 006510 170267 172750 STFPS FPS ;GET STATUS

```

FPU INSTR EXERCISER MACY11 27(1006) 09-FEB-77 10:28 PAGE 49
DOFPCA.P11 09-FEB-77 10:26 T21 TEST OF LDCFD,LDCDF

2085 006514 022767 000001 172450
2086 006522 001401
2087 006524 104010
2088

CMP #1,SREG1
BEQ +4
ERROR i0

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

```

2089
2090
2091
2092 006526 000004
2093 006530 170127 040200
2094 006534 172667 173016
2095 006540 170001
2096 006542 176202
2097 006544 170011
2098 006546 174267 172416
2099 006552 173667 173150
2100 006556 170000
2101 006560 001401
2102 006562 104010
2103
2104 006564 172767 173154
2105 006570 176303
2106 006572 174367 172372
2107 006576 173767 173134
2108 006602 170000
2109 006604 001401
2110 006606 104010
2111
2112 006610 172467 173130
2113 006614 170127 000040
2114 006620 176000
2115 006622 170011
2116 006624 174067 172340
2117 006630 173467 173070
2118 006634 170000
2119 006636 001401
2120 006640 104010
2121

```

```

*****
;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
STCFD 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,$REG0 ;GET RESULT
CMPD D0100X,AC3 ;CHECK RESULT
CFCC ;GET CC
BEQ .+4
ERROR 10 ;$REG2 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)

```

K04

```

2122 .....
2123 ;*TEST 23 TEST OF LDCIF,LDCID,STCFI,STCDI
2124 .....
2125 006642 000004          †ST23: SCOPE
2126 006644 170127 047400 LDFPS #47400 ;FLOATING MODE
2127 006650 177027 000005 LDCIF #5,AC0 ;STORE A 5
2128 006654 170267 172604 STFPS FPS ;STORE FLOATING POINT STATUS
2129 006660 022767 047400 172576 CMP #47400,FPS ;CHECK FLOATING POINT STATUS
2130 006666 001401 BEQ .+4 ;BRANCH IF OK
2131 006670 104001 ERROR 1 ;FPS NOT EQUAL TO 47400
2132
2133 006672 174067 172272 STF ACO,$REGO ;GET THE RESULT
2134 006676 173427 040640 CMPF #5,ACO ;CHECK IT
2135 006702 170000 CFCC ;GET CC
2136 006704 001401 BEQ .+4
2137 006706 104011 ERROR 11 ;ACO NOT 5.0
2138
2139 006710 175400 STCFI ACO,RO ;CONVERT IT BACK
2140 006712 022700 000005 CMP #5,RO ;CHECK RESULT
2141 006716 001401 BEQ .+4
2142 006720 104013 ERROR 13 ;RO NOT 5
2143
2144 006722 170011 SETD ;SET DOUBLE MODE
2145 006724 172567 172630 LDD 00101,AC1 ;LOAD JUNK
2146 006730 012700 001556 MOV #01010,RO ;LOAD ADDRESS OF DATA
2147 006734 177120 LDCID (RO)+,AC1 ;CONVERT TO -1.0
2148 006736 022700 001560 CMP #01010+2,RO ;CHECK ADDRESS
2149 006742 001401 BEQ .+4
2150 006744 104005 ERROR 5 ;RO NOT #01010+2
2151
2152 006746 170267 172512 STFPS FPS ;STORE FLOATING POINT STATUS
2153 006752 022767 047610 172504 CMP #47610,FPS ;CHECK FLOATING POINT STATUS
2154 006760 001401 BEQ .+4 ;BRANCH IF OK
2155 006762 104001 ERROR 1 ;FPS NOT EQUAL TO 47610
2156
2157 006764 174167 172200 STD ACO,$REGO ;GET RESULT
2158 006770 173527 140200 CMPD #-1,ACO ;CHECK RESULT
2159 006774 170000 CFCC ;GET CC
2160 006776 001401 BEQ .+4
2161 007000 104010 ERROR 10 ;ACO NOT -1.0
2162
2163 007002 175500 STCDI ACO,RO ;CONVERT IT BACK
2164 007004 022700 177777 CMP #-1,RO ;CHECK RESULT
2165 007010 001401 BEQ .+4
2166 007012 104013 ERROR 13 ;RO NOT -1
2167
2168 007014 170001 SETF ;SET FLOATING MODE
2169 007016 177227 054321 LDCIF #54321,AC2 ;LOAD 54321
2170 007022 174267 172142 STF AC2,$REGO ;GET RESULT
2171 007026 170267 172432 STFPS FPS ;GET STATUS
2172 007032 173667 172724 CMPF F5T01,AC2 ;CHECK IT
2173 007036 170000 CFCC ;CHECK CC
2174 007040 001401 BEQ .+4
2175 007042 104011 ERROR 11 ;AC2 NOT 54321.
2176
2177 007044 052737 000017 177776 BIS #17,2#PS ;SET PS

```

L04

FPU INSTR EXERCISER MACY11 27(1006) 09-FEB-77 10:28 PAGE 52
DQFPCA.P11 09-FEB-77 10:26 T23 TEST OF LDCIF,LDCID,STCFI,STCDI

2178	007052	175667	172112	STCFI	AC2,\$REG0	; CONVERT IT BACK
2179	007056	013700	177776	MOV	@#PS,RO	; GET PS
2180	007062	042700	177760	BIC	#177760,RO	; CLEAR JUNK
2181	007066	001401		BEQ	.+4	; SKIP IF OK
2182	007070	104004		ERROR	4	; PS NOT 0
2183	007072	170267	172366	STFPS	FPS	; GET STATUS
2184						
2185	007076	022767	054321 172064	CMP	#54321,\$REG0	; CHECK RESULT
2186	007104	001401		BEQ	.+4	
2187	007106	104012		ERROR	12	; \$REG0 NOT 54321
2188						

M04

```

2189
2190
2191
2192 007110 000004
2193 007112 170127 047500
2194 007116 177327 177773
2195 007122 000401
2196 007124 104000
2197
2198 007126 170267 172332
2199 007132 022767 047510 172324
2200 007140 001401
2201 007142 104001
2202
2203 007144 174367 172020
2204 007150 173727
2205 007152 144640
2206 007154 170000
2207 007156 001401
2208 007160 104011
2209
2210 007162 012700 001632
2211 007166 175730
2212 007170 022700 001634
2213 007174 001401
2214 007176 104005
2215
2216 007200 170267 172260
2217 007204 022767 177773 171756
2218 007212 001401
2219 007214 104011
2220
2221 007216 005767 171750
2222 007222 001401
2223 007224 104011
2224
2225 007226 170011
2226 007230 177067 172520
2227 007234 170267 172224
2228 007240 022767 047700 172216
2229 007246 001401
2230 007250 104001
2231
2232 007252 174067 171712
2233 007256 173467 172500
2234 007262 170000
2235 007264 001401
2236 007266 104010
2237
2238 007270 012700 001174
2239 007274 175440
2240 007276 022700 001170
2241 007302 001401
2242 007304 104005
2243
2244 007306 170267 172152

```

```

;*****
;TEST 24 TEST OF LDCLF,LDCLD,STCFL,STCDL
;*****
†ST24: 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
CFCC ;GET CC
BEQ .+4
ERROR 11 ;AC3 NOT -5

MOV #A$REG0,RO ;SET UP ADDRESS OF ADDRESS
STCFL AC3,(RO)+ ;STORE IN $REG0
CMP #A$REG0+2,RO ;CHECK ADDRESS
BEQ .+4
ERROR 5 ;ADDRESS IN FPU NOT A$REG0+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 #A$REG2,RO ;GET IT BACK
STCDL ACO,-(RO) ;CHECK ADDRESS
CMP #A$REG0,RO
BEQ .+4
ERROR 5 ;RO NOT #A$REG0

STFPS FPS ;STORE FLOATING POINT STATUS

```

N04

FPU INSTR EXERCISER MACY11 27(1006) 09-FEB-77 10:28 PAGE 54
DGFPCA.P11 09-FEB-77 10:26 T24 TEST CF LOCLF, LOCLD, STCFL, STCDL

2245	007312	022767	047700	172144	CMP	#47700, FPS	;CHECK FLOATING POINT STATUS
2246	007320	001401			BEQ	.+4	;BRANCH IF OK
2247	007322	104001			ERROR	i	;FPS NOT EQUAL TO 47700
2248							
2249	007324	170267	172134		STFPS	FPS	;GET STATUS
2250	007330	026767	172420	171632	CMP	DST01, \$REG0	;CHECK LEFT
2251	007336	001401			BEQ	.+4	
2252	007340	104011			ERROR	11	; \$REG0 NOT
2253							
2254	007342	026767	172410	171622	CMP	DST01+2, \$REG1	;CHECK RIGHT
2255	007350	001401			BEQ	.+4	
2256	007352	104011			ERROR	11	; \$REG1 NOT
2257							

```

2258
2259
2260
2261 007354 000004
2262 007356 170127 047400
2263 007362 172467 172260
2264 007366 171427 040200
2265 007372 170267 172066
2266 007376 022767 047404 172060
2267 007404 001401
2268 007406 104001
2269
2270 007410 174067 171554
2271 007414 170500
2272 007416 170000
2273 007420 001401
2274 007422 104011
2275
2276 007424 174167 171540
2277 007430 173567 172212
2278 007434 170000
2279 007436 001401
2280 007440 104011
2281
2282 007442 172627 040200
2283 007446 171667 172242
2284 007452 170267 172006
2285 007456 022767 047400 172000
2286 007464 001401
2287 007466 104001
2288
2289 007470 174267 171474
2290 007474 173667 172214
2291 007500 170000
2292 007502 001401
2293 007504 104011
2294
2295 007506 174367 171456
2296 007512 170503
2297 007514 170000
2298 007516 001401
2299 007520 104011
2300
2301 007522 170011
2302 007524 172467 172154
2303 007530 171467 172140
2304 007534 174067 171430
2305 007540 173467 172120
2306 007544 170000
2307 007546 001401
2308 007550 104010
2309
2310 007552 174367 171412
2311 007556 170503
2312 007560 170000
2313 007562 001401

```

```

*****
;TEST 25 TEST OF MODF AND MODD
*****
TST25: SCOPE
LDFPS #47400 ;SET FLOATING MODE
LDF DALTB+2,ACO ;LOAD 52525,52525 INTO ACO
MODF #1,ACO ;MOD BY A ! *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

```

C05

FPU INSTR EXERCISER MACY11 27(1006) 09-FEB-77 10:28 PAGE 56
DOFPCA.P11 09-FEB-77 10:26 T25 TEST OF MODF AND MODD

2314 007564 104010
2315

ERROR 10

;RESULT NOT 0

E05

FPU INSTR EXERCISER MACY11 27(1006) 09-FEB-77 10:28 PAGE 58
DQFPCA.P11 09-FEB-77 10:26 T27 MULF ERROR - OVERFLOW

```

2372 010002 170267 171456          STFPS  FPS          ;STORE FLOATING POINT STATUS
2373 010006 170367 171454          STST   FEC          ;STORE EXCEPTION CODES
2374 010012 022767 101002 171444  CMP    #101002,FPS  ;CHECK FLOATING POINT STATUS
2375 010020 001401                   BEQ    .+4          ;BRANCH IF OK
2376 010022 104001                   ERROR  1           ;FPS NOT EQUAL TO 101002
2377 010024 022767 000010 171434  CMP    #10,FEC     ;CHECK FLOATING EXCEPTION CODE
2378 010032 001401                   BEQ    .+4          ;BRANCH IF OK
2379 010034 104001                   ERROR  1           ;FEC NOT EQUAL TO 10
2380 010036 022767 007760 171424  CMP    #7760,FEA   ;CHECK FLOATING EXCEPTION ADDRESS
2381 010044 001401                   BEQ    .+4          ;BRANCH IF OK
2382 010046 104001                   ERROR  1           ;FEA NOT EQUAL TO 7760
2383
2384
2385
2386
2387 010050 000004          †ST30: SCOPE
2388 010052 170127 002000          LDFPS  #2000        ;FLOATING/UNDERFLOW
2389 010056 005001          CLR    R1           ;CLEAR FLAG WORD
2390 010060 172427 002252          LDF    #1E-36,AC0  ;LOAD A SMALL NUMBER
2391 010064 174427 076101 1$:  DIVF  #1E36,AC0  ;DIVIDE BY A LARGE NUMBER
2392 010070 174067 171074          STF    AC0,$REG0   ;GET FOR TYPING
2393 010074 170267 171364          STFPS  FPS         ;GET STATUS
2394 010100 005701          TST    R1           ;DID IT TRAP?
2395 010102 001001          BNE    3$          ;SKIP IF SET
2396 010104 104007          ERROR  7           ;DID NOT TRAP ON UNDERFLOW
2397 010106
2398 010106 170267 171352          STFPS  FPS         ;STORE FLOATING POINT STATUS
2399 010112 170367 171350          STST   FEC         ;STORE EXCEPTION CODES
2400 010116 022767 102000 171340  CMP    #102000,FPS ;CHECK FLOATING POINT STATUS
2401 010124 001401                   BEQ    .+4          ;BRANCH IF OK
2402 010126 104001                   ERROR  1           ;FPS NOT EQUAL TO 102000
2403 010130 022767 000012 171330  CMP    #12,FEC     ;CHECK FLOATING EXCEPTION CODE
2404 010136 001401                   BEQ    .+4          ;BRANCH IF OK
2405 010140 104001                   ERROR  1           ;FEC NOT EQUAL TO 12
2406 010142 022767 010064 171320  CMP    #10064,FEA  ;CHECK FLOATING EXCEPTION ADDRESS
2407 010150 001401                   BEQ    .+4          ;BRANCH IF OK
2408 010152 104001                   ERROR  1           ;FEA NOT EQUAL TO 10064
2409
2410
2411
2412
2413 010154 000004          †ST31: SCOPE
2414 010156 170127 000400          LDFPS  #400         ;FLOATING/INTEGER/CONVERSION
2415 010162 005001          CLR    R1           ;CLEAR FLAG WORD
2416 010164 172527 076101          LDF    #1E36,AC1  ;LOAD LARGE NUMBER
2417 010170 175567 170774 1$:  STCFI AC1,$REG0   ;TRY TO STUFF INTO 16 BITS
2418 010174 170267 171264          STFPS  FPS         ;GET STATUS
2419 010200 005701          TST    R1           ;TRAP FLAG SET?
2420 010202 001001          BNE    3$          ;SKIP IF SET
2421 010204 104007          ERROR  7           ;DID NOT TRAP ON CONVERT
2422 010206
2423 010206 170267 171252          STFPS  FPS         ;STORE FLOATING POINT STATUS
2424 010212 170367 171250          STST   FEC         ;STORE EXCEPTION CODES
2425 010216 022767 100405 171240  CMP    #100405,FPS ;CHECK FLOATING POINT STATUS
2426 010224 001401                   BEQ    .+4          ;BRANCH IF OK
2427 010226 104001                   ERROR  1           ;FPS NOT EQUAL TO 100405

```

F05

FPU INSTR EXERCISER MACY11 27(1006) 09-FEB-77 10:28 PAGE 59
 DQFPCA.P11 09-FEB-77 10:26 T31 STCFI ERROR - CONVERSION(6)

2428	010230	022767	000006	171230	CMP	#6,FEC	;CHECK FLOATING EXCEPTION CODE
2429	010236	001401			BEQ	+.4	;BRANCH IF OK
2430	010240	104001			ERROR	1	;FEC NOT EQUAL TO 6
2431	010242	022767	010170	171220	CMP	#10170,FEA	;CHECK FLOATING EXCEPTION ADDRESS
2432	010250	001401			BEQ	+.4	;BRANCH IF OK
2433	010252	104001			ERROR	1	;FEA NOT EQUAL TO 10170
2434							
2435							
2436							
2437							
2438	010254	000004			ST32:	SCOPE	
2439	010256	170127	000000		LDFPS	#0	;FLOATING
2440	010262	005001			CLR	R1	;CLEAR FLAG
2441	010264	174527	000000		1\$:	DIVF	#0,AC1
2442	010270	170267	171170		STFPS	FPS	;GET STATUS
2443	010274	005701			TST	R1	;CHECK FLAG
2444	010276	001001			BNE	3\$;SKIP IF SET
2445	010300	104007			ERROR	7	;DIVIDE BY 0 DID NOT TRAP
2446	010302				3\$:		
2447	010302	170267	171156		STFPS	FPS	;STORE FLOATING POINT STATUS
2448	010306	170367	171154		STST	FEC	;STORE EXCEPTION CODES
2449	010312	022767	100000	171144	CMP	#100000,FPS	;CHECK FLOATING POINT STATUS
2450	010320	001401			BEQ	+.4	;BRANCH IF OK
2451	010322	104001			ERROR	1	;FPS NOT EQUAL TO 100000
2452	010324	022767	000004	171134	CMP	#4,FEC	;CHECK FLOATING EXCEPTION CODE
2453	010332	001401			BEQ	+.4	;BRANCH IF OK
2454	010334	104001			ERROR	1	;FEC NOT EQUAL TO 4
2455	010336	022767	010264	171124	CMP	#10264,FEA	;CHECK FLOATING EXCEPTION ADDRESS
2456	010344	001401			BEQ	+.4	;BRANCH IF OK
2457	010346	104001			ERROR	1	;FEA NOT EQUAL TO 10264
2458							
2459							
2460							
2461							
2462	010350	000004			ST33:	SCOPE	
2463	010352	170127	004000		LDFPS	#4000	;FLOATING/UNDEFINED VARIABLE
2464	010356	005001			CLR	R1	;CLEAR FLAG
2465	010360	172667	171236		1\$:	LDF	DMZERO,PC2
2466	010364	170267	171074		STFPS	FPS	;LOAD AN UNDEFINED VARIABLE
2467	010370	005701			TST	R1	;GET STATUS
2468	010372	001001			BNE	3\$;CHECK FLAG
2469	010374	104007			ERROR	7	;SKIP IF SET
2470	010376				3\$:		;LOAD OF -0 DID NOT TRAP
2471	010376	170267	171062		STFPS	FPS	;STORE FLOATING POINT STATUS
2472	010402	170367	171060		STST	FEC	;STORE EXCEPTION CODES
2473	010406	022767	104014	171050	CMP	#104014,FPS	;CHECK FLOATING POINT STATUS
2474	010414	001401			BEQ	+.4	;BRANCH IF OK
2475	010416	104001			ERROR	1	;FPS NOT EQUAL TO 104014
2476	010420	022767	000014	171040	CMP	#14,FEC	;CHECK FLOATING EXCEPTION CODE
2477	010426	001401			BEQ	+.4	;BRANCH IF OK
2478	010430	104001			ERROR	1	;FEC NOT EQUAL TO 14
2479	010432	022767	010360	171030	CMP	#10360,FEA	;CHECK FLOATING EXCEPTION ADDRESS
2480	010440	001401			BEQ	+.4	;BRANCH IF OK
2481	010442	104001			ERROR	1	;FEA NOT EQUAL TO 10360
2482							
2483							

```

2484
2485
2486 010444 000004
2487 010446 170127 000000
2488 010452 005001
2489 010454 177707
2490 010456 170267 171002
2491 010462 005701
2492 010464 001001
2493 010466 104007
2494 010470
2495 010470 170267 170770
2496 010474 170367 170766
2497 010500 022767 100000 170756
2498 010506 001401
2499 010510 104001
2500 010512 022767 000002 170746
2501 010520 001401
2502 010522 104001
2503 010524 022767 010454 170736
2504 010532 001401
2505 010534 104001
2506
2507
2508
2509
2510 010536 000004
2511 010540 170127 001000
2512 010544 005001
2513 010546 172767 171040
2514 010552 172367 171034
2515 010556 170267 170702
2516 010562 174367 170402
2517 010566 005701
2518 010570 001001
2519 010572 104007
2520 010574
2521 010574 170267 170664
2522 010600 170367 170662
2523 010604 022767 101006 170652
2524 010612 001401
2525 010614 104001
2526 010616 022767 000010 170642
2527 010624 001401
2528 010626 104001
2529 010630 022767 010552 170632
2530 010636 001401
2531 010640 104001
2532
2533
2534
2535
2536 010642 000004
2537 010644 170127 002000
2538 010650 005001
2539 010652 172427 000430

```

```

; *TEST 34 OPCODE ERROR
; *****
†ST34: SCOPE
LDFPS #0 ; FLOATING
CLR R1 ; CLEAR FLAG
1$: 177707 ; ILLEGAL OPCODE
STFPS FPS ; GET STATUS
TST R1 ; CHECK FLAG
BNE 3$ ; SKIP IF SET
ERROR 7 ; NOT AN ILLEGAL OPCODE

3$: 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 #10454,FEA ; CHECK FLOATING EXCEPTION ADDRESS
BEQ .+4 ; BRANCH IF OK
ERROR 1 ; FEA NOT EQUAL TO 10454

```

```

; *****
; *TEST 35 ADDF ERROR - OVERFLOW
; *****
†ST35: SCOPE
LDFPS #1000 ; FLOATING/OVERFLOW
CLR R1 ; CLEAR FLAG
1$: LDF DBIG,AC3 ; LOAD A BIG NUMBER
ADDF DBIG,AC3 ; MAKE OVERFLOW
STFPS FPS ; GET STATUS
STF AC3,$REGO ; GET RESULT
TST R1 ; FLAG SET?
BNE 3$ ; SKIP IF SET
ERROR 7 ; DID NOT TRAP ON OVERFLOW

3$: 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 #10552,FEA ; CHECK FLOATING EXCEPTION ADDRESS
BEQ .+4 ; BRANCH IF OK
ERROR 1 ; FEA NOT EQUAL TO 10552

```

```

; *****
; *TEST 36 SUBF ERROR UNDERFLOW
; *****
†ST36: SCOPE
LDFPS #2000 ; FLOATING/UNDERFLOW
CLR R1 ; CLEAR FLAG
LDF #.07E-37,AC0 ; LOAD SMALL NUMBER

```

2540	010656	173027	000504	15:	SUBF	#.09E-37,AC0	;SUBF SMALL NUMBER
2541	010662	174067	170302		STF	AC0,\$REG0	;GET RESULT
2542	010666	170267	170572		STFPS	FPS	;GET STATUS
2543	010672	005701			TST	R1	;FLAG SET?
2544	010674	001001			BNE	3\$;SKIP IF SET
2545	010676	104007			ERROR	7	;NO TRAP ON UNDERFLOW
2546	010700			35:			
2547	010700	170267	170560		STFPS	FPS	;STORE FLOATING POINT STATUS
2548	010704	170367	170556		STST	FEC	;STORE EXCEPTION CODES
2549	010710	022767	102014 170546		CMP	#102014,FPS	;CHECK FLOATING POINT STATUS
2550	010716	001401			BEQ	.+4	;BRANCH IF OK
2551	010720	104001			ERROR	1	;FPS NOT EQUAL TO 102014
2552	010722	022767	000012 170536		CMP	#12,FEC	;CHECK FLOATING EXCEPTION CODE
2553	010730	001401			BEQ	.+4	;BRANCH IF OK
2554	010732	104001			ERROR	1	;FEC NOT EQUAL TO 12
2555	010734	022767	010656 170526		CMP	#10656,FEA	;CHECK FLOATING EXCEPTION ADDRESS
2556	010742	001401			BEQ	.+4	;BRANCH IF OK
2557	010744	104001			ERROR	1	;FEA NOT EQUAL TO 10656

```

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

```

2563	010746	000004		TST37:	SCOPE		
2564	010750	012704	014300		MOV	#FEC02,R4	; PTR TO FPP FEC CODE GENERATOR
2565	010754	012705	011002		MOV	#FPPN1,R5	; PTR TO TEST DATA SET
2566	010760	012767	014300 170512		MOV	#FII02,EXPFEA	; ADDR(FPP BAD INSTR) EXPECTED
2567	010766	012767	014302 170506		MOV	#NXT02,EXPTS	; ADDR(NEXT INSTR) EXPECTED
2568							
2569	010774	004737	013744		JSR	PC,@TRPTST	; GO TEST

```

645: BR TST40 ;;
FPPN1: ; TEST DATA SET FPPN-1:
.WORD 000157,000040,000157 ; PSW'S: BEFORE, LOADED, AFTER
.WORD 047417,147417,100002 ; OLDFPS/NEWFPS/FECCODE

```

```

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

```

2582	011016	000004		TST40:	SCOPE		
2583	011020	012704	014310		MOV	#FEC04,R4	; PTR TO FPP FEC CODE GENERATOR
2584	011024	012705	011052		MOV	#FPPN2,R5	; PTR TO TEST DATA SET
2585	011030	012767	014314 170442		MOV	#FII04,EXPFEA	; ADDR(FPP BAD INSTR) EXPECTED
2586	011036	012767	014320 170436		MOV	#NXT04,EXPTS	; ADDR(NEXT INSTR) EXPECTED
2587							
2588	011044	004737	013744		JSR	PC,@TRPTST	; GO TEST

```

645: BR TST41 ;;
FPPN2: ; TEST DATA SET FPPN-2:
.WORD 000157,000040,000157 ; PSW'S: BEFORE, LOADED, AFTER
.WORD 047417,147400,100004 ; OLDFPS/NEWFPS/FECCODE

```

2590	011050						
2591	011050	000406					
2592							
2593	011052						
2594	011052	000157	000040 000157				
2595	011060	047417	147400 100004				

```

2596
2597
2598
2599
2600
2601 011066 000004
2602 011070 012704 014326
2603 011074 012705 011122
2604 011100 012767 014332 170372
2605 011106 012767 014336 170366
2606
2607 011114 004737 013744
2608
2609 011120
2610 011120 000406
2611
2612 011122
2613 011122 000155 000042 000142
2614 011130 047412 147405 100006
2615
2616
2617
2618
2619
2620 011136 000004
2621 011140 012704 014344
2622 011144 012705 011172
2623 011150 012767 014354 170322
2624 011156 012767 014356 170316
2625
2626 011164 004737 013744
2627
2628 011170
2629 011170 000406
2630
2631 011172
2632 011172 000151 000046 000151
2633 011200 047411 147406 100010
2634
2635
2636
2637
2638
2639 011206 000004
2640 011210 012704 014364
2641 011214 012705 011242
2642 011220 012767 014370 170252
2643 011226 012767 014374 170246
2644
2645 011234 004737 013744
2646
2647 011240
2648 011240 000406
2649
2650 011242
2651 011242 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,R5 ; PTR TO TEST DATA SET
MOV #FII06,EXPFEA ; ADDR(FPP BAD INSTR) EXPECTED
MOV #NXT06,EXPTS ; 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,R5 ; PTR TO TEST DATA SET
MOV #FII10,EXPFEA ; ADDR(FPP BAD INSTR) EXPECTED
MOV #NXT10,EXPTS ; 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,R5 ; PTR TO TEST DATA SET
MOV #FII12,EXPFEA ; ADDR(FPP BAD INSTR) EXPECTED
MOV #NXT12,EXPTS ; 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

```

J05

FPU INSTR EXERCISER MACY11 27(1006) 09-FEB-77 10:28 PAGE 63
DQFPCA.P11 09-FEB-77 10:26

T43 TEST OF FEC-12, WITH NO TRAP, INTERRUPT DISABLED

2652 011250 047417 147400 100012 .WORD 047417,147400,100012 ; OLDFPS/NEWFPS/FECCODE

2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707

; *TEST 44 TEST OF FEC-14, WITH NO TRAP, INTERRUPT DISABLED

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,EXPTS ; ADDR(NEXT INSTR) EXPECTED
JSR PC,@TRPTST ; GO TEST

645: BR TST45 ;;

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

; *TEST 45 TEST OF FEC-16, WITH NO TRAP, INTERRUPT DISABLED

TST45: SCOPE
; //////////////////////////////////////
; THIS TEST IS EXECUTED FOR HFP ONLY, NOT IN WFP MODE

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

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

; //////////////////////////////////////
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,EXPTS ; ADDR(NEXT INSTR) EXPECTED

JSR PC,@TRPTST ; GO TEST

645: BR TST46 ;;

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

K05

FPU INSTR EXERCISER MACY11 27(1006) 09-FEB-77 10:28 PAGE 64
DQFPCA.P11 09-FEB-77 10:26

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

2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763

011422 000004
011424 012704 014300
011430 012705 011456
011434 012767 014300 170036
011442 012767 014302 170032
011450 004737 013744
011454
011454 000406
011456
011456 000353 000104 000353
011464 000017 100017 140002
011472 000004
011474 012704 014310
011500 012705 011526
011504 012767 014314 167766
011512 012767 014320 167762
011520 004737 013744
011524
011524 000406
011526
011526 000350 000107 000350
011534 000017 100000 140004
011542 000004
011544 012704 014326
011550 012705 011576
011554 012767 014332 167716
011562 012767 014336 167712

```
;/;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;/
:
: 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

64$: 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

64$: 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
```

L05

FPU INSTR EXERCISER MACY11 27(1006) 09-FEB-77 10:28 PAGE 65
 DOFPCA.P11 09-FEB-77 10:26 T50

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

2764 011570 004737 013744

JSR PC, @TRPTST ; GO TEST

2765

2766 011574

64\$:

2767 011574 000406

BR TST51 ;;

2768

2769 011576

FPPN12: ; TEST DATA SET FPPN-12:

2770 011576 000353 000104 000345

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

2771 011604 007012 007005 000000

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

2772

2773

2774

2775

2776

2777 011612 000004

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

2778 011614 012704 014326

TST51: SCOPE

2779 011620 012705 011646

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

2780 011624 012767 014332 167646

MOV #FPPT13,R5 ; PTR TO TEST DATA SET

2781 011632 012767 014336 167642

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

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

2782

2783 011640 004737 013744

JSR PC, @TRPTST ; GO TEST

2784

2785 011644

64\$:

2786 011644 000406

BR TST52 ;;

2787

2788 011646

FPPT13: ; TEST DATA SET FPPT-13:

2789 011646 000353 000104 000345

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

2790 011654 000412 100405 140006

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

2791

2792

2793

2794

2795

2796 011662 000004

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

2797 011664 012704 014344

TST52: SCOPE

2798 011670 012705 011716

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

2799 011674 012767 014354 167576

MOV #FPPN14,R5 ; PTR TO TEST DATA SET

2800 011702 012767 014356 167572

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

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

2801

2802 011710 004737 013744

JSR PC, @TRPTST ; GO TEST

2803

2804 011714

64\$:

2805 011714 000406

BR TST53 ;;

2806

2807 011716

FPPN14: ; TEST DATA SET FPPN-14:

2808 011716 000350 000107 000350

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

2809 011724 006411 006406 000000

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

2810

2811

2812

2813

2814

2815 011732 000004

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

2816 011734 012704 014344

TST53: SCOPE

2817 011740 012705 011766

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

2818 011744 012767 014354 167526

MOV #FPPT15,R5 ; PTR TO TEST DATA SET

2819 011752 012767 014356 167522

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

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

M05

FPU INSTR EXERCISER MACY11 27(1006) 09-FEB-77 10:28 PAGE 66
DGFPCA.P11 09-FEB-77 10:26 T53 TEST OF FEC-10, WITH TRAP, INTERRUPT ENABLED

```

2820
2821 011760 004737 013744          JSR    PC,@TRPTST    ; GO TEST
2822
2823 011764          64$:          BR     TST54          ;;
2824 011764 000406
2825
2826 011766          FPPT15: ; TEST DATA SET FPPT-15:
2827 011766 000350 000107 000350      .WORD 000350,000107,000350 ; PSW'S: BEFORE, LOADED, AFTER
2828 011774 001011 101006 140010      .WORD 001011,101006,140010 ; OLDFPS/NEWFPS/FECODE
2829
2830
2831
2832 ;*****
2833 ;*TEST 54      TEST OF FEC-12, WITH NO TRAP, INTERRUPT ENABLED
2834 ;*****
2835 †TST54: SCOPE
2836      MOV     #FEC12,R4      ; PTR TO FPP FEC CODE GENERATOR
2837      MOV     #FPPN16,R5    ; PTR TO TEST DATA SET
2838      MOV     #FII12,EXPFEA ; ADDR(FPP BAD INSTR) EXPECTED
2839      MOV     #NXT12,EXPTS  ; ADDR(NEXT INSTR) EXPECTED
2840 012030 004737 013744          JSR    PC,@TRPTST    ; GO TEST
2841
2842 012034          64$:          BR     TST55          ;;
2843 012034 000406
2844
2845 012036          FPPN16: ; TEST DATA SET FPPN-16:
2846 012036 000350 000107 000350      .WORD 000350,000107,000350 ; PSW'S: BEFORE, LOADED, AFTER
2847 012044 005413 005404 000000      .WORD 005413,005404,000000 ; OLDFPS/NEWFPS/FECODE
2848
2849
2850
2851 ;*****
2852 ;*TEST 55      TEST OF FEC-12, WITH TRAP, INTERRUPT ENABLED
2853 ;*****
2854 †TST55: SCOPE
2855      MOV     #FEC12,R4      ; PTR TO FPP FEC CODE GENERATOR
2856      MOV     #FPPT17,R5    ; PTR TO TEST DATA SET
2857      MOV     #FII12,EXPFEA ; ADDR(FPP BAD INSTR) EXPECTED
2858      MOV     #NXT12,EXPTS  ; ADDR(NEXT INSTR) EXPECTED
2859 012100 004737 013744          JSR    PC,@TRPTST    ; GO TEST
2860
2861 012104          64$:          BR     TST56          ;;
2862 012104 000406
2863
2864 012106          FPP17: ; TEST DATA SET FPPT-17:
2865 012106 000350 000107 000350      .WORD 000350,000107,000350 ; PSW'S: BEFORE, LOADED, AFTER
2866 012114 002017 102000 140012      .WORD 002017,102000,140012 ; OLDFPS/NEWFPS/FECODE
2867
2868
2869
2870 ;*****
2871 ;*TEST 56      TEST OF FEC-14, WITH NO TRAP, INTERRUPT ENABLED
2872 ;*****
2873 †TST56: SCOPE
2874      MOV     #FEC14,R4      ; PTR TO FPP FEC CODE GENERATOR
2875      MOV     #FPPN20,R5    ; PTR TO TEST DATA SET
2876      MOV     #FII14,EXPFEA ; ADDR(FPP BAD INSTR) EXPECTED

```


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

```

2932
2933
2934 012314 076600 000022      MED      ,RWHAMI      ;GET WHAMI INTO RO
2935 012320 032700 000020      BIT      #BIT04,RO   ;HFP IN SYSTEM ?
2936 012324 001421              BEQ      64$         ;BR IF NONE
2937
2938 012326 076600 000144      MED      ,RFLAG      ;GET FLAGS INTO RO
2939 012332 032700 010000      BIT      #BIT12,RO   ;HFP ENABLED ?
2940 012336 001414              BEQ      64$         ;BR IF NOT
2941
2942 ;////////////////////////////////////
2943
2944 012340 012704 014414      MOV      #FEC16,R4   ; PTR TO FPP FEC CODE GENERATOR
2945 012344 012705 012372      MOV      #FPPT23,R5  ; PTR TO TEST DATA SET
2946 012350 012767 014422 167122  MOV      #FI116,EXPFER ; ADDR(FPP BAD INSTR) EXPECTED
2947 012356 012767 014424 167116  MOV      #NXT16,EXPTS  ; ADDR(NEXT INSTR) EXPECTED
2948
2949 012364 004737 013744      JSR      PC,#TRPTST  ; GO TEST
2950
2951 012370              64$:
2952 012370 000406              BR      TST62        ;;
2953
2954 012372              FPPT23: ; TEST DATA SET FPPT-23:
2955 012372 000357 000152 000341  .WORD   000357,000152,000341 ; PSW'S: BEFORE, LOADED, AFTER
2956 012400 000020 100020 140016  .WORD   000020,100020,140016 ; OLOFPS/NEWFPS/FECODE
2957
2958

```

C06

```

2959
2960
2961
2962 012406 000004
2963 012410 170127 047600
2964 012414 177327 000004
2965 012420 177227 000002
2966 012424 172302
2967 012426 173302
2968 012430 175737 001170
2969 012434 022737 000004 001170
2970 012442 170267 167016
2971 012446 001401
2972 012450 104012
2973
2974
2975
2976
2977 012452 000004
2978 012454 177027 000005
2979 012460 174005
2980 012462 172605
2981 012464 172200
2982 012466 173205
2983 012470 175637 001170
2984 012474 022737 000005 001170
2985 012502 170267 166756
2986 012506 001401
2987 012510 104012
2988

```

```

*****
;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, @SREGO ;@SREGO SHOULD =4
CMP #4, @SREGO ;DOES @SREGO =4
STFPS FPS ;GET STATUS
BEQ .+4 ;YES
ERROR 12 ;@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, @SREGO ;PUT ANS INTO @SREGO
CMP #5, @SREGO ;WERE THE TWO AC'S EQUAL
STFPS FPS ;GET STATUS
BEQ .+4 ;YES
ERROR 12 ;SREGO SHOULD = 5

```

```

2989
2990
2991
2992 012512 000004
2993 012514 012702 001200
2994 012520 177227 000052
2995 012524 174267 166450
2996 012530 177327 000025
2997 012534 171312
2998 012536 174712
2999 012540 171312
3000 012542 172312
3001 012544 012704 000025
3002 012550 173367 166424
3003 012554 005304
3004 012556 001374
3005 012560 175737 001170
3006 012564 170267 166674
3007 012570 022737 000052 001170
3008 012576 001401
3009 012600 104012

```

```

*****
;TEST 64 MULD 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
DIVD (R2),AC3 ;DIV 52 INTO RESULT IN AC3
MULD (R2),AC3 ;MUL 52X25
ADD (R2),AC3 ;ADD AC4 TO AC3 TO MAKE 53 TIMES
MOV #25,R4 ;SET UP COUNTER
SUBT: 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 12 ;SREG0 SHOULD BE 52

```

```

3010
3011
3012
3013 012602 000004
3014 012604 177027 025252
3015 012610 177127 000025
3016 012614 171100
3017 012616 174137 001170
3018 012622 012702 000024
3019 012626 174005
3020 012630 172005
3021 012632 005302
3022 012634 001375
3023 012636 174037 001200
3024 012642 170267 166616
3025 012646 173437 001170
3026 012652 170000
3027 012654 001401
3028 012656 104010
3029 012660 177327 000002
3030 012664 012702 000013
3031 012670 174304
3032 012672 171304
3033 012674 005302
3034 012676 001375
3035 012700 175703
3036 012702 010337 001170
3037 012706 022703 010000
3038 012712 170267 166546
3039 012716 001401
3040 012720 104012
3041 012722 177227 010000
3042 012726 177127 000002
3043 012732 012702 000013
3044 012736 174601
3045 012740 005302
3046 012742 001375
3047 012744 175603
3048 012746 010237 001170
3049 012752 022703 000002
3050 012756 170267 166502
3051 012762 001401
3052 012764 104012

```

```

*****
;TEST 65 EXERCISER TEST
*****
†ST65: 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,#$REGO ;SAV ANS IN #SREGO
MOV #24,R2 ;SET UP COUNT
STD AC0,AC5 ;PUT 25252 INTO AC5
AAD: ADDD AC5,AC0 ;ADD 25252 TO 25252
DEC R2 ;DO 25 TIMES
BNE AAD ;DONE?
STD AC0,#$REG4 ;LOAD TO PRINT
STFPS FPS ;GET STATUS
CMPD #SREGO,AC0 ;IS ANS CORRECT?
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,#$REGO ;NOW PUT IT INTO #SREGO
CMP #10000,R3 ;ANS SHOULD BE 10000
STFPS FPS ;GET STATUS
BEQ .+4 ;CONT. IF ANS IS CORRECT
ERROR 12 ;ANS SHOULD BE 10000
LDCID #10000,AC2 ;LOAD AC2 WITH 10000
LDCID #2,AC1 ;LOAD AC1 WITH A 2
MOV #13,R2 ;SET UP COUNTER
DDIV: DIVD AC1,AC2 ;DIVD 2 INTO 65536 16 TIMES
DEC R2 ;COUNT NO OF TIMES
BNE DDIV ;ARE WE DONE?
STCDI AC2,R3 ;STOR ANS INTO R3
MOV R2,#$REGO ;PUT IT INTO #SREGO FOR TYPING
CMP #2,R3 ;IS ANS CORRECT
STFPS FPS ;GET STATUS
BEQ .+4 ;ANS IS CORRECT
ERROR 12 ;SREGO SHOULD EQUAL =2

```

```

3053
3054
3055
3056 012766 000004
3057 012770 170127 047400
3058 012774 177327 000005
3059 013000 012702 001170
3060 013004 177227 000007
3061 013010 174322
3062 013012 173737 001170
3063 013016 170267 166442
3064 013022 170000
3065 013024 001401
3066 013026 104011
3067 013030 171242
3068 013032 012703 000006
3069 013036 173212
3070 013040 005303
3071 013042 001375
3072 013044 175637 001170
3073 013050 022737 000005 001170
3074 013056 170267 166402
3075 013062 001401
3076 013064 104012
3077
3078
3079
3080
3081 013066 000004
3082 013070 170127 047600
3083 013074 177027 000252
3084 013100 177227 052525
3085 013104 174204
3086 013106 171002
3087 013110 012702 000251
3088 013114 172204
3089 013116 005302
3090 013120 001375
3091 013122 174237 001170
3092 013126 170267 166332
3093 013132 173402
3094 013134 170000
3095 013136 001401
3096 013140 104010

```

```

*****
; *TEST 66 MODE ONE TEST
*****
↑ST66: SCOPE
LDFPS #47400
LDCIF #5, AC3 ; AC3=5
MOV #SREGO, R2 ; R2=SREGO
LDCIF #7, AC2 ; LOAD 7 INTO AC2
STF AC3, (R2)+ ; SREGO SHOULD =5=R2
CMPF @#SREGO, AC3 ; DOES @#SREGO = 5
STFPS FPS ; GET STATUS
CFCC
BEQ .+4 ; YES BRANCH
ERROR 11 ; SREGO 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, @#SREGO ; @#SREGO SHOULD =5
CMP #5, @#SREGO ; DOES @#SREGO =5
STFPS FPS ; GET STATUS
BEQ .+4 ; BRANCH IF YES
ERROR 12 ; ANS SHOULD =5

```

```

*****
; *TEST 67 ADD EXERCISER
*****
↑ST67: SCOPE
LDFPS #47600
LDCID #252, ACO ; LOAD ACO WITH 252
LDCID #52525, AC2 ; LOAD AC2 WITH 52525
STD AC2, AC4 ; AC4=52525
MULD AC2, ACO ; ACO=52525 X 252
MOV #251, R2 ; SET UP COUNTER
AADDM: ADD AC4, AC2 ; ADD 52525 TO 252
DEC R2 ; DO 252 TIMES
BNE AADDM ; DONE?
STD AC2, @#SREGO ; GET FOR PRINTING
STFPS FPS ; GET STATUS
CMPD AC2, ACO ; DOES AC2 = ACO?
CFCC
BEQ .+4 ; BRANCH IF EQUAL
ERROR 10 ; ANS SHOULD BE.....

```

```

3097
3098
3099
3100 013142 000004
3101 013144 170127 047600
3102 013150 177027 077777
3103 013154 177127 000000
3104 013160 174401
3105 013162 175437 001170
3106 013166 170267 166272
3107 013172 170337 001466
3108 013176 022737 077777 001170
3109 013204 001401
3110 013206 104012
3111 013210 022737 000004 001466
3112 013216 001401
3113 013220 104002
3114
3115
3116
3117
3118 013222 000004
3119 013224 170127 047400
3120 013230 170011
3121 013232 177027 077777
3122 013236 177127 002525
3123 013242 012702 000012
3124 013246 174401
3125 013250 171001
3126 013252 172001
3127 013254 173001
3128 013256 005302
3129 013260 001372
3130 013262 175437 001170
3131 013266 170267 166172
3132 013272 022737 077777 001170
3133 013300 001401
3134 013302 104012
3135 013304 172437 001556
3136 013310 012702 001170
3137 013314 171022
3138 013316 172537 001556
3139 013322 174122
3140 013324 172737 001200
3141 013330 170267 166130
3142 013334 173737 001170
3143 013340 170000
3144 013342 001401
3145 013344 104010
3146 013346 172537 001560
3147 013352 174142
3148 013354 172012
3149 013356 173042
3150 013360 173001
3151 013362 175437 001170
3152 013366 022737 000000 001170

```

```

*****
;TEST 70 TEST DIVIDE BY 0
*****
†ST70: 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 12 ;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
*****
†ST71: 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 12
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 10 ;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

```

H06

FPU INSTR EXERCISER MACY11 27(1006) 09-FEB-77 10:28 PAGE 74
DQFPCA.P11 09-FEB-77 10:26 T71 EXERCISER FOR ADD, SUBD, MULD AND DIVD

3153	013374	170267	166064		STFPS	FPS		;GET STATUS
3154	013400	001401			BEQ	.+4		
3155	013402	104012			ERROR	12		
3156	013404	170001		MORE:	SETF			;SET FLOATING MODE
3157	013406	177027	000525		LDCIF	#525, ACO		;LOAD ACP WITH 525
3158	013412	177127	000252		LDCIF	#252, AC1		;LOAD AC1 WITH 252
3159	013416	174104			STF	AC1, AC4		;LOAD AC4 WITH 252
3160	013420	172104			ADDF	AC4, AC1		;ADD 252 TO 252
3161	013422	172701			LDF	AC1, AC3		;PUT ANS IN AC3=524
3162	013424	173003			SUBF	AC3, ACO		;SUB 524 FROM 525
3163	013426	175037	001170		STEXP	ACO, @#\$REGO		;1 IN \$REGO
3164	013432	170267	166026		STFPS	FPS		;GET STATUS
3165	013436	022737	000001	001170	CMP	#1, @#\$REGO		;CORRECT ANS SHOULD BE 1
3166	013444	001401			BEQ	.+4		
3167	013446	104012			ERROR	12		;ANS SHOULD BE 5
3168	013450	177027	000021		LDCIF	#21, ACO		;LOAD ACO WITH 21
3169	013454	171000			MULF	ACO, ACO		;21 TIMES 21 ACO = 441
3170	013456	174427	040400		DIVF	#2, ACO		;DIV BY 2
3171	013462	012701	001552		MOV	#FLTONE+4, R1		;ADDR(DATA)+4
3172	013466	171441			MOVF	-(R1), ACO		;MUL 1 * 441.
3173	013470	170267	165770		STFPS	FPS		;GET STATUS
3174	013474	175537	001170		STCFI	AC1, @#\$REGO		;PUT \$REGO IN \$REGO
3175	013500	022737	000220	001170	CMP	#220, @#\$REGO		;IS IT EQUAL?
3176	013506	001401			BEQ	.+4		;YES
3177	013510	104012			ERROR	12		;SHOULD = 220
3178	013512	171427	041040		MOVF	#10, ACO		;GET FRACTION
3179	013516	175537	001170		STCFI	AC1, @#\$REGO		;PUT ACO INTO \$REGO
3180	013522	022737	000005	001170	CMP	#5, @#\$REGO		;\$REGO SHOULD = 5
3181	013530	001401			BEQ	.+4		
3182	013532	104012			ERROR	12		;\$REGO SHOULD = 5

;*****

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

3191	013534	105767	165756		TSTB	CLKPRS		;KW11-L PRESENT ?
3192	013540	100002			BPL	NKWL2		;NO
3193	013542	005037	177546		CLR	@#LKS		;YES, CLEAR IT
3194								
3195	013546	005767	165744	NKWL2:	TST	CLKPRS		;KW11-P PRESENT ?
3196	013552	100002			BPL	NKWP2		;NO
3197	013554	005037	172540		CLR	@#PLKS		;YES, CLEAR IT
3198	013560			NKWP2:				
3199								

```

3200
3201 ;*****
3202 .ENABL AMA ;ASSEMBEL ALL PC RELATIVE REFERENCES AS ABSOLUTE
3203 ;*****
3204
3205 ;*****
3206 .SBTTL SUB PASS END CONTROL
3207
3208 013560 000004 SCOPE ;CHECK FOR TEST ITERATIONS HERE
3209
3210 ;IF TEST ONLY EITHER HFP OR WFP, ENTER "EOP" ROUTINE DIRECTLY
3211
3212 ; IF IN ALTERNATE HFP/WFP MODE,
3213 ; COMPLEMENT FLAG<5>, HFP ENABLE BIT,
3214 ; ENTER EOP ROUTINE ONLY IF ABOUT TO TEST HFP NEXT,
3215 ; TESTING SEQUENCE IS: PASS#1 HFP SUB-PASS
3216 ; PASS#1 WFP SUB-PASS
3217 ; PASS#2 HFP SUB-PASS
3218 ; ...
3219
3220 013562 076600 000022 MED RWHAMI ;GET WHAMI INTO RO
3221 013566 032700 000020 BIT #BIT04,RO ;1=HFP PRESENT, 0=NONE
3222 013572 001423 BEQ $EOP ;EXIT IF NONE
3223
3224 013574 032777 000002 165344 BIT #SW01,@SWR ;1=HFP OR WFP TEST ONLY
3225 013602 001017 BNE $EOP ;0=ALTERNATE HFP AND WFP TESTS
3226
3227 013604 012701 010000 MOV #BIT12,R1 ;HFP PRESENT, AND IN ALTERNATE MODE;
3228 013610 076600 000144 MED RFLAG ;SO READ FLAGS
3229 013614 030100 BIT R1,RO ;COMPLEMENT FLAG<5>=BIT12=HFP ENABLE FLAG
3230 013616 001402 BEQ 1$
3231 013620 040100 BIC R1,RO ;CLEAR BIT 12
3232 013622 000401 BR 2$
3233 013624 050100 1$: BIS R1,RO ;SET BIT 12
3234 013626 076600 000344 2$: MED ,WFLAG ;REWRITE FLAGS
3235
3236 013632 030100 BIT R1,RO ;HFP OR WFP NEXT ?
3237 013634 001002 BNE $EOP ;IF HFP AGAIN, START NEW PASS
3238 013636 000137 002756 JMP @#SUBPAS ;IF WFP, NEXT SUBPASS
3239
3240 ;*****
3241 .SBTTL END OF PASS ROUTINE (MODIFIED SYSMAC)
3242
3243 ;*INCREMENT THE PASS NUMBER ($PASS)
3244 ;*IF SW<10>=0, DING BELL ON PASS END
3245 ;*IF THERE'S A MONITOR, GO TO IT
3246 ;* ELSE JUMP TO NEWPAS
3247
3248
3249
3250 $EOP:
3251 013642 005037 001104 CLR $ERFLG ;ZERO ERROR COUNT
3252 013646 005037 001102 CLP $YSTNM ;ZERO TEST NUMBER
3253 013652 005037 001230 CLR $TIMES ;ZERO NUMBER OF ITERATIONS
3254 013656 005237 001252 INC $PASS ;INCREMENT PASS COUNT
3255 013662 042737 100000 001252 BIC #100000,$PASS ; BUT NEVER LET IN GO NEGATIVE

```

3256	013670	005327			DEC	(PC)+		; PASS LOOP ?
3257	013672	000001			\$EOPCT: WORD	1		; YES
3258	013674	003021			BGT	\$DOAGN		; RESTORE COUNTER
3259	013676	012737			MOV	(PC)+, 2(PC)+		
3260	013700	000001			\$ENDCT: WORD	1		
3261	013702	013672			\$EOPCT			
3262	013704	032777	002000	165234	BIT	#SW10, 2SWR		; BELL ON PASS END ?
3263	013712	001002			BNE	\$GET42		; NO
3264	013714	104401	001234		TYPE	, \$BELL		; YES
3265								
3266	013720	013700	000042		\$GET42: MOV	2#42, R0		; GET MONITOR ADDRESS
3267	013724	001405			BEG	\$DOAGN		; NO MONITOR
3268	013726	000005			RESET			; CLEAR WORLD
3269								
3270	013730	004710			\$ENDAD: JSR	PC, (R0)		; GO TO MONITOR
3271	013732	000240			NOP			
3272	013734	000240			NOP			; RESERVED FOR ACT11
3273	013736	000240			NOP			
3274								
3275	013740	000137	002720		\$DOAGN: JMP	2#NEWPAS		; RETURN
3276								
3277								
3278								

K06

```

.SBTTL TRAP/FEC TESTER SUBR
3279
3280
3281 013744 013746 000244 TRPTST: MOV 2#FPPVEC, -(SP) ; SAVE OLD FPPVEC PC
3282 013750 013746 000246 MOV 2#FPPVEC+2, -(SP) ; AND PS
3283 013754 012737 014052 000244 MOV #FPPTRP, 2#FPPVEC ; NEW VECTOR PC FOR TEST
3284 013762 016537 000002 000246 MOV 2(R5), 2#FPPVEC+2 ; NEW VECTOR PS FOR TEST
3285 013770 012700 000006 MOV #6, R0 ; LOAD STMPD-5
3286 013774 010501 MOV R5, R1 ; WITH TEST DATA SETS
3287 013776 012702 001210 MOV #STMPD, R2 ; FOR DISPLAY LATER
3288 014002 012122 MOV (R1)+, (R2)+
3289 014004 077002 SOB R0, -2
3290 014006 012737 014016 001114 MOV #FLPERR, $LPERR ; ERROR LOOP TO HERE
3291 014014 010602 MOV SP, R2 ; SAVE GOOD SP
3292
3293 014016 012700 000010 FLPERR: MOV #8, R0 ; DELAY CTR, 8(10) INSTR EXEC
3294 014022 010206 MOV R2, SP ; RESET TO GOOD SP
3295 014024 170165 000006 LDFPS 6(R5) ; SET UP INITIAL FPS
3296 014030 011537 177776 MOV (R5), 2#PSW ; SET UP INITIAL PSW
3297
3298
3299
3300
3301
3302
3303 014034 000114 JMP (R4) ; GO TEST W/O USING STACK
3304
3305
3306 014036 032765 040000 000012 FPPNTP: BIT #BIT14, 12(R5) ; NO TRAP EXPECTED ?
3307 014044 001455 BEQ FCONT1 ; BR IF YES
3308 014046 104014 ERROR 14 ; DIDNT TRAP, SHOULD HAVE
3309
3310 014050 000453 BR FCONT1 ; CONTINUE WITH TEST
3311
3312
3313 014052 013737 177776 001512 FPPTRP: BIT #BIT14, 12(R5) ; TRAP EXPECTED ?
3314 014060 011637 001506 MOV 2#PSW, NPSLOD ; SAVE LOADED PSW
3315 014064 016637 000002 001510 MOV (SP), OPCRCV ; GET PUSHED PC
3316 014072 010637 001514 MOV 2(SP), OPSRCV ; GET PUSHED PS
3317 014076 012716 014104 MOV SP, OSPRCV ; GET SP AFTER TRAP
3318 014102 000002 RTI ; FUDGE RETURN
3319
3320 014104 032765 040000 000012 2$: BIT #BIT14, 12(R5) ; TRAP EXPECTED ?
3321 014112 001001 BNE 1$ ; BR IF YES
3322 014114 104015 ERROR 15 ; TRAPPED, SHOULDNT HAVE
3323
3324 014116 010237 001504 1$: MOV R2, EXPSP ; GENERATE EXPECTED SP CONTENTS
3325 014122 162737 000004 001504 SUB #4, EXPSP ; AFTER TRAP
3326 014130 023737 001514 001504 CMP OSPRCV, EXPSP ; IS SP WHERE IT SHOULD BE?
3327 014136 001401 BEQ 64$ ;
3328 014140 104016 ERROR 16 ; NOT EQUAL, SIGNAL ERROR
3329 014142 64$:
3330
3331 014142 023737 001502 001506 CMP EXPRTS, OPCRCV ; CHECK STORED PC IS OK
3332 014150 001401 BEQ 65$ ;
3333 014152 104017 ERROR 17 ; NOT EQUAL, SIGNAL ERROR
3334 014154 65$:

```

```

3335 014154 026537 000004 001510      CMP      4(R5),OPSRCV      ; CHECK STORED PS IS OK
3336 014162 001401                      BEQ      66$              ;
3337 014164 104020                      ERROR    20              ; NOT EQUAL, SIGNAL ERROR
3338 014166                                66$:
3339
3340 014166 026537 000002 001512      CMP      2(R5),NPSLOD     ; CHECK LOADED PS IS OK
3341 014174 001401                      BEQ      67$              ;
3342 014176 104021                      ERROR    21              ; NOT EQUAL, SIGNAL ERROR
3343 014200                                67$:
3344
3345      ;----- CONTINUE -----
3346 014200 170237 001464      FCONT1: STFPS   FPS      ; STORE FPS AFTER
3347 014204 170337 001466      STST    FEC      ; STORE FEC/FEA AFTER
3348
3349 014210 023765 001464 000010      CMP      FPS,10(R5)      ; CHECK FPS
3350 014216 001401                      BEQ      65$              ; FPS IS OK
3351 014220 104023                      ERROR    23              ; FPS BAD
3352 014222 005765 000012      65$:  TST      12(R5)      ; DOES FEC/FEA APPLY?
3353 014226 100014                      BPL      66$              ; NO - SKIP TEST
3354 014230 013737 001500 001500      MOV      EXPFEA,EXPFEA    ; GET EXPECTED FEA
3355 014236 123765 001466 000012      CMPB     FEC,12(R5)      ; COMPARE FEC-S
3356 014244 001004                      BNE      64$              ; NOT EQUAL
3357 014246 023737 001470 001500      CMP      FEA,EXPFEA      ; COMPARE FEA-S
3358 014254 001401                      BEQ      66$              ; FEC, FEA OK
3359 014256 104022                      ERROR    22              ; FEC OR FEA ARE BAD
3360 014260                                64$:
3361                                66$:
3362 014260 010206                      MOV      R2,SP           ; RESTORE GOOD SP AFTER TEST, IN CASE
3363
3364 014262 012637 000246                      MOV      (SP)+,0#FPPVEC+2 ; RESET STANDARD FPP VECTOR
3365 014266 012637 000244                      MOV      (SP)+,0#FPPVEC  ;
3366
3367 014272 005037 177776                      CLR      0#PSW           ; SETUP FOR INTERRUPT ENABLE, PRO
3368 014276 000207                      RTS      PC              ; RETURN TO TEST CALLER
3369
3370      ;----- LOCAL FEC-CODE GENERATING SUBR -----
3371
3372 014300      FEC02:
3373 014300 170406      FII02: 170406           ; ILLEGAL FPP INSTR
3374 014302 077001      NXT02: SOB   RO         ; DELAY FOR FPP TO SETTLE
3375 014304 000137 014036      JMP     FPPNTP         ;
3376
3377 014310 172437 014442      FEC04: LDF   FLT001,AC0  ;
3378 014314 174437 014432      FII04: DIVF  FLTZER,AC0  ; X/0.0 = ?
3379 014320 077001      NXT04: SOB   RO         ; DELAY FOR FPP TO SETTLE
3380 014322 000137 014036      JMP     FPPNTP         ;
3381
3382 014326 172537 014442      FEC06: LDF   FLT001,AC1  ;
3383 014332 175537 001170      FII06: STCFI AC1,$REGO   ; FLT001 > LGST I
3384 014336 077001      NXT06: SOB   RO         ; DELAY FOR FPP TO SETTLE
3385 014340 000137 014036      JMP     FPPNTP         ;
3386
3387 014344 172637 014442      FEC10: LDF   FLT001,AC2  ;
3388 014350 172737 014446      LDF   FLT002,AC3        ;
3389 014354 171203      FII10: MULF  AC3,AC2     ; LG * LG = OVFLW
3390 014356 077001      NXT10: SOB   RO, .      ; DELAY FOR FPP TO SETTLE
  
```

```

3391 014360 000137 014036          JMP      FPPNTP          ;
3392
3393 014364 172637 014452          FEC12:  LDF      FLT003,AC2      ;
3394 014370 171237 014456          FII12:  MULF     FLT004,AC2      ; SM * SM = UNDFLW
3395 014374 077001                    NXT12:  SOB      RO              ; DELAY FOR FPP TO SETTLE
3396 014376 000137 014036          JMP      FPPNTP          ;
3397
3398 014402                    FEC14:
3399 014402 172737 014436          FII14:  LDF      FLTMZR,AC3      ; -0.0 -> AC3
3400 014406 077001                    NXT14:  SOB      RO              ; DELAY FOR FPP TO SETTLE
3401 014410 000137 014036          JMP      FPPNTP          ;
3402
3403 014414 012703 000342          FEC16:  MOV      #342,R3        ; HFP U-ADDR FOR "LDF/D" INSTR
3404 014420 170003                    LDU8
3405 014422 172400                    FII16:  LDF      ACO,ACO        ; U-BREAK TRAP
3406 014424 077001                    NXT16:  SOB      RO              ; DELAY FOR FPP TO SETTLE
3407 014426 000137 014036          JMP      FPPNTP          ;
3408
3409                    ----- CONSTANTS -----
3410 014432 000000 000000          FLTZER:  .WORD   000000,000000  ; +0.0
3411 014436 100000 000000          FLTMZR:  .WORD   100000,000000  ; -0.0
3412 014442 044000 000000          FLT001:  .WORD   044000,000000  ; .1E+20 = 65536
3413 014446 074377 177777          FLT002:  .WORD   074377,177777  ; .111 . . . 1E+161
3414                    ; NOTE FLT001*FLT002 = .111 . . . 1E+200 (OVERFLOW)
3415 014452 030000 000000          FLT003:  .WORD   030000,000000  ; .1E-40
3416 014456 010000 000000          FLT004:  .WORD   010000,000000  ; .1E-140
3417                    ; NOTE FLT003*FLT004 = .1E-201 (UNDERFLOW)

```

N06

FPU INSTR EXERCISER MACY11 27(1006) 09-FEB-77 10:28 PAGE 80
 DIFFPCA.P11 09-FEB-77 10:26

LINE CLOCK INTERRUPT SERVICE ROUTINES

3418
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449

```

014462 005037 177546
014466 005237 001520
014472 042737 100000 001520
014500 052737 000100 177546
014506 000002

014510 005037 172540
014514 005237 001522
014520 042737 100000 001522

014526 005046
014530 006237 001524
014534 006116
014536 032737 000004 001524
014544 001403
014546 005116
014550 042716 177776
014554 000316
014556 006316
014560 052637 001524
014564 042737 176000 001524
014572 001003
014574 012737 001777 001524

014602 013737 001524 172542
014610 052737 000101 172540
014616 000002
  
```

.SBTTL LINE CLOCK INTERRUPT SERVICE ROUTINES

```

IKW11L: CLR    2#PLKS            ; CLEAR STATUS
          INC    KW11LC           ; BUMP COUNTER
          BIC    #BIT15, KW11LC   ; NEVER OVERFLOW
          BIS    #BIT6, 2#PLKS   ; RESTART CLOCK
          RTI                    ; AND RETURN

IKW11P: CLR    2#PLKS            ; CLEAR STATUS
          INC    KW11PC           ; BUMP COUNTER
          BIC    #BIT15, KW11PC   ; NEVER OVERFLOW

          ; THE NEXT 13 INSTRUCTIONS GENERATE A 10 BIT RANDOM
          ; INTEGER FROM 0001-1777 INCLUSIVE TO FEED INTO THE
          ; KW11-P COUNT SET REGISTER.
          CLR    -(SP)            ; TEMP STORAGE FOR SHIFTED OUT BIT
          ASR    KW11PR           ; SFT OLD CONTENTS RITE 1; LOB INTO C
          ROL    (SP)            ; PUT OLD LOB ON STACK
          BIT    #BIT2, KW11PR   ; WAS OTHER BIT ON ?
          BEQ    1$              ; NO
          COM    (SP)            ; YES, COMPLEMENT OLD LOB
          BIC    #177776, (SP)   ; ELIMINATE JUNK
          SWAB    (SP)           ; FAST SHIFT LOB TO BIT9 POSITION
          ASL    (SP)            ;
          BIS    (SP)+, KW11PR   ; PUT NEW HOB INTO COUNTER
          BIC    #176000, KW11PR ; NO HIGH ORDER JUNK
          BNE    2$              ; ZERO IS AN ILLEGAL STATE,
          MOV    #1777, KW11PR   ; SO RESET TO START

1$:
          MOV    KW11PR, 2#PLKR   ; SET NEW COUNT
          BIS    #BIT6+BIT0, 2#PLKS ; RESTART CLOCK
          RTI                    ; AND RETURN

2$:
          MOV    KW11PR, 2#PLKR   ; SET NEW COUNT
          BIS    #BIT6+BIT0, 2#PLKS ; RESTART CLOCK
          RTI                    ; AND RETURN
  
```



```

3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3480
3481
3482
3483 014704
3484 014704
3485 014704 032777 040000 164234
3486 014712 001114
3487
3488 014714 000416
3489
3490 014716 013746 000004
3491 014722 012737 014742 000004
3492 014730 005737 177060
3493 014734 012637 000004
3494 014740 000463
3495 014742 022626
3496 014744 012637 000004
3497 014750 000423
3498 014752
3499 014752 032777 000400 164166
3500 014760 001404
3501 014762 023737 001112 001102
3502 014770 001465
3503 014772 005737 001104
3504 014776 001421
3505 015000 023737 001122 001104
3506 015006 101015
3507 015010 032777 001000 164130
3508 015016 001404
3509 015020 013737 001114 001110
3510 015026 000446
3511 015030 005037 001104
3512 015034 005037 001230
3513 015040 000415
3514 015042 032777 004000 164076
3515 015050 001011
3516 015052 005737 001252
3517 015056 001406
3518 015060 005237 001106
3519 015064 023737 001230 001106
3520 015072 002024
3521 015074 012737 000001 001106
3522 015102 013737 015160 001230
3523 015110 005237 001102
3524 015114 013737 001102 001250
3525 015122 011637 001110

```

```

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

$SCOPE:
64$:
1$: BIT #BIT14,$SWR ;;LOOP ON PRESENT TEST?
BNE $OVER ;;YES IF SW14=1
*****START OF CODE FOR THE XOR TESTER*****
$XTSTR: BR 6$
MOV @ERRVEC, -(SP) ;;IF RUNNING ON THE "XOR" TESTER CHANGE
MOV #5$,@ERRVEC ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
TST @177060 ;;SAVE THE CONTENTS OF THE ERROR VECTOR
MOV (SP)+,@ERRVEC ;;SET FOR TIMEOUT
BR $SVLAD ;;TIME OUT ON XOR?
5$: CMP (SP)+,(SP)+ ;;RESTORE THE ERROR VECTOR
MOV (SP)+,@ERRVEC ;;GO TO THE NEXT TEST
BR 7$ ;;CLEAR THE STACK AFTER A TIME OUT
6$: *****END OF CODE FOR THE XOR TESTER*****
BIT #BIT08,$SWR ;;LOOP ON SPEC. TEST?
BEQ 2$ ;;BR IF NO
CMP $LPTST,$STNM ;;ON THE RIGHT TEST?
BEQ $OVER ;;BR IF YES
2$: TST $ERFLG ;;HAS AN ERROR OCCURRED?
BEQ 3$ ;;BR IF NO
CMP $ERMAX,$ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
BHI 3$ ;;BR IF NO
BIT #BIT09,$SWR ;;LOOP ON ERROR?
BEQ 4$ ;;BR IF NO
7$: MOV $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
BR $OVER
4$: CLR $ERFLG ;;ZERO THE ERROR FLAG
CLR $TIMES ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
BR 1$ ;;ESCAPE TO THE NEXT TEST
3$: BIT #BIT11,$SWR ;;INHIBIT ITERATIONS?
BNE 1$ ;;BR IF YES
TST $PASS ;;IF FIRST PASS OF PROGRAM
BEQ 1$ ;;INHIBIT ITERATIONS
INC $ICNT ;;INCREMENT ITERATION COUNT
CMP $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
BGE $OVER ;;BR IF MORE ITERATION REQUIRED
1$: MOV #1,$ICNT ;;REINITIALIZE THE ITERATION COUNTER
MOV $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
$SVLAD: INC $STNM ;;COUNT TEST NUMBERS
MOV $STNM,$STESTN ;;SET TEST NUMBER IN APT MAILBOX
MOV (SP),$LPADR ;;SAVE SCOPE LOOP ADDRESS

```

3526	015126	011637	001114		MOV	(SP), \$LPERR	:: SAVE ERROR LOOP ADDRESS
3527	015132	005037	001232		CLR	\$ESCAPE	:: CLEAR THE ESCAPE FROM ERROR ADDRESS
3528	015136	012737	000001	001122	MOV	#1, \$ERMAX	:: ONLY ALLOW ONE(1) ERROR ON NEXT TEST
3529	015144	013777	001102	163776	\$OVER: MOV	\$TSTNM, @DISPLAY	:: DISPLAY TEST NUMBER
3530	015152	013716	001110		MOV	\$LPADR, (SP)	:: FUDGE RETURN ADDRESS
3531	015156	000002			RTI		:: FIXES PS
3532	015160	003720			\$MXCNT: 2000.		:: MAX. NUMBER OF ITERATIONS

;

```

3533 .SBTTL ERROR HANDLER ROUTINE
3534
3535 ;*****
3536 ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
3537 ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
3538 ;*AND GO TO $TYPERR ON ERROR
3539 ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
3540 ;*SW15=1 HALT ON ERROR
3541 ;*SW13=1 INHIBIT ERROR TYPEOUTS
3542 ;*SW10=1 BELL ON ERROR
3543 ;*SW09=1 LOOP ON ERROR
3544 ;*CALL
3545 ;* ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER
3546
3547 $ERROR:
3548 015162 010037 001526 MOV R0, EREG0 ; DISPLAY R0
3549 015166 010137 001530 MOV R1, EREG1 ; R1
3550 015172 010237 001532 MOV R2, EREG2 ; R2
3551 015176 010337 001534 MOV R3, EREG3 ; R3
3552 015202 010437 001536 MOV R4, EREG4 ; R4
3553 015206 010537 001540 MOV R5, EREG5 ; R5
3554 015212 010637 001542 MOV R6, EREG6 ; GET R6(SP) BEFORE TRAP
3555 015216 062737 000004 001542 ADD #4, EREG6
3556 015224 011637 001544 MOV (SP), EREG7 ; PC -> ERROR CALL INSTR
3557 015230 005237 001104 75: INC $ERFLG ; SET THE ERROR FLAG
3558 015234 001775 BEQ 75 ; DON'T LET THE FLAG GO TO ZERO
3559 015236 013777 001102 163704 MOV $STNM, @DISPLAY ; DISPLAY TEST NUMBER
3560 015244 032777 002000 163674 BIT #BIT10, @SWR ; BELL ON ERROR?
3561 015252 001402 BEQ 15 ; NO - SKIP
3562 015254 104401 001234 TYPE $BELL ; RING BELL
3563 015260 005237 001116 15: INC $ERTL ; COUNT THE NUMBER OF ERRORS
3564 015264 011637 001124 MOV (SP), $ERRPC ; GET ADDRESS OF ERROR INSTRUCTION
3565 015270 162737 000002 001124 SUB #2, $ERRPC
3566 015276 117737 163622 001120 MOVB @ERRPC, $ITEMB ; STRIP AND SAVE THE ERROR ITEM CODE
3567 015304 032777 020000 163634 BIT #BIT13, @SWR ; SKIP TYPEOUT IF SET
3568 015312 001004 BNE 20$ ; SKIP TYPEOUTS
3569 015314 004737 015424 JSR PC, $TYPERR ; GO TO USER ERROR ROUTINE
3570 015320 104401 001241 TYPE , $CRLF
3571 015324 20$:
3572 015324 122737 000001 001264 CMPB #APTENV, $ENV ; RUNNING IN APT MODE
3573 015332 001007 BNE 25 ; NO SKIP APT ERROR REPORT
3574 015334 113737 001120 015346 MOVB $ITEMB, 21$ ; SET ITEM NUMBER AS ERROR NUMBER
3575 015342 004737 016140 JSR PC, $ATY4 ; REPORT FATAL ERROR TO APT
3576 015346 000 21$: .BYTE 0
3577 015347 000 .BYTE 0
3578 015350 000777 22$: BR 22$ ; APT ERROR LOOP
3579 015352 005777 163570 25: TST @SWR ; HALT ON ERROR
3580 015356 100001 BPL 35 ; SKIP IF CONTINUE
3581 015360 000000 HALT ; HALT ON ERROR!
3582 015362 032777 001000 163556 35: BIT #BIT09, @SWR ; LOOP ON ERROR SWITCH SET?
3583 015370 001402 BEQ 45 ; BR IF NO
3584 015372 013716 001114 MOV $LPERR, (SP) ; FUDGE RETURN FOR LOOPING
3585 015376 005737 001232 45: TST $ESCAPE ; CHECK FOR AN ESCAPE ADDRESS
3586 015402 001402 BEQ 55 ; BR IF NONE
3587 015404 013716 001232 55: MOV $ESCAPE, (SP) ; FUDGE RETURN ADDRESS FOR ESCAPE
3588 015410

```

F07

FPU INSTR EXERCISER MACY11 27(1006) 09-FEB-77 10:28 PAGE 85
DQFPCA.P11 09-FEB-77 10:26 ERROR HANDLER ROUTINE

3589	015410	022737	013730	000042	CMP	#SENDAD,2#42	::ACT-11 AUTO-ACCEPT?
3590	015416	001001			BNE	6S	::BRANCH IF NO
3591	015420	000000			HALT		::YES
3592	015422			6S:			
3593	015422	000002		64S:	RTI		:RETURN

```

3594
3595
3596
3597
3598
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608 015424
3609 015424 104401
3610 015426 001241
3611 015430 010046
3612 015432 010146
3613 015434 005000
3614 015436 153700 001120
3615 015442 001004
3616
3617 015444 013746 001124
3618 015450 104402
3619 015452 000452
3620 015454 005300
3621 015456 006300
3622 015460 010001
3623 015462 006300
3624 015464 060100
3625 015466 062700 001274
3626 015472 012037 015502
3627 015476 001404
3628 015500 104401
3629 015502 000000
3630 015504 104401 001241
3631 015510 104401 015620
3632 015514 012037 015524
3633 015520 001402
3634 015522 104401
3635 015524 000000
3636 015526 104401 001241
3637 015532 017746 000054
3638 015536 104402
3639 015540 104401 015616
3640 015544 017746 000044
3641 015550 104402
3642 015552 104401 015616
3643 015556 011000
3644 015560 001407
3645 015562 013046
3646 015564 104402
3647 015566 005710
3648 015570 001403
3649 015572 104401 015616

```

```

;*****
.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 HCRIZONTAL TAB STOP. ALSO, THE DATA FORMAT
;*POINTER HAS BEEN ELIMINATED FROM THE ERROR VECTOR. THIS ROUTINE
;*ALSO ALWAYS PRINTS $TESTN AND SERRPC AS THE FIRST TWO DATA ELEMENTS
;*(WITH APPROPRIATE HEADERS).

$TYPERR:
HOTWRM: TYPE TYPE "HOT" OR "WARM"
        .WORD SCRLF PTR TO MESSAGE
        MOV RO,-(SP) SAVE RO
        MOV R1,-(SP) SAVE R1
        CLR RO PICKUP ITEM INDEX
        BISB @#$ITEMB,RO
        BNE 1$ IF ITEM NUMBER FROM ERROR 0,
                JUST TYPE PC OF ERPOR
                GET ERROR PC FOR TYPEOUT
                TYPE OCTAL, ALL DIGITS
                EXIT
                ADJUST ERROR # FOR TABLE INDEX
                OF 6 BYTES/ENTRY

        MOV SERRPC,-(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 ,SCRLF
    "TEST # ERR PC" HEADER
    PICKUP "DATA HEADER" PTR
    SKIP TYPEOUT IF NULL
    TYPE "DATA HEADER"
    "DATA HEADER" PTR HERE
    CR & LF

3$: TYPE 11$
    MOV (RO)+,4$
    BEQ 5$

4$: .WORD 0
5$: TYPE SCRLF
    MOV @8$,-(SP)
    ($TESTN)
    OCTAL W/ LEADING ZEROS
    <HT>
    MOV @9$,-(SP)
    ($SERRPC)
    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

6$: MOV @2(R0)+,-(SP)
    TYPOC
    TST (RO)
    BEQ 7$
    TYPE ,10$

```

H07

FPU INSTR EXERCISER
DQFPCA.P11 09-FEB-77

MACY11 27(1006)
10:26

09-FEB-77 10:28 PAGE 87
ERROR MESSAGE TYPEOUT ROUTINE (MODIFIED SYSMAC)

3650	015576	000771		
3651	015600	012601		
3652	015602	012600		
3653	015604	104401	001241	
3654	015610	000207		
3655	015612	001250		
3656	015614	001124		
3657	015616	000011		
3658	015620	042524	052123	021440
3659	015626	042411	051122	050040
3660	015634	004503	000	
3661		015640		

7S:	BR	6S	:	LOOP ON DATA TABLE VECTOR
	MOV	(SP)+,R1	:	RESTORE R1
	MOV	(SP)+,R0	:	RESTORE R0
	TYPE	\$CRLF	:	CR & LF
	RTS	PC	:	RETURN
8S:	.WORD	\$TESTN	:	
9S:	.WORD	\$ERRPC	:	
10S:	.ASCIZ	<11>	:	<HT>
11S:	.ASCIZ	"TEST # ERR PC	:	
	.EVEN			

.SBTTL TYPE ROUTINE

3662
3663
3664
3665
3666
3667
3668
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3710
3711
3712
3713
3714
3715
3716
3717

015640 105737 001165
015644 100002
015646 000000
015650 000430
015652 010046
015654 017600 000002
015660 122737 000001 001264
015666 001011
015670 132737 000100 001265
015676 001405
015700 010037 015710
015704 004737 016130
015710 000000
015712 132737 000040 001265
015720 001003
015722 112046
015724 001005
015726 005726
015730 012600
015732 062716 000002
015736 000002
015740 122716 000011
015744 001430
015746 122716 000200
015752 001006
015754 005726
015756 104401
015760 001241
015762 105037 016116
015766 000755
015770 004737 016052
015774 123726 001164
016000 001350
016002 013746 001162
016006 105366 000001
016012 002770
016014 004737 016052
016020 105337 016116

```
*****
*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
*
$TYPE: TSTB $STPFLG ;; IS THERE A TERMINAL?
BPL 1$ ;; BR IF YES
HALT ;; HALT HERE IF NO TERMINAL
BR 3$ ;; LEAVE
1$: MOV RO, -(SP) ;; SAVE RO
MOV 02(SP), RO ;; GET ADDRESS OF ASCIZ STRING
CMPB #APTENV, $ENV ;; RUNNING IN APT MODE
BNE 62$ ;; NO GO CHECK FOR APT CONSOLE
BITB #APTPOOL, $ENVM ;; SPOOL MESSAGE TO APT
BEQ 62$ ;; NO GO CHECK FOR CONSOLE
MOV RO, 61$ ;; SETUP MESSAGE ADDRESS FOR APT
JSR PC, $ATY3 ;; SPOOL MESSAGE TO APT
61$: .WORD 0 ;; MESSAGE ADDRESS
62$: BITB #APTCSUP, $ENVM ;; APT CONSOLE SUPPRESSED
BNE 60$ ;; YES, SKIP TYPE OUT
2$: MOVB (RO)+, -(SP) ;; PUSH CHARACTER TO BE TYPED ONTO STACK
BNE 4$ ;; BR IF IT ISN'T THE TERMINATOR
TST (SP)+ ;; IF TERMINATOR POP IT OFF THE STACK
60$: MOV (SP)+, RO ;; RESTORE RO
3$: ADD #2, (SP) ;; ADJUST RETURN PC
4$: CMPB #HT, (SP) ;; BRANCH IF <HT>
BEQ 8$
CMPB #CRLF, (SP) ;; BRANCH IF NOT <CRLF>
BNE 5$
TST (SP)+ ;; POP <CR><LF> EQUIV
TYPE ;; TYPE A CR AND LF
$CRLF
CLRB $CHARCNT ;; CLEAR CHARACTER COUNT
BR 2$ ;; GET NEXT CHARACTER
5$: JSR PC, $TYPEC ;; GO TYPE THIS CHARACTER
6$: CMPB $FILLC, (SP)+ ;; IS IT TIME FOR FILLER CHARS.?
BNE 2$ ;; IF NO GO GET NEXT CHAR.
MOV $NULL, -(SP) ;; GET # OF FILLER CHARS. NEEDED
AND THE NULL CHAR.
7$: DECB 1(SP) ;; DOES A NULL NEED TO BE TYPED?
BLT 6$ ;; BR IF NO--GO POP THE NULL OFF OF STACK
JSR PC, $TYPEC ;; GO TYPE A NULL
DECB $CHARCNT ;; DO NOT COUNT AS A COUNT
```

```

3718 016024 000770          BR      7$          ;;LOOP
3719
3720          ;HORIZONTAL TAB PROCESSOR
3721
3722 016026 112716 000040 8$:      MOVB      #' (SP)          ;; REPLACE TAB WITH SPACE
3723 016032 004737 016052 9$:      JSR      PC,$TYPEC          ;; TYPE A SPACE
3724 016036 132737 000007 016116      BITB      #7,$SCHARCNT          ;; BRANCH IF NOT AT
3725 016044 001372          BNE      9$          ;; TAB STOP
3726 016046 005726          TST      (SP)+          ;; POP SPACE OFF STACK
3727 016050 000724          BR      2$          ;; GET NEXT CHARACTER
3728 016052 105777 163100 $TYPEC: TSTB      @STPS          ;; WAIT UNTIL PRINTER IS READY
3729 016056 100375          BPL      $TYPEC
3730 016060 116677 000002 163072      MOVB      2(SP),@STPB          ;; LOAD CHAR TO BE TYPED INTO DATA REG.
3731 016066 122766 000015 000002      CMPB      #CR,2(SP)          ;; IS CHARACTER A CARRIAGE RETURN?
3732 016074 001003          BNE      1$          ;; BRANCH IF NO
3733 016076 105037 016116      CLRB      $SCHARCNT          ;; YES--CLEAR CHARACTER COUNT
3734 016102 000406          BR      $TYPEX          ;; EXIT
3735 016104 122766 000012 000002 1$:      CMPB      #LF,2(SP)          ;; IS CHARACTER A LINE FEED?
3736 016112 001402          BEQ      $TYPEX          ;; BRANCH IF YES
3737 016114 105227          INCB      (PC)+          ;; COUNT THE CHARACTER
3738 016116 000000      $SCHARCNT: .WORD 0          ;; CHARACTER COUNT STORAGE
3739 016120 000207      $TYPEX:  RTS      PC
3740

```

K07

FPU INSTR EXERCISER MACY11 27(1006) 09-FEB-77 10:28 PAGE 90
DQFPCA.P11 09-FEB-77 10:26 APT COMMUNICATIONS ROUTINE

.SBTTL APT COMMUNICATIONS ROUTINE

```

3741
3742
3743
3744 016122 112737 000001 016366 $ATY1: MOVB #1,$FFLG ;; TO REPORT FATAL ERROR
3745 016130 112737 000001 016364 $ATY3: MOVB #1,$MFLG ;; TO TYPE A MESSAGE
3746 016136 000403
3747 016140 112737 000001 016366 $ATY4: MOVB #1,$FFLG ;; TO ONLY REPORT FATAL ERROR
3748 016146
3749 016146 010046
3750 016150 010146
3751 016152 105737 016364 TSTB $MFLG ;; SHOULD TYPE A MESSAGE?
3752 016156 001450 BEQ 5$ ;; IF NOT: BR
3753 016160 122737 000001 001264 CMPB $APTENV,$ENV ;; OPERATING UNDER APT?
3754 016166 001031 BNE 3$ ;; IF NOT: BR
3755 016170 132737 000100 001265 BITB $APTPOOL,$ENVM ;; SHOULD SPOOL MESSAGES?
3756 016176 001425 BEQ 3$ ;; IF NOT: BR
3757 016200 017600 000004 MOV #4(SP),RO ;; GET MESSAGE ADDR.
3758 016204 062766 000002 000004 ADD #2,4(SP) ;; BUMP RETURN ADDR.
3759 016212 005737 001244 1$: TST $MSGTYPE ;; SEE IF DONE W/ LAST XMISSION?
3760 016216 001375 BNE 1$ ;; IF NOT: WAIT
3761 016220 010037 001260 MOV RO,$MSGAD ;; PUT ADDR IN MAILBOX
3762 016224 105720 2$: TSTB (RO)+ ;; FIND END OF MESSAGE
3763 016226 001376 BNE 2$
3764 016230 163700 001260 SUB $MSGAD,RO ;; SUB START OF MESSAGE
3765 016234 006200 ASR RO ;; GET MESSAGE LNTH IN WORDS
3766 016236 010037 001262 MOV RO,$MSGLGT ;; PUT LENGTH IN MAILBOX
3767 016242 012737 000004 001244 MOV #4,$MSGTYPE ;; TELL APT TO TAKE MSG.
3768 016250 000413 BR 5$
3769 016252 017637 000004 016276 3$: MOV #4(SP),4$ ;; PUT MSG ADDR IN JSR LINKAGE
3770 016260 062766 000002 000004 ADD #2,4(SP) ;; BUMP RETURN ADDRESS
3771 016266 013746 177776 MOV 177776,-(SP) ;; PUSH 177776 ON STACK
3772 016272 004737 015640 JSR PC,$TYPE ;; CALL TYPE MACRO
3773 016276 000000 4$: .WORD 0
3774 016300 5$:
3775 016300 105737 016366 10$: TSTB $FFLG ;; SHOULD REPORT FATAL ERROR?
3776 016304 001416 BEQ 12$ ;; IF NOT: BR
3777 016306 005737 001264 TST $ENV ;; RUNNING UNDER APT?
3778 016312 001413 BEQ 12$ ;; IF NOT: BR
3779 016314 005737 001244 11$: TST $MSGTYPE ;; FINISHED LAST MESSAGE?
3780 016320 001375 BNE 11$ ;; IF NOT: WAIT
3781 016322 017637 000004 001246 MOV #4(SP),$FATAL ;; GET ERROR #
3782 016330 062766 000002 000004 ADD #2,4(SP) ;; BUMP RETURN ADDR.
3783 016336 005237 001244 INC $MSGTYPE ;; TELL APT TO TAKE ERROR
3784 016342 105037 016366 12$: CLRB $FFLG ;; CLEAR FATAL FLAG
3785 016346 105037 016365 CLRB $LFLG ;; CLEAR LOG FLAG
3786 016352 105037 016364 CLRB $MFLG ;; CLEAR MESSAGE FLAG
3787 016356 012601 MOV (SP)+,R1 ;; POP STACK INTO R1
3788 016360 012600 MOV (SP)+,RO ;; POP STACK INTO RO
3789 016362 000207 RTS PC ;; RETURN
3790 016364 000 $MFLG: .BYTE 0 ;; MESSG. FLAG
3791 016365 000 $LFLG: .BYTE 0 ;; LOG FLAG
3792 016366 000 $FFLG: .BYTE 0 ;; FATAL FLAG
3793 016370 .EVEN
3794 000200 APTSIZE=200
3795 000001 APTENV=001
3796 000100 APTPOOL=100

```

L07

FPU INSTR EXERCISER MACY11 27(1006) 09-FEB-77 10:28 PAGE 91
DQFPCA.P11 09-FEB-77 10:26 APT COMMUNICATIONS ROUTINE

3797

000040

APTCSUP=040

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

3798
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3820
3821
3822
3823 016370 017646 000000
3824 016374 116637 000001 016613
3825 016402 112637 016615
3826 016406 062716 000002
3827 016412 000406
3828 016414 112737 000001 016613
3829 016422 112737 000006 016615
3830 016430 112737 000005 016612
3831 016436 010346
3832 016440 010446
3833 016442 010546
3834 016444 113704 016615
3835 016450 005404
3836 016452 062704 000006
3837 016456 110437 016614
3838 016462 113704 016613
3839 016466 016605 000012
3840 016472 005003
3841 016474 006105 15:
3842 016476 000404
3843 016500 006105 25:
3844 016502 006105
3845 016504 006105
3846 016506 010503
3847 016510 006103 35:
3848 016512 105337 016614
3849 016516 100016
3850 016520 042703 177770
3851 016524 001002
3852 016526 005704
3853 016530 001403

```

```

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

```

3854	016532	005204		
3855	016534	052703	000060	
3856	016540	052703	000040	
3857	016544	110337	016610	
3858	016550	104401	016610	
3859	016554	105337	016612	
3860	016550	003347		
3861	016562	002402		
3862	016564	005204		
3863	016566	000744		
3864	016570	012605		
3865	016572	012604		
3866	016574	012603		
3867	016576	016666	000032	000004
3868	016604	012616		
3869	016606	000002		
3870	016610	000		
3871	016611	000		
3872	016612	000		
3873	016613	000		
3874	016614	000000		

```

48: INC R4
    BIS 0'D,R3
58: BIS 0' R3
    MOV8 R3,B8
    TYPE B8
78: DECB $OCNT
    BGT 28
    BLT 68
    INC R4
    BR 28
68: MOV (SP)+,R5
    MOV (SP)+,R4
    MOV (SP)+,R3
    MOV 2(SP),4(SP)
    MOV (SP)+,(SP)
    RTI
88: .BYTE 0
    .BYTE 0
$OCNT: .BYTE 0
$DFILL: .BYTE 0
$OMODE: .WORD 0

```

```

:: DON'T SUPPRESS ANYMORE 0'S
:: MAKE THIS DIGIT ASCII
:: MAKE ASCII IF NOT ALREADY
:: SAVE FOR TYPING
:: GO TYPE THIS DIGIT
:: COUNT BY 1
:: BR IF MORE TO DO
:: BR IF DONE
:: INSURE LAST DIGIT ISN'T A BLANK
:: GO DO THE LAST DIGIT
:: RESTORE R5
:: RESTORE R4
:: RESTORE R3
:: SET THE STACK FOR RETURNING
:: RETURN
:: STORAGE FOR ASCII DIGIT
:: TERMINATOR FOR TYPE ROUTINE
:: OCTAL DIGIT COUNTER
:: ZERO FILL SWITCH
:: NUMBER OF DIGITS TO TYPE

```

3875
 3876
 3877
 3878
 3879
 3880
 3881
 3882
 3883 016616 010046
 3884 016620 016600 000002
 3885 016624 005740
 3886 016626 111000
 3887 016630 006300
 3888 016632 016000 016652
 3889 016636 000200
 3890
 3891
 3892
 3893
 3894 016640 011646
 3895 016642 016666 000004 000002
 3896 016650 000002
 3897
 3898
 3899
 3900
 3901
 3902
 3903
 3904
 3905 016652 016640
 3906 016654 015640
 3907 016656 016414
 3908 016660 016370
 3909 016662 016430
 3910
 3911

.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                    ;; 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
\$TYPE	:::CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
\$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

3912
3913
3914
3915
3916
3917
3918
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958

016664 012737 017036 000024
016672 012737 000340 000026
016700 010046
016702 010146
016704 010246
016706 010346
016710 010446
016712 010546
016714 017746 162226
016720 010637 017042
016724 012737 016736 000024
016732 000000
016734 000776

016736 012737 017036 000024
016744 013706 017042
016750 005037 017042
016754 005237 017042
016760 001375
016762 011600
016764 076600 000226
016770 012677 162152
016774 012605
016776 012604
017000 012603
017002 012602
017004 012601
017006 012600
017010 012737 016664 000024
017016 012737 000340 000026
017024 104401
017026 017044
017030 012716
017032 002100
017034 000002
017036 000000
017040 000776
017042 000000
017044 005015 047520 042527
017052 000122

POWER DOWN ROUTINE

```
$PWRDN: MOV $SILLUP, @#PWRVEC ;; SET FOR FAST UP
MOV #340, @#PWRVEC+2 ;; PRIO:7
MOV R0, -(SP) ;; PUSH R0 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 @SWR, -(SP) ;; PUSH @SWR ON STACK
MOV SP, $SAVR6 ;; SAVE SP
MOV @SPWRUP, @#PWRVEC ;; SET UP VECTOR
HALT
BR .-2 ;; HANG UP
```

POWER UP ROUTINE

```
$PWRUP: MOV $SILLUP, @#PWRVEC ;; SET FOR FAST DOWN
MOV $SAVR6, SP ;; GET SP
CLR $SAVR6 ;; WAIT LOOP FOR THE TTY
15: INC $SAVR6 ;; WAIT FOR THE INC
BNE 15 ;; OF WORD
MOV (SP), R0 ;; GET SAVED SWR OFF STACK
MED 226 ;; RESTORE SWR CONTENTS
MOV (SP)+, @SWR ;; POP STACK INTO @SWR
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)+, R0 ;; POP STACK INTO R0
MOV @SPWRDN, @#PWRVEC ;; SET UP THE POWER DOWN VECTOR
MOV #340, @#PWRVEC+2 ;; PRIO:7
TYPE $POWER ;; REPORT THE POWER FAILURE
$PWRMG: .WORD $POWER ;; POWER FAIL MESSAGE POINTER
MOV (PC)+, (SP) ;; RESTART AT START
$PWRAD: .WORD START ;; RESTART ADDRESS
RTI
$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 MACY11 27(1006) 09-FEB-77 10:28 PAGE 96
 DQFPCA.P11 09-FEB-77 10:26 ERROR MESSAGES, DATA HEADERS, DATA VECTORS, ETC

```

3959 .SBTTL ERROR MESSAGES, DATA HEADERS, DATA VECTORS, ETC
3960 ;MESSAGE PREFIXES
3961 ;
3962 017054 047510 035124 000040 ASCHOT: .ASCIZ "HOT: "
3963 017062 040527 046522 020072 ASCWRM: .ASCIZ "WARM: "
3964 017070 000
3965
3966 ;ERROR MESSAGES HERE
3967 017071 122 041505 044505 EMA: .ASCIZ "RECEIVED FPS IS BAD"
3968 017076 042526 020104 050106
3969 017104 020123 051511 041040
3970 017112 042101 000
3971 017115 122 041505 044505 EMB: .ASCIZ "RECEIVED FEC IS BAD"
3972 017122 042526 020104 042506
3973 017130 020103 051511 041040
3974 017136 042101 000
3975 017141 122 041505 044505 EMC: .ASCIZ "RECEIVED FEA IS BAD"
3976 017146 042526 020104 042506
3977 017154 020101 051511 041040
3978 017162 042101 000
3979 017165 122 041505 044505 EMD: .ASCIZ "RECEIVED PSW CC-S ARE BAD"
3980 017172 042526 020104 051520
3981 017200 020127 041503 051455
3982 017206 040440 042522 041040
3983 017214 042101 000
3984 017217 102 042101 040440 EME: .ASCIZ "BAD ADDRESS IN RC"
3985 017224 042104 042522 051523
3986 017232 044440 020116 030122
3987 017240 000
3988 017241 102 042101 051440 EMF: .ASCIZ "BAD STACK POINTER"
3989 017246 040524 045503 050040
3990 017254 044517 052116 051105
3991 017262 000
3992 017263 122 051505 046125 EMH: .ASCIZ "RESULT OF FPP ARITHMETIC OPERATION IS BAD"
3993 017270 020124 043117 043040
3994 017276 050120 040440 044522
3995 017304 044124 042515 044524
3996 017312 020103 050117 051105
3997 017320 052101 047511 020116
3998 017326 051511 041040 042101
3999 017334 000
4000 017335 122 041505 044505 EMI: .ASCIZ "RECEIVED FEC/FEA IS BAD"
4001 017342 042526 020104 042506
4002 017350 027503 042506 020101
4003 017356 051511 041040 042101
4004 017364 000
4005 017365 125 042516 050130 EMJ: .ASCIZ "UNEXPECTED FPP TRAP, IGNORED AND CONTINUING"
4006 017372 041505 042524 020104
4007 017400 050106 020120 051124
4008 017406 050101 020054 043511
4009 017414 047516 042522 020104
4010 017422 047101 020104 047503
4011 017430 052116 047111 044525
4012 017436 043516 000
4013 017441 103 052520 042040 EMP: .ASCIZ "CPU DID NOT TRAP ON FPP EXCEPTION, BUT IT SHOULD HAVE"
4014 017446 042111 047040 052117

```

E08

FPU INSTR EXERCISER MACY11 27(1006) 09-FEB-77 10:28 PAGE 97
DQFPCA.P11 09-FEB-77 10:26 ERROR MESSAGES, DATA HEADERS, DATA VECTORS, ETC

4015	017454	052040	040522	020120
4016	017462	047117	043040	050120
4017	017470	042440	041530	050105
4018	017476	044524	047117	020054
4019	017504	052502	020124	052111
4020	017512	051440	047510	046125
4021	017520	020104	040510	042526
4022	017526	000		
4023	017527	103	052520	052040
4024	017534	040522	050120	042105
4025	017542	047440	020116	050106
4026	017550	020120	054105	042503
4027	017556	052120	047511	026116
4028	017564	041040	052125	044440
4029	017572	020124	044123	052517
4030	017600	042114	047040	052117
4031	017606	044040	053101	000105
4032	017614	043101	042524	020122
4033	017622	051124	050101	020054
4034	017630	052123	041501	020113
4035	017636	047520	047111	042524
4036	017644	020122	051511	044440
4037	017652	041516	051117	042522
4038	017660	052103	000	
4039	017663	101	052106	051105
4040	017670	052040	040522	026120
4041	017676	051440	047524	042522
4042	017704	020104	041520	044440
4043	017712	020123	047111	047503
4044	017720	051122	041505	000124
4045	017726	043101	042524	020122
4046	017734	051124	050101	020054
4047	017742	052123	051117	042105
4048	017750	050040	020123	051511
4049	017756	044440	041516	051117
4050	017764	042522	052103	000
4051	017771	101	052106	051105
4052	017776	052040	040522	026120
4053	020004	046040	040517	042504
4054	020012	020104	051520	044440
4055	020020	020123	047111	047503
4056	020026	051122	041505	000124
4057				
4058				
4059				
4060				
4061				
4062	020034	043040	051520	000
4063	020041	040	042506	000103
4064	020046	043040	040505	000
4065	020053	040	050106	004523
4066	020060	020040	030122	000
4067	020065	040	051040	000060
4068	020072	020040	030522	020011
4069	020100	051440	000120	
4070	020104	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"	

F08

FPU INSTR EXERCISER
DQFPCA.P11 09-FEB-77

MACY11 27(1006)
10:26

09-FEB-77 10:28 PAGE 98
ERROR MESSAGES, DATA HEADERS, DATA VECTORS, ETC

4071	020112	051044	043505	004460	DHM:	.ASCIZ	"\$REGO \$REG1"				
4072	020120	051044	043505	000061							
4073	020126	051044	043505	004460	DHI:	.ASCIZ	"\$REGO \$REG1 \$REG2 \$REG3"				
4074	020134	051044	043505	004461							
4075	020142	051044	043505	004462							
4076	020150	051044	043505	000063							
4077	020156	054105	023520	026504	DHJ:	.ASCIZ	"EXP'D-FPS-RCV'D"				
4078	020164	050106	026523	041522							
4079	020172	023526	000104								
4080	020176	054105	023520	026504	DHK:	.ASCIZ	"EXP'D-FEC-RCV'D	EXP'D-FEA-RCV'D"			
4081	020204	042506	026503	041522							
4082	020212	023526	004504	054105							
4083	020220	023520	026504	042506							
4084	020226	026501	041522	023526							
4085	020234	000104									
4086	020236	054105	023520	004504	DHL:	.ASCIZ	"EXP'D RCV'D"				
4087	020244	041522	023526	000104							
4088	020252	043040	051520	020011	DHM:	.ASCIZ	" FPS FEC FEA OLD PS OLD PC SP AFTER"				
4089	020260	042506	004503	043040							
4090	020266	040505	047411	042114							
4091	020274	050040	004523	046117							
4092	020302	020104	041520	051411							
4093	020310	020120	043101	042524							
4094	020316	000122									
4095											
4096											
4097											
4098											
4099	020320	001466	000000								
4100	020324	001464			DTB:	.WORD	FEC,0				
4101	020326	001526	000000		DTD:	.WORD	FPS				
4102	020332	001530	001542	000000	DTE:	.WORD	EREGO,0				
4103	020340	001170	000000		DTF:	.WORD	EREG1,EREG6,0				
4104	020344	001170	001172	000000	DTG:	.WORD	\$REGO,0				
4105	020352	001170	001172	001174	DTH:	.WORD	\$REGO,\$REG1,0				
4106	020360	001176	000000		DTI:	.WORD	\$REGO,\$REG1,\$REG2,\$REG3,0				
4107	020364	001220			DTJ:	.WORD	\$TMP4				
4108	020366	001464	000000		DTA:	.WORD	FPS,0				
4109	020372	001222	001466	001500	DTK:	.WORD	\$TMP5,FEC,EXPFEA				
4110	020400	001470	000000		DTC:	.WORD	FEA,0				
4111	020404	001464	001466	001470	DTL:	.WORD	FPS,FEC,FEA,FPPOPS,FPPOPC,OSPRCV,0				
4112	020412	001474	001472	001514							
4113	020420	000000									
4114	020422	001504	001514	000000	DTAG:	.WORD	EXPSP,OSPRCV,0				
4115	020430	001502	001506	000000	DTAH:	.WORD	EXPRTS,OPCRCV,0				
4116	020436	001214	001510	000000	DTAI:	.WORD	\$TMP2,OPSRCV,0				
4117	020444	001212	001512	000000	DTAJ:	.WORD	\$TMP1,NPSLOD,0				
4118											
4119											
4120		000001									

;DATA VECTORS HERE

.EVEN

DTB: .WORD FEC,0
 DTD: .WORD FPS
 DTE: .WORD EREGO,0
 DTF: .WORD EREG1,EREG6,0
 DTG: .WORD \$REGO,0
 DTH: .WORD \$REGO,\$REG1,0
 DTI: .WORD \$REGO,\$REG1,\$REG2,\$REG3,0
 DTJ: .WORD \$TMP4
 DTA: .WORD FPS,0
 DTK: .WORD \$TMP5,FEC,EXPFEA
 DTC: .WORD FEA,0
 DTL: .WORD FPS,FEC,FEA,FPPOPS,FPPOPC,OSPRCV,0
 DTAG: .WORD EXPSP,OSPRCV,0
 DTAH: .WORD EXPRTS,OPCRCV,0
 DTAI: .WORD \$TMP2,OPSRCV,0
 DTAJ: .WORD \$TMP1,NPSLOD,0

;THE END
.END

AAD	012630	3020#	3022		
AADM	013114	3088#	3090		
ABASE =	000000	1027			
ACDM1 =	000000	1027			
ACDM2 =	000000	1027			
ACPUOP =	000000	1027	1042		
ADDW0 =	000000	1027			
ADDW1 =	000000	1027			
ADDW10 =	000000	1027			
ADDW11 =	000000	1027			
ADDW12 =	000000	1027			
ADDW13 =	000000	1027			
ADDW14 =	000000	1027			
ADDW15 =	000000	1027			
ADDW2 =	000000	1027			
ADDW3 =	000000	1027			
ADDW4 =	000000	1027			
ADDW5 =	000000	1027			
ADDW6 =	000000	1027			
ADDW7 =	000000	1027			
ADDW8 =	000000	1027			
ADDW9 =	000000	1027			
ADEVCT =	000000	1027	1033		
ADEVH =	000000	1027			
AD101	001604	1135#	1510	2034	
AD100	001610	1137#	1546	1548	
AD1001	001606	1136#			
ADST01	001760	1164#			
RENV =	000000	1027	1038		
RENVH =	000000	1027	1039		
AFATAL =	000000	1027	1030		
AMADR1 =	000000	1027			
AMADR2 =	000000	1027			
AMADR3 =	000000	1027			
AMADR4 =	000000	1027			
AMAMS1 =	000000	1027			
AMAMS2 =	000000	1027			
AMAMS3 =	000000	1027			
AMAMS4 =	000000	1027			
AMSGAD =	000000	1027	1035		
AMSGLG =	000000	1027	1036		
AMSGTY =	000000	1027	1029		
AMTYP1 =	000000	1027			
AMTYP2 =	000000	1027			
AMTYP3 =	000000	1027			
AMTYP4 =	000000	1027			
APASS =	000000	1027	1032		
APRIOR =	000000	1027			
APTC SU =	000040	3692	3797#		
APTEMV =	000001	3572	3685	3753	3795#
APTSIZ =	000200	1231	3794#		
APTSP0 =	000100	3687	3755	3796#	
ASCHOT	017054	1342	3962#		
ASCHRM	017062	1344	3963#		
ASWREG =	000000	1027	1040		
ATESTN =	000000	1027	1031		

AUNIT = 000000	1027	1034							
AUSMR = 000000	1027	1041							
AVECT1= 000000	1027								
AVECT2= 000000	1027								
ASREG0 001632	1142#	1589*	1941*	1954*	2210	2212			
BGNMES 001772	1170#	1270							
BIT0 = 000001	869#	1355	3448						
BIT00 = 000001	859#	869							
BIT01 = 000002	858#	868							
BIT02 = 000004	857#	867							
BIT03 = 000010	856#	866							
BIT04 = 000020	855#	865	1276	1324	2683	2935	3221		
BIT05 = 000040	854#	864							
BIT06 = 000100	853#	863							
BIT07 = 000200	852#	862							
BIT08 = 000400	851#	861	3499						
BIT09 = 001000	850#	860	3507	3582					
BIT1 = 000002	868#								
BIT10 = 002000	849#	3560							
BIT11 = 004000	848#	3514							
BIT12 = 010000	847#	1308	1334	1336	1339	2687	2939	3227	
BIT13 = 020000	846#	3567							
BIT14 = 040000	845#	3306	3320	3485					
BIT15 = 100000	844#	1258	3422	3428					
BIT2 = 000004	867#	3436							
BIT3 = 000010	866#								
BIT4 = 000020	865#								
BIT5 = 000040	864#								
BIT6 = 000100	863#	1350	1355	3423	3448				
BIT7 = 000200	862#	1248							
BIT8 = 000400	861#								
BIT9 = 001000	860#								
BPTVEC= 000014	876#								
CLKDON 002536	1264	1267#							
CLKPRS 001516	1110#	1241*	1248*	1258*	1348	1353	3191	3195	
CR = 000015	784#	3731	3741						
CRLF = 000200	785#	3702	3741						
DALTA 001634	1143#	1947	1960						
DALTB 001644	1145#	1948	2263	2277					
DALTC 001654	1147#	1967							
DBIG 001612	1138#	1624	1830	2513	2514				
DOISP = 177570	791#	990	1219						
DOIV 012736	3044#	3046							
DHA 020034	1069	4062#							
DHB 020041	1070	4063#							
DHC 020046	1071	4064#							
DHD 020053	1072	4065#							
DHE 020065	1073	1079	4067#						
DHF 020072	1074	4068#							
DHG 020104	1078	4070#							
DHH 020112	1077	4071#							
DHI 020126	1076	4073#							
DHJ 020156	1088	4077#							
DHK 020176	1087	4080#							
DHL 020236	1083	1084	1085	1086	4086#				
DHM 020252	1090	4088#							

CMFFLT	744#														
COMMEN	884#														
COMM00	744#														
COMM01	744#														
COMM02	744#														
COMM03	744#														
COMM04	744#														
COMM05	744#														
COMM06	744#														
COMM07	744#														
COMM1	744#														
COMM10	744#														
COMM11	744#														
COMM12	744#														
COMM13	744#														
COMM14	744#														
COMM15	744#														
COMM16	744#														
COMM17	744#														
COMM2	744#														
COMM20	744#														
COMM21	744#														
COMM22	744#														
COMM23	744#														
COMM24	744#														
COMM25	744#														
COMM26	744#														
COMM27	744#														
COMM3	744#														
COMM30	744#														
COMM31	744#														
COMM32	744#														
COMM33	744#														
COMM34	744#														
COMM35	744#														
COMM36	744#														
COMM37	744#														
COMM4	744#														
COMM40	744#														
COMM41	744#														
COMM42	744#														
COMM43	744#														
COMM44	744#														
COMM45	744#														
COMM46	744#														
COMM47	744#														
ENDCOM	884#														
ERRCMP	744#	3326	3331	3335	3340										
ERRLUR	744#	3593													
ERROR	778#	1373	1379	1385	1391	1397	1403	1414	1422	1430	1440	1448	1453	1463	1466
	1469	1472	1479	1483	1493	1499	1502	1505	1508	1518	1530	1534	1537	1544	1550
	1555	1573	1577	1587	1594	1600	1606	1616	1622	1628	1639	1645	1651	1655	1659
	1664	1665	1666	1670	1683	1688	1692	1695	1702	1707	1714	1719	1730	1736	1742
	1746	1750	1758	1764	1772	1778	1791	1795	1799	1807	1814	1820	1828	1833	1845
	1851	1858	1862	1870	1875	1879	1884	1890	1896	1908	1914	1920	1926	1939	1945
	1952	1958	1964	1970	1981	1987	1993	1999	2010	2016	2022	2028	2039	2044	2050

H09

FPU INSTR EXERCISER MACY11 27(1006) 09-FEB-77 10:28 PAGE 115
DQFPCA.P11 09-FEB-77 10:26 CROSS REFERENCE TABLE -- MACRO NAMES

.SAPTH	7448	938
.SAPTY	7448	3741
.SCATC	7448	915
.SCMTA	7448	961
.SEOP	7448	3240
.SERRO	7448	3533
.SPOWE	7448	3912
.SSCOP	7448	3470
.STRAP	7448	3875
.STYER	7448	3594
.STYPE	7448	3662
.STYPO	7448	3798

. ABS. 020452 000

ERRORS DETECTED: 0
DEFAULT GLOBALS GENERATED: 0

DSKZ: DQFPCA, DSKZ: DQFPCA. SEQ/SOL/CRF=DQFPCA.MEM, DQFPCA.MAC, DQFPCA.P11
RUN-TIME: 20 16 1 SECONDS
RUN-TIME RATIO: 122/38=3.1
CORE USED: 31K (62 PAGES)