

.REM_

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56

IDENTIFICATION

PRODUCT CODE: AC-T038B-MC
PRODUCT NAME: CJKDJBO 11/23B CPU CLSTR DIAG
PRODUCT DATE: APRIL-82
MAINTAINER: DIAGNOSTIC ENGINEERING

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

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

COPYRIGHT (C): 1982 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	DEC/X11

CJKDJB0 11/23-B CPU CLUSTER DIAG.
CJKDJB.P11 26-MAY-82 11:14

DNMAC X24.07-563 26-MAY-82^{C 1} 11:18 PAGE 3

SEQ 0002

HISTORY

57
58
59
60
61
62
63
64
65
66

REVISION A
REVISION B

FIRST RELEASE OF DIAGNOSTIC
CORRECTED MEDIA PROBLEMS

TABLE OF CONTENTS

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
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122

1.0 GENERAL PROGRAM INFORMATION
1.1 ABSTRACT
1.2 SYSTEM REQUIREMENTS
1.3 RELATED DOCUMENTS AND STANDARDS
1.4 DIAGNOSTIC HIERARCHY PREREQUISITES
1.5 ASSUMPTIONS

2.0 OPERATING INSTRUCTIONS
2.1 LOADING AND STARTING PROCEDURE
2.2 PROGRAM OPTIONS
2.3 EXECUTION TIMES

3.0 ERROR INFORMATION
3.1 ERROR REPORTING PROCEDURES
3.2 ERROR HALTS

4.0 PERFORMANCE AND PROGRESS REPORTS
4.1 PERFORMANCE REPORTS
4.2 PROGRESS REPORTS

5.0 DEVICE INFORMATION TABLES

6.0 PROGRAM DESCRIPTION
6.1 PROGRAM EXECUTION CHARACTERISTICS
6.2 SUBTEST SUMMARIES
6.3 SPECIAL SUBROUTINE DESCRIPTION

7.0 LISTING

1.0 ABSTRACT

THIS PROGRAM IS A GO-NOGO TEST FOR THE 11/23-B CPU BOARD. IT TESTS THE CPU INCLUDING EIS, THE MMU, THE FPP, THE LTC AND BOTH SLU'S. IT DOES NOT CONTAIN THE CAPABILITIES OF SCOPE LOOPING, ERROR RECOVERY OR PRINTING OF ERROR INFORMATION. ERROR HALTS DO INDICATE WHICH DEVICE FAILED TO ALLOW THE TECHNICIAN TO DETERMINE WHICH DIAGNOSTIC TO USE TO FIX THE BOARD OR WHAT FIELD REPLACEABLE UNIT (FRU) MAY FIX THE BOARD. THE PROGRAM WILL RUN UNDER THE ACT AND APT MANUFACTURING SYSTEMS AND IS CHAINABLE UNDER XXDP.

1.1 SYSTEM REQUIREMENTS

A. HARDWARE REQUIREMENTS

- KDF11-B CPU MODULE

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
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178

- 32K OF MEMORY
- THE SECOND SLU MUST HAVE TURN AROUND CONNECTOR.

B. SOFTWARE ENVIRONMENTS

- APT (MULTI-CPU TESTER)
- ACT
- XXDP (SLIDE)
- STAND-ALONE

1.2 RELATED DOCUMENTS AND STANDARDS

- ASSEMBLED WITH SYSMAC; SEE FIRST PAGE OF LISTING FOR REVISION NUMBER.
- MXXXX MODULE SPECIFICATION
- DIAGNOSTIC ENGINEERING FUNCTIONAL SPECIFICATION FOR SPECIAL MANUFACTURING TEST BGI-79-003-00-U.
- DIAGNOSTIC ENGINEERING STANDARDS AND CONVENTIONS 175-003-009-02.

1.3 PREREQUISITE DIAGNOSTICS

NONE

1.4 ASSUMPTIONS

THIS PROGRAM ASSUMES THE MACHINE IS UP SUFFICIENTLY TO ALLOW PROPER OPERATION OF THE MICRO-ODT OF THE DCF11-AA CHIP SET.

THE SYSTEM MUST HAVE PARITY MEMORY LOCATED IN THE FIRST 32K BLOCK.

THE SOFTWARE ASSUMES THAT THERE IS NO MEMORY OR DEVICES LOCATED AT OR BEYOND ADDRESS BIT 17 (64 KW). IF MEMORY IS THERE THE PROGRAM WILL FAIL WHEN IN THE EXTENDED ADDRESS TESTS. IF BIT 7 IS SET IN THE SWITCH REGISTER (176) THIS FAILURE CAN BE PREVENTED SINCE THAT PARTICULAR TEST WILL BE BYPASSED.

SEE PARAGRAPH 2.2.

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
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234

2.0 OPERATING INSTRUCTIONS

2.1 LOADING AND STARTING PROCEDURES

TO LOAD AND START THIS PROGRAM USE THE STANDARD PROCEDURES FOR THE DIAGNOSTIC SOFTWARE ENVIRONMENT THAT IS BEING USED.

2.2 PROGRAM OPTIONS

THIS PROGRAM USES THE SOFTWARE SWITCH LOCATION 176 IF PROGRAM IS NOT BEING RUN UNDER APT MODE (BIT 0 SET OF LOCATION \$ENV). IF PROGRAM IS BEING RUN IN APT MODE THE LOCATION \$SWREG IN THE APT ETABLE IS USED TO STORE OPERATING SWITCHES.

WARNING****THIS PROGRAM IS SET TO DO MINIMUM TESTING UNLESS CORRECTIVE ACTION IS TAKEN VIA THE SOFTWARE SWITCH REGISTER (176). BITS 1, 6,7-10 HAVE BEEN SET UP SUCH THAT THE PROGRAM WILL BYPASS CERTAIN TESTS UNLESS THE SWITCH REGISTER BIT IS SET. THIS CONDITION ALSO APPLIES WHEN UNDER CONTROL OF APT. THE APT SWITCH REGISTER, LOCATION 1022, MUST BE CORRECTLY SET AT APT LOAD TIME.

BIT #	DEFINITION
15-11	NOT USED
10	1 - TEST E102 SWITCHES 0 - INHIBIT TESTING E102 SWITCHES
9	1 - TEST PARITY ERROR DETECTION 0 - INHIBIT TESTING PARITY ERROR DETECTION
8	1 - USE THE Q22BE 0 - USE THE QBE IN PLACE OF THE Q22BE
7	1 - TEST THE UPPER 5 ADDRESS BITS FOR TIME OUT 0 - INHIBIT TESTING THE UPPER 5 ADRS BITS
6	1 - TEST USING A Q BUS EXERCISER (QBE OR Q22BE) 0 - INHIBIT TESTS THAT USE A Q BUS EXERCISER
5	0 - PROGRAM RESERVED -- PROGRAM WILL CLEAR IF CIS CHIP SET NOT ON BOARD 1 - PROGRAM RESERVED -- PROGRAM WILL SET IF CIS CHIP SET IS ON BOARD

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
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290

- 4 0 - TEST SLU2 OF 11/23-B
1 - INHIBIT TESTING OF SLU2
- 3 0 - TEST LTC OF 11/23-B
1 - INHIBIT TESTING OF LTC
- 2 0 - TEST SLU1 OF 11/23-B
1 - INHIBIT TESTING OF SLU1
- 1 1 - TEST FPP INSTRUCTION SET
0 - INHIBIT TESTING OF FPP
- 0 0 - TEST MEMORY MANAGEMENT UNIT
1 - INHIBIT TESTING OF MMEMORY MANAGEMENT UNIT

2.3 EXECUTION TIMES

FIRST PASS RUNTIME (WORST CASE).....45 SEC
LONGEST TEST TIME.....30 SEC
ADDITIONAL RUNTIME (EXTRA UNITS).....NONE
LONGEST PASS TIME.....45 SEC

3.0 ERROR INFORMATION

3.1 ERROR REPORTING PROCEDURES

THE PROGRAM DOES NOT TYPE OUT ANY ERROR REPORTS OF ITS OWN BUT TAKES ADVANTAGE OF THE HARDWARE FEATURE THAT TYPES THE PC WHEN A HALT OCCURS. WHEN AN ERROR IS DETECTED THE PROGRAM JUMPS TO ONE OF SEVEN HALT ROUTINES. THE ROUTINES SIMPLY MOVE A FATAL ERROR NUMBER INTO LOCATION \$FATAL, SET THE FATAL ERROR FLAG IN LOCATION \$MSGTY AND EITHER HALT OR IF ON APT DO A BRANCH DOT. THE OPERATOR HAS THREE WAYS TO DETERMINE THE FAILING DEVICE; 1) BY EXAMINING LOCATION \$FATAL, 2) BY DETERMINING THE HALT ADDRESS AND LOOKING UP THE ADDRESS IN THE LISTING AND 3) BY EXAMINING LOCATION \$TESTN WHICH WILL CONTAIN THE TEST NUMBER BEING EXECUTED.

3.2 ERROR HALTS

FOR DISCUSSION SEE SECTION 3.1. THE LABELS FOR THE HALTS AND THE DEVICE THEY INDICATE HAVING FAILED ARE:

CPUHLT: CPU
MMUHLT: MMU
FPPHLT: FPP
LTCHLT: LTC
SL1HLT: SLU1

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
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

SLU1 RBUF

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	I	I	I	I	I				I	I	I	I	I	I	I	I
		I	I	I	I											I
		I	I	I	I											I
ERROR-			I	I	I											I
OVERRUN			I	I	I											I
FRAME ERROR				I	I											I
RECEIVE PARITY																I
ERROR																I
RECEIVED DATA BITS (8)	-----															

SLU1 XCSR

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	I							I	I	I						I
								I	I							
TRANSMITTER READY									I	I						I
TRANSMITTER INTERRUPT ENABLE										I						I

SLU1 XBUF

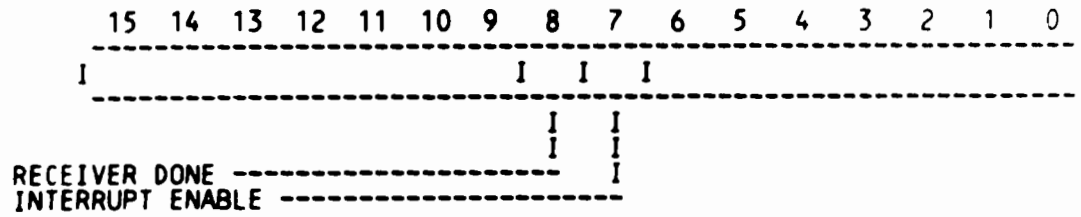
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	I								I	I	I	I	I	I	I	I
																I
																I
TRANSMITTER DATA BITS (8)	-----															

LTC CSR

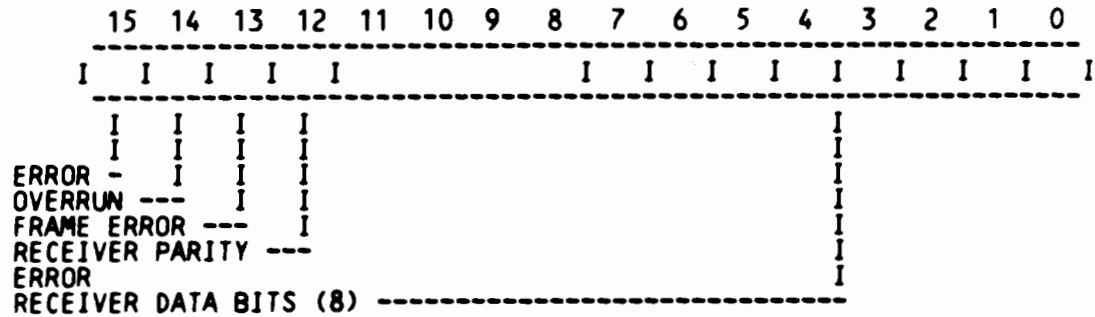
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	I									I	I					
																I
																I
LINE CLOCK	-----															
INTERRUPT ENABLE	-----															

SLU2 RCSR

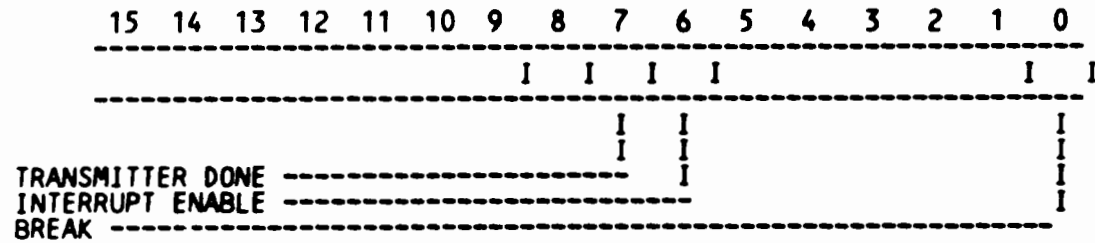
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458



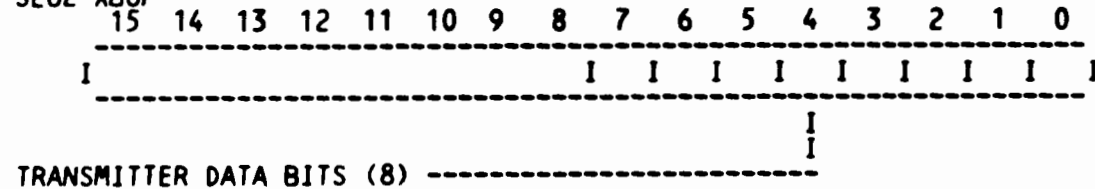
SLU2 RBUF



SLU2 XCSR



SLU2 XBUF



6.0 PROGRAM DESCRIPTION

459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478

6.1 PROGRAM EXECUTION CHARACTERISTICS

THIS PROGRAM RUNS THE SAME UNDER ALL DIAGNOSTIC MONITORS. WHEN THE TEST IS STARTED AT ADDRESS 200 OCTAL THE TESTING IS DONE AND ON COMPLETION THE TITLE IS TYPED AS PART OF THE END OF PASS MESSAGE.

6.2 SUB-TEST SUMMARIES

6.2.1 CENTRAL PROCESSING UNIT SUBTEST -

THESE TESTS CHECK THE BASIC INSTRUCTION SET AND ADDRESSING MODES, THE EXTENDED ELEVEN INSTRUCTION SET (EIS) AND TRAPS TESTING. IT IS EQUIVALENT TO CJKDB.

479
480
481
482
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

6.2.2 MEMORY MANAGEMENT UNIT SUBTEST -

THESE TESTS ARE THE SAME AS IN CJKDA, THE KEF11-AA TEST. THE PROGRAM BEGINS BY TESTING SOME OF THE INTERNAL CPU DATA AND ADDRESS PATHS AND ADDRESS DETECTION LOGIC. NEXT THE MEMORY MANAGEMENT REGISTERS ARE CHECKED FOR DATA RELIABILITY, THEN RELOCATION CAPABILITIES (FORMATION OF A PHYSICAL ADDRESS FROM A VIRTUAL ADDRESS AND ASSOCIATED PAGE DESCRIPTOR (PDR) INFORMATION). FINALLY THE ABORT AND STATUS SEGMENTS OF THE LOGIC ARE CHECKED.

6.2.3 EXTENDED ADDRESS BIT TESTING AND PARITY ERROR LOGIC TEST

6.2.4 Q22BE BR LEVEL TESTING

6.2.5 BDV TESTS

6.2.6 FLOATING POINT PROCESSOR SUBTEST -

THE FLOATING POINT PROCESSOR SUBTEST CHECKS FLOATING POINT REGISTERS FIRST USING A LIMITED NUMBER OF FLOATING POINT INSTRUCTIONS. IT THEN VERIFIES THE REST OF THE FLOATING POINT INSTRUCTION SET USING A NUMBER OF DATA PATTERNS FOR EACH INSTRUCTION.

6.2.7 SERIAL LINE UNIT (SLU1) SUBTEST -

THESE TESTS CHECK THE SLU'S REGISTERS FOR ADDRESSING AND DATA HANDLING.

6.2.8 LINE TIME CLOCK (LTC) SUBTEST -

FIRST THE REGISTER IS CHECKED FOR ADDRESSING AND BIT SETTING CAPABILITIES THEN THE INTERRUPT LOGIC IS CHECKED. THERE IS ALSO A

6.2.9 SERIAL LINE UNIT 2 SUBTEST -

THE TESTING DONE HERE IS SIMILIAR AS FOR THE SLU1. AN EXTERNAL JUMPER, TURN AROUND CONNECTOR, MUST BE PRESENT.

535
536
537
538
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

6.2.10 BLAST SUBTEST

THIS TEST CHECKS THE ABILITY OF THE 11/23-B TO HANDLE SYSTEM INTER-
ACTION. THE CPU HAS TO HANDLE DEVICES AT DIFFERENT PRIORITY LEVELS
AND ARBITRATE BETWEEN THEM AND ITS OWN PRIORITY. THE TEST SETS UP
ALL DEVICES TO INTERRUPT THEN ENABLES THEM ALL AT ONCE. THE SLU'S
TRANSFER DATA UNTIL THEY TRANSFER 400(8) BYTES OR UNTIL ONE SECOND
(60 TICKS) OF THE LINE CLOCK HAS BEEN RECEIVED. THE PROGRAM THEN
VERIFIES THE NUMBER OF TRANSMITTER INTERRUPTS IS EQUAL TO THE NUM-
BER OF RECEIVER INTERRUPTS. FINALLY THE DATA TRANSFERRED BY EACH
DEVICE IS CHECKED.

6.3 SPECIAL SUBROUTINE DESCRIPTIONS

THE ONLY SPECIAL SUBROUTINES ARE THE ERROR ROUTINES EACH SUBTEST
HAS IS OWN. THE ROUTINES SIMPLY SET THE FATAL ERROR FLAG IN THE
APT MAILBOX AND EITHER 'BRANCH SELF' OR 'HALT'. THIS CHOICE IS
DETERMINED IN THE INITIALIZE PORTION OF THE PROGRAM AND IS A
'BRANCH SELF' IF RUNNING UNDER APT OR A 'HALT' IF RUNNING UNDER
ANY OTHER MONITOR.

```
.TITLE CJKDJBO 11/23-B CPU CLUSTER DIAG.  
.ENABLE ABS  
.NLIST CND,MC,MD  
.LIST ME  
SCOPE=NOP  
R7=%7  
R6=%6  
PS=177776  
TPS=177564  
TPB=177566  
USRM=140000  
PUSRM=30000  
SP=%6  
R6=%6  
TAB=%3  
LAST=%1  
FIRST=%5  
R2=%2  
HLT=HALT  
TRT=3  
ITRAP5=4  
RTRAP5=4  
RTRAP4=14  
RTRAP3=30  
RTRAP2=20  
RTRAP1=34  
TTCSR=177564  
TRCSR=177560  
TKB=177562  
TPB=177566  
;RESERVED INST AND ILLEGAL ADDRESSES  
;FOR TRACE TRAP  
;FOR EMULATOR TRAP  
;FOR IOT TRAP  
;FOR TRAP INST
```

591 000240
592 000240
593 177776
594 000077
595 000010
596 004700
597 000100
598
599 177520
600 177524
601 177522
602
603 177776
604 001000
605 000600
606 104377
607 104777
608 000001

BELL=240
NOP=240
STATUS=177776
TRAPA=77
RTRAP=10
ILLA=004700
ILLB=100

PCR=177520
LSREG=177524
RWREG=177522

CC=177776
KERSTK=STBOT
USESTK=STBOT-200
EMTA=104377
TRAPC=104777
APTENV=1

.SBTTL ACT11 HOOKS

;HOOKS REQUIRED BY ACT11

613 000400
614 000046
615 000046 133212
616 000052
617 000052 000000
618 000400
619 001000

SSVPC=. ;SAVE PC
. =46
SENDAD ;:1)SET LOC.46 TO ADDRESS OF SENDAD IN .SEOP
. =52
.WORD 0 ;:2)SET LOC.52 TO ZERO
. =SSVPC ;: RESTORE PC
. =1000

.SBTTL APT MAILBOX-ETABLE

623 .EVEN
624 001000
625 001000 000000
626 001002 000000
627 001004 000000
628 001006 000000
629 001010 000000
630 001012 000000
631 001014 000000
632 001016 000000
633 001020
634 001020 000
635 001021 000
636 001022 000000
637 001024 000000
638 001026 000000

SMAIL: ;: APT MAILBOX
\$MSGTY: .WORD AMSGTY ;: MESSAGE TYPE CODE
\$FATAL: .WORD AFATAL ;: FATAL ERROR NUMBER
\$TESTN: .WORD ATESTN ;: TEST NUMBER
\$PASS: .WORD APASS ;: PASS COUNT
\$DEVCT: .WORD ADEVCT ;: DEVICE COUNT
\$UNIT: .WORD AUNIT ;: I/O UNIT NUMBER
\$MSGAD: .WORD AMSGAD ;: MESSAGE ADDRESS
\$MSGLG: .WORD AMSGLG ;: MESSAGE LENGTH
\$ETABLE: ;: APT ENVIRONMENT TABLE
\$ENV: .BYTE AENV ;: ENVIRONMENT BYTE
\$ENVM: .BYTE AENVM ;: ENVIRONMENT MODE BITS
\$SWREG: .WORD ASWREG ;: APT SWITCH REGISTER
\$USWR: .WORD AUSWR ;: USER SWITCHES
\$CPUOP: .WORD ACPUOP ;: CPU TYPE, OPTIONS
* ;: BITS 15-11=CPU TYPE
* ;: 11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
* ;: 11/70=06,PDQ=07,Q=10
* ;: BIT 10=REAL TIME CLOCK
* ;: BIT 9=FLOATING POINT PROCESSOR
* ;: BIT 8=MEMORY MANAGEMENT

645 001030
646

\$ETEND:
.MEXIT

647
648
649
650
651
652 001030
653 000024
654 000024 000200
655 000044 000044
656 000044 001030
657 001030
658
659
660
661
662 001030
663 001030 000000
664 001032 001000
665 001034 000010
666 001036 000025
667 001040 000000
668 001042 000014
669
670
671
672 000004
673 000004 021336
674 000006 000000
675 000010 021340
676 000012 000000
677 000014 021342
678 000016 000000
679 000020 021344
680 000022 000000
681 000030 000030
682 000030 021346
683 000032 000000
684 000034 021350
685 000036 000000
686 000114 000114
687 000114 021352
688 000116 000000
689 000244 000244
690 000244 021354
691 000246 000000
692 000250 021356
693 000252 000000
694
695 000172
696 000172 000000
697 000174 000000
698 000176 000000
699
700
701
702

```
.SBTTL APT PARAMETER BLOCK

;*****
;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
;*****
.$X=. ;;SAVE CURRENT LOCATION
.=24 ;;SET POWER FAIL TO POINT TO START OF PROGRAM
200 ;;FOR APT START UP
.=44 ;;POINT TO APT INDIRECT ADDRESS PNTR.
$APTHDR ;;POINT TO APT HEADER BLOCK
.=.$X ;;RESET LOCATION COUNTER

;*****
;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
;INTERFACE SPEC.

$APTHD:
$HIBTS: .WORD 0 ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MBADR: .WORD $MAIL ;;ADDRESS OF APT MAILBOX (BITS 0-15)
$STMT: .WORD 10 ;;RUN TIM OF LONGEST TEST
$PASTM: .WORD 25 ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
$UNITM: .WORD 0 ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
        .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)

;*****
;SOME POINTERS TO CPU TRAP HANDLERS
;*****
.=4
T04
0
T010
0
T014
0
T020
0
.=30
T030
0
T034
0
.=114
T0114
0
.=244
T0244
0
T0250
0

.=172
MTFLAG: 0 ;;MULTI-TESTER ACTIVE BIT
DISPREG: 0 ;;SOFTWARE DISPLAY REGISTER
SWREG: 0 ;;SOFTWARE SWITCH REGISTER

;*****
;DATA TABLE FOR USE IN ADDRESSING MODE TESTS
;*****
```

```
703          000370          .=370
704 000370 000000 000000 000000 0,0,0,0,0,0
705 000376 000000 000000 000000
706 000404 000001 000001 177777 1,1,-1
707
708 ;*****
709 ;SET UP STARTING ADDRESS
710 001000 001000 000000 .=1000
711 STBOT: .WORD 0 ;STACK POINTER
712 ;* *****
713
714          000510          .=510 ;Q22BE DEVICE VECTOR
715 ;AREA
716
717          000200          .=200
718 000200 000167 000774 JMP START
719 000204 012706 001000 MOV #STBOT,R6 ;SET STACK POINTER
720 000210 012702 001004 MOV #STESTN,R2 ;SET MAILBOX POINTER
721 000214 000137 JMP a(PC)+ ;JUMP TO SUBTEST
722 000216 000000 0 ;ADDR. OF SUBTEST GOES HERE
723
724          001200          .=1200
725 .SBTIL **STARTING OF CPU TEST **
726 001200 032737 000001 001020 START: BIT #1,a#SENV ;UNDER APT
727 001206 001004 BNE CONTIN
728 001210 012700 133431 MOV #STRMSG,R0 ;START MESSAGE
729 001214 004767 132110 JSR PC,TYPE
730 001220 012737 000000 001006 CONTIN: MOV #0,a#SPASS ;CLEAR PASS COUNT
731 001226 012737 133264 000024 RESTR: MOV #PWRDN,a#24 ;SET UP FOR POWER FAIL
732 001234 012706 001000 MOV #STBOT,R6 ;SET UP STACK
733 001240 012737 001270 000004 MOV #1$,a#4 ;SET UP FOR TIMEOUT IF NO MULTI TESTER
734 001246 012737 000340 000006 MOV #340,a#6
735 001254 012737 000002 164000 MOV #2,a#164000 ;SET BIT1 FOR MULTI TESTER
736 001262 012737 000001 000172 MOV #1,a#MTFLAG ;SET FLAG TO INDICATE MULTI-TESTER
737 001270 012737 021336 000004 1$: MOV #T04,a#4 ;SET TRAP CATCHER
738 001276 012737 000000 000006 MOV #0,a#6 ;SET HALT BACK IN LOCATION 6
739 001304 012706 001000 MOV #STBOT,R6 ;INITIALIZE STACK POINTER
740 001310 012737 000001 001004 MOV #1,a#STESTN ;SET TEST NUMBER TO 1
741 001316 012737 000000 001002 MOV #0,a#SFATAL ;CLEAR ERROR INDICATOR
742 001324 012737 000000 001000 MOV #0,a#SMSGTY ;CLEAR MESSAGE TYPE(FOR APT)
743
744 001332 005067 176166 CLR LSREG ;THIS WILL TURN ON THE 4 LEDS
745
746 001336 105737 001020 TSTB a#SENV ;RUNNING ON APT
747 001342 001036 BNE TS1 ;IF YES DO BRANCH SELF ON ERROR
748 001344 012737 000000 002164 MOV #0,a#CPUHLT ;IF NOT THEN PUT A HALT IN ON ERROR
749 001352 012737 000000 050470 MOV #0,a#MMUHLT
750 001360 012737 000000 051404 MOV #0,a#EXADHT
751 001366 012737 000000 053214 MOV #0,a#Q22HLT
752 001374 012737 000000 124532 MOV #0,a#FPHLT
753 001402 012737 000000 126674 MOV #0,a#LTCHLT
754 001410 012737 000000 125754 MOV #0,a#SL1HLT
755 001416 012737 000000 131540 MOV #0,a#SL2HLT
756 001424 012737 000000 132504 MOV #0,a#COMHLT
757 001432 012737 000000 054472 MOV #0,a#BDVHLT
758
```

```

759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784 001440
785 001440 012700 021242
786 001444 012704 021300
787 001450 012767 000017 000130
788 001456 012067 000110
789 001462 012401
790 001464 012767 177777 000074
791 001472 012703 000020
792 001476 005267 000064
793 001502 032701 100000
794 001506 013705 177776
795 001512 042705 177773
796 001516 000165 001522
797 001522 000167 000020
798 001526 012767 001610 000042
799 001534 012767 001604 000040
800 001542 000167 000014
801 001546 012767 001604 000022
802 001554 012767 001610 000020
803 001562 006101
804
805 001564 012737
806 001566 000000
807 001570 177776
808 001572 000000
809 001574 000137
810 001576 000000
811 001600 000137
812 001602 000000
813 001604
814 001604 000551
  
```

```

:*****
:
: THIS TEST EXECUTES EVERY POSSIBLE BRANCH WITH EVERY POSSIBLE
: CONDITION CODE COMBINATION.
: THE ROUTINE USES TWO TABLES. THE BRANCH TABLE HOLDS ALL THE
: POSSIBLE BRANCH INSTRUCTIONS, THE OTHER TABLE (YNTAB) HOLDS BIT MAPS FOR
: EACH BRANCH. A ONE IN THE BIT MAP INDICATES THAT THE CORRESPONDING
: BRANCH INSTRUCTION SHOULD BRANCH FOR THE CONDITION CODE SETTING WHICH
: CORRESPONDS TO THE BIT POSITION WITHIN THE MAP. FOR EXAMPLE IF THE LEFT
: MOST BIT IS A ONE THEN THE CORRESPONDING BRANCH INSTRUCTION SHOULD BRANCH
: WHEN THE CONDITION CODES ARE 0.
: THE ROUTINE CONSISTS OF NESTED LOOPS; THE OUTER LOOP SETS UP
: ALL THE POSSIBLE BRANCH INSTRUCTIONS. THE INNER LOOP SETS UP EVERY POSSIBLE
: CONDITION CODE FOR EACH BRANCH.
: THE BIT MAP IS USED TO SET THE ADDRESS LOCATION IN TWO
: JUMP MODE 3 INSTRUCTIONS. THE ADDRESSES ARE CHANGED TO ALLOW THE
: PROGRAM TO CONTINUE OR JUMP TO AN ERROR ROUTINE DEPENDING UPON
: WHETHER IT HANDLED THE BRANCH INSTRUCTION CORRECTLY.
: AT ANY ERROR HALT, LOCATION, BRH, HOLDS THE BRANCH INSTRUCTION
: UNDER TEST AND LOCATION, CC, HOLDS THE VALUE OF THE CONDITION CODES
: AT THE TIME THE BRANCH WAS EXECUTED.
:
:*****
: TEST 1 TEST THE BRANCH ROM
:*****
TS1:
SETUP: MOV #BRTAB,R0 ;INITIALIZE BRANCH TABLE POINTER
      MOV #YNTAB,R4 ;INITIALIZE YES/NO BRANCH MAP POINTER
      MOV #15.,BRCT ;INITIALIZE BRANCH TABLE COUNT
SETBR: MOV (R0)+,BRH ;GET NEXT BRANCH INST.
      MOV (R4)+,R1 ;GET NEXT BRANCH MAP
      MOV #-1,CC1 ;INITIALIZE CONDITION CODE VALUE
      MOV #16.,R3 ;INITIALIZE CONDITION CODE COUNT
SETCC: INC CC1 ;SET FOR NEXT CC VALUE
      BIT #100000,R1 ;SEE IF SHOULD BR W/ THESE CC'S
      MOV @#177776,R5 ;SIMULATE A JNE
      BIC #177773,R5 ;
      JMP .+4(R5) ; (JUMP NOT EQUAL)
      JMP SET2BR ; TO SET2BR
      MOV #CONT,NBR ;SET TO CONTINUE IF NO BRANCH
      MOV #ER,YBR ;SET TO REPORT ERROR IF BRANCH
      JMP AROUND ;GO AROUND OPPOSITE CONDITION
SET2BR: MOV #ER,NBR ;SET TO REPORT ERROR IF NO BRANCH
      MOV #CONT,YBR ;SET TO CONTINUE IF BRANCH
AROUN: ROL R1 ;UPDATE BIT MAP
      MOV (PC)+,@(PC)+ ;SET CONDITION CODE
CC1: 0 ;NEW CC VALUE GOES HERE
      177776
BRH: 0 ;BRANCH INST. GOES HERE
      JMP @ (PC)+ ;THIS JUMP IF NO BRANCH
NBR: 0 ;WHERE TO GO IF NO BRANCH OCCURS
      JMP @ (PC)+ ;THIS JUMP IF BRANCH OCCURS
YBR: 0 ;WHERE TO GO IF BRANCH OCCURS
ER: BR ERROR1 ;
  
```


E 2

815	001606	000000	
816	001610	005303	
817	001612	013705	177776
818	001616	042705	177773
819	001622	000165	001626
820	001626	000167	177644
821	001632	005367	177750
822	001636	013705	177776
823	001642	042705	177773
824	001646	000165	001652
825	001652	000167	177600
826	001656	012700	000357

```

BRCT: 0
CONT: DEC R3 ;CC'S DONE?
      MOV @#177776,R5 ;SIMULATE A JNE
      BIC #177773,R5 ; (JUMP NOT EQUAL)
      JMP .+4(R5) ; TO SETCC
      JMP SETCC
      DEC BRCT ;BR'S DONE?
      MOV @#177776,R5 ;SIMULATE A JNE
      BIC #177773,R5 ; (JUMP NOT EQUAL)
      JMP .+4(R5) ; TO SETBR
      JMP SETBR
      MOV #357,R0 ;IF THIS TEST IS DONE SET UP R0 FOR THE NEXT
                    ;SEVEN TESTS. THIS IS SAVING 4 LOCATIONS PER
                    ;TEST WHICH I NEED BECAUSE BRANCHES WERE OUT
                    ;OF BOUNDS.

```

827
828
829
830
831

```

;*****
;TEST 2 TEST TRAP OF RESERVED INSTRUCTION
;*****

```

833			
834			
835	001662		
836	001662	012706	001000
837	001666	012767	001710
838	001674	010067	176112
839	001700	005067	176072
840	001704	000077	
841	001706		
842	001706	000510	
843	001710	020067	176062
844	001714	001105	
845	001716	020627	000774
846	001722	001102	
847	001724	022767	001706
848	001732	001076	
849	001734	005767	177036
850	001740	001073	
851	001742	062706	000004
852	001746	012767	001770
853	001754	005067	176032
854	001760	010037	177776
855	001764	000077	
856	001766	000460	
857	001770	005767	176002
858	001774	001055	
859	001776	020067	176774
860	002002	001052	

```

TS2:
      MOV #STBOT,SP ;INITIALIZE THE STACK POINTER
      MOV #1$,RTRAP ;SET UP NEW PC IN VECTOR
      MOV R0,RTRAP+2 ;SET UP NEW PSW IN VECTOR
      CLR STATUS ;CLEAR PRESENT (OLD) STATUS
      TRAPA ;DO TRAP
8$: BR ERROR1 ;INSTRUCTION FAILED TO TRAP
1$: CMP R0,STATUS ;IS NEW STATUS CORRECT
   BNE ERROR1 ;NEW STATUS WRONG
2$: CMP SP,#STBOT-4 ;DID STACK DECREMENT CORRECTLY
   BNE ERROR1 ;STACK DID NOT DECREMENT CORRECTLY
3$: CMP #8$,STBOT-4 ;WAS PROPER PC SAVED
   BNE ERROR1 ;PROPER PC WAS NOT SAVED
4$: TST STBOT-2 ;WAS OLD PSW SAVED
   BNE ERROR1 ;WRONG PSW SAVED
5$: ADD #4,SP ;RESET STACK POINTER
   MOV #6$,RTRAP ;SET UP NEW PC IN VECTOR
   CLR RTRAP+2 ;SET UP NEW PSW IN VECTOR
   MOV R0,@#STATUS ;SET UP OLD STATUS FOR COMPARISON AFTER TRAP
   TRAPA ;DO TRAP
6$: BR ERROR1 ;INSTRUCTION FAILED TO TRAP
   TST STATUS ;IS NEW PSW CORRECT
   BNE ERROR1 ;NEW PSW WRONG
7$: CMP R0,STBOT-2 ;WAS OLD STATUS STORED
   BNE ERROR1 ;OLD STATUS WRONG

```

861
862
863

```

;*****
;TEST 3 TEST TRAP OF TRAP INSTRUCTION
;*****

```

864	002004		
865	002004	012706	001000
866	002010	012767	002032
867	002016	010067	176014
868	002022	005067	175750
869	002026	104400	
870	002030		

```

TS3:
      MCV #STBOT,SP ;INITIALIZE THE STACK POINTER
      MOV #1$,RTRAP1 ;SET UP NEW PC IN VECTOR
      MOV R0,RTRAP1+2 ;SET UP NEW PSW IN VECTOR
      CLR STATUS ;CLEAR PRESENT (OLD) STATUS
      TRAP ;DO TRAP
8$:

```

```
871 002030 000437          BR      ERROR1          ;INSTRUCTION FAILED TO TRAP
872 002032 020067 175740  1$:  CMP      RO,STATUS      ;IS NEW STATUS CORRECT
873 002036 001034          BNE     ERROR1          ;NEW STATUS WRONG
874 002040 020627 000774  2$:  CMP      SP,#STBOT-4    ;DID STACK DECREMENT CORRECTLY
875 002044 001031          BNE     ERROR1          ;STACK DID NOT DECREMENT CORRECTLY
876 002046 022767 002030 176720 3$:  CMP      #8$,STBOT-4    ;WAS PROPER PC SAVED
877 002054 001025          BNE     ERROR1          ;PROPER PC WAS NOT SAVED
878 002056 005767 176714  4$:  TST      STBOT-2      ;WAS OLD PSW SAVED
879 002062 001022          BNE     ERROR1          ;WRONG PSW SAVED
880 002064 062706 000004  5$:  ADD      #4,SP          ;RESET STACK POINTER
881 002070 012767 002112 175736  MOV      #6$,RTRAP1      ;SET UP NEW PC IN VECTOR
882 002076 005067 175734  CLR      RTRAP1+2        ;SET UP NEW PSW IN VECTOR
883 002102 010037 177776  MOV      RO,@#STATUS     ;SET UP OLD STATUS FOR COMPARISON AFTER TRAP
884 002106 104777          TRAPC                    ;DO TRAP WITH LOWER BYTE ALL ONES
885 002110 000407          BR      ERROR1          ;INSTRUCTION FAILED TO TRAP
886 002112 005767 175660  6$:  TST      STATUS        ;IS NEW PSW CORRECT
887 002116 001004          BNE     ERROR1          ;NEW PSW WRONG
888 002120 020067 176652  7$:  CMP      RO,STBOT-2    ;WAS OLD STATUS STORED
889 002124 001001          BNE     ERROR1          ;OLD STATUS WRONG
890 002126 000434          BR      CPUHLT+34       ;GET OVER ERROR CALL TO NEXT TEST IF NO ERROR
891                                     ;*****
892                                     ;THIS ERROR IS USED FOR THE ENTIRE CPU,TRAPS AND EIS PORTION OF
893                                     ;THIS TEST
894                                     ;*****
895 002130 012737 000001 001002  ERROR1: MOV      #1,@#$FATAL ;SET UP FATAL ERROR NUMBER
896 002136 012767 000001 176634  MOV      #1,$MSGTY       ;SET FATAL ERROR FLAG
897 002144 032737 000001 001020  BIT      #1,@#SENV       ;UNDER APT
898 002152 001004          BNE     CPUHLT          ;YES, THEN DO NOT PRINT
899 002154 012700 002166  MOV      #CPUMSG,RO
900 002160 004767 131144  JSR      PC,TYPE        ;TYPE MSG
901 002164 000777          CPUHLT: BR      .
902
903 002166 040506 046111 042105  CPUMSG: .ASCIZ  /FAILED DURING CPU TESTS/<12><15>
904 002174 042040 051125 047111
905 002202 020107 050103 020125
906 002210 042524 052123 005123
907 002216 000015
908                                     .EVEN
909
910                                     ;*****
911                                     ;TEST 4 TEST TRAP OF IOT INSTRUCTION
912                                     ;*****
913 002220          TS4:
914 002220 012706 001000          MOV      #STBOT,SP      ;INITIALIZE THE STACK POINTER
915 002224 012767 002246 175566  MOV      #1$,RTRAP2    ;SET UP NEW PC IN VECTOR
916 002232 010067 175564  MOV      RO,RTRAP2+2   ;SET UP NEW PSW IN VECTOR
917 002236 005067 175534  CLR      STATUS        ;CLEAR PRESENT (OLD) STATUS
918 002242 000004          IOT                    ;DO TRAP
919 002244          8$:
920 002244 000731          BR      ERROR1          ;INSTRUCTION FAILED TO TRAP
921 002246 020067 175524  1$:  CMP      RO,STATUS      ;IS NEW STATUS CORRECT
922 002252 001326          BNE     ERROR1          ;NEW STATUS WRONG
923 002254 020627 000774  2$:  CMP      SP,#STBOT-4    ;DID STACK DECREMENT CORRECTLY
924 002260 001323          BNE     ERROR1          ;STACK DID NOT DECREMENT CORRECTLY
925 002262 022767 002244 176504 3$:  CMP      #8$,STBOT-4    ;WAS PROPER PC SAVED
926 002270 001317          BNE     ERROR1          ;PROPER PC WAS NOT SAVED
```

927 002272 005767 176500
928 002276 001314
929 002300 062706 000004
930 002304 012767 002326 175506
931 002312 005067 175504
932 002316 010037 177776
933 002322 000004
934 002324 000701
935 002326 005767 175444
936 002332 001276
937 002334 020067 176436
938 002340 001273

4\$: TST STBOT-2 ;WAS OLD PSW SAVED
BNE ERROR1 ;WRONG PSW SAVED
5\$: ADD #4,SP ;RESET STACK POINTER
MOV #6\$,RTRAP2 ;SET UP NEW PC IN VECTOR
CLR RTRAP2+2 ;SET UP NEW PSW IN VECTOR
MOV RO,#STATUS ;SET UP OLD STATUS FOR COMPARISON AFTER TRAP
IOT ;DO TRAP
BR ERROR1 ;INSTRUCTION FAILED TO TRAP
6\$: TST STATUS ;IS NEW PSW CORRECT
BNE ERROR1 ;NEW PSW WRONG
7\$: CMP RO,STBOT-2 ;WAS OLD STATUS STORED
BNE ERROR1 ;OLD STATUS WRONG

:TEST 5 TEST TRAP OF EMT INSTRUCTION

940
941
942 002342
943 002342 012706 001000
944 002346 012767 002370 175454
945 002354 010067 175452
946 002360 005067 175412
947 002364 104000
948 002366
949 002366 000660
950 002370 020067 175402
951 002374 001255
952 002376 020627 000774
953 002402 001252
954 002404 022767 002366 176362
955 002412 001246
956 002414 005767 176356
957 002420 001243
958 002422 062706 000004
959 002426 012767 002450 175374
960 002434 005067 175372
961 002440 010037 177776
962 002444 104377
963 002446 000630
964 002450 005767 175322
965 002454 001225
966 002456 020067 176314
967 002462 001222
968 002464 000401

TS5:
MOV #STBOT,SP ;INITIALIZE THE STACK POINTER
MOV #1\$,RTRAP3 ;SET UP NEW PC IN VECTOR
MOV RO,RTRAP3+2 ;SET UP NEW PSW IN VECTOR
CLR STATUS ;CLEAR PRESENT (OLD) STATUS
EMT ;DO TRAP
8\$: BR ERROR1 ;INSTRUCTION FAILED TO TRAP
1\$: CMP RO,STATUS ;IS NEW STATUS CORRECT
BNE ERROR1 ;NEW STATUS WRONG
2\$: CMP SP,#STBOT-4 ;DID STACK DECREMENT CORRECTLY
BNE ERROR1 ;STACK DID NOT DECREMENT CORRECTLY
3\$: CMP #8\$,STBOT-4 ;WAS PROPER PC SAVED
BNE ERROR1 ;PROPER PC WAS NOT SAVED
4\$: TST STBOT-2 ;WAS OLD PSW SAVED
BNE ERROR1 ;WRONG PSW SAVED
5\$: ADD #4,SP ;RESET STACK POINTER
MOV #6\$,RTRAP3 ;SET UP NEW PC IN VECTOR
CLR RTRAP3+2 ;SET UP NEW PSW IN VECTOR
MOV RO,#STATUS ;SET UP OLD STATUS FOR COMPARISON AFTER TRAP
EMTA ;DO TRAP WITH LOWER BYTE ALL ONES
BR ERROR1 ;INSTRUCTION FAILED TO TRAP
6\$: TST STATUS ;IS NEW PSW CORRECT
BNE ERROR1 ;NEW PSW WRONG
7\$: CMP RO,STBOT-2 ;WAS OLD STATUS STORED
BNE ERROR1 ;OLD STATUS WRONG
BR ERROR2+2 ;WE MUST GET OVER ERROR CALL AT END OF THIS TEST

:THIS ERROR IS NEEDED BECAUSE BRANCHES IN TRAP TESTS BEYOND HERE CAN NOT
:REACH ERROR1.

969
970
971
972
973 002466 000620
974
975
976
977

ERROR2: BR ERROR1

:TEST 6 TEST TRAP OF TRACE-TRAP INSTRUCTION

978
979 002470
980 002470 012706 001000
981 002474 012767 002516 175312
982 002502 010067 175310

TS6:
MOV #STBOT,SP ;INITIALIZE THE STACK POINTER
MOV #1\$,RTRAP4 ;SET UP NEW PC IN VECTOR
MOV RO,RTRAP4+2 ;SET UP NEW PSW IN VECTOR

```
983 002506 005067 175264          CLR      STATUS          ;CLEAR PRESENT (OLD) STATUS
984 002512 000003                   TRT                    ;DO TRAP
985 002514                   8$:
986 002514 000764                   BR      ERROR2          ;INSTRUCTION FAILED TO TRAP
987 002516 020067 175254          1$:  CMP      R0,STATUS    ;IS NEW STATUS CORRECT
988 002522 001361                   BNE     ERROR2          ;NEW STATUS WRONG
989 002524 020627 000774          2$:  CMP      SP,#STBOT-4   ;DID STACK DECREMENT CORRECTLY
990 002530 001356                   BNE     ERROR2          ;STACK DID NOT DECREMENT CORRECTLY
991 002532 022767 002514 176234  3$:  CMP      #8$,STBOT-4    ;WAS PROPER PC SAVED
992 002540 001352                   BNE     ERROR2          ;PROPER PC WAS NOT SAVED
993 002542 005767 176230          4$:  TST      STBOT-2        ;WAS OLD PSW SAVED
994 002546 001347                   BNE     ERROR2          ;WRONG PSW SAVED
995 002550 062706 000004          5$:  ADD      #4,SP          ;RESET STACK POINTER
996 002554 012767 002576 175232  MOV      #6$,RTRAP4      ;SET UP NEW PC IN VECTOR
997 002562 005067 175230          CLR      RTRAP4+2        ;SET UP NEW PSW IN VECTOR
998 002566 010037 177776          MOV      R0,#STATUS      ;SET UP OLD STATUS FOR COMPARISON AFTER TRAP
999 002572 000003                   TRT                    ;DO TRAP
1000 002574 000734                   BR      ERROR2          ;INSTRUCTION FAILED TO TRAP
1001 002576 005767 175174          6$:  TST      STATUS        ;IS NEW PSW CORRECT
1002 002602 001331                   BNE     ERROR2          ;NEW PSW WRONG
1003 002604 020067 176166          7$:  CMP      R0,STBOT-2    ;WAS OLD STATUS STORED
1004 002610 001326                   BNE     ERROR2          ;OLD STATUS WRONG
1005                                     ;PDP-11 ILLEGAL AND ADDRESS INSTRUCTION TEST
1006                                     ;ALL INSTRUCTIONS THAT ARE RESERVED
1007                                     ;SHOULD TRAP TO LOCATION 4, AND THE
1008                                     ;PC THAT POINTS TO THE TRAPPING INSTRUCTION
1009                                     ;SHOULD BE PLACED ON THE STACK
1010
1011                                     ;*****
1012                                     ;TEST 7 TEST TRAP OF ILLEGAL INSTRUCTION
1013                                     ;*****
1014 002612                   TS7:
1015 002612 012706 001000          MOV      #STBOT,SP      ;INITIALIZE THE STACK POINTER
1016 002616 012767 002640 175160  MOV      #1$,RTRAP5     ;SET UP NEW PC IN VECTOR
1017 002624 010067 175156          MOV      R0,RTRAP5+2    ;SET UP NEW PSW IN VECTOR
1018 002630 005067 175142          CLR      STATUS        ;CLEAR PRESENT (OLD) STATUS
1019 002634 000100                   JMP      %0              ;DO TRAP
1020 002636                   8$:
1021 002636 000713                   BR      ERROR2          ;INSTRUCTION FAILED TO TRAP
1022 002640 020067 175132          1$:  CMP      R0,STATUS    ;IS NEW STATUS CORRECT
1023 002644 001310                   BNE     ERROR2          ;NEW STATUS WRONG
1024 002646 020627 000774          2$:  CMP      SP,#STBOT-4   ;DID STACK DECREMENT CORRECTLY
1025 002652 001305                   BNE     ERROR2          ;STACK DID NOT DECREMENT CORRECTLY
1026 002654 022767 002636 176112  3$:  CMP      #8$,STBOT-4    ;WAS PROPER PC SAVED
1027 002662 001301                   BNE     ERROR2          ;PROPER PC WAS NOT SAVED
1028 002664 005767 176106          4$:  TST      STBOT-2        ;WAS OLD PSW SAVED
1029 002670 001276                   BNE     ERROR2          ;WRONG PSW SAVED
1030 002672 062706 000004          5$:  ADD      #4,SP          ;RESET STACK POINTER
1031 002676 012767 002720 175100  MOV      #6$,RTRAP5     ;SET UP NEW PC IN VECTOR
1032 002704 005067 175076          CLR      RTRAP5+2        ;SET UP NEW PSW IN VECTOR
1033 002710 010037 177776          MOV      R0,#STATUS      ;SET UP OLD STATUS FOR COMPARISON AFTER TRAP
1034 002714 000100                   JMP      %0              ;DO TRAP
1035 002716 000600                   BR      ERROR2          ;INSTRUCTION FAILED TO TRAP
1036 002720 005767 175052          6$:  TST      STATUS        ;IS NEW PSW CORRECT
1037 002724 001260                   BNE     ERROR2          ;NEW PSW WRONG
1038 002726 020067 176044          7$:  CMP      R0,STBOT-2    ;WAS OLD STATUS STORED
```

```
1039 002732 001255          BNE      ERROR2          ;OLD STATUS WRONG
1040                          ;*****
1041                          ;TEST 10      TEST TRAP OF ALL ILLEGAL INSTRUCTION
1042                          ;*****
1043                          TS10:
1044 002734 012706 001000      MOV      #STBOT,SP        ;INITIALIZE THE STACK POINTER
1045 002740 012767 002762 175036  MOV      #1$,RTRAP5      ;SET UP NEW PC IN VECTOR
1046 002746 010067 175034      MOV      R0,RTRAP5+2    ;SET UP NEW PSW IN VECTOR
1047 002752 005067 175020      CLR      STATUS          ;CLEAR PRESENT (OLD) STATUS
1048 002756 004000              JSR      %0,%0          ;DO TRAP
1049 002760 8$:
1050 002760 000642              BR       ERROR2        ;INSTRUCTION FAILED TO TRAP
1051 002762 020067 175010 1$:  CMP      R0,STATUS      ;IS NEW STATUS CORRECT
1052 002766 001237              BNE     ERROR2        ;NEW STATUS WRONG
1053 002770 020627 000774 2$:  CMP      SP,#STBOT-4    ;DID STACK DECREMENT CORRECTLY
1054 002774 001234              BNE     ERROR2        ;STACK DID NOT DECREMENT CORRECTLY
1055 002776 022767 002760 175770 3$:  CMP      #8$,STBOT-4    ;WAS PROPER PC SAVED
1056 003004 001230              BNE     ERROR2        ;PROPER PC WAS NOT SAVED
1057 003006 005767 175764 4$:  TST     STBOT-2        ;WAS OLD PSW SAVED
1058 003012 001225              BNE     ERROR2        ;WRONG PSW SAVED
1059 003014 062706 000004 5$:  ADD     #4,SP          ;RESET STACK POINTER
1060 003020 012767 003042 174756  MOV      #6$,RTRAP5      ;SET UP NEW PC IN VECTOR
1061 003026 005067 174754      CLR      RTRAP5+2      ;SET UP NEW PSW IN VECTOR
1062 003032 010037 177776      MOV      R0,#STATUS     ;SET UP OLD STATUS FOR COMPARISON AFTER TRAP
1063 003036 004000              JSR      %0,%0          ;DO TRAP
1064 003040 000612              BR       ERROR2        ;INSTRUCTION FAILED TO TRAP
1065 003042 005767 174730 6$:  TST     STATUS         ;IS NEW PSW CORRECT
1066 003046 001207              BNE     ERROR2        ;NEW PSW WRONG
1067 003050 020067 175722 7$:  CMP      R0,STBOT-2    ;WAS OLD STATUS STORED
1068 003054 001204              BNE     ERROR2        ;OLD STATUS WRONG
1069
1070                          ;*****
1071                          ;SBTTL DATA PATH TESTS
1072                          ;
1073                          ; THE DATA PATH TESTS ARE USED TO VERIFY THAT VARIOUS
1074                          ; DATA PATTERNS CAN BE SUCCESSFULLY MOVED THROUGH THE DATA PATHS
1075                          ; MOVE AND COMPARE MODE 2,3 INSTRUCTIONS ARE USED TO PASS AND
1076                          ; TEST VARIOUS DATA PATTERNS IN THE DATA PATHS.
1077                          ; THE TEST EXERCISES THE INTERNAL DATA PATHS, AND THE UNIBUS
1078                          ; DATA TRANSCIEVERS.
1079                          ; IF THESE TESTS FAIL, EXAMINE THE TARGET LOCATION (LOC. 0)
1080                          ; TO SEE WHICH BITS OF THE DATA PATH ARE FAILING.
1081 003056 012737 002130 000030  MOV      #ERROR1,#30    ;SET UP VECTOR FOR ERROR CALLS
1082 003064 012737 000340 000032  MOV      #340,#32      ;SET UP NEW PSW
1083 003072 012737 021336 000004  MOV      #T04,#4       ;SET UP FOR UNEXPECTED TRAP TO 4
1084 003100 012737 021340 000010  MOV      #T010,#10     ;SET UP FOR UNEXPECTED TRAP TO 10
1085 003106 012737 021342 000014  MOV      #T014,#14     ;SET UP FOR UNEXPECTED TRAP TO 14
1086 003114 012737 021350 000034  MOV      #T034,#34     ;SET UP FOR UNEXPECTED TRAP TO 34
1087
1088                          ;*****
1089                          ;TEST 11      TEST OF ZEROES IN THE DATA PATH
1090                          ;*****
1091                          TS11:
1092 003122 012737 000000 000000  MOV      #0,#0         ;MOVE ZEROES THRU ADDRESS LINES, DATA
1093 003122 012737 000000 000000  ;LINES AND INTERNAL PATHS
1094
```

1095 003130 005737 000000
1096 003134 001401
1097 003136 104000
1098
1099
1100
1101
1102 003140
1103 003140 012737 125252 000000
1104
1105 003146 022737 125252 000000
1106 003154 001401
1107 003156 104000
1108
1109
1110
1111
1112 003160
1113 003160 012737 052525 000000
1114
1115 003166 022737 052525 000000
1116 003174 001401
1117 003176 104000
1118
1119
1120
1121
1122 003200
1123 003200 012737 177777 000000
1124 003206 022737 177777 000000
1125 003214 001401
1126 003216 104000
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145 003220
1146 003220 000241
1147 003222 012737 000001 000000
1148 003230 006137 000000
1149 003234 022737 000002 000000
1150 003242 001401

TST @#0 ;SUCCESSFUL?
BEQ TS12
EMT ;DATA INCORRECT
:*****
:TEST 12 TEST OF PATTERN 125252 IN DATA PATH
:*****
TS12:
MOV #125252,@#0 ;MOVE ALTERNATING ONES AND ZEROES
;THRU DATA PATHS
CMP #125252,@#0 ;SUCCESSFUL
BEQ TS13
EMT ;DATA INCORRECT
:*****
:TEST 13 TEST OF PATTERN 052525 IN DATA PATH
:*****
TS13:
MOV #052525,@#0 ;MOVE ALTERNATING ZEROES AND ONES
;THRU DATA PATH
CMP #052525,@#0 ;SUCCESSFUL?
BEQ TS14
EMT ;DATA INCORRECT
:*****
:TEST 14 TEST OF ALL ONES IN DATA PATH
:*****
TS14:
MOV #177777,@#0 ;MOVE ONES THRU DATA PATH
CMP #177777,@#0 ;SUCCESSFUL
BEQ TS15
EMT ;DATA INCORRECT
:*****
:SBTTL B-REGISTER TEST
:
: THE B-REGISTER (LOCATION 0) SHIFTING LOGIC TESTS ARE USED
: TO TEST THAT THE B-REGISTER CAN HOLD VARIOUS DATA PATTERNS AND THAT
: THE ASSOCIATED LOGIC SUPPORTS THE SHIFTING FUNCTIONS WITHIN THE
: B-REGISTER AND C-BIT.
: A ONE IS SHIFTED THROUGH EVERY BIT IN THE B-REGISTER AND C-BIT IN
: BOTH DIRECTIONS.
: THE B-REGISTER ITSELF IS TESTED IN ITS ABILITY AS A BUFFER AND AS
: A SHIFT REGISTER. DATA IS ALSO PASSED THROUGH THE DATA PATH AND ALU,
: IF THESE TESTS FAIL, EXAMINE THE TARGET LOCATION (LOC. 0) TO SEE
: WHICH BITS OF THE B-REGISTER MAY BE FAILING.
:*****
:TEST 15 SHIFT BIT 0 TO BIT 1
:*****
TS15:
CLC ;CLEAR CARRY BIT
MOV #1,@#0 ;LOAD A 1
ROL @#0 ;SHIFT LEFT
CMP #2,@#0 ;SUCCESSFUL
BEQ TS16

```
1151 003244 104000 EMT ;BIT 1 NOT SET
1152
1153 :*****
1154 :TEST 16 SHIFT CARRY INTO BIT 0
1155 :*****
1156 003246 TS16:
1157 003246 012737 000000 000000 MOV #0,@#0 ;CLEAR LOCATION
1158 003254 000261 SEC ;SET CARRY
1159 003256 006137 000000 ROL @#0 ;ROTATE CARRY BIT TO BIT 0
1160 003262 103001 BCC CARRY1
1161 003264 104000 EMT ;CARRY CLEAR
1162 003266 022737 000001 000000 CARRY1: CMP #1,@#0 ;BIT 0 SET
1163 003274 001401 BEQ TS17
1164 003276 104000 EMT ;BIT 0 NOT SET
1165
1166 :*****
1167 :TEST 17 LEFT SHIFT FROM BIT 0 TO C-BIT
1168 :*****
1169 003300 TS17:
1170 003300 012737 000001 000000 MOV #1,@#0 ;SET BIT 0
1171 003306 012700 177757 MOV #-21,R0 ;SET BIT COUNTER
1172 003312 000241 CLC ;CLEAR C-BIT
1173 003314 005200 SHL: INC R0 ;INCREMENT BIT COUNTER
1174 003316 001404 BEQ SHLE ;BR TO ERROR HALT IF BIT IS LOST
1175 003320 006137 000000 ROL @#0 ;SHIFT LEFT ONE POSITION
1176 003324 103373 BCC SHL ;BRANCH IF C-BIT NOT SET
1177 003326 001401 BEQ TS20
1178 003330 SHLE:
1179 003330 104000 EMT ;LEFT SHIFTING LOGIC FAILED
1180
1181 :*****
1182 :TEST 20 SHIFT BIT 15 TO BIT 14
1183 :*****
1184 003332 TS20:
1185 003332 012737 100000 000000 MOV #100000,@#0 ;SET BIT 15
1186 003340 000241 CLC ;CLEAR CARRY
1187 003342 006037 000000 ROR @#0 ;SHIFT BIT 15 TO BIT 14
1188 003346 022737 040000 000000 CMP #40000,@#0 ;SUCCESSFUL
1189 003354 001401 BEQ TS21
1190 003356 104000 EMT ;BIT 14 NOT SET
1191
1192 :*****
1193 :TEST 21 RIGHT SHIFT FROM BIT 15 TO C-BIT
1194 :*****
1195 003360 TS21:
1196 003360 012737 100000 000000 MOV #100000,@#0 ;SET BIT 15
1197 003366 012700 177757 MOV #-21,R0 ;SET BIT COUNTER
1198 003372 000241 CLC ;CLEAR C-BIT
1199 003374 005200 SHR: INC R0 ;INCREMENT BIT COUNTER
1200 003376 001404 BEQ SHRE ;BR TO ERROR HALT IF BIT IS LOST
1201 003400 006037 000000 ROR @#0 ;ROTATE RIGHT ONE POSITION
1202 003404 103373 BCC SHR ;BRANCH IF C-BIT CLEAR
1203 003406 001401 BEQ TS22
1204 003410 SHRE:
1205 003410 104000 EMT ;RIGHT SHIFT LOGIC FAILED
1206
```

1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230 003412
1231
1232 003412 012700 000000
1233 003416 005700
1234 003420 001401
1235 003422 104000
1236
1237
1238
1239
1240 003424
1241 003424 012700 125252
1242 003430 020027 125252
1243 003434 001401
1244 003436 104000
1245
1246
1247
1248
1249 003440
1250 003440 012700 052525
1251 003444 020027 052525
1252 003450 001401
1253 003452 104000
1254
1255
1256
1257
1258 003454
1259 003454 012700 177777
1260 003460 020027 177777
1261 003464 001401
1262 003466 104000

```
*****  
:SBTTL  SCRATCH PAD TESTS  
:  
:      THE SCRATCH PAD TESTS ARE USED TO VERIFY THAT VARIOUS  
:DATA PATTERNS CAN BE SUCCESSFULLY HELD IN THE SCRATCH PAD  
:CIRCUITRY.  MOVE AND COMPARE INSTRUCTIONS ARE USED TO TEST THAT  
:RO CAN HOLD VARIOUS DATA PATTERNS.  EACH DATA PATTERN IS  
:MOVED AND TESTED IN A SMALL LOOP CONVENIENT FOR SCOPING.  THE  
:SUCCESSFUL COMPLETION OF THESE TESTS SHOULD VERIFY THE CIRCUITRY EXTERNAL  
:TO THE SCRATCH PAD ITSELF.  
:      THE REMAINDER OF THE GENERAL REGISTERS ARE TESTED BY MOVING  
:A BIT INTO BIT 0 OF THE REGISTER AND SHIFTING IT LEFT ONE  
:BIT AT A TIME INTO THE CARRY BIT.  THE RESULT IS THEN CHECKED TO INSURE THAT  
:NO BITS WERE PICKED.  THE PROCEDURE IS THEN REPEATED UNDER OPPOSITE  
:CONDITIONS.  THE GENERAL REGISTER AND THE CARRY BIT ARE SET TO  
:ALL ONES, AND A ZERO IS SHIFTED LEFT FROM BIT 0 INTO THE CARRY BIT.  
:THE RESULT IS THEN CHECKED TO INSURE THAT NO ZEROES WERE PICKED.  
:      AT THIS POINT ALL OF THE GENERAL REGISTERS HAVE BEEN EXERCISED  
:AS WELL AS REGISTER 11.  
:  
:*****  
:TEST 22  TEST IF RO CAN HOLD ALL ZEROES  
:*****  
TS22:  
      MOV      #0,RO          ;MOVE ZEROES TO RO  
      TST      RO            ;SUCCESSFUL?  
      BEQ      TS23          ;RO NOT 0  
      EMT  
:  
:*****  
:TEST 23  TEST IF RO CAN HOLD ONES AND ZEROES  
:*****  
TS23:  
      MOV      #125252,RO     ;MOVE ALTERNATING ONES AND ZEROES TO RO  
      CMP      RO,#125252    ;SUCCESSFUL?  
      BEQ      TS24          ;RO NOT 125252  
      EMT  
:  
:*****  
:TEST 24  TEST IF RO CAN HOLD ZEROES AND ONES  
:*****  
TS24:  
      MOV      #052525,RO     ;MOVE ALTERNATING ZEROES AND ONES TO RO  
      CMP      RO,#052525    ;SUCCESSFUL?  
      BEQ      TS25          ;RO NOT 52525  
      EMT  
:  
:*****  
:TEST 25  TEST IF RO CAN HOLD ALL ONES  
:*****  
TS25:  
      MOV      #177777,RO     ;MOVE ALL ONES TO RO  
      CMP      RO,#177777    ;SUCCESSFUL?  
      BEQ      TS26          ;RO NOT 177777  
      EMT
```


1263
1264
1265
1266
1267 003470
1268 003470 012701 000001
1269 003474 012700 177757
1270 003500 000241
1271 003502 005200
1272 003504 001403
1273 003506 006101
1274 003510 103374
1275 003512 001401
1276 003514
1277 003514 104000
1278
1279
1280
1281
1282 003516
1283 003516 012701 177776
1284 003522 012700 177757
1285 003526 000261
1286 003530 005200
1287 003532 001405
1288 003534 006101
1289 003536 103774
1290 003540 012701 177777
1291 003544 001401
1292 003546
1293 003546 104000
1294
1295
1296
1297 003550
1298 003550 012702 000001
1299 003554 012700 177757
1300 003560 000241
1301 003562 005200
1302 003564 001403
1303 003566 006102
1304 003570 103374
1305 003572 001401
1306 003574
1307 003574 104000
1308
1309
1310
1311
1312 003576
1313 003576 012702 177776
1314 003602 012700 177757
1315 003606 000261
1316 003610 005200
1317 003612 001405
1318 003614 006102

```
*****  
:TEST 26 TEST IF R1 CAN HOLD A ONE IN ALL BITS  
*****  
TS26:  
MOV #1,R1 ;SET BIT 0  
MOV #-21,R0 ;SET BIT COUNTER  
CLC ;CLEAR C-BIT  
REG1: INC R0 ;INCREMENT BIT COUNTER  
BEQ REG1E ;BR TO ERROR HALT IF BIT IS LOST  
ROL R1 ;ROTATE 1 POSITION  
BCC REG1 ;ALL DONE  
BEQ TS27  
REG1E: EMT ;FAILURE WITH R1  
*****  
:TEST 27 TEST IF R1 CAN HOLD A ZERO IN ALL BITS  
*****  
TS27:  
MOV #-2,R1 ;SET ALL ONES IN R1 EXCEPT FOR BIT 0  
MOV #-21,R0 ;SET BIT COUNTER  
SEC ;SET C-BIT  
REG1A: INC R0 ;INCREMENT COUNTER  
BEQ R1ERR ;BR TO ERROR HALT IF COUNTER=0  
ROL R1 ;ROTATE 1 POSITION  
BCS REG1A ;CONTINUE UNTIL C-BIT IS CLEAR  
CMP #-1,R1 ;CHECK DATA IN R1  
BEQ TS30  
R1ERR: EMT ;FAILURE WITH R1  
*****  
:TEST 30 TEST IF R2 CAN HOLD A ONE IN ALL BITS  
*****  
TS30:  
MOV #1,R2 ;SET BIT 0  
MOV #-21,R0 ;SET BIT COUNTER  
CLC ;CLEAR C-BIT  
REG2: INC R0 ;INCREMENT BIT COUNTER  
BEQ REG2E ;BR TO ERROR HALT IF BIT IS LOST  
ROL R2 ;ROTATE 1 POSITION  
BCC REG2 ;ALL DONE  
BEQ TS31  
REG2E: EMT ;FAILURE WITH R2  
*****  
:TEST 31 TEST IF R2 CAN HOLD A ZERO IN ALL BITS  
*****  
TS31:  
MOV #-2,R2 ;SET ALL ONES IN R2 EXCEPT FOR BIT 0  
MOV #-21,R0 ;SET BIT COUNTER  
SEC ;SET C-BIT  
REG2B: INC R0 ;INCREMENT BIT COUNTER  
BEQ R2ERR ;BR TO ERROR HALT IF COUNTER=0  
ROL R2 ;ROTATE 1 POSITION
```

1319 003616 103774
1320 003620 022702 177777
1321 003624 001401
1322 003626
1323 003626 104000
1324

BCS REG2B ;CONTINUE UNTIL C-BIT IS CLEAR
CMP #1,R2 ;CHECK DATA IN R2
BEQ TS32
R2ERR: EMT ;FAILURE WITH R2

1325
1326
1327

:TEST 32 TEST IF R3 CAN HOLD A ONE IN ALL BITS

TS32:

1328 003630
1329 003630 012703 000001
1330 003634 012700 177757
1331 003640 000241
1332 003642 005200
1333 003644 001403
1334 003646 006103
1335 003650 103374
1336 003652 001401
1337 003654
1338 003654 104000
1339

MOV #1,R3 ;SET BIT 0
MOV #-21,R0 ;SET BIT COUNTER
CLC ;CLEAR C-BIT
REG3: INC R0 ;INCREMENT BIT COUNTER
BEQ REG3E ;BR TO ERROR HALT IF BIT IS LOST
ROL R3 ;ROTATE 1 POSITION
BCC REG3 ;ALL DONE
BEQ TS33
REG3E: EMT ;FAILURE WITH R3

1340
1341
1342

:TEST 33 TEST IF R3 CAN HOLD A ZERO IN ALL BITS

TS33:

1343 003656
1344 003656 012703 177776
1345 003662 012700 177757
1346 003666 000261
1347 003670 005200
1348 003672 001405
1349 003674 006103
1350 003676 103774
1351 003700 022703 177777
1352 003704 001401
1353 003706
1354 003706 104000
1355

MOV #-2,R3 ;SET ALL ONES IN R3 EXCEPT FOR BIT 0
MOV #-21,R0 ;SET BIT COUNTER
SEC ;SET C-BIT
REG3A: INC R0 ;INCREMENT BIT COUNTER
BEQ R3ERR ;BR TO ERROR HALT IF COUNTER=0
ROL R3 ;ROTATE 1 POSITION
BCS REG3A ;CONTINUE UNTIL C-BIT IS CLEAR
CMP #-1,R3 ;CHECK DATA
BEQ TS34
R3ERR: EMT ;FAILURE WITH R3

1356
1357
1358

:TEST 34 TEST IF R4 CAN HOLD A ONE IN ALL BITS

TS34:

1359 003710
1360 003710 012704 000001
1361 003714 012700 177757
1362 003720 000241
1363 003722 005200
1364 003724 001403
1365 003726 006104
1366 003730 103374
1367 003732 001401
1368 003734
1369 003734 104000
1370

MOV #1,R4 ;SET BIT 0
MOV #-21,R0 ;SET BIT COUNTER
CLC ;CLEAR C-BIT
REG4: INC R0 ;INCREMENT BIT COUNTER
BEQ REG4E ;BR TO ERROR HALT IF BIT IS LOST
ROL R4 ;ROTATE 1 POSITION
BCC REG4 ;ALL DONE
BEQ TS35
REG4E: EMT ;FAILURE WITH R4

1371
1372
1373
1374 003736

:TEST 35 TEST IF R4 CAN HOLD A ZERO IN ALL BITS

TS35:

1375 003736 012704 177776
1376 003742 012700 177757
1377 003746 000261
1373 003750 005200
1379 003752 001405
1380 003754 006104
1381 003756 103774
1382 003760 022704 177777
1383 003764 001401
1384 003766
1385 003766 104000
1386
1387
1388
1389
1390
1391 003770
1392 003770 012705 000001
1393 003774 012700 177757
1394 004000 000241
1395 004002 005200
1396 004004 001403
1397 004006 006105
1398 004010 103374
1399 004012 001401
1400 004014
1401 004014 104000
1402
1403
1404
1405
1406 004016
1407 004016 012705 177776
1408 004022 012700 177757
1409 004026 000261
1410 004030 005200
1411 004032 001405
1412 004034 006105
1413 004036 103774
1414 004040 022705 177777
1415 004044 001401
1416 004046
1417 004046 104000
1418
1419
1420
1421
1422 004050
1423 004050 012706 000001
1424 004054 012700 177757
1425 004060 000241
1426 004062 005200
1427 004064 001403
1428 004066 006106
1429 004070 103374
1430 004072 001401

MOV #2,R4 ;SET ALL ONES IN R4 EXCEPT FOR BIT 0
MOV #-21,R0 ;SET BIT COUNTER
SEC ;SET C-BIT
REG4A: INC R0 ;INCREMENT BIT COUNTER
BEQ R4ERR ;BR TO ERROR HALT IF COUNTER=0
ROL R4 ;ROTATE 1 POSITION
BCS REG4A ;CONTINUE UNTIL C-BIT IS CLEAR
CMP #-1,R4 ;CHECK DATA
BEQ TS36
R4ERR: EMT ;FAILURE WITH R4

:TEST 36 TEST IF R5 CAN HOLD A ONE IN ALL BITS

TS36:

MOV #1,R5 ;SET BIT 0
MOV #-21,R0 ;SET BIT COUNTER
CLC ;CLEAR C-BIT
REG5: INC R0 ;INCREMENT BIT COUNTER
BEQ REG5E ;BR TO ERROR HALT IF BIT IS LOST
ROL R5 ;ROTATE 1 POSITION
BCC REG5 ;ALL DONE
BEQ TS37
REG5E: EMT ;FAILURE WITH R5

:TEST 37 TEST IF R5 CAN HOLD A ZERO IN ALL BITS

TS37:

MOV #-2,R5 ;SET ALL ONES IN R5 EXCEPT FOR BIT 0
MOV #-21,R0 ;SET BIT COUNTER
SEC ;SET C-BIT
REG5A: INC R0 ;INCREMENT BIT COUNTER
BEQ R5ERR ;BR TO ERROR HALT IF COUNTER=0
ROL R5 ;ROTATE 1 POSITION
BCS REG5A ;CONTINUE UNTIL C-BIT IS CLEAR
CMP #-1,R5 ;CHECK DATA
BEQ TS40
R5ERR: EMT ;FAILURE WITH R5

:TEST 40 TEST IF R6 CAN HOLD A ONE IN ALL BITS

TS40:

MOV #1,R6 ;SET BIT 0
MOV #-21,R0 ;SET BIT COUNTER
CLC ;CLEAR C-BIT
REG6: INC R0 ;INCREMENT BIT COUNTER
BEQ REG6E ;BR TO ERROR HALT IF BIT IS LOST
ROL R6 ;ROTATE 1 POSITION
BCC REG6 ;ALL DONE
BEQ TS41

1431 004074
1432 004074 104000
1433
1434
1435
1436
1437 004076
1438 004076 012706 177776
1439 004102 012700 177757
1440 004106 000261
1441 004110 005200
1442 004112 001405
1443 004114 006106
1444 004116 103774
1445 004120 022706 177777
1446 004124 001401
1447 004126
1448 004126 104000
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467 004130
1468 004130 012706 001000
1469 004134 012737 000000 177776
1470 004142 005737 177776
1471 004146 001401
1472 004150 104000
1473
1474
1475
1476
1477 004152
1478 004152 012737 000252 177776
1479 004160 023727 177776 000252
1480 004166 001401
1481 004170 104000
1482
1483
1484
1485
1486 004172

```
REG6E:      EMT                ;FAILURE WITH R6

:*****
:TEST 41     TEST IF R6 CAN HOLD A ZERO IN ALL BITS
:*****
TS41:
MOV    #-2,R6      ;SET ALL ONES IN R6 EXCEPT FOR BIT 0
MOV    #-21,R0     ;SET BIT COUNTER
SEC                    ;SET C-BIT
REG6A:  INC    R0      ;INCREMENT BIT COUNT
BEQ    R6ERR       ;BR TO ERROR HALT IF COUNTER=0
ROL    R6          ;ROTATE 1 POSITION
BCS    REG6A       ;CONTINUE UNTIL C-BIT IS CLEAR
CMP    #-1,R6      ;CHECK DATA
BEQ    TS42

R6ERR:      EMT                ;FAILURE WITH R6

:*****
:SBTTL  PSW TESTS
:
:      THE PSW TESTS ARE USED TO VERIFY THAT VARIOUS DATA
: PATTERNS CAN BE SUCCESSFULLY HELD IN THE PSW AND THAT THE
: PSW ADDRESSING LOGIC IS FUNCTIONING. MOVE AND COMPARE INSTRUCTIONS
: ARE USED TO TEST THAT THE PSW CAN HOLD VARIOUS DATA PATTERNS.
: EACH DATA PATTERN IS MOVED AND TESTED IN A SMALL LOOP CONVENIENT FOR
: SCOPING.
:      THE PSW REGISTER IS TESTED, THE CC INPUTS ARE TESTED
: LATER IN THE MICROCODE TESTS. SETTING OF THE T-BIT BY THE
: TEST PATTERNS IS PURPOSELY AVOIDED. TESTING OF THE
: T-BIT TRAP CIRCUITRY IS LEFT FOR THE TRAP TEST.
:*****
:TEST 42     TEST IF PSW WILL HOLD ZEROES
:*****
TS42:
MOV    #STBOT,R6   ;SET PSW TO ZERO
MOV    #0,@#PS     ;SUCCESSFUL
TST    @#PS
BEQ    TS43
EMT                    ;PSW NOT 0

:*****
:TEST 43     TEST IF PSW WILL HOLD ONES AND ZEROES
:*****
TS43:
MOV    #252,@#PS   ;MOVE ALT. ONES AND ZEROES TO PSW
CMP    @#PS,#252   ;SUCCESSFUL?
BEQ    TS44
EMT                    ;PSW NOT 252

:*****
:TEST 44     TEST IF PSW (EXCEPT T-BIT) WILL HOLD ZEROES AND ONES
:*****
TS44:
```

1487 004172 012737 000105 177776
1488 004200 023727 177776 000105
1489 004206 001401
1490 004210 104000
1491
1492
1493
1494
1495 004212
1496 004212 012737 000357 177776
1497 004220 023727 177776 000357
1498 004226 001401
1499 004230 104000
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534 004232
1535 004232 005000
1536 004234 001401
1537 004236 104000
1538 004240 005200
1539 004242 005100
1540 004244 005200
1541 004246 100401
1542 004250 104000

```
MOV #105,@#PS ;MOVE ALT. ONES AND ZEROES TO PSW
CMP @#PS,#105 ;SUCCESSFUL?
BEQ TS45
EMT ;PSW NOT 105

:*****
:TEST 45 TEST IF PSW (EXCEPT T-BIT) WILL HOLD ALL ONES
:*****
TS45:
MOV #357,@#PS ;MOVE ONES TO PSW
CMP @#PS,#357 ;SUCCESSFUL
BEQ TS46
EMT ;PSW NOT 357

:*****
:SBTTL MICROCODE TESTS
:
: THE TEST EXERCISES BRANCHES IN THE MICROCODE BY
: TESTING AT LEAST ONE INSTRUCTION FROM EVERY CLASS OF INSTRUCTION IN
: ALL POSSIBLE MODES. FOR EXAMPLE, TO TEST THE SINGLE OPERAND INSTRUCTIONS,
: AT LEAST ONE SINGLE OPERAND INSTRUCTION IS VERIFIED IN ALL UNIQUE
: ADDRESSING MODES. BYTE MODES ARE ALSO TESTED. AS EACH NEW
: MODE IS INTRODUCED THE SAME INSTRUCTION IS TRIED AND TESTED IN
: A SMALL LOOP CONVENIENT FOR SCOPING. THE TEST IS SET UP USING
: ONLY INSTRUCTIONS AND ADDRESSING MODES WHICH HAVE BEEN PREVIOUSLY
: VERIFIED.
: IF THESE TESTS FAIL, CHECK THE RESULTS FOR A CLUE TO THE
: FAULT.
:*****

:*****
: THE CLR INSTRUCTION IS USED TO INTRODUCE EACH ADDRESSING
: MODE WITH THE SINGLE OPERAND INSTRUCTION. FOLLOWING THE SEQUENCE CHECK,
: THE CLR INSTRUCTION IS EXECUTED AND A BRANCH TEST IS EXECUTED WHICH
: CHECKS THAT THE Z-BIT WAS PROPERLY SET. THIS TEST CAN CHECK IR DECODE
: AND MICROCODE FOR SOP INSTRUCTIONS WITH MODE 0. FOLLOWING THIS TEST
: SEVERAL OTHER SOP INSTRUCTIONS ARE INTRODUCED WITH MODE 0. THESE
: INSTRUCTIONS MAINPULATE DATA AND SERVE TO CHECK THE DATA RESULTS
: OF THE SOP INSTRUCTIONS IN THIS TEST. THE DATA IN THIS TEST IS
: OPERATED ON BY EACH INSTRUCTION WITHOUT REINITIALIZING.
:*****
:TEST 46 TEST MODE 0 USING SOP INST.
:*****
TS46:
CLR RO ;TRY THE CLEAR INST.
BEQ SOPOA
EMT
SOP0A: INC RO ;CLR DID NOT SET Z-BIT
COM RO ;TRY THE INCREMENT INST.
INC RO ;TRY COMPLEMENT
BMI SOPOB
EMT ;NEGATE DID NOT SET N-BIT
```

1543 004252 005100
1544 004254 001401
1545 004256 104000

1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559

1560 004260
1561 004260 005000
1562 004262 005300
1563 004264 100401
1564 004266 104000
1565 004270 000261
1566 004272 005500
1567 004274 001007
1568 004276 000261
1569 004300 005600
1570 004302 100004
1571 004304 005100
1572 004306 005200
1573 004310 005300
1574 004312 001401
1575 004314
1576 004314 104000

1577
1578
1579

1580
1581
1582
1583
1584
1585
1586
1587 004316
1588 004316 105000
1589 004320 001401
1590 004322 104000
1591 004324 105100
1592 004326 100002
1593 004330 105200
1594 004332 001401
1595 004334
1596 004334 104000

1597
1598

SOP0B: COM R0 ;TRY COMPLEMENT INST.
BEQ TS47
EMT ;CUMMULATIVE RESULT OF CLR,INC,NEG AND COM INSTS. FAILED

: THIS TEST INTRODUCES THE REMAINING SOP INSTRUCTIONS AND TESTS
: THEM IN MODE 0. THE PURPOSE IS TO PROVIDE A BASELINE OF
: INSTRUCTIONS FOR USE IN THE SUBSEQUENT TESTS. SINCE THE MICROCODE FOR
: THESE INSTRUCTIONS IS IDENTICAL TO THAT ALREADY TESTED, ANY TROUBLE
: SHOOTING EFFORTS SHOULD BE AIMED AT THE ACTUAL IR DECODE AND ALU
: FUNCTIONING.

: TEST 47 TEST REMAINDER OF SOP INSTS IN MODE 0
: *****
TS47:

CLR R0 ;INITIALIZE
DEC R0 ;TRY DECREMENT INST.
BMI SOPOC
EMT ;N-BIT NOT SET ON DEC
SOP0C: SEC ;INITIALIZE CARRY
ADC R0 ;TRY ADD CARRY INST
BNE SOPOD
SEC ;INITIALIZE CARRY
SBC R0 ;TRY SUBTRACT-CARRY INST
BPL SOPOD
COM R0
INC R0
DEC R0
BEQ TS50
SOP0D: EMT ; CUMMULATIVE RESULT OF ADC,SBC,COM,INC AND DEC INSTS. F

: THIS TEST INTRODUCES THE BYTE CONTROL LOGIC OF THE PROCESSOR.
: THE MODE 0 BYTE MICROCODE IS TESTED. THE METHOD AND SEQUENCE
: OF TESTING IS THE SAME AS THAT USED IN THE SOP MODE 0 TESTS.

: TEST 50 TEST MODE 0 EVEN BYTE USING SOP INST
: *****
TS50:

CLRB R0 ;TRY CLEARING EVEN BYTE OF REGISTER
BEQ SOPBOA
EMT ;CLRB DID NOT SET Z-BIT
SOPBOA: COMB R0 ;TRY SETTING EVEN BYTE OF REGISTER
BPL SOPBOB
INCB R0 ;TRY INCREMENTING EVEN BYTE OF REGISTER>>
BEQ TS51
SOPBOB: EMT ;TEST CUMMULATIVE RESULT OF ABOVE BYTE INST.

1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609 004336
1610 004336 005000
1611 004340 005010
1612 004342 001401
1613 004344 104000
1614 004346 005310
1615 004350 100003
1616 004352 000261
1617 004354 005510
1618 004356 001401
1619 004360
1620 004360 104000
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632 004362
1633 004362 005000
1634 004364 005010
1635 004366 005110
1636 004370 105010
1637 004372 001401
1638 004374 104000
1639 004376 005210
1640 004400 100005
1641 004402 105110
1642 004404 105210
1643 004406 100002
1644 004410 105210
1645 004412 001401
1646 004414
1647 004414 104000
1648
1649
1650
1651
1652
1653
1654

```

:
: THIS TEST USES THE CLR INSTRUCTION TO INTRODUCE AND TEST
: SINGLE OPERAND MODE 1 INSTRUCTIONS. AGAIN, THE CLR INSTRUCTION
: IS USED TO INTRODUCE THE MICROCODE AND TO TEST THAT THE PROPER
: CONDITION CODES ARE SET. OTHER SOP INSTRUCTIONS ARE USED TO MANIPULATE
: COMMON DATA TO VERIFY THAT THE CORRECT DATA IS PRODUCED.
:
:*****
:TEST 51 TEST MODE 1 USING SOP INST.
:*****
TS51:
      CLR      R0          ;INITIALIZE R0
      CLR      (R0)       ;TRY CLEAR INST W/MODE 1
      BEQ      SOP1A
SOP1A: EMT              ;CLR DID NOT SET Z-BIT
      DEC      (R0)       ;TRY DECREMENT INST W/MODE 1
      BPL      SOP1B
      SEC
      ADC      (R0)       ;INITIALIZE CARRY
      BEQ      TS52       ;TRY ADD-CARRY W/MODE 1
SOP1B: EMT              ;TEST CUMMULATIVE RESULT OF ABOVE INST
:*****
:
: THIS TEST VERIFIES THE BYTE INSTRUCTION MICROCODE FOR MODE 1
: SINGLE OPERAND INSTRUCTIONS.
: THIS IS THE FIRST PLACE THE SIGN EXTEND LOGIC IS EXERCISED
: AND VERIFIED.
:
:*****
:TEST 52 TEST MODE 1 EVEN BYTE USING SOP INST
:*****
TS52:
      CLR      R0          ;INITIALIZE R0
      CLR      (R0)       ;INITIALIZE LOC. 0
      COM      (R0)
      CLRB     (R0)       ;TRY TO CLEAR BYTE 0
      BEQ      SOPB1A
SOPB1A: EMT              ;CLRB DID NOT SET Z-BIT
      INC      (R0)       ;INCREMENT TO TEST WORD
      BPL      SOPB1B
      COMB     (R0)       ;COMPLEMENT: ODD BYTE = 376
      INCB     (R0)       ;INC: ODD BYTE = 377
      BPL      SOPB1B
      INCB     (R0)       ;INCREMENT ODD BYTE=0
      BEQ      TS53
SOPB1B: EMT              ;CHECK CUMMULATIVE RESULT OF ABOVE INST
:*****
:
: THIS TEST VERIFIES THAT SINGLE OPERAND BYTE INSTRUCTIONS WILL
: FUNCTION CORRECTLY FOR ODD BYTES.
: THIS IS THE FIRST TIME THAT ADDRESS LINE 0 HAS BEEN
:
```

1655
 1656
 1657
 1658
 1659
 1660
 1661
 1662 004416
 1663 004416 005000
 1664 004420 005010
 1665 004422 005110
 1666 004424 005200
 1667 004426 105010
 1668 004430 001401
 1669 004432 104000
 1670 004434 005300
 1671 004436 005210
 1672 004440 005200
 1673 004442 105110
 1674 004444 105210
 1675 004446 100002
 1676 004450 105210
 1677 004452 001401
 1678 004454
 1679 004454 104000
 1680
 1681
 1682
 1683
 1684
 1685
 1686
 1687
 1688
 1689
 1690
 1691
 1692
 1693
 1694 004456
 1695 004456 005000
 1696 004460 105100
 1697 004462 005200
 1698 004464 005010
 1699 004466 005110
 1700 004470 005020
 1701 004472 001401
 1702 004474 104000
 1703 004476 005300
 1704 004500 005300
 1705 004502 005120
 1706 004504 100004
 1707 004506 005300
 1708 004510 005300
 1709 004512 005220
 1710 004514 001401

```

;EXERCISED. CHECKS ARE MADE THAT THE PROPER BYTE IS MODIFIED AND
;THE CONDITION CODES ARE CHECKED. IT IS ALSO VERIFIED THAT THE UNADDRESSED
;BYTE IS NOT ALTERED BY THE INSTRUCTION.
:
:*****
;TEST 53          TEST MODE 1 ODD BYTE USING SOP INST
:*****
TS53:
      CLR      RO          ;INITIALIZE RO
      CLR      (RO)       ;INITIALIZE LOC. 0
      COM      (RO)
      INC      RO          ;RO=ODD BYTE
      CLRB     (RO)       ;TRY TO CLEAR BYTE 1
      BEQ      SOPB1C
      EMT
SOPB1C: DEC      RO          ;CLRB DID NOT SET Z-BIT
        INC      (RO)       ;RO=WORD ADDR.
        INC      RO          ;INCREMENT TO TEST WORD
        INC      RO          ;RO=ODD BYTE
        COMB     (RO)       ;TRY TO COMPLEMENT BYTE 1
        INCB     (RO)
        BPL      SOPB1D
        INCB     (RO)       ;TRY TO INCREMENT BYTE 1
        BEQ      TS54
SOPB1D: EMT                ;TEST CUMMULATIVE RESULT OF ABOVE INST.
:
:*****
: THIS TEST VERIFIES MODE 2 SINGLE-OPERAND INSTRUCTIONS. PREVIOUSLY
;TESTED INSTRUCTIONS ARE USED TO SET A POINTER IN RO TO LOC. 400.
;LOC. 400 IS INITIALIZED TO -1 BEFORE A CLR MODE 2 IS EXECUTED.
; THEN RO IS DECREMENTED BY TWO TO AGAIN POINT TO 400 BEFORE EACH
;OF SEVERAL MODE 2 INSTRUCTIONS ARE USED TO VERIFY THE DATA RESULTS OF
;THE TEST. THIS PROCEDURE ALSO VERIFIES THE PROPER INCREMENTING OF THE
;REGISTER.
:
:*****
;TEST 54          TEST MODE 2 USING SOP INST.
:*****
TS54:
      CLR      RO          ;SET RO=400
      COMB     RO
      INC      RO
      CLR      (RO)       ;CLEAR 400
      COM      (RO)       ;INITIALIZE: 400=-1
      CLR      (RO)+      ;TRY CLEARING WITH MODE 2
      BEQ      SOPZA
      EMT
SOPZA: DEC      RO          ;CLR INST DID NOT SET Z-BIT
        DEC      RO          ;RESET RO
        COM      (RO)+      ;TRY COMPLEMENTING WITH MODE 2
        BPL      SOP2B
        DEC      RO          ;RESET RO
        DEC      RO
        INC      (RO)+      ;TRY INCREMENTING WITH MODE 2
        BEQ      TS55
  
```



```

1711 004516
1712 004516 104000
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727 004520
1728 004520 005000
1729 004522 105100
1730 004524 005200
1731 004526 005010
1732 004530 005110
1733 004532 105020
1734 004534 001401
1735 004536 104000
1736 004540 005300
1737 004542 005210
1738 004544 105110
1739 004546 105220
1740 004550 100003
1741 004552 005300
1742 004554 105220
1743 004556 001401
1744 004560
1745 004560 104000
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755 004562
1756 004562 005000
1757 004564 105100
1758 004566 005200
1759 004570 005010
1760 004572 005110
1761 004574 005200
1762 004576 105020
1763 004600 001401
1764 004602 104000
1765 004604 005300
1766 004606 005300
  
```

```

SOP2B:
  EMT                                ;CHECK CUMMULATIVE RESULT OF ABOVE INST

:*****
:
:      THIS TEST VERIFIES MODE 2 SINGLE OPERAND INSTRUCTIONS WHICH
:ADDRESS EVEN BYTES.  RO IS SET TO 400 AND USED TO INITIALIZE LOCATION
:400 TO -1.  CLRB INSTRUCTION IS THEN EXECUTED ON BYTE 400 WITH
:MODE 2.
:      RO IS THEN DECREMENTED BEFORE EACH OF SEVERAL MODE 2 INSTRUCTIONS
:WHICH ARE USED TO VERIFY THE DATA RESULTS OF THE TEST.  THIS PROCEDURE ALSO
:VERIFIES THE PROPER INCREMENTING OF THE REGISTER.
:*****
:TEST 55      TEST MODE 2 EVEN BYTE USING SOP INST.
:*****
TS55:
  CLR      RO                        ;SET RO=400
  COMB    RO
  INC     RO
  CLR     (RO)                       ;CLEAR 400
  COM     (RO)                       ;INITIALIZE: 400=-1
  CLRB    (RO)+                      ;TRY TO CLEAT 400 W/MODE 2
  BEQ     SOPB2A
  EMT
  SOPB2A: DEC     RO                  ;CLR DID NOT SET Z-BIT
          INC     (RO)               ;RESULT RO=400
          COMB    (RO)               ;INC 400 TO TEST WORD
          INCB   (RO)+               ;TRY TO INC EVEN BYTE
          BPL    SOPB2B
          DEC     RO                  ;RESET RO=400
          INCB   (RO)+               ;TRY INCREMENT OF EVEN BYTE
          BEQ     TS56
  SOPB2B: EMT                        ;TEST CUMMULATIVE RESULT OF ABOVE INST.

:*****
:
:      THIS TEST FOLLOWS THE SAME PROCEDURE DESCRIBED IN THE PREVIOUS
:TEST.  HERE, THE BYTE INSTRUCTION IS USED TO ADDRESS AN ODD BYTE.
:*****
:TEST 56      TEST MODE 2 ODD BYTE USING SOP INST.
:*****
TS56:
  CLR      RO                        ;SET RO=400
  COMB    RO
  INC     RO
  CLR     (RO)                       ;CLEAR LOC 400
  COM     (RO)                       ;INITIALIZE: 400=-1
  INC     RO                          ;RO=ODD BYTE
  CLRB    (RO)+                      ;TRY TO CLEAR ODD BYTE
  BEQ     SOPB2C
  EMT
  SOPB2C: DEC     RO                  ;CLRB DID NOT SET Z-BIT
          DEC     RO                  ;RO=WORD ADDR.
  
```

1767 004610 005220
1768 004612 005300
1769 004614 105110
1770 004616 105220
1771 004620 100003
1772 004622 005300
1773 004624 105220
1774 004626 001401
1775 004630
1776 004630 104000
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786 004632
1787 004632 005000
1788 004634 005200
1789 004636 005400
1790 004640 100003
1791 004642 001402
1792 004644 102401
1793 004646 103401
1794 004650
1795 004650 104000
1796
1797 004652 005200
1798 004654 001401
1799 004656 104000
1800
1801 004660 105100
1802 004662 105400
1803 004664 100403
1804 004666 001402
1805 004670 102401
1806 004672 103401
1807 004674
1808 004674 104000
1809 004676 005300
1810 004700 001401
1811 004702 104000
1812
1813
1814
1815 004704
1816 004704 005000
1817 004706 005010
1818 004710 005210
1819 004712 005410
1820 004714 100003
1821 004716 001402
1822 004720 102401

```
INC (R0)+ ;INCREMENT WORD
DEC R0 ;POINT TO ODD BYTE
COMB (R0) ;COMPLEMENT ODD BYTE
INCB (R0)+ ;TRY TO INCREMENT ODD BYTE
BPL SOPB2D
DEC R0 ;RESET R0 TO ODD BYTE
INCB (R0)+ ;TRY TO INCREMENT ODD BYTE
BEQ TS57

SOPB2D: EMT ;TEST CUMMULATIVE RESULT OF ABOVE INST.

:*****
: THESE TESTS CHECK THE NEGATE INSTRUCTION IN ALL MODES. PREVIOUSLY
: TESTED SINGLE-OPERAND INSTRUCTIONS ARE USED TO TEST THE NEGATE INSTRUCTION.
:*****
:TEST 57 TEST MODE 0 USING NEGATE INSTRUCTION
:*****
TS57:
CLR R0 ;SET R0=0
INC R0 ; R0=1
NEG R0 ;TRY NEGATE MODE 0: R0=-1
BPL NEG00 ;CC=1001?
BEQ NEG00
BVS NEG00
BCS NEG01

NEG00: EMT ;NEGATE DID NOT SET CC'S CORRECTLY

NEG01: INC R0 ;TEST DATA RESULT
BEQ NEG02
EMT ;DATA RESULT OF NEGATE INCORRECT

NEG02: COMB R0 ;R0=377
NEGB R0 ;R0=1
BMI NEG03 ;CC=0001?
BEQ NEG03
BVS NEG03
BCS NEG04

NEG03: EMT ;NEGB DID NOT SET CC'S CORRECTLY

NEG04: DEC R0 ;TEST DATA RESULT
BEQ TS60
EMT ;DATA RESULT OF NEGB INCORRECT

:*****
:TEST 60 TEST MODE 1 USING NEGATE INST.
:*****
TS60:
CLR R0 ;POINT TO LOC. 0
CLR (R0) ;CLEAR LOC. 0
INC (R0) ;LOC. 0=1
NEG (R0) ;TRY NEG. LOC. 0=-1
BPL NEG10 ;CC=1001
BEQ NEG10
BVS NEG10
```

1823 004722 103401
1824 004724
1825 004724 104000
1826
1827 004726 005237 000000
1828 004732 001401
1829 004734 104000
1830 004736 105110
1831 004740 105410
1832 004742 100403
1833 004744 001402
1834 004746 102401
1835 004750 103401
1836 004752
1837 004752 104000
1838 004754 005337 000000
1839 004760 001401
1840 004762 104000

BCS NEG11
NEG10: EMT ;NEGATE DID NOT SET CC'S CORRECTLY
NEG11: INC @#0 ;TEST DATA RESULT
BEQ NEG12
EMT ;DATA RESULT OF NEGATE INCORRECT
NEG12: COMB (R0) ;LOC. 0=377
NEGB (R0) ;TRY NEGB LOC. 0=1
BMI NEG13 ;CC=0001?
BEQ NEG13
BVS NEG13
BCS NEG14
NEG13: EMT ;NEGB DID NOT SET CC'S CORRECTLY
NEG14: DEC @#0 ;TEST DATA RESULT
BEQ TS61
EMT ;DATA RESULT OF NEGB INCORRECT

1841
1842
1843
1844 004764
1845 004764 005000
1846 004766 005010
1847 004770 005210
1848 004772 005420
1849 004774 100003
1850 004776 001402
1851 005000 102401
1852 005002 103401
1853 005004
1854 005004 104000
1855 005006 105300
1856 005010 105300
1857 005012 105420
1858 005014 105420
1859 005016 105340
1860 005020 005300
1861 005022 001401
1862 005024 104000
1863 005026 005337 000000
1864 005032 001401
1865 005034 104000
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878

;TEST 61 TEST MODE 2 USING NEGATE INSTRUCTION

TS61:
CLR R0 ;POINT TO LOC. 0
CLR (R0) ;CLEAR LOC. 0
INC (R0) ;LOC. 0=1
NEG (R0)+ ;TRY NEG.: LOC. 0=-1
BPL NEG20 ;CC=1001?
BEQ NEG20
BVS NEG20
BCS NEG21
NEG20: EMT ;NEGATE DID NOT SET CC'S CORRECTLY
NEG21: DECB R0 ;RO=LOC. 0
DECB R0
NEGB (R0)+ ;BYTE 0=1 RO=1
NEGB (R0)+ ;BYTE 1=1 RO=2
DECB -(R0) ;RO=1 LOC. 0=01
DEC R0 ;RO=0
BEQ NEG22
EMT ;REGISTER NOT INCREMENTED CORRECTLY
NEG22: DEC @#0 ;LOC. 0=0
BEQ TS62
EMT ;NEG BYTE INSTRUCTIONS FAILED

: THIS TEST VERIFIES MODE 3 SINGLE OPERAND INSTRUCTIONS. IT
: USES LOCATION 0 AS ITS TARGET DATA. A TABLE LOCATED AT LOC. 400
: THRU 402 IS USED TO SUPPLY THE ADDRESS OF LOCATION 0 TO THE
: INSTRUCTIONS UNDER TEST.
: RO IS SET TO 400, THE START OF THE ADDRESS TABLE, AND A CLR
: INSTRUCTION IS EXECUTED WITH MODE 3 TO CLEAR LOC. 0. THEN RO
: IS DECREMENTED BY TWO AND TWO OTHER MODE 3 INSTRUCTIONS OPERATE ON
: LOC. 0 TO VERIFY THE DATA RESULTS OF THE TEST. THE PROPER INCREMENTING
: OF THE REGISTER IS ALSO VERIFIED IN THIS MANNER.
: IF A FAILURE IS DETECTED BE SURE TO VERIFY THAT THE TABLE

1879
1880
1881
1882
1883
1884 005036
1885 005036 005000
1886 005040 105100
1887 005042 005200
1888 005044 005010
1889 005046 005030
1890 005050 001401
1891 005052 104000
1892 005054 005300
1893 005056 005300
1894 005060 005130
1895 005062 100002
1896 005064 005230
1897 005066 001401
1898 005070
1899 005070 104000
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916 005072
1917 005072 005004
1918 005074 105104
1919 005076 005204
1920 005100 005000
1921 005102 005010
1922 005104 005110
1923 005106 105034
1924 005110 001401
1925 005112 104000
1926 005114 005304
1927 005116 005304
1928 005120 005234
1929 005122 100006
1930 005124 105434
1931 005126 100004
1932 005130 005304
1933 005132 005304
1934 005134 105234

```
;(LOC. 400-402) HAS THE PROPER VALUES (0).  
:*****  
:TEST 62 TEST MODE 3 USING SOP INST.  
:*****  
TS62:  
CLR R0 ;SET R0=400  
COMB R0  
INC R0  
CLR (R0) ;CLEAR LOC 400  
CLR @(R0)+ ;TRY TO CLEAR LOC 0 USING MODE 3 ;R0=402  
BEQ SOP3A  
EMT ;CLR DID NOT SET Z-BIT  
SOP3A: DEC R0 ;RESET R0=400  
DEC R0  
COM @(R0)+ ;TRY TO COMPLEMENT LOC 0 OF MODE 3 ;R0=402  
BPL SOP3B  
INC @(R0)+ ;TRY TO INCREMENT LOC 0 W/MODE 3 ;R0=404  
BEQ TS63  
SOP3B: EMT ;CUMMULATIVE RESULT OF ABOVE INST FAILED  
:*****  
: THIS TEST VERIFIES MODE 3 SINGLE OPERAND BYTE INSTRUCTIONS  
: WHICH ADDRESS EVEN BYTES. AGAIN, THE TARGET LOCATION 0 IS USED  
: AND THE SAME TABLE AT 400 IS EMPLOYED.  
: AFTER POINTING R4 TO THE TABLE (400) AND SETTING LOCATION  
: 0 TO -1, A CLRB INSTRUCTION IS USED TO CLEAR BYTE 0.  
: SEVERAL OTHER MODE 3 INSTRUCTIONS ARE THEN USED WITH THE TABLE  
: TO VERIFY THE DATA RESULTS AND THE PROPER INCREMENTING OF THE REGISTER.  
: IF A FAILURE IS DETECTED, BE SURE THAT THE TABLE (LOCATION 400-402) HAS  
: THE PROPER VALUES (0).  
:*****  
:TEST 63 TEST MODE 3 EVEN BYTE USING SOP INST.  
:*****  
TS63:  
CLR R4 ;SET R4=400  
COMB R4  
INC R4  
CLR R0 ;INITIALIZE LOC. 0=-1  
CLR (R0)  
COM (R0) ;LOC. 0=-1  
CLRB @(R4)+ ;TRY TO CLEAR EVEN BYTE ;LOC. 0=177400 R4=402  
BEQ SOPB3A  
EMT ;CLRB DID NOT SET Z-BIT  
SOPB3A: DEC R4 ;RESET POINTER R4=400  
DEC R4  
INC @(R4)+ ;TRY INCREMENTING WORD LOC.0=177401 R4=402  
BPL SOPB3B  
NEGB @(R4)+ ;TRY TO NEGATE EVEN BYTE ;LOC.0=-1 R4=404  
BPL SOPB3B  
DEC R4 ;R4=402  
DEC R4  
INCB @(R4)+ ;TRY TO INCREMENT EVEN BYTE ;LOC. 0=17400
```

CJKDJBO 11/23-B CPU CLUSTER DIAG.
CJKDJB.P11 26-MAY-82 11:14

DNMAC X24.07-563 26-MAY-82^{L 3} 11:18 PAGE 38
T63 TEST MODE 3 EVEN BYTE USING SOP INST.

SEQ 0037

1935 005136 001401
1936 005140
1937 005140 104000
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951

BEQ TS64
SOPB3B: EMT ;CUMMULATIVE RESULT OF ABOVE INST FAILED

: THIS TEST VERIFIES MODE 3 SINGLE OPERAND BYTE INSTRUCTIONS
: WHICH ADDRESS ODD BYTES. THE TARGET IS BYTE 1. A TABLE AT
: LOC. 400-406 IS USED. RO SERVES AS THE TABLE POINTER.
: RO IS INITIALIZED TO 400. LOC. 0 IS SET TO -1 USING THE
: FIRST TWO TABLE ENTRIES. A CLRB MODE 3 IS EXECUTED ON BYTE 1 USING
: TABLE ADDRESS AT 404. RO IS DECREMENTED TO 402 AND SEVERAL SOP
: MODE 3 INSTRUCTIONS ARE USED TO VERIFY DATA RESULTS AND PROPER
: REGISTER INCREMENTING.
: THE TABLE (400-406) SHOULD CONTAIN 0,0,1,1 BEFORE AND
: AFTER THE TEST IS RUN.
:

CJKDJBO 11/23-B CPU CLUSTER DIAG.
CJKDJB.P11 26-MAY-82 11:14

M 3

DNMAC X24.07-563 26-MAY-82 11:18 PAGE 39
T63 TEST MODE 3 EVEN BYTE USING SOP INST.

SEQ 0038

1952
1953
1954
1955 005142
1956 005142 005000
1957 005144 105100
1958 005146 005200
1959 005150 005030
1960 005152 005130
1961 005154 105030
1962 005156 001401
1963 005160 104000
1964 005162 005300
1965 005164 005300
1966 005166 005300
1967 005170 005300
1968 005172 005230

```
*****
:TEST 64          TEST MODE 3 ODD BYTE USING SOP INST.
*****
TS64:
      CLR        R0          ;SET R0=400
      COMB       R0
      INC        R0
      CLR        @(R0)+     ;INITIALIZE
      COM        @(R0)+     ;LOC 0=-1 R0=404
      CLRB       @(R0)+     ;TRY TO CLEAR ODD BYTE LOC. 0=377 R0=406
      BEQ        SOPB3C
      EMT
SOPB3C: DEC       R0        ;CLRB DID NOT SET Z-BIT
      DEC       R0        ;RESET R0=402
      DEC       R0
      DEC       R0        ;POINT TO EVEN BYTE ADDR.
      INC       @(R0)+     ;INCREMENT WORD LOC. 0=400 R0=404
```

1969 005174 105430
1970 005176 100002
1971 005200 105230
1972 005202 001401
1973 005204
1974 005204 104000
1975
1976
1977
1978 005206
1979 005206 005000
1980 005210 105100
1981 005212 005200
1982 005214 005010
1983 005216 005004
1984 005220 005014
1985 005222 005214
1986 005224 005430
1987 005226 100003
1988 005230 001402
1989 005232 102401
1990 005234 103401
1991 005236
1992 005236 104000
1993 005240 005214
1994 005242 001401
1995 005244 104000
1996 005246 105137 000001
1997 005252 005237 000000
1998 005256 105430
1999 005260 100401
2000 005262 104000
2001 005264 105430
2002 005266 100001
2003 005270 104000
2004 005272 105137 000001
2005 005276 105237 000001
2006 005302 005214
2007 005304 001401
2008 005306 104000
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021 005310
2022 005310 005000
2023 005312 105100
2024 005314 005200

NEGB @ (R0)+ ;TRY TO NEGATE ODD BYTE LOC. 0=177400 R0=406
BPL SOPB3D
INCB @ (R0)+ ;TRY TO INCREMENT ODD BYTE LOC.0=0 R0=410
BEQ TS65
SOPB3D:
EMT ;CUMMULATIVE RESULT OF ABOVE INSTS FAILED
:*****
:TEST 65 TEST MODE 3 USING NEGATE INSTRUCTION
:*****
TS65:
CLR R0 ;R0=400
COMB R0
INC R0
CLR (R0) ;LOC. 400=0
CLR R4 ;R4=0
CLR (R4) ;LOC. 0=0
INC (R4) ;LOC. 0=1
NEG @ (R0)+ ;TRY NEGATE LOC. 0=-1 R0=402
BPL NEG30 ;CC=1001?
BEQ NEG30
BVS NEG30
BCS NEG31
NEG30:
EMT ;NEG DID NOT SET CC'S CORRECTLY
NEG31:
INC (R4) ;LOC. 0=0
BEQ NEG32
NEG32:
COMB @#1 ;DATA RESULT OF NEG INCORRECT
INC @#0 ;LOC 0=177400
NEGB @ (R0)+ ;LOC. 0=177401
BMI NEG33 ;TRY NEGB LOC. 0=177777 R0=404
EMT ;NEGB FAILED WITH EVEN BYTE
NEG33:
NEGB @ (R0)+ ;TRY NEGB LOC.0=777 R0=406
BPL NEG34
EMT ;NEGB FAILED WITH ODD BYTE
NEG34:
COMB @#1 ;LOC. 0=177377
INCB @#1 ;LOC. 0=177777
INC (R4) ;LOC. 0=0
BEQ TS66
EMT ;DATA RESULT OF NEGB'S INCORRECT
:*****
: THIS TEST VERIFIES MODE 4 SINGLE OPERAND INSTRUCTIONS.
:RO IS SET TO 400. A CLR INSTRUCTION IS EXECUTED IN MODE 4 TO CLEAR
:LOC. 376. RO IS RESET TO 400 AND A COM INSTRUCTION USING MODE 4
:COMPLEMENTS LOC.376.
: TWO INC INSTRUCTIONS AND A MODE 4 INSTRUCTION ARE EXECUTED
:TO COMPLETE THE TEST.
:*****
:TEST 66 TEST MODE 4 USING SOP INSTS
:*****
TS66:
CLR R0 ;SET R0=400
COMB R0
INC R0

2025 005316 005040
2026 005320 001401
2027 005322 104000
2028 005324 005200
2029 005326 005200
2030 005330 005140
2031 005332 100004
2032 005334 005200
2033 005336 005200
2034 005340 005240
2035 005342 001401
2036 005344
2037 005344 104000

CLR -(R0) ;TRY TO CLEAR USING MODE 4
BEQ SOP4A
SOP4A: INC R0 ;CLR DID NOT SET Z-BIT
INC R0 ;RESET R0
COM -(R0) ;TRY TO COMPLEMENT USING MODE 4
BPL SOP4B
INC R0 ;MOVE POINTER
INC R0
INC -(R0)
BEQ TS67
SOP4B: EMT ;CHECK CUMMULATIVE RESULT OF ABOVE INST.

2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057 005346
2058 005346 012700 000370
2059 005352 005020
2060 005354 005020
2061 005356 005020
2062 005360 005010
2063 005362 005000
2064 005364 005020
2065 005366 105400
2066 005370 005050
2067 005372 001401
2068 005374 104000
2069 005376 005200
2070 005400 005200
2071 005402 005150
2072 005404 100002
2073 005406 005250
2074 005410 001401
2075 005412
2076 005412 104000
2077
2078
2079
2080

: THIS TEST VERIFIES MODE 5 SINGLE OPERAND INSTRUCTIONS. IT
: USES LOCATION 0 AS ITS TARGET DATA. A TABLE LOCATED AT LOC. 372
: THRU 374 IS USED TO SUPPLY THE ADDRESS OF LOCATION 0 TO THE
: INSTRUCTIONS UNDER TEST.
: R0 IS SET TO 376, (THE START OF THE ADDRESS TABLE) +2,
: AND A CLR INSTRUCTION IS EXECUTED WITH MODE 3 TO CLEAR
: LOC. 0. THEN R0 IS INCREMENTED BY TWO AND TWO OTHER MODE 3
: INSTRUCTIONS OPERATE ON LOC. 0 TO VERIFY THE DATA RESULTS OF
: THE TEST. THE PROPER DECREMENTING OF THE REGISTER IS ALSO
: VERIFIED IN THIS MANNER.
: IF A FAILURE IS DETECTED BE SURE TO VERIFY THAT THE TABLE
: (LOC. 372 THRU 374) HAS THE PROPER VALUES (0).
: *****
: TEST 67 TEST MODE 5 USING SOP INSTS
: *****
TS67: MOV #370,R0 ;CLEAR LOCATION 370-376
CLR (R0)+ ;370
CLR (R0)+ ;372
CLR (R0)+ ;374
CLR (R0) ;376
CLR R0 ;SET R0=376 (LOW BYTE)
CLR (R0)+
NEGB R0
CLR @-(R0) ;TRY TO CLEAR LOC 0 W/MODE 5
BEQ SOP5A
EMT ;CLR DID NOT SET Z-BIT
SOP5A: INC R0 ;RESET R0
INC R0
COM @-(R0) ;TRY TO COMPLEMENT LOC. 0 W/MODE 5
BPL SOP5B
INC @-(R0) ;TRY TO INCREMENT LOC. 0 W/MODE 5
BEQ TS70
SOP5B: EMT ;TEST CUMMULATIVE RESULT OF ABOVE INSTS

: THIS TEST VERIFIES MODE 6 SINGLE OPERAND INSTRUCTIONS. IT

2081
2082
2083
2084
2085
2086
2087
2088
2089 005414
2090 005414 005000
2091 005416 105100
2092 005420 005200
2093 005422 005060 177400
2094 005426 001401
2095 005430 104000
2096 005432 005160 177400
2097 005436 100003
2098 005440 005260 177400
2099 005444 001401
2100 005446
2101 005446 104000
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115 005450
2116 005450 005000
2117 005452 105100
2118 005454 005200
2119 005456 005210
2120 005460 005070 000002
2121 005464 001401
2122 005466 104000
2123 005470 005170 000002
2124 005474 100003
2125 005476 005270 000002
2126 005502 001401
2127 005504
2128 005504 104000
2129
2130
2131
2132
2133 005506
2134 005506 005000
2135 005510 005010
2136 005512 005120

:USES LOCATION 0 AS ITS TARGET DATA. R0 IS SET TO 400 USING
:PREVIOUSLY TESTED INSTRUCTIONS AND A MODE 6 CLR INSTRUCTION IS
:EXECUTED ON LOC. 0 USING R0 AND A -400 OFFSET. COM AND INC
:INSTRUCTIONS ARE THEN USED TO VERIFY THE DATA.

:TEST 70 TEST MODE 6 USING SOP INSTS

TS70:

CLR R0 ;SET R0=400
COMB R0
INC R0
CLR -400(R0) ;TRY TO CLEAR LOCATION 0 W/MODE 6
BEQ SOP6A
EMT ;CLR DID NOT SET Z-BIT
SOP6A: COM -400(R0) ;TRY TO COMPLEMENT LOCATION 0 W/MODE 6
BPL SOP6B
INC -400(R0) ;TRY TO INCREMENT LOCATION 0 W/MODE 6
BEQ TS71
SOP6B: EMT ;TEST CUMMULATIVE RESULT OF ABOVE INSTS

: THIS TEST VERIFIES MODE 7 SINGLE OPERAND INSTRUCTIONS. IT USES
: THE POINTER TO LOC. 0 WHICH IS STORED AT LOC. 402.
: R0 IS SET TO 400 AND A MODE 7 CLR INSTRUCTION IS
: EXECUTED WITH A +2 OFFSET TO CLEAR LOC. 0.
: SEVERAL OTHER MODE 7 INSTRUCTIONS ARE THEN USED ON THE COMMON
: LOCATION TO VERIFY THE DATA RESULTS.

:TEST 71 TEST MODE 7 USING SOP INST.

TS71:

CLR R0 ;SET R0=400
COMB R0
INC R0
INC (R0) ;R0=1
CLR @2(R0) ;TRY TO CLEAR LOC. 0 W/MODE 7
BEQ SOP7A
EMT ;CLR DID NOT SET Z-BIT
SOP7A: COM @2(R0) ;TRY TO COMPLEMENT LOC. 0 W/MODE 7
BPL SOP7B
INC @2(R0) ;TRY TO INCREMENT LOC. 0 W/MODE 7
BEQ TS72
SOP7B: EMT ;TEST CUMMULATIVE RESULT OF ABOVE INSTS.

:TEST 72 TEST MODE 4 WITH NEGATE INSTRUCTION

TS72:

CLR R0
CLR (R0)
COM (R0)+ ;LOC. 0=177777, R0=2

2137 005514 005440
2138 005516 100403
2139 005520 001402
2140 005522 102401
2141 005524 103401
2142 005526
2143 005526 104000
2144 005530 005400
2145 005532 001401
2146 005534 104000
2147 005536 005310
2148 005540 001401
2149 005542 104000
2150
2151
2152
2153 005544
2154 005544 005000
2155 005546 005010
2156 005550 105100
2157 005552 005200
2158 005554 005010
2159 005556 005004
2160 005560 005314
2161 005562 005450
2162 005564 100403
2163 005566 001402
2164 005570 102401
2165 005572 103401
2166 005574
2167 005574 104000
2168 005576 005314
2169 005600 001401
2170 005602 104000
2171 005604 105100
2172 005606 005300
2173 005610 001401
2174 005612 104000
2175
2176
2177
2178 005614
2179 005614 005000
2180 005616 005004
2181 005620 105100
2182 005622 005014
2183 005624 105024
2184 005626 105114
2185 005630 005460 177401
2186 005634 100403
2187 005636 001402
2188 005640 102401
2189 005642 103401
2190 005644
2191 005644 104000
2192 005646 105314

```
NEG -(R0) ;TRY NEGATE, LOC. 0=1
BMI NEG40 ;CC=0001?
BEQ NEG40
BVS NEG40
BCS NEG41
NEG40:
EMT ;NEG DID NOT SET CC'S CORRECTLY
NEG41: NEG R0 ;TST R0 WITH A NEG.
BEQ NEG42
EMT ;R0 NOT DECREMENTED PROPERLY
NEG42: DEC (R0) ;TEST DTA RESULT OF NEG
BEQ TS73
EMT ;DATA RESULT OF NEG INCORRECT
:*****
:TEST 73 TEST MODE 5 WITH NEGATE INSTRUCTION
:*****
TS73:
CLR R0 ;R0=0
CLR (R0) ;LOC. 0=0
COMB R0 ;R0=377
INC R0 ;R0=400
CLR (R0) ;SET 400 = 0
CLR R4 ;R4=0
DEC (R4) ;LOC. 0=177777
NEG @-(R0) ;TRY NEGATE: LOC. 0=1
BMI NEG50 ;CC=0001?
BEQ NEG50
BVS NEG50
BCS NEG51
NEG50:
EMT ;NEG DID NOT SET CC'S CORRECTLY
NEG51: DEC (R4)
BEQ NEG52
EMT ;DATA RESULT OF NEG INCORRECT
NEG52: COMB R0
DEC R0
BEQ TS74
EMT ;REGISTER NOT DECREMENTED PROPERLY
:*****
:TEST 74 TEST MODE 6 WITH NEGATE
:*****
TS74:
CLR R0 ;R0=0
CLR R4 ;R4=0
COMB R0 ;R0=377
CLR (R4) ;LOC. 0=0
CLRB (R4)+ ;LOC. 0=177777, R4=1
COMB (R4) ;LOC. 0=177400
NEG -377(R0) ;LOC. 0=400
BMI NEG60 ;CC=0001
BEQ NEG60
BVS NEG60
BCS NEG61
NEG60:
EMT ;NEG DID NOT SET CC'S CORRECTLY
NEG61: DECB (R4)
```

2193 005650 001401
2194 005652 104000
2195
2196
2197
2198 005654
2199 005654 005000
2200 005656 005010
2201 005660 005110
2202 005662 105100
2203 005664 105470 000005
2204 005670 100403
2205 005672 001402
2206 005674 102401
2207 005676 103401
2208 005700
2209 005700 104000
2210 005702 105100
2211 005704 105120
2212 005706 105310
2213 005710 005467 172064
2214 005714 001401
2215 005716 104000
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228 005720
2229 005720 005027
2230 005722 177777
2231 005724 001401
2232 005726 104000
2233 005730 005237 005722
2234 005734 005467 177762
2235 005740 100003
2236 005742 005277 000004
2237 005746 001402
2238 005750
2239 005750 104000
2240 005752 005722
2241
2242
2243
2244
2245
2246
2247
2248

```
BEQ TS75
EMT ;DATA RESULT OF NEG INCORRECT
;*****
;TEST 75 TEST MODE 7 W/ NEGATE
;*****
TS75:
CLR R0 ;R0=0
CLR (R0) ;LOC. 0=0
COM (R0) ;LOC. 0=177777
COMB R0 ;R0=377
NEGB @5(R0) ;R0+5=404, 404=1, LOC. 0=777
BMI NEG70 ;CC=0001?
BEQ NEG70
BVS NEG70
BCS NEG71
NEG70:
EMT ;NEG DID NOT SET CC'S CORRECTLY
NEG71: COMB R0 ;R0=0
COMB (R0)+ ;LOC. 0=400, R0=1
DECB (R0) ;LOC. 0=0
NEG 0 ;USE NEG MODE 67 TO TST FOR ZERO
BEQ TS76
EMT ;DATA RESULT OF NEG WAS INCORRECT
;*****
;
; THIS TEST VERIFIES PROGRAM COUNTER ADDRESSING WITH SOP
; INSTRUCTIONS. CLR MODE 77 IS USED TO CLEAR THE LOCATION FOLLOWING THE
; INSTRUCTION (SOPX). THEN SINGLE OPERAND INSTRUCTIONS WITH MODES 37, 67, AND
; 77, USING INDIRECT POINTER SOPXAD ARE USED TO VERIFY THE DATA RESULTS
; OF THESE INSTRUCTIONS.
;*****
;TEST 76 TEST SOP INSTRUCTIONS MODES 2,3,6,7 WITH REGISTER 7
;*****
TS76:
CLR (R7)+ ;CLEAR NEXT LOCATION: (SOPX)
SOPX: -1 ;USE MODE 27
BEQ SOPA
EMT ;CLR DID NOT SET Z-BIT
SOPA: INC @SOPX ;INC SOPX W/MODE 37
NEG SOPX ;NEGATE SOPX W/MODE 67
BPL SOPB
INC @SOPXAD ;INC SOPX W/MODE 77
BEQ TS77
SOPB:
EMT ;INC DID NOT SET Z-BIT
SOPXAD: SOPX ;INDIRECT ADDRESS OF SOPX
;*****
;
; THIS TEST VERIFIES SINGLE OPERAND NON-MODIFYING INSTRUCTIONS
; USING MODE 0. R0 IS SET TO ZERO AND THE CONDITION CODES ARE SET
; TO THE COMPLEMENT OF THAT EXPECTED BY THE INSTRUCTION. A TST INSTRUCTION
; IS EXECUTED AND CONDITIONAL BRANCHES ARE USED TO TEST THE CONDITION
; CODES.
```

2249
2250
2251
2252
2253 005754
2254 005754 005000
2255 005756 000277
2256 005760 000244
2257 005762 005700
2258 005764 102403
2259 005766 100402
2260 005770 103401
2261 005772 001401
2262 005774
2263 005774 104000
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276 005776
2277 005776 005000
2278 006000 105100
2279 006002 000277
2280 006004 000250
2281 006006 105700
2282 006010 102402
2283 006012 101401
2284 006014 100401
2285 006016
2286 006016 104000
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299 006020
2300 006020 005000
2301 006022 005010
2302 006024 000277
2303 006026 000244
2304 006030 005710

```
*****
:TEST 77      TEST MODE 0 SOP NON-MODIFYING
*****
TS77:
      CLR      RO          ;INITIALIZE RO=0
      SCC          ;SET CC=1011
      CLZ
      TST      RO          ;TRY TST W/ MODE 0
      BVS      SNMOA       ;CHECK T; T CC=0100
      BMI      SNMOA
      BCS      SNMOA
      BEQ      TS100
SNMOA:
      EMT          ;CONDITION CODES NOT SET PROPERLY
*****
:
:      THIS TEST VERIFIES SINGLE OPERAND NON-MODIFYING BYTE INSTRUCTIONS WITH MODE 0.
:RO IS SET TO 377 AND COMPLEMENT OF THE EXPECTED CONDITION CODES
:IS LOADED IN PSW. A TSTB INSTRUCTION IS EXECUTED AND THE RESULTS
:ARE CHECKED WITH SEVERAL CONDITIONAL BRANCH INSTRUCTIONS.
:      THIS VERIFIES THAT THE PROPER BYTE WAS TESTED.
*****
:TEST 100     TEST MODE 0 EVEN BYTE W/ SOP NON-MODIFYING
*****
TS100:
      CLR      RO          ;INITIALIZE
      COMB     RO          ;RO=377
      SCC          ;SET CC=0111
      CLN
      TSTB     RO          ;TRY TST EVEN BYTE
      BVS      SNMBOA      ;CHECK CC=1000
      BLOS     SNMBOA
      BMI      TS101
SNMBOA:
      EMT          ;CONDITION CODES NOT SET PROPERLY
*****
:
:      THIS TEST VERIFIES SINGLE OPERAND INSTRUCTIONS WITH MODE 1.
:RO IS USED TO POINT TO AND CLEAR LOC. 0. THE COMPLEMENT OF THE
:EXPECTED CONDITION CODES ARE LOADED IN THE PSW. A TST INSTRUCTION
:IS THEN EXECUTED ON LOC. 0 USING RO AND CONDITIONAL BRANCHES TEST
:THE RESULTS.
*****
:TEST 101     TEST MODE 1 SOP NON-MODIFYING
*****
TS101:
      CLR      RO          ;POINT TO LOC 0
      CLR      (RO)        ;CLEAR LOC 0
      SCC          ;INITIALIZE
      CLZ          ;CC=1011
      TST      (RO)        ;TRY TST W/ MODE 1
```

2305 006032 102403
2306 006034 103402
2307 006036 100401
2308 006040 001401
2309 006042
2310 006042 104000
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322 006044
2323 006044 005000
2324 006046 005010
2325 006050 105110
2326 006052 000277
2327 006054 000250
2328 006056 105710
2329 006060 102402
2330 006062 101401
2331 006064 100401
2332 006066
2333 006066 104000
2334 006070 005000
2335 006072 005200
2336 006074 000277
2337 006076 000244
2338 006100 105710
2339 006102 102403
2340 006104 103402
2341 006106 100401
2342 006110 001401
2343 006112
2344 006112 104000
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356 006114
2357 006114 005000
2358 006116 005010
2359 006120 000277
2360 006122 000244

BVS SNM1A ;CHECK CC=0100
BCS SNM1A
BMI SNM1A
BEQ TS102
SNM1A:
EMT ;CC'S NOT SET PROPERLY

: THIS TEST SETS LOCATION 0 TO 377 AND THEN USES R0 TO TEST
: THE EVEN BYTE AND THE ODD BYTE USING SOP BYTE INSTRUCTIONS WITH MODE 1.
: AGAIN, CONDITIONAL BRANCHES ARE USED TO VERIFY THE SETTING OF THE
: PROPER CONDITION CODE BITS.

: TEST 102 TEST MODE 1 BYTE INST. NON-MODIFYING

TS102:

CLR R0 ;POINT TO LOC 0
CLR (R0) ;CLEAR LOC 0
COMB (R0) ;COMPLEMENT BYTE 0
SCC ;SET CC=0111
CLN
TSTB (R0) ;TRY TST ON EVEN BYTE
BVS SNMB1A
BLOS SNMB1A
BMI SNMB1B

SNMB1A:
EMT ;CC'S NOT CORRECT

SNMB1B: CLR R0
INC R0
SCC ;SET CC=1011
CLZ
TSTB (R0) ;TRY TO TST AN ODD BYTE
BVS SNMB1C ;CHECK CC=0100
BCS SNMB1C
BMI SNMB1C
BEQ TS103

SNMB1C:
EMT ;CC'S NOT CORRECT

: THIS TEST VERIFIES THE SINGLE-OPERAND NON-MODIFYING INSTRUCTIONS
: USING MODE 2. IT USES THE IDENTICAL PROCEDURE EMPLOYED IN THE
: MODE 1 TESTS. ADDITIONALLY, THE REGISTER IS CHECKED TO ASSURE THAT
: IT IS INCREMENTED PROPERLY.

: TEST 103 TEST MODE 2 WITH SOP NON-MODIFYING

TS103:

CLR R0 ;INITIALIZE R0=0
CLR (R0) ;CLEAR LOC 0
SCC ;SET CC=1011
CLZ

2361 006124 005720
 2362 006126 102403
 2363 006130 103402
 2364 006132 100401
 2365 006134 001401
 2366 006136
 2367 006136 104000
 2368 006140 005300
 2369 006142 005300
 2370 006144 001401
 2371 006146 104000
 2372
 2373
 2374
 2375
 2376
 2377
 2378
 2379
 2380
 2381
 2382
 2383
 2384 006150
 2385 006150 005000
 2386 006152 005010
 2387 006154 105110
 2388 006156 000277
 2389 006160 000250
 2390 006162 105720
 2391 006164 102402
 2392 006166 101401
 2393 006170 100401
 2394 006172
 2395 006172 104000
 2396 006174 005300
 2397 006176 001401
 2398 006200 104000
 2399 006202 005200
 2400 006204 000277
 2401 006206 000244
 2402 006210 105720
 2403 006212 102403
 2404 006214 103402
 2405 006216 100401
 2406 006220 001401
 2407 006222
 2408 006222 104000
 2409 006224 005300
 2410 006226 005300
 2411 006230 001401
 2412 006232 104000
 2413
 2414
 2415
 2416

TST (R0)+ ;TRY TST W/ MODE 2
 BVS SNM2A ;CHECK CC=0100
 BCS SNM2A
 BMI SNM2A
 BEQ SNM2B
 SNM2A:
 EMT ;CC'S NOT CORRECT
 SNM2B: DEC R0 ;RESET R0
 DEC R0
 BEQ TS104
 EMT ;MODE 2 DID NOT INC REG CORRECTLY

 :
 : THIS TEST VERIFIES MODE 2 SINGLE OPERAND NON-MODIFYING BYTE
 : INSTRUCTIONS IT USES R0 TO POINT TO LOC. 0. WITH LOCATION 0
 : SET TO 377, THE EVEN AND ODD BYTE IS TESTED WITH TSTB INSTRUCTIONS
 : TO VERIFY THE CORRECT CC ARE SET. THE REGISTER IS CHECKED FOR
 : PROPER INCREMENTING.
 :
 : *****

TEST 104 TEST MODE 2 - BYTE W/ SOP NON-MODIFYING

 :
 : TS104:
 : *****
 CLR R0 ;CLEAR R0
 CLR (R0) ;CLEAR LOC 0
 COMB (R0) ;SET LOC 0=377
 SCC ;SET CC=0111
 CLN
 TSTB (R0)+ ;TRY TST OF EVEN BYTE
 BVS SNMB2A
 BLOS SNMB2A
 BMI SNMB2B
 SNMB2A:
 EMT ;CC'S NOT SET CORRECTLY
 SNMB2B: DEC R0 ;DECREMENT R0
 BEQ SNMB2C
 EMT ;MODE 2 DID NOT INC REG CORRECTLY
 SNMB2C: INC R0 ;POINT TO ODD BYTE
 SCC ;SET CC=1011
 CLZ
 TSTB (R0)+ ;TRY TST OF ODD BYTE
 BVS SNMB2D ;CHECK CC'S=0100
 BCS SNMB2D
 BMI SNMB2D
 BEQ SNMB2E
 SNMB2D:
 EMT ;CC'S NOT CORRECT
 SNMB2E: DEC R0
 DEC R0
 BEQ TS105
 EMT ;R0 DID NOT INCREMENT PROPERLY

 :
 : THIS TEST VERIFIES MODE 3 SINGLE OPERAND NON-MODIFYING INSTRUCTIONS.
 :
 : *****

2417
2418
2419
2420
2421
2422
2423
2424 006234
2425 006234 005000
2426 006236 005010
2427 006240 105100
2428 006242 005300
2429 006244 000277
2430 006246 000244
2431 006250 005730
2432 006252 102403
2433 006254 103402
2434 006256 100401
2435 006260 001401
2436 006262
2437 006262 104000
2438 006264 005300
2439 006266 105100
2440 006270 001401
2441 006272 104000
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455 006274
2456 006274 005000
2457 006276 005010
2458 006300 105110
2459 006302 105100
2460 006304 005200
2461 006306 005720
2462 006310 000277
2463 006312 000250
2464 006314 105730
2465 006316 102402
2466 006320 101401
2467 006322 100401
2468 006324
2469 006324 104000
2470 006326 000277
2471 006330 000244
2472 006332 105730

:A POINTER IN A TABLE AT LOC. 376 IS USED TO TEST LOCATION 0.
:THE CC'S AND THE REGISTER ARE CHECKED FOLLOWING THE
:TST MODE 3 INSTRUCTION.

:TEST 105 TEST MODE 3 W/ SOP NON-MODIFYING INSTS

TS105:
CLR R0 ;R0=0
CLR (R0) ;CLEAR LOC 0
COMB R0 ;R0=376
DEC R0
SCC ;SET CC=1011
CLZ
TST @(R0)+ ;TRY TST W/ MODE 3
BVS SNM3A ;CHECK CC=0100
BCS SNM3A
BMI SNM3A
BEQ SNM3B
SNM3A: EMT ;CC'S NOT CORRECT
SNM3B: DEC R0 ;R0=377
COMB R0 ;R0=0
BEQ TS106
EMT ;MODE 3 DID NOT INC REG CORRECTLY

: THIS TEST VERIFIES SOP NON-MODIFYING BYTE INSTRUCTIONS MODE 3
: LOC. 0 IS SET TO 377. TABLE AT LOC. 402-404 IS USED TO TEST
: BYTE 0 AND BYTE 1. THE REGISTER IS CHECKED FOR PROPER INCREMENTING AND
: THE CC'S ARE VERIFIED.
: THE TABLE AT LOC. 402-404 SHOULD CONTAIN 0 AND 1 BEFORE AND
: AFTER THE TEST IS RUN.

:TEST 106 TEST MODE 3 - BYTES W/ SOP NON-MODIFYING INSTS.

TS106:
CLR R0 ;R0=0
CLR (R0) ;CLEAR LOC 0
COMB (R0) ;LOC. 0 =377
COMB R0
INC R0
TST (R0)+ ;R0=402
SCC ;CC=0111
CLN
TSTB @(R0)+ ;TRY TST OF EVEN BYTE
BVS SNM3A ;CHECK CC=1000
BLOS SNM3A
BMI SNM3B
SNM3A: EMT ;CC'S NOT CORRECT
SNM3B: SCC ;SET CC=1011
CLZ
TSTB @(R0)+ ;TRY TST OF ODD BYTE

2473 006334 102403
2474 006336 103402
2475 006340 100401
2476 006342 001401
2477 006344
2478 006344 104000
2479 006346 005720
2480 006350 005710
2481 006352 100401
2482 006354 104000

BVS SNMB3C ;CHECK CC=0100
BCS SNMB3C
BMI SNMB3C
BEQ SNMB3D
SNMB3C: EMT ;CC'S NOT CORRECT
SNMB3D: TST (R0)+ ;R0=410
TST (R0)
BMI TS107
EMT ;TSTB DID NOT INCREMENT R0 CORRECTLY

2483
2484
2485
2486
2487
2488
2489
2490

THIS TEST VERIFIES MODE 4 SOP NON-MODIFYING INSTRUCTIONS.
LOC. 0 IS SET TO -1 AND THE CC'S ARE SET TO THE COMPLEMENT OF THE
EXPECTED RESULTS. R0 AND SET TO 2 AND A TST MODE 4 IS EXECUTED.
THE CC'S ARE CHECKED WITH CONDITIONAL BRANCH INSTRUCTIONS AND THE REGISTER
IS CHECKED FOR PROPER DECREMENTING.

2491
2492
2493

TEST 107 TEST MODE 4 W/ SOP NON-MODIFYING INSTS

2494 006356
2495 006356 005000
2496 006360 005010
2497 006362 005120
2498 006364 000277
2499 006366 000244
2500 006370 005740
2501 006372 102402
2502 006374 101401
2503 006376 100401
2504 006400
2505 006400 104000
2506 006402 005700
2507 006404 001401
2508 006406 104000
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519

TS107: CLR R0 ;R0=0
CLR (R0) ;LOC 0=0
COM (R0)+ ;LOC 0=-1
SCC ;SET CC=1011
CLZ
TST -(R0) ;TRY TST W/ MODE 4
BVS SNM4A ;CHECK CC=0100
BLOS SNM4A
BMI SNM4B
SNM4A: EMT ;CC'S NOT CORRECT
SNM4B: TST R0
BEQ TS110
EMT ;TST MODE 4 DID NOT DEC R0 CORRECTLY

2511
2512
2513
2514
2515
2516
2517
2518
2519

THIS TEST VERIFIES MODE 5 SOP NON-MODIFYING INSTRUCTIONS.
IT USES A POINTER AT LOC. 376 TO TEST LOC. 0. R0 IS SET
TO 400, A TST MODE 5 INSTRUCTION IS EXECUTED AND THE CC'S CHECKED.
R0 IS CHECKED TO INSURE PROPER DECREMENTING.

2520 006410
2521 006410 005000
2522 006412 005010
2523 006414 005110
2524 006416 105100
2525 006420 005200
2526 006422 000277
2527 006424 000250
2528 006426 005750

TEST 110 TEST MODE 5 W/ SOP NON-MODIFYING INSTS

TS110: CLR R0 ;R0=0
CLR (R0) ;LOC 0=0
COM (R0) ;LOC 0=-1
COMB R0 ;R0=377
INC R0 ;R0=400
SCC ;SET CC=0111
CLN
TST @-(R0) ;TRY TST W/ MODE 5

2529 006430 102402
2530 006432 101401
2531 006434 100401
2532 006436
2533 006436 104000
2534 006440 005200
2535 006442 105100
2536 006444 001401
2537 006446 104000

BVS SNM5A :CHECK CC=1000
BLOS SNM5A
BMI SNM5B
SNM5A: EMT :CC'S NOT SET PROPERLY
SNM5B: INC RO :RO=377
COMB RO :RO=0
BEQ TS111
EMT :MODE 5 DID NOT DEC RO CORRECTLY

2538
2539
2540
2541
2542
2543
2544
2545
2546
2547

: THIS TEST VERIFIES MODE 6 SOP NON-MODIFYING INSTRUCTIONS.
: RO IS SET TO 377 AND A MODE 6 TST INSTRUCTION IS EXECUTED
: USING RO AND AN OFFSET OF -377. THE CC'S ARE CHECKED AS WELL
: AS RO TO INSURE IT WAS NOT ALTERED.

2548
2549 006450
2550 006450 005000
2551 006452 005010
2552 006454 005110
2553 006456 105100
2554 006460 000277
2555 006462 000250
2556 006464 005760 177401
2557 006470 102402
2558 006472 101401
2559 006474 100401
2560 006476
2561 006476 104000
2562 006500 105100
2563 006502 001401
2564 006504 104000

: TEST 111 TEST MODE 6 W/ SOP NON-MODIFYING INSTS

TS111:
CLR RO :RO=0
CLR (RO) :LOC 0=0
COM (RO) :LOC 0=-1
COMB RO :RO=377
SCC :SET CC=0111
CLN
TST -377(RO) :TRY TST W/ MODE 6
BVS SNM6A :CHECK CC=1000
BLOS SNM6A
BMI SNM6B
SNM6A: EMT :CC'S INCORRECT
SNM6B: COMB RO :RO=0
BEQ TS112
EMT :TST MODE 6 INCORRECTLY CHANGED RO

2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576 006506
2577 006506 005000
2578 006510 005010
2579 006512 005110
2580 006514 105100
2581 006516 000277
2582 006520 000250
2583 006522 005770 000001
2584 006526 102402
2585 006530 101401
2586 006532 100401
2587 006534
2588 006534 104000
2589 006536 105100
2590 006540 001401
2591 006542 104000
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602 006544
2603 006544 005000
2604 006546 005100
2605 006550 005004
2606 006552 060004
2607 006554 005204
2608 006556 001401
2609 006560 104000
2610
2611
2612
2613
2614
2615
2616
2617
2618 006562
2619 006562 005000
2620 006564 005004

: THIS TEST VERIFIES MODE 7 SOP NON-MODIFYING INSTRUCTIONS.
: IT USES A POINTER TO LOC. 0 STORED AT LOC. 400 TO TST LOC. 0.
: R0 IS SET TO 377 AND LOC. 0 IS TESTED THRU THE POINTER AT 400 USING
: R0 AND AN OFFSET OF 1.

: TEST 112 TEST MODE 7 W/ SOP NON-MODIFYING INSTS.

TS112:
CLR R0 ;R0=0
CLR (R0) ;LCC 0=0
COM (R0) ;LOC 0=-1
COMB R0 ;R0=377
SCC ;CC=0111
CLN
TST @1(R0) ;TRY TST W/ MODE 7
BVS SNM7A ;CHECK CC=1000
BLOS SNM7A
BMI SNM7B
SNM7A:
EMT ;CC'S NOT CORRECT
SNM7B: COMB R0 ;R0=0
BEQ TS113
EMT ;TST MODE 7 INCORRECTLY CHANGED R0

: THIS TEST VERIFIES MODE 0 DOUBLE OPERAND INSTRUCTIONS. IT SETS
: DATA IN R0 AND R4 AND USES THE ADD INSTRUCTION TO TEST THE DOP
: MICROCODE.

: TEST 113 TEST MODE 0 DOUBLE-OPERAND (DOP) INSTS.

TS113:
CLR R0 ;R0=0
COM R0 ;R0=-1
CLR R4 ;R4=0
ADD R0,R4 ;TRY ADD: R4=-1
INC R4 ;R4=0
BEQ TS114
EMT ;ADD INST. FAILED W/ MODE 0

: THIS TEST VERIFIES THE MOVE INSTRUCTION WITH MODE 0 TO MODE 0.

: TEST 114 MOV MODE 0 TO MODE 0

TS114:
CLR R0 ;R0=0
CLR R4 ;R4=0

2621 006566 005100
2622 006570 010004
2623 006572 005204
2624 006574 001401
2625 006576 104000
2626
2627
2628
2629
2630
2631
2632
2633
2634 006600
2635 006600 005000
2636 006602 005004
2637 006604 005204
2638 006606 160400
2639 006610 100003
2640 006612 001402
2641 006614 102401
2642 006616 103401
2643 006620
2644 006620 104000
2645 006622 005200
2646 006624 001401
2647 006626 104000
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660 006630
2661 006630 005000
2662 006632 010004
2663 006634 001401
2664 006636 104000
2665 006640 005200
2666 006642 005100
2667 006644 005104
2668 006646 040004
2669 006650 005304
2670 006652 001401
2671 006654 104000
2672 006656 050004
2673 006660 005204
2674 006662 005204
2675 006664 001401
2676 006666 104000

```
COM R0 ;R0=-1
MOV R0,R4 ;TRY MOVE -1 TO R4
INC R4 ;INC R4
BEQ TS115
EMT ;MOVE FAILED MODE 0 TO MODE 0

:*****
: THIS TEST VERIFIES THE SUBTRACT INSTRUCTION WITH MODE 0,0.
:*****
:TEST 115 TEST SUB MODE 0,0
:*****
TS115:
CLR R0 ;R0=0
CLR R4 ;R4=0
INC R4 ;R4=1
SUB R4,R0 ;TRY SUB 0,0 R0=-1
BPL SUB0 ;CC=1001
BEQ SUB0
BVS SUB0
BCS SUB0A

SUB0:
EMT ;CONDITION CODE FAILED ON SUB

SUB0A:
INC R0
BEQ TS116
EMT ;DATA RESULT OF SUB FAILED

:*****
: THIS TEST QUICKLY VERIFIES THE REMAINING DOP MODIFYING INSTRUCTIONS
: WITH MODE 0,0 TO PROVIDE A BASELINE FOR SUBSEQUENT TESTS.
: SINGLE OPERAND INSTRUCTIONS ARE USED TO SET UP DATA IN R0 AND R4
: BEFORE EACH OF THE SEVERAL DOP MODIFYING INSTRUCTIONS ARE USED AND
: VERIFIED.
:*****
:TEST 116 TEST ALL THE DOP INSTRUCTIONS W/ SOURCE MODE 0,0
:*****
TS116:
CLR R0 ;R0=0
MOV R0,R4 ;TRY MOVE MODE 0,0
BEQ DOPOA
EMT ;Z-BIT NOT SET
DOPOA:
INC R0 ;R0=1
COM R0 ;R0=177776
COM R4 ;R4=177777
BIC R0,R4 ;TRY BIC: R4=1
DEC R4 ;R4=0
BEQ DOPOB
EMT ;BIC CLEAR RESULT INCORRECT
DOPOB:
BIS R0,R4 ;TRY BIS: R4=177777
INC R4
INC R4 ;R4=0
BEQ DOPOC
EMT ;RESULT OF BIS INCORRECT
```


2733 007004 020004
2734 007006 002401
2735 007010 104000
2736 007012 005200
2737 007014 020400
2738 007016 001401
2739 007020 104000
2740 007022 005000
2741 007024 005100
2742 007026 005004
2743 007030 030004
2744 007032 001401
2745 007034 104000
2746 007036 005304
2747 007040 030004
2748 007042 100401
2749 007044 104000

DNM1: CMP R0,R4 ;TRY COMPARE R0 TO R4
BLT DNM2 ;
EMT ;CC'S NOT CORRECT FOR CMP
DNM2: INC R0 ;R0=1
CMP R4,R0 ;TRY COMPARE R4=1 TO R0=1
BEQ DNM3 ;
EMT ;CC'S NOT CORRECT (Z=1) FOR CMP
DNM3: CLR R0 ;R0=0
COM R0 ;R0=177777
CLR R4 ;R4=0
BIT R0,R4 ;TRY BIT R0 TO R4
BEQ DNM4 ;
EMT ;CC'S NOT CORRECT FOR BIT
DNM4: DEC R4 ;R4=177777
BIT R0,R4 ;TRY BIT AGAIN
BMI TS121 ;
EMT ;CC'S NOT CORRECT FOR BIT

2750
2751
2752
2753
2754
2755
2756
2757

: THIS TEST VERIFIES MODE 0,X DOUBLE OPERAND NON-MODIFYING INSTRUCTIONS.
: IT SETS DATA IN R0 AND LOCATION 0 AND COMPARES THEM USING DOPNM INSTRUCTIONS.
: *****
: TEST 121 TEST MODE 0,X DOUBLE-OPERAND NON-MODIFYING INSTS.
: *****

2758 007046
2759 007046 005000
2760 007050 005010
2761 007052 005110
2762 007054 005200
2763 007056 020037 000000
2764 007062 100403
2765 007064 001402
2766 007066 102401
2767 007070 103401
2768 007072
2769 007072 104000
2770 007074 005300
2771 007076 001002
2772 007100 005210
2773 007102 001401
2774 007104
2775 007104 104000

TS121:
CLR R0 ;R0=0
CLR (R0) ;LOC. 0=0
COM (R0) ;LOC. 0=177777
INC R0 ;R0=1
CMP R0,#0 ;TRY CMP MODE 0,3
BMI DNM03A ;CC=0001
BEQ DNM03A
BVS DNM03A
BCS DNM03B
DNM03A: EMT ;CC'S NOT SET CORRECTLY
DNM03B: DEC R0
BNE DNM03C
INC (R0)
BEQ TS122
DNM03C: EMT ;DATA INCORRECTLY MODIFIED BY CMP

2776
2777
2778
2779
2780
2781
2782
2783
2784
2785

: THIS TEST VERIFIES MODE 1 DOP INSTRUCTIONS. R0 IS SET TO -1
: AND LOC 0 TO 1. R4 IS THEN CLEARED AND USED TO POINT TO LOC 0.
: IN THE ADD MODE 1 INSTRUCTION, LOC 0 IS ADDED TO R0 AND THE
: RESULTS VERIFIED.
: *****
: TEST 122 TEST MODE 1 W/ DOP INST.
: *****

2786 007106
2787 007106 005000
2788 007110 005100

TS122:
CLR R0 ;R0=0
COM R0 ;R0=177777

2789 007112 005004
2790 007114 005014
2791 007116 005214
2792 007120 061400
2793 007122 001401
2794 007124 104000

CLR R4 ;R4=0
CLR (R4) ;LOC 0=0
INC (R4) ;LOC 0=1
ADD (R4),R0 ;TRY ADD SOURCE MODE 1
BEQ TS123
EMT ;RESULT OF ADD INCORRECT

: THIS TEST VERIFIES MODE 1 DOP BYTE INSTRUCTIONS WHICH ADDRESS
: EVEN BYTES. LOC. 0 IS SET TO -1 AND R4 IS CLEARED. THEN R4 IS
: SET TO -1 USING A BISB THRU R0 WITH MODE 1.
: *****

2795
2796
2797
2798
2799
2800
2801

TEST 123 TEST MODE 1 - EVEN BYTE W/ DOP INSTS.

2802
2803
2804

TS123:

2805 007126
2806 007126 005000
2807 007130 005010
2808 007132 005110
2809 007134 005004
2810 007136 151004
2811 007140 105104
2812 007142 001401
2813 007144 104000

CLR R0 ;R0=0
CLR (R0) ;LOC. 0=0
COM (R0) ;LOC. 0=177777
CLR R4 ;R4=0
BISB (R0),R4 ;TRY MODE 1- EVEN BYTE W/ DOP
COMB R4 ;R4=0
BEQ TS124
EMT ;RESULT OF BISB IS INCORRECT

2814
2815
2816
2817
2818
2819
2820
2821
2822

: THIS TEST VERIFIES MODE 1 DOP NON-MODIFYING INSTRUCTIONS
: WHICH ADDRESS EVEN BYTES. LOC. 0 IS SET TO -1 AND R0 IS CLEARED
: AND USED AS THE ADDRESSING REGISTER. R4 IS SET TO 377 AND A
: MODE 1.0 CMPB INSTRUCTION IS USED THE RESULTS VERIFIED.
: *****

2823
2824

TEST 124 TEST MODE 1 - EVEN BYTE W/ DOP NON-MODIFYING INST.

2825 007146
2826 007146 005000
2827 007150 005010
2828 007152 005110
2829 007154 005004
2830 007156 105104
2831 007160 121004
2832 007162 001401
2833 007164 104000

TS124:

CLR R0 ;R0=0
CLR (R0) ;LOC 0=0
COM (R0) ;LOC 0=177777
CLR R4 ;R4=0
COMB R4 ;R4=377
CMPB (R0),R4 ;TRY MODE 1 - EVEN BYTE W/ DOP NON-MODIFYING
BEQ TS125
EMT ;RESULT OF CMPB INCORRECT

2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844

: THIS TEST VERIFIES MODE 1.0 MOVB INSTRUCTIONS
: WHICH ADDRESS EVEN BYTES. LOC. 0 IS SET TO 177400, R0 IS CLEARED AND
: R4 IS SET TO -1. MOVB ARE USED TO MOVE BYTE 0 TO R4. THIS
: VERIFIES THAT THE PROPER BYTE WAS SELECTED AND THAT THE SIGN-X-TEND
: FUNCTION WITH MODE 0.
: THEN LOC. 0 IS COMPLEMENTED AND THE SAME PROCEDURE EXERCISES
: THE LOGIC FOR COMPLEMENTARY DATA.
: THIS TEST EXERCISES UNIQUE MICROCODE.
: *****

2845
2846
2847
2848
2849 007166
2850 007166 005000
2851 007170 005010
2852 007172 105110
2853 007174 005110
2854 007176 005004
2855 007200 005104
2856 007202 111004
2857 007204 005704
2858 007206 001401
2859 007210 104000
2860 007212 005110
2861 007214 111004
2862 007216 100401
2863 007220 104000
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875 007222
2876 007222 005000
2877 007224 005010
2878 007226 005004
2879 007230 005204
2880 007232 105114
2881 007234 151410
2882 007236 005210
2883 007240 001401
2884 007242 104000
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896 007244
2897 007244 005000
2898 007246 005010
2899 007250 005110
2900 007252 012004

: TEST 125 TEST MOV INSTRUCTION MODE 1,0 EVEN BYTE
:*****
TS125:

```
CLR R0 ;R0=0
CLR (R0) ;LOC 0=0
COMB (R0) ;LOC 0=177400
COM (R0)
CLR R4 ;R4=0
COM R4 ;R4=177777
MOVB (R0),R4 ;R4=0
TST R4 ;CHECK SIGN OF WORD
BEQ DOP1
EMT ;MOVB SHOULD SIGN X-TEND
DOP1: COM (R0) ;LOC 0=177777
MOVB (R0),R4 ;DO MOVB W/ EVEN BYTE
BMI TS126
EMT ;MOVB SHOULD SIGN X-TEND
```

: THIS TEST VERIFIES MODE 1 DOP INSTRUCTIONS WHICH REFERENCE
: ODD BYTES. LOC. 0 IS SET TO 177400. R0 IS SET TO 0 AND R4 IS
: SET TO 1. THE BISB INSTRUCTION USES THE DATA IN BYTE 1 TO SET BYTE 0.
: THE RESULT IS CHECKED BY INCREMENTING THE WORD (LOC. 0) TO ZERO.
:*****

: TEST 126 TEST MODE 1-ODD BYTE W/ DOP INSTS.
:*****
TS126:

```
CLR R0 ;R0=0
CLR (R0) ;LOC. 0=0
CLR R4 ;R4=0
INC R4 ;R4=1
COMB (R4) ;LOC. 0=177400
BISB (R4),(R0) ;TRY TO BIS LOW ORDER BITS W/ MODE 1
INC (R0) ;CHECK RESULT
BEQ TS127
EMT ;RESULT OF BISB INCORRECT
```

: THIS TEST VERIFIES MODE 2 DOP INSTRUCTIONS. LOC. 0 IS SET TO -1.
: R0 IS CLEARED AND USED AS THE MODE 2 ADDRESSING REGISTER TO MOVE LOC. 0
: TO R7. THE DATA RESULTS ARE VERIFIED AND THE INCREMENTING OF THE REGISTER
: IS CHECKED.
:*****

: TEST 127 TEST MODE 2 W/ DOP INSTS.
:*****
TS127:

```
CLR R0 ;R0=0
CLR (R0) ;LOC. 0=0
COM (R0) ;LOC. 0=177777
MOV (R0)+,R4 ;TRY MOVE MODE 2,0
```

2901 007254 005204
2902 007256 001401
2903 007260 104000
2904 007262 005300
2905 007264 005300
2906 007266 001401
2907 007270 104000
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921 007272
2922 007272 005000
2923 007274 010010
2924 007276 005110
2925 007300 142010
2926 007302 105737 000001
2927 007306 001401
2928 007310 104000
2929 007312 105137 000000
2930 007316 001401
2931 007320 104000
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942 007322
2943 007322 005000
2944 007324 005004
2945 007326 005010
2946 007330 005110
2947 007332 105120
2948 007334 112004
2949 007336 005204
2950 007340 001401
2951 007342 104000
2952 007344 005740
2953 007346 005700
2954 007350 001401
2955 007352 104000
2956

```
INC R4 ;CHECK R4
BEQ DOP2
EMT ;RESULT OF MOV INST INCORRECT
DOP2: DEC R0 ;TEST R0 AFTER MODE 2
      DEC R0
      BEQ TS130
      EMT ;REGISTER NOT INCREMENTED IN MODE 2

:*****
:
: THIS TEST VERIFIES MODE 2 DOP BYTE INSTRUCTIONS WHICH ADDRESS
: EVEN BYTES. LOC. 0 IS SET TO -1. R0 IS CLEARED AND USED AS THE
: ADDRESSING REGISTER IN A TEST WHICH TRIES TO CLEAR BYTE 1 USING
: BYTE 0 DATA AND A BICB. UNIQUE IN THIS TEST IS USE OF THE
: SAME ADDRESSING REGISTER FOR BOTH SOURCE AND DESTINATION. THE SOURCE AND
: DESTINATION IS CHECKED TO INSURE PROPER FUNCTIONING.
:*****
: TEST 130 TEST MODE 2 - EVEN BYTE W/ DOP INST.
:*****
TS130:
      CLR R0 ;R0=0
      MOV R0,(R0) ;LOC. 0=0
      COM (R0) ;LOC. 0=177777
      BICB (R0)+,(R0) ;TRY TO CLEAR BYTE 1 FROM BYTE 0 W/ BICB
      TSTB #1 ;CHECK RESULT
      BEQ DOPB2A
      EMT ;BICB DESTINATION INCORRECT
DOPB2A: COMB #0 ;CHECK BICB SOURCE
      BEQ TS131
      EMT ;BICB SOURCE INCORRECTLY CHANGED

:*****
:
: THIS TEST VERIFIES MODE 2 DOP BYTE INSTRUCTIONS WHICH REFERENCE
: ODD BYTES. R0 IS SET TO 1, LOC. 0 IS SET TO 177400, AND R4 IS CLEARED.
: A MODE 2 MOVVB USES R0 TO MOVE BYTE 1 TO R4. AN INCREMENT
: IS USED TO CHECK THAT THE PROPER BYTE WAS MOVED AND SIGN X-TENDED.
:*****
: TEST 131 TEST MODE 2 - ODD BYTE W/ DOP INST.
:*****
TS131:
      CLR R0 ;R0=0
      CLR R4 ;R4=0
      CLR (R0) ;LOC. 0=0
      COM (R0) ;LOC. 0=177777
      COMB (R0)+ ;LOC 0=177400; R0=1
      MOVVB (R0)+,R4 ;TRY DOP MODE 2 W/ ODD BYTE
      INC R4 ;CHECK RESULT OF MOVVB
      BEQ DOPB2B
      EMT ;RESULT OF MOVVB INCORRECT
DOPB2B: TST -(R0) ;BUMP R0 DOWN BY 2
      TST R0 ;CHECK R0
      BEQ TS132
      EMT ;MODE 2 BYTE DID NOT INCREMENT REG. CORRECTLY

:*****
```


2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967 007354
2968 007354 012737 052525 000000
2969 007362 012700 125252
2970 007366 053700 000000
2971 007372 005200
2972 007374 001401
2973 007376 104000
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984 007400
2985 007400 012737 052652 000000
2986 007406 005000
2987 007410 153700 000000
2988 007414 022700 000252
2989 007420 001401
2990 007422 104000
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001 007424
3002 007424 012737 052652 000000
3003 007432 005000
3004 007434 153700 000001
3005 007440 022700 000125
3006 007444 001401
3007 007446 104000
3008
3009
3010
3011
3012 007450

: THIS TEST VERIFIES MODE 3 DOUBLE-OPERAND INSTRUCTIONS.
: LOC. 0 IS LOADED WITH ALTERNATING ZEROES AND ONES; AND RO IS LOADED
: WITH ALTERNATING ONES AND ZEROES. A MODE 3 BIS IS USED TO SET RO
: TO -1 BY USING LOC. 0 AS THE SOURCE TO BIS THE ZEROES IN RO. THE
: RESULT IS TESTED BY INCREMENTING RO AND CHECKING FOR ZERO.
: *****
: TEST 132 TEST MODE 3 W/ DOP INSTS.
: *****
TS132:
MOV #052525,@#0 ;MOVE 52525 TO LOC. 0
MOV #125252,RO ;SET ALT. ONE AND ZERO IN RO
BIS @#0,RO ;TRY TO SET ALL OTHER BITS W/ MODE 3
INC RO ;TEST RESULT
BEQ TS133
EMT ;BIS W/ MODE 3 INCORRECT RESULT
: *****
: THIS TEST VERIFIES MODE 3 DOUBLE OPERAND BYTE INSTRUCTIONS WHICH
: ADDRESS EVEN BYTES. BYTE 0 IS SET TO ALTERNATING 1'S AND 0'S; BYTE 1,
: ALTERNATING 0'S AND 1'S. RO IS CLEARED AND A BISB IS USED TO
: SET THE LOW BYTE OF RO TO 252.
: *****
: TEST 133 TEST MODE 3 - EVEN BYTE W/ DOP INSTS.
: *****
TS133:
MOV #52652,@#0 ;MOVE 1'S AND 0' PATTERN TO LOC. 0
CLR RO ;RO=0
BISB @#0,RO ;TRY RO=252 W/ MODE 3 - EVEN BYTE
CMP #252,RO ;BISB W/ EVEN BYTE SUCCESSFUL?
BEQ TS134
EMT ;BISB W/ MODE 3 - EVEN BYTE FAILED
: *****
: THIS TEST VERIFIES MODE 3 DOUBLE OPERAND BYTE INSTRUCTIONS
: WHICH ADDRESS ODD BYTES. THE SAME PROCEDURE USED IN PREVIOUS
: TEST IS USED HERE. THIS TIME BYTE 1 IS USED AS THE SOURCE BYTE.
: THE EXPECTED RESULT IS: RO = 125.
: *****
: TEST 134 TEST MODE 3 - ODD BYTE W/ DOP INSTS.
: *****
TS134:
MOV #52652,@#0 ;MOVE 1'S AND 0'S PATTERN TO LOC 0
CLR RO ;RO=0
BISB @#1,RO ;TRY RO=152 W/ MODE 3 - ODD BYTE
CMP #125,RO ;RO=125?
BEQ TS135
EMT ;BISB W/ MODE 3 - ODD BYTE FAILED
: *****
: TEST 135 TEST DEST. MODE 0-BYTE W/ DOP NON-MODIFYING MST
: *****
TS135:

3013 007450 005000
3014 007452 105100
3015 007454 000263
3016 007456 132700 000200
3017 007462 001403
3018 007464 102402
3019 007466 103001
3020 007470 100401
3021 007472
3022 007472 104000
3023 007474 105100
3024 007476 001401
3025 007500 104000
3026
3027

```
CLR R0 ;RO=0
COMB R0 ;RO=377
+SEC!SEV ;SET C AND V BITS
BITB #200,R0 ;TRY DOPNM DEST. MODE 0-BYTE
BEQ DNMB0A ;BR TO ERROR IF Z BIT SET
BVS DNMB0A ;BR TO ERROR IF V BIT SET
BCC DNMB0A ;BR TO ERROR IF C BIT CLEAR.
BMI DNMB0B

DNMB0A:
EMT ;CC'S INCORRECT
DNMB0B: COMB R0 ;CHECK DESTINATION DATA
BEQ TS136
EMT ;DEST. DATA MODIFIED
```

3028
3029
3030 007502
3031 007502 005000
3032 007504 005010
3033 007506 000241
3034 007510 032710 177777
3035 007514 100403
3036 007516 102402
3037 007520 103401
3038 007522 001401
3039 007524
3040 007524 104000
3041 007526 005710
3042 007530 001401
3043 007532 104000
3044
3045

```
*****
;TEST 136 TEST DEST. MODE 1 W/ DOP NON-MODIFYING INST
*****
TS136:
CLR R0 ;RO=0
CLR (R0) ;LOC. 0=0
CLC ;CLEAR C BIT
BIT #177777,(R0) ;TRY DOPNM DEST. MODE 1
BMI DNMB1A ;BR TO ERROR IF N BIT SET
BVS DNMB1A ;BR TO ERROR IF V BIT SET
BCS DNMB1A ;BR TO ERROR IF C BIT SET
BEQ DNMB1B

DNMB1A:
EMT ;COND. CODES INCORRECT
DNMB1B: TST (R0) ;CHECK TEST DATA
BEQ TS137
EMT ;DESTINATION DATA MODIFIED
```

3046
3047
3048 007534
3049 007534 005000
3050 007536 005010
3051 007540 052710 125252
3052 007544 032720 077777
3053 007550 102402
3054 007552 001401
3055 007554 100001
3056 007556
3057 007556 104000
3058 007560 005300
3059 007562 005300
3060 007564 001401
3061 007566
3062 007566 104000
3063 007570 022710 125252
3064 007574 001401
3065 007576 104000
3066
3067
3068

```
*****
;TEST 137 TEST DEST, MODE 2 W/ DOP NON-MODIFYING INST.
*****
TS137:
CLR R0 ;RO=0
CLR (R0) ;LOC. 0=0
BIS #125252,(R0) ;LOC. 0=125252
BIT #77777,(R0)+ ;TRY DOPNM INST W/ MODE 2
BVS DNMB2A ;BR TO ERROR IF V BIT SET
BEQ DNMB2A ;BR TO ERROR IF Z-BIT SET
BPL DNMB2B

DNMB2A:
EMT ;COND. CODES INCORRECT
DNMB2B: DEC R0 ;DECREMENT R0 TO CHECK IT.
DEC R0
BEQ DNMB2D

DNMB2C:
EMT ;MODE 2 REGISTER NOT INCREMENTED BY 2
DNMB2D: CMP #125252,(R0) ;CHECK DEST. DATA
BEQ TS140
EMT ;DEST. DATA MODIFIED
```

```
*****
;TEST 140 TEST DEST. MODE 2-BYTE, W/DOP NON-MODIFYING INST
*****
```

3069
3070 007600
3071 007600 005000
3072 007602 005010
3073 007604 052710 052652
3074 007610 000263
3075 007612 132720 000201
3076 007616 001403
3077 007620 103002
3078 007622 102401
3079 007624 100401
3080 007626
3081 007626 104000
3082 007630 005300
3083 007632 001401
3084 007634 104000
3085 007636 005200
3086 007640 132720 000201
3087 007644 001402
3088 007646 102401
3089 007650 100001
3090 007652
3091 007652 104000
3092 007654 005300
3093 007656 005300
3094 007660 001401
3095 007662 104000
3096 007664 022710 052652
3097 007670 001401
3098 007672 104000
3099

```
*****  
TS140:  
CLR R0 ;R0=0  
CLR (R0) ;LOC. 0=0  
BIS #52652,(R0) ;LOC. 0=52652  
+SEC!SEV ;SET C AND V BITS  
BITB #201,(R0)+ ;TRY DOPNM INST. W/ MODE 2 EVEN BYTE  
BEQ DNMB2A ;BR TO ERROR IF Z-BIT SET  
BCC DNMB2A ;BR TO ERROR IF C-BIT CLEAR  
BVS DNMB2A ;BR TO ERROR IF V-BIT SET  
BMI DNMB2B  
  
DNMB2A: EMT ;COND. CODES INCORRECT  
DNMB2B: DEC R0 ;CHECK DEST. REGISTER.  
BEQ DNMB2C  
EMT ;DEST. REGISTER NOT INCREMENTED BY 1  
DNMB2C: INC R0 ;R0=1  
BITB #201,(R0)+ ;TRY DOPNM INST. W/MODE 2-ODD BYTE  
BEQ DNMB2D ;BR TO ERROR IF Z-BIT SET  
BVS DNMB2D ;BR TO ERROR IF V-BIT SET  
BPL DNMB2E  
  
DNMB2D: EMT ;COND. CODES INCORRECT  
DNMB2E: DEC R0 ;DEC R0 TO CHECK IT.  
DEC R0  
BEQ DNMB2F  
EMT ;DEST. REGISTER NOT INCREMENTED BY 1  
DNMB2F: CMP #52652,(R0) ;CHECK DEST. DATA IS UNMODIFIED  
BEQ TS141  
EMT ;DEST. DATA WAS MODIFIED.
```

3100
3101
3102
3103
3104 007674
3105 007674 005000
3106 007676 005010
3107 007700 052710 125125
3108 007704 105100
3109 007706 005200
3110 007710 005010
3111 007712 000263
3112 007714 132730 000201
3113 007720 001403
3114 007722 102402
3115 007724 103001
3116 007726 100001
3117 007730
3118 007730 104000
3119 007732 022700 000402
3120 007736 001401
3121 007740 104000
3122 007742 005200
3123 007744 005200
3124 007746 132730 000201

```
*****  
:TEST 141 TEST DEST, MODE 3-BYTES W/DOP NON-MODIFYING INST.  
*****  
TS141:  
CLR R0 ;R0=0  
CLR (R0) ;LOC. 0=0  
BIS #125125,(R0) ;LOC. 0=125125  
COMB R0 ;R0=377  
INC R0 ;R0=400  
CLR (R0) ;LOC. 400=0  
+SEC!SEV ;C-BIT=V-BIT=1  
BITB #201,@(R0)+ ;TRY DOPNM W/MODE 3-EVEN BYTE  
BEQ DNMB3A ;BR TO ERROR IF Z BIT SET  
BVS DNMB3A ;BR TO ERROR IF V BIT SET  
BCC DNMB3A ;BR TO ERROR IF C BIT CLEAR  
BPL DNMB3B  
  
DNMB3A: EMT ;COND. CODES INCORRECT  
DNMB3B: CMP #402,R0 ;CHECK DEST. REGISTER INC. BY 2 AND INC BY 2 AGAIN  
BEQ DNMB3C  
EMT ;DEST. REGISTER NOT INCREMENTED BY 2  
DNMB3C: INC R0 ;R0=404  
INC R0  
BITB #201,@(R0)+ ;TRY DOPNM DEST MODE 3-BYTE(ODD)
```

3125 007752 001402
3126 007754 102401
3127 007756 100401
3128 007760
3129 007760 104000
3130 007762 005004
3131 007764 022714 125125
3132 007770 001401
3133 007772 104000
3134
3135

BEQ DNMB3D ;BR TO ERROR IF Z BIT SET
BVS DNMB3D ;BR TO ERROR IF V BIT SET
BMI DNMB3E
DNMB3D: EMT ;COND. CODES INCORRECT
DNMB3E: CLR R4 ;R4=0
CMP #125125,(R4) ;CHECK DEST. DATA
BEQ TS142
EMT ;DEST. DATA MODIFIED

:TEST 142 TEST DEST. MODE 4 W/DOP NON-MODIFYING INST.

TS142:

3138 007774
3139 007774 005000
3140 007776 005010
3141 010000 052710 125252
3142 010004 052700 000002
3143 010010 000277
3144 010012 032740 020000
3145 010016 100403
3146 010020 102402
3147 010022 103001
3148 010024 001001
3149 010026
3150 010026 104000
3151 010030 005700
3152 010032 001401
3153 010034 104000
3154 010036 022737 125252 000000
3155 010044 001401
3156 010046 104000
3157
3158
3159

CLR R0 ;R0=0
CLR (R0) ;LOC. 0=0
BIS #125252,(R0) ;LOC. 0=125125
BIS #2,R0 ;R0=2
SCC ;SET ALL COND. CODE BITS
BIT #20000,-(R0) ;TRY DOPNM W/ MODE 4
BMI DNMB4A ;BR TO ERROR IF N-BIT SET
BVS DNMB4A ;BR TO ERROR IF V-BIT SET
BCC DNMB4A ;BR TO ERROR IF C-BIT CHAR
BNE DNMB4B
DNMB4A: EMT ;COND. CODES INCORRECT
DNMB4B: TST R0 ;CHECK DEST. REGISTER
BEQ DNMB4C
EMT ;DEST. REGISTER NOT DECREMENTED BY 2
DNMB4C: CMP #125252,#0 ;CHECK DEST. DATA
BEQ TS143
EMT ;DEST. DATA MODIFIED

:TEST 143 TEST DEST. MODE 4-BYTE W/ DOP NON-MODIFYING INST.

TS143:

3161 010050
3162 010050 005000
3163 010052 005010
3164 010054 052710 052652
3165 010060 052700 000002
3166 010064 000257
3167 010066 132740 000201
3168 010072 102403
3169 010074 001402
3170 010076 103401
3171 010100 001001
3172 010102
3173 010102 104000
3174 010104 022700 000001
3175 010110 001401
3176 010112 104000
3177 010114 132740 000201
3178 010120 001401
3179 010122 100401
3180 010124

CLR R0 ;R0=0
CLR (R0) ;LOC. 0=0
BIS #52652,(R0) ;LOC. 0=52652
BIS #2,R0 ;R0=2
CCC ;COND. CODES=0
BITB #201,-(R0) ;TRY DOPNM INST W/MODE 4 ODD BYTE
BVS DNMB4A ;BR TO ERROR IF V BIT SET
BEQ DNMB4A ;BR TO ERROR IF Z BIT SET
BCS DNMB4A ;BR TO ERROR IF C BIT SET
BNE DNMB4B
DNMB4A: EMT ;COND. CODES INCORRECT
DNMB4B: CMP #1,R0 ;CHECK DEST. REGISTER
BEQ DNMB4C
EMT ;DEST REG. NOT DECREMENTED BY 1
DNMB4C: BITB #201,-(R0) ;TRY DOPNM INST. W/MODE 4 EVEN BYTE
BEQ DNMB4D ;BR TO ERROR IF Z-BIT SET
BMI DNMB4E
DNMB4D:

3181 010124 104000
3182 010126 005700
3183 010130 001401
3184 010132 104000
3185 010134 022710 052652
3186 010140 001401
3187 010142 104000
3188
3189
3190
3191
3192 010144
3193 010144 005000
3194 010146 005010
3195 010150 052710 100000
3196 010154 052700 000402
3197 010160 000277
3198 010162 032750 100000
3199 010166 102403
3200 010170 103002
3201 010172 001401
3202 010174 100401
3203 010176
3204 010176 104000
3205 010200 022700 000400
3206 010204 001401
3207 010206 104000
3208 010210 022737 100000 000000
3209 010216 001401
3210 010220 104000
3211
3212
3213
3214
3215 010222
3216 010222 005000
3217 010224 005010
3218 010226 052710 000001
3219 010232 005100
3220 010234 032760 000001 000001
3221 010242 001403
3222 010244 102402
3223 010246 103001
3224 010250 100001
3225 010252
3226 010252 104000
3227 010254 022700 177777
3228 010260 001401
3229 010262 104000
3230 010264 022737 000001 000000
3231 010272 001401
3232 010274 104000
3233
3234
3235
3236

```
DNMB4E: EMT ;COND. CODES INCORRECT
          TST R0 ;CHECK DEST. REGISTER
          BEQ DNMB4F
DNMB4F: EMT ;DEST. REG. NOT DECREMENTED BY 1
          CMP #52652,(R0) ;CHECK DESTINATION DATA
          BEQ TS144
          EMT ;DEST. DATA MODIFIED

:*****
:TEST 144 TEST DEST MODE 5 W/DOP NON-MODIFYING INST.
:*****
TS144:
          CLR R0 ;R0=0
          CLR (R0) ;LOC 0=0
          BIS #100000,(R0) ;LOC. 0=100000
          BIS #402,R0 ;R0=2
          SCC ;SET ALL COND. CODE BITS
          BIT #100000,a-(R0) ;TRY DOPNM W/MODE 5
          BVS DNMB5A ;BR TO ERROR IF V-BIT SET
          BCC DNMB5A ;BR TO ERROR IF C-BIT CLEAR
          BEQ DNMB5A ;BR TO ERROR IF Z-BIT SET
          BMI DNMB5B
DNMB5A: EMT ;COND. CODES INCORRECT
          CMP #400,R0 ;CHECK DEST. REGISTER
          BEQ DNMB5C
          EMT ;DEST. REGISTER NOT DECREMENTED BY 2
DNMB5C: CMP #100000,a#0 ;CHECK DESTINATION DATA
          BEQ TS145
          EMT ;DEST. DATA INCORRECTLY MODIFIED

:*****
:TEST 145 TEST DEST. MODE 6 W/DOP NON-MODIFYING INST.
:*****
TS145:
          CLR R0 ;R0=0
          CLR (R0) ;LOC> 0=0
          BIS #1,(R0) ;LOC. 0=1
          COM R0 ;R0=-1 C-BIT=1
          BIT #1,(R0) ;TRY DOPNM W/MODE 6
          BEQ DNMB6A ;BR TO ERROR IF Z-BIT SET
          BVS DNMB6A ;BR TO ERROR IF V-BIT SET
          BCC DNMB6A ;BR TO ERROR IF C-BIT CLEAR
          BPL DNMB6B
DNMB6A: EMT ;COND. CODES INCORRECT
          CMP #-1,R0 ;CHECK DEST. REGISTER
          BEQ DNMB6C
          EMT ;DEST. REGISTER MODIFIED
DNMB6C: CMP #1,a#0 ;CHECK DEST. DATA
          BEQ TS146
          EMT ;DEST. DATA MODIFIED

:*****
:TEST 146 TEST DEST MODE 7 W/DOP NON-MODIFYING INST.
:*****
```

3237 010276
3238 010276 005000
3239 010300 005010
3240 010302 052710 125125
3241 010306 052700 000001
3242 010312 132770 000125 000403
3243 010320 102403
3244 010322 100402
3245 010324 103401
3246 010326 001401
3247 010330
3248 010330 104000
3249 010332 022700 000001
3250 010336 001401
3251 010340 104000
3252 010342 022737 125125 000000
3253 010350 001401
3254 010352 104000
3255
3256
3257
3258
3259
3260
3261
3262
3263
3264
3265 010354
3266 010354 005000
3267 010356 005010
3268 010360 005100
3269 010362 005004
3270 010364 010014
3271 010366 102402
3272 010370 001401
3273 010372 100401
3274 010374
3275 010374 104000
3276 010376 005704
3277 010400 001401
3278 010402 104000
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289 010404
3290 010404 005000
3291 010406 005001
3292 010410 005010

TS146:
CLR R0 ;R0=0
CLR (R0) ;LOC. 0=0 C-BIT=0
BIS #125125,(R0) ;LOC. 0=125125
BIS #1,R0 ;R0=1
BITB #125,@403(R0) ;TRY DOPNM W/MODE 7
BVS DNM7A ;BR TO ERROR IF V-BIT SET
BMI DNM7A ;BR TO ERROR IF N-BIT SET
BCS DNM7A ;BR TO ERROR IF C-BIT SET
BEQ DNM7B
DNM7A: EMT ;COND. CODES INCORRECT
DNM7B: CMP #1,R0 ;CHECK DEST. REGISTER
BEQ DNM7C
EMT ;DESTINATION REGISTER MODIFIED
DNM7C: CMP #125125,@#0 ;CHECK DEST. DATA
BEQ TS147
EMT ;DEST. DATA INCORRECT

: THIS TEST VERIFIES THE MOV DESTINATION MODE 1 INSTRUCTION.
: DATA IS SET IN R0 USING SOP INSTRUCTIONS AND THEN MOVED TO LOC. 0
: USING MOV SRC MODE 0, DEST. MODE 1.
: *****

TEST 147 TEST MOV DESTINATION MODE 1

TS147:

CLR R0 ;R0=0
CLR (R0) ;LOC. 0=0
COM R0 ;R0=-1
CLR R4 ;R4 POINTS TO LOC. 0
MOV R0,(R4) ;TRY MOVE MODE 0,1
BVS MDM1A ;BR TO ERROR IF V SET
BEQ MDM1A ;BR TO ERROR IF Z SET
BMI MDM1B
MDM1A: EMT ;CONDITION CODE NOT CORRECT
MDM1B: TST R4
BEQ TS150
EMT ;DESTINATION REGISTER INCORRECTLY ALTERED

: THIS TEST VERIFIES THE MOV DESTINATION MODE 2 INSTRUCTION.
: DATA IS SET IN R0 USING SOP INSTRUCTIONS AND THEN MOVED
: TO LOCATION 0 USING MOV SRC MODE 0, DEST. MODE 1.
: *****

TEST 150 TEST MOV DESTINATION MODE 2

TS150:

CLR R0 ;R0=0
CLR R1 ;R1=0
CLR (R0) ;LOC.0=0

3293 010412 005110
3294 010414 010120
3295 010416 100402
3296 010420 102401
3297 010422 001401
3298 010424
3299 010424 104000
3300 010426 005300
3301 010430 005300
3302 010432 001401
3303 010434
3304 010434 104000
3305 010436 005737 000000
3306 010442 001401
3307 010444 104000
3308
3309
3310
3311
3312
3313
3314
3315
3316
3317 010446
3318 010446 005000
3319 010450 005010
3320 010452 112720 000125
3321 010456 102402
3322 010460 001401
3323 010462 100001
3324 010464
3325 010464 104000
3326 010466 022700 000001
3327 010472 001401
3328 010474 104000
3329 010476 112720 000252
3330 010502 102402
3331 010504 001401
3332 010506 100401
3333 010510
3334 010510 104000
3335 010512 022700 000002
3336 010516 001401
3337 010520 104000
3338 010522 022737 125125 000000
3339 010530 001401
3340 010532 104000
3341
3342
3343
3344
3345
3346
3347
3348

COM (R0) ;LOC. 0= 1
MOV R1,(R0)+ ;TRY MOVE MODE 0,2
BMI MDM2A ;BR TO ERROR IF N SET
BVS MDM2A ;BR TO ERROR IF V SET
BEQ MDM2B
MDM2A: EMT ;CC'S INCORRECT
MDM2B: DEC R0
DEC R0
BEQ MDM2D
MDM2C: EMT ;DESTINATION REGISTER NOT INCREMENTED PROPERLY
MDM2D: TST @#0
BEQ TS151
EMT ;DESTINATION DATA INCORRECT

: THIS TEST VERIFIES DESTINATION MODE 2 W/MOVB INSTS. TWO DIFFERENT MOVB
: INSTRUCTIONS ARE USED TO MOVE A TEST PATTERN FIRST TO BYTE 0 THEN TO BYTE 1.
: *****
: TEST 151 TEST MOV-BYTE DESTINATION MODE 2
: *****

TS151:
CLR R0 ;R0=0
CLR (R0) ;LOC. 0=0
MOVB #125,(R0)+ ;TRY DESTINATION MODE 2 W/EVEN BYTE
BVS MBDM2A ;BR TO ERROR IF V SET
BEQ MBDM2A ;BR TO ERROR IF Z SET
BPL MBDM2B
MBDM2A: EMT ;CC'S INCORRECT
MBDM2B: CMP #1,R0
BEQ MBDM2C
EMT ;REGISTER NOT INCREMENTED BY ONE
MBDM2C: MOVB #252,(R0)+ ;TRY DESTINATION MODE 2 W/ODD BYTE
BVS MBDM2D
BEQ MBDM2D
BMI MBDM2E
MBDM2D: EMT ;CC'S NOT SET CORRECT
MBDM2E: CMP #2,R0
BEQ MBDM2F
EMT ;REGISTER NOT INCREMENTED BY ONE
MBDM2F: CMP #125125,@#0 ;CHECK DATA
BEQ TS152
EMT ;DESTINATION DATA INCORRECT

: THIS TEST VERIFIES MOV DESTINATION MODE 3. R0 IS USED TO PICK UP
: AN ADDRESS AT LOC. 400. LOC 400 POINTS TO LOC. 0 THE EFFECTIVE DEST. ADDR.. ALSO, MOVB
: INST. ARE USED W/ EVEN AND ODD BYTES TO CHECK MOV BYTES INST AND MODE 37 DESTINATIONS.
: *****
: TEST 152 TEST MOV(B) DESTINATION MODE 3

3349
3350 010534
3351 010534 012700 000400
3352 010540 005010
3353 010542 005037 000000
3354 010546 012730 125252
3355 010552 102402
3356 010554 001401
3357 010556 100401
3358 010560
3359 010560 104000
3360 010562 022700 000402
3361 010566 001401
3362 010570 104000
3363 010572 022737 125252 000000
3364 010600 001401
3365 010602 104000
3366 010604 112737 000125 000000
3367 010612 022737 125125 000000
3368 010620 001401
3369 010622 104000
3370 010624 112737 000525 000001
3371 010632 022737 052525 000000
3372 010640 001401
3373 010642 104000

```
*****  
TS152:  
MOV #400,R0 ;R0=400  
CLR (R0) ;LOC. 400 POINTS TO LOC. 0  
CLR @#0 ;LOC. 0=0  
MOV #125252,@(R0)+ ;TRY MOV DESTINATION MODE 2  
BVS MDM3A ;BR TO ERROR IF V SET  
BEQ MDM3A ;BR TO ERROR IF Z SET  
BMI MDM3B  
  
MDM3A: EMT ;CC'S INCORRECT  
MDM3B: CMP #402,R0 ;CHECK DEST. MODE REGISTER  
BEQ MDM3C  
EMT ;REGISTER NOT INCREMENTED BY 2  
MDM3C: CMP #125252,@#0 ;CHECK DESTINATION DATA  
BEQ MDM3D  
EMT ;DESTINATION DATA INCORRECT  
MDM3D: MOVB #125,@#0 ;TRY MOVB DESTINATION MODE 2 EVEN BYTE  
CMP #125125,@#0 ;CHECK DATA  
BEQ MDM3E  
EMT ;DESTINATION DATA INCORRECT  
MDM3E: MOVB #525,@#1 ;TRY MOVB DESTINATION MODE 2 ODD BYTE  
CMP #52525,@#0 ;CHECK DATA  
BEQ TS153  
EMT  
:
```

3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385 010644
3386 010644 005000
3387 010646 005010
3388 010650 012704 000002
3389 010654 012744 012345
3390 010660 102402
3391 010662 001401
3392 010664 100001
3393 010666
3394 010666 104000
3395 010670 005704
3396 010672 001401
3397 010674 104000
3398 010676 022710 012345
3399 010702 001401
3400 010704 104000
3401
3402
3403
3404

```
*****  
: THIS TEST VERIFIES THE MOV DESTINATION MODE 4 INSTRUCTION.  
: SOP INSTRUCTIONS ON R0 ARE USED TO CLEAR TARGET LOCATION 0.  
: R4 IS USED AS THE MODE 4 ADDRESSING REGISTER, AND  
: CONDITIONAL BRANCHES ARE USED TO VERIFY THE DATA.  
: *****  
: TEST 153 TEST MOV DESTINATION MODE 4  
: *****  
TS153:  
CLR R0 ;R0=0  
CLR (R0) ;LOC 0=0  
MOV #2,R4 ;R4=2  
MOV #12345,-(R4) ;TRY MOV DEST. MODE 4  
BVS MDM4A ;BR TO ERROR IF V-BIT SET  
BEQ MDM4A ;BR TO ERROR IF Z-BIT SET  
BPL MDM4B  
  
MDM4A: EMT ;CC'S NOT CORRECT  
MDM4B: TST R4 ;CHECK DECREMENTING OF MODE 4 REG.  
BEQ MDM4C  
EMT ;DESTINATION MODE REGISTER NOT DECREMENTED BY 2  
MDM4C: CMP #12345,(R0) ;CHECK DESTINATION DATA  
BEQ TS154  
EMT ;DESTINATION DATA INCORRECT
```

```
*****  
: THIS TEST VERIFIES THE MOVB DESTINATION MODE 4 INSTRUCTION  
:
```



```

3405
3406
3407
3408
3409
3410
3411
3412
3413 010706
3414 010706 005004
3415 010710 005014
3416 010712 012700 000002
3417 010716 112740 125125
3418 010722 020027 000001
3419 010726 001401
3420 010730 104000
3421 010732 021427 052400
3422 010736 001401
3423 010740 104000
3424 010742 112740 125125
3425 010746 102402
3426 010750 001401
3427 010752 100001
3428 010754
3429 010754 104000
3430 010756 005700
3431 010760 001401
3432 010762 104000
3433 010764 021427 052525
3434 010770 001401
3435 010772 104000
3436
3437
3438
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449 010774
3450 010774 005004
3451 010776 005014
3452 011000 012700 000400
3453 011004 012750 004321
3454 011010 102402
3455 011012 001401
3456 011014 100001
3457 011016
3458 011016 104000
3459 011020 022700 000376
3460 011024 001401

```

```

: ON BOTH ODD AND EVEN BYTES. SOP INSTRUCTIONS ON R4 ARE
: USED TO CLEAR TARGET LOCATION 0. R0 IS USED AS THE MODE 4
: ADDRESSING REGISTER, AND CMP AND CONDITIONAL BRANCH
: INSTRUCTIONS ARE USED TO VERIFY THE DATA.
:*****
:TEST 154 TEST MOVDESTINATION MODE 4
:*****
TS154:
      CLR      R4           ;R4=0
      CLR      (R4)        ;LOC. 0=0
      MOV      #2,R0       ;R0 = 2
      MOVB     #125125,-(R0) ;TRY MOVB DEST. MODE 4-ODD BYTE
      CMP      R0,#1       ;CHECK THAT DEST. REG. WAS DECREMENTED
      BEQ      MBDM4A
      EMT
      MBDM4A: CMP      (R4),#52400 ;DESTINATION REG. NOT DECREMENTED BY 1
      BEQ      MBDM4B       ;CHECK DEST. DATA
      EMT
      MBDM4B: MOVB     #125125,-(R0) ;DEST. DATA NOT CORRECT
      BVS      MBDM4C       ;TRY MOVB DEST. MODE 4--EVEN BYTE
      BEQ      MBDM4C       ;BR. TO ERROR IF V-BIT SET
      BPL      MBDM4D       ;BR TO ERROR IF Z-BIT SET
      MBDM4C:
      EMT
      MBDM4D: TST      R0           ;COND. CODES INCORRECT
      BEQ      MBDM4E       ;CHECK MODE 4 DEST. REGISTER
      EMT
      MBDM4E: CMP      (R4),#52525 ;DESTINATION REG NOT DECREMENTED BY 1
      BEQ      TS155        ;CHECK DEST. DATA
      EMT
      ;DESTINATION DATA INCORRECT
:*****
:
: THIS TEST VERIFIES THE MOV DESTINATION MODE 5 AND THE MOVB
: DESTINATION MODE 5 - EVEN BYTE INSTRUCTIONS. R4 IS A
: POINTER TO TARGET LOCATION 0 AND R0 IS SETUP TO
: POINT TO LOCATION 376 FOR THE MOV, AND LOCATION 404 FOR
: THE MOVB INSTRUCTIONS. CMP INSTRUCTIONS ARE USED TO VERIFY
: PROPER ADDRESSING AND DATA.
:*****
:TEST 155 TEST MOV DESTINATION MODE 5
:*****
TS155:
      CLR      R4           ;R4=0
      CLR      (R4)        ;LOC. 0 = 0
      MOV      #400,R0      ;R0=400
      MOV      #4321,a-(R0) ;TRY MOV DEST. MODE 5
      BVS      MDM5A       ;BR TO ERROR IF V-BIT SET
      BEQ      MDM5A       ;BR TO ERROR IF Z-BIT SET
      BPL      MDM5B
      MDM5A:
      EMT
      MDM5B: CMP      #376,R0      ;COND. CODES INCORRECT
      BEQ      MDM5C       ;CHECK MODE 5 REG. WAS DECREMENTED

```

3461 011026 104000
3462 011030 022714 004321
3463 011034 001401
3464 011036 104000
3465 011040 012700 000406
3466 011044 112750 000377
3467 011050 022700 000404
3468 011054 001401
3469 011056 104000
3470 011060 022714 177721
3471 011064 001401
3472 011066 104000
3473
3474
3475
3476
3477
3478
3479
3480
3481
3482
3483
3484
3485 011070
3486 011070 005000
3487 011072 005010
3488 011074 005200
3489 011076 012760 052525 177777
3490 011104 102402
3491 011106 001401
3492 011110 100001
3493 011112
3494 011112 104000
3495 011114 022700 000001
3496 011120 001401
3497 011122 104000
3498 011124 022737 052525 000000
3499 011132 001401
3500 011134 104000
3501 011136 012700 000002
3502 011142 112760 000377 177777
3503 011150 022700 000002
3504 011154 001401
3505 011156 104000
3506 011160 022737 177525 000000
3507 011166 001401
3508 011170 104000
3509
3510
3511
3512
3513
3514
3515
3516

MDM5C: EMT ;MODE 5 REGISTER NOT DECREMENTED BY 2
CMP #4321,(R4) ;CHECK DEST. DATA
BEQ MDM5D
MDM5D: EMT ;DEST. DATA INCORRECT
MOV #406,R0 ;R0=406
MOVB #377,@-(R0) ;TRY MOV DEST. MODE 5 --EVEN BYTE
CMP #404,R0 ;CHECK MODE 5 REG.
BEQ MDM5E
MDM5E: EMT ;MODE 5 REGISTER NOT DECREMENTED BY 2
CMP #177721,(R4) ;CHECK DEST. DATA
BEQ TS156
EMT ;DEST. DATA INCORRECT

: THIS TEST VERIFIES THE MOV DESTINATION MODE 6 AND MOVB - EVEN BYTE
: DESTINATION MODE 6 INSTRUCTIONS. R0 IS USED TO SETUP TARGET LOC.0
: FOR BOTH TESTS. PATTERNS OF ONES AND ZEROES ARE MOVED INTO LOC.0
: BY MODE 6 INSTRUCTIONS, AND CMP INSTRUCTIONS ARE USED TO VERIFY
: PROPER ADDRESSING AND DATA.

: TEST 156 TEST MOV DESTINATION MODE 6
: *****
TS156:

CLR R0 ;R0=0
CLR (R0) ;LOC. 0=0
INC R0 ;R0=1
MOV #052525,-1(R0) ;TRY MOV DEST. MODE 6
BVS MDM6A ;BR TO ERROR IF V-BIT SET
BEQ MDM6A ;BR TO ERROR IF Z-BIT SET
BPL MDM6B
MDM6A: EMT ;COND. CODES INCORRECT
MDM6B: CMP #1,R0 ;CHECK DEST. REGISTER UNALTERED
BEQ MDM6C
EMT ;DEST. REGISTER INCORRECTLY ALTERED
MDM6C: CMP #52525,@#0 ;CHECK DEST. DATA
BEQ MDM6D
EMT ;DEST. DATA INCORRECT
MDM6D: MOV #2,R0 ;R0=2
MOVB #377,-1(R0) ;TRY MOVB DEST. MODE 6
CMP #2,R0 ;CHECK DEST. REGISTER UNALTERED
BEQ MDM6E
EMT ;DEST. REGISTER INCORRECTLY ALTERED
MDM6E: CMP #177525,@#0 ;CHECK DEST. DATA
BEQ TS157
EMT ;DEST. DATA INCORRECT

: THIS TEST VERIFIES THE MOV DESTINATION MODE 7 AND MOVB - ODD BYTE
: DESTINATION MODE 7 INSTRUCTIONS. R4 POINTS TO TARGET LOC.0 AND R0
: IS USED AS THE MODE 7 ADDRESSING REGISTER. CMP INSTRUCTIONS ARE
: USED TO VERIFY PROPER ADDRESSING AND DATA.

3517
3518
3519
3520 011172
3521 011172 005004
3522 011174 005014
3523 011176 012700 000403
3524 011202 012770 070707 177777
3525 011210 102402
3526 011212 001401
3527 011214 100001
3528 011216
3529 011216 104000
3530 011220 022700 000403
3531 011224 001401
3532 011226 104000
3533 011230 022737 070707 000000
3534 011236 001401
3535 011240 104000
3536 011242 112770 107070 000001
3537 011250 022700 000403
3538 011254 001401
3539 011256 104000
3540 011260 022737 034307 000000
3541 011266 001401
3542 011270 104000
3543
3544
3545
3546
3547
3548
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559 011272
3560 011272 012700 011336
3561 011276 014037 011336
3562 011302 064037 011336
3563 011306 144037 011336
3564 011312 154037 011337
3565 011316 024037 011336
3566 011322 001406
3567 011324
3568 011324 104000
3569
3570 011326 125252
3571 011330 052652
3572 011332 053125

:TEST 157 TEST MOV DESTINATION MODE 7

TS157:
CLR R4 ;R4=0
CLR (R4) ;LOC.0=0
MOV #403,R0 ;R0=403
MOV #70707,a-1(R0) ;TRY MOV W/DEST MODE 7
BVS MDM7A ;BR. TO ERROR IF V-BIT SET
BEQ MDM7A ;BR TO ERROR IF Z-BIT SET
BPL MDM7B
MDM7A: EMT ;COND. CODES INCORRECT
MDM7B: CMP #403,R0 ;CHECK DEST. REGISTER
BEQ MDM7C
EMT ;DEST. REGISTER INCORRECTLY ALTERED
MDM7C: CMP #70707,a#0 ;CHECK DEST. DATA
BEQ MDM7D
EMT ;DEST. DATA INCORRECT
MDM7D: MOVB #107070,a1(R0) ;TRY MOVB W/DEST MODE 7--ODD BYTE
CMP #403,R0 ;CHECK MODE 7 DEST. REG.
BEQ MDM7E
EMT ;DEST. DATA INCORRECT
MDM7E: CMP #34307,a#0 ;CHECK DEST. DATA
BEQ TS160
EMT ;DESTINATION DATA INCORRECT

: THIS TEST VERIFIES MODE 4 DOUBLE OPERAND INSTRUCTIONS.
: THE TEST USES MODE 4 ADDRESSING WITH REGISTER 0 TO MOVE THRU A
: TABLE OF OPERANDS. THE TABLE OF OPERANDS AND THE WORK LOCATION IS
: STORED FOLLOWING THE TEST CODE. A SERIES OF 5 DOP INSTRUCTIONS UTILIZES
: THE DATA IN THE TABLE TO CYCLE THE WORK LOCATION THRU A SET OF
: VALUE. THE DATA HAS BEEN CHOSEN TO INSURE THAT NO SINGLE ERROR WILL
: GO UNDETECTED. WORD AND BYTE INSTRUCTION ACCESSING BOTH EVEN AND
: ODD ADDRESSES ARE USED IN THE TEST. THE LISTING SHOWS THE
: EXPECTED INTERMEDIATE RESULT AS EACH INSTRUCTION IS EXECUTED.

:TEST 160 TEST MODE 4 W/ DOP INSTS.

TS160:
MOV #TBL1,R0 ;INITIALIZE R0
MOV -(R0),a#TBL1 ;TBL1=125252
ADD -(R0),a#TBL1 ;TBL1=000377
BICB -(R0),a#TBL1 ;TBL1=000252
BISB -(R0),a#TBL1+1 ;TBL1=125252
CMP -(R0),a#TBL1 ;CHECK RESULT
BEQ TS161
DOP4: EMT ;RESULT OF MODE 4 INSTS. INCORRECT
125252
52652
53125

3573 011334 125252
3574 011336 000000
3575
3576
3577
3578
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588 011340
3589 011340 012700 011406
3590 011344 015037 011336
3591 011350 065037 011336
3592 011354 145037 011336
3593 011360 155037 011337
3594 011364 025037 011336
3595 011370 001406
3596 011372
3597 011372 104000
3598 011374 011326
3599 011376 011330
3600 011400 011331
3601 011402 011332
3602 011404 011334
3603
3604
3605
3606
3607
3608
3609
3610
3611
3612
3613
3614
3615
3616 011406
3617 011406 012700 011332
3618 011412 016037 000002 011336
3619 011420 066037 000000 011336
3620 011426 146037 177777 011336
3621 011434 156037 177776 011337
3622 011442 026037 177774 011336
3623 011450 001401
3624 011452 104000
3625
3626
3627
3628

125252
TBL1: 0

: THIS TEST VERIFIES MODE 5 DOUBLE OPERAND INSTRUCTIONS.
: THE TEST USES AN ADDRESS TABLE STORED FOLLOWING THE TEST CODE.
: THIS TABLE IS SIMPLY A TABLE OF ADDRESS POINTERS WHICH ADDRESS
: THE DATA TABLE USED IN THE PREVIOUS TEST. THE TEST IS IDENTICAL TO
: THE PREVIOUS TEST EXCEPT THE DATA IS REFERENCED USING THIS ADDRESS
: TABLE AND MODE 5 ADDRESSING. (SEE PREVIOUS TEST).
: *****

: TEST 161 TEST MODE 5 W/ DOP INSTS.
: *****

TS161:
MOV #TBL2+2,R0 ;INITIALIZE R0
MOV @-(R0),@#TBL1 ;TBL1=125252
ADD @-(R0),@#TBL1 ;TBL1=000377
BICB @-(R0),@#TBL1 ;TBL1=000252
BISB @-(R0),@#TBL1+1 ;TBL1=125252
CMP @-(R0),@#TBL1 ;CHECK RESULT
BEQ TS162

DOP5:
EMT ;RESULT OF MODE 5 INSTS. INCORRECT
TBL1-10
TBL1-6
TBL1-5
TBL1-4

TBL2: TBL1-2

: THIS TEST VERIFIES MODE 6 DOUBLE OPERAND INSTRUCTIONS.
: IT USES THE SAME DATA AS THAT USED IN THE MODE 4 TESTS.
: THIS TIME THE DATA IS ACCESSED USING MODE 6. R0 IS SET
: TO POINT TO THE MIDDLE OF THE TABLE. THE TABLE IS ACCESSED FROM
: BOTTOM TO TOP BY VARYING THE OFFSET IN THE MODE 6 INSTRUCTIONS.
: THE DATA RESULTS ARE IDENTICAL TO THOSE EXPECTED IN THE MODE 4
: TESTS.
: *****

: TEST 162 TEST MODE 6 W/ DOP INSTS.
: *****

TS162:
MOV #TBL1-4,R0 ;INITIALIZE R0
MOV 2(R0),@#TBL1 ;TBL1=125252
ADD 0(R0),@#TBL1 ;TBL1=000377
BICB -1(R0),@#TBL1 ;TBL1=000252
BISB -2(R0),@#TBL1+1 ;TBL1=125252
CMP -4(R0),@#TBL1 ;CHECK RESULT
BEQ TS163

EMT ;RESULT OF MODE 6 INSTS. INCORRECT

: THIS TEST VERIFIES MODE 7 DOUBLE OPERAND INSTRUCTIONS.
: THIS TEST USES THE SAME ADDRESS TABLE AND DATA TABLE USED BY

3629
3630
3631
3632
3633
3634
3635
3636
3637
3638 011454
3639 011454 012700 011400
3640 011460 017037 000004 011336
3641 011466 067037 000002 011336
3642 011474 147037 000000 011336
3643 011502 157037 177776 011337
3644 011510 027037 177774 011336
3645 011516 001401
3646 011520 104000
3647
3648
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658 011522
3659 011522 012700 125252
3660 011526 000261
3661 011530 006100
3662 011532 102004
3663 011534 103003
3664 011536 022700 052525
3665 011542 001401
3666 011544
3667 011544 104000
3668 011546 012700 125252
3669 011552 000261
3670 011554 106100
3671 011556 102004
3672 011560 103003
3673 011562 022700 125125
3674 011566 001401
3675 011570
3676 011570 104000
3677
3678
3679
3680
3681
3682
3683
3684

:THE MODE 5 TESTS. THIS TIME THE DATA IS ACCESSED USING MODE 7.
:RO IS SET TO POINT TO THE MIDDLE OF THE ADDRESS TABLE IN THE MODE 5
:TEST. THE TABLE IS ACCESSED FROM BOTTOM TO TOP BY VARYING THE OFFSET
:IN THE MODE 7 INSTRUCTIONS. THE DATA RESULTS ARE IDENTICAL TO
:THOSE EXPECTED IN THE MODE 5 TESTS.

:TEST 163 TEST MODE 7 W/ DOP INSTS.

TS163:

```
MOV #TBL2-4,R0 ;INITIALIZE R0
MOV @4(R0),@#TBL1 ;TBL1=125252
ADD @2(R0),@#TBL1 ;TBL1=000377
BICB @0(R0),@#TBL1 ;TBL1=000252
BISB @-2(R0),@#TBL1+1 ;TBL1=125252
CMP @-4(R0),@#TBL1 ;CHECK RESULT
BEQ TS164
EMT ;RESULT OF MODE 7 INSTS INCORRECT
```

: THIS TEST VERIFIES THE ROTATE MODE 0 INSTRUCTIONS.
:RO IS LOADED WITH A DATA PATTERN, THE C-BIT IS LOADED, AND
:AN ROL INSTRUCTION IS EXECUTED WITH MODE 0. THE OPERATION IS CHECKED
:BY TESTING THE RESULTING DATA AND THE STATE OF THE C AND V BITS.
:NEXT, THE SAME PROCEDURE IS EXECUTED TO TEST MODE 0 BYTE INSTRUCTIONS.

:TEST 164 TEST ROTATE INSTRUCTIONS OF MODE 0

TS164:

```
MOV #125252,R0 ;INITIALIZE DATA
SEC ;SET C-BIT
ROL R0 ;TRY ROL W/ MODE 0
BVC R0TOA ;CC=0011
BCC R0TOA
CMP #052525,R0 ;CHECK DATA
BEQ R0TOB
```

R0TOA:

```
EMT ;ROL MODE 0 FAILED
```

R0TOB:

```
MOV #125252,R0 ;INITIALIZE DATA
SEC ;SET C-BIT
ROLB R0 ;TRY ROL W/ MODE 0 EVEN BYTE
BVC R0TOC ;CC=0011
BCC R0TOC
CMP #125125,R0 ;CHECK DATA
BEQ TS165
```

R0TOC:

```
EMT ;ROLB MODE 0 FAILED
```

: THIS TEST VERIFIES THE ROTATE MODE 1 INSTRUCTIONS.
:THE DATA TO BE ROTATED IS IN LOC 0. RO IS USED AS THE
:ADDRESSING REGISTER. THE C-BIT IS LOADED AND AN ROL IS EXECUTED.
:THE RESULTS ARE CHECKED BY COMPARING THE DATA RESULTS AND TESTING
:THE C AND V BITS. THIS PROCEDURE IS THEN REPEATED TWICE MORE

```

3685
3686
3687
3688
3689
3690 011572
3691 011572 005000
3692 011574 012710 052525
3693 011600 000241
3694 011602 006110
3695 011604 102005
3696 011606 103404
3697 011610 023727 000000 125252
3698 011616 001401
3699 011620
3700 011620 104000
3701 011622 000261
3702 011624 012710 125252
3703 011630 106110
3704 011632 102005
3705 011634 103004
3706 011636 022737 125125 000000
3707 011644 001401
3708 011646
3709 011646 104000
3710 011650 012710 125252
3711 011654 005000
3712 011656 005200
3713 011660 000261
3714 011662 106110
3715 011664 102005
3716 011666 103004
3717 011670 022737 052652 000000
3718 011676 001401
3719 011700
3720 011700 104000
3721
3722
3723
3724
3725
3726
3727
3728
3729
3730
3731
3732 011702
3733 011702 005000
3734 011704 012710 173737
3735 011710 000241
3736 011712 006120
3737 011714 103007
3738 011716 022737 167676 000000
3739 011724 001003
3740 011726 005300
  
```

:TO TEST THE BYTE ROTATES. FIRST ON BYTE 0, THEN ON BYTE 1.

```

:*****
:TEST 165 TEST ROTATE INSTRUCTIONS W/ MODE 1
:*****
TS165:
  
```

```

      CLR      RO      ;POINT TO LOC. 0
      MOV      #52525,(RO) ;INITIALIZE DATA
      CLC      ;CLEAR C-BIT
      ROL      (RO)      ;TRY ROL W/ MODE 1
      BVC      ROT1A     ;CC=1010
      BCS      ROT1A
      CMP      @#0,#125252 ;CHECK RESULT
      BEQ      ROT1B
ROT1A:
      EMT      ;ROL MODE 1 FAILED
ROT1B:
      SEC      ;INITIALIZE DATA
      MOV      #125252,(RO) ;TRY ROLB W/ MODE 1 EVEN BYTE
      ROLB     (RO)      ;CC=1011
      BVC      ROT1C
      BCC      ROT1C
      CMP      #125125,@#0 ;TEST RESULT
      BEQ      ROT1D
ROT1C:
      EMT      ;ROLB W/ MODE 1 EVEN BYTE FAILED
ROT1D:
      MOV      #125252,(RO) ;POINT TO ODD BYTE
      CLR      RO
      INC      RO
      SEC      ;SET C-BIT
      ROLB     (RO)      ;TRY ROLB W/ MODE 1 ODD BYTE
      BVC      ROT1E
      BCC      ROT1E
      CMP      #052652,@#0 ;CHECK DATA
      BEQ      TS166
ROT1E:
      EMT      ;ROLB W/ MODE 1 ODD BYTE FAILED
  
```

```

:*****
:
: THIS TEST VERIFIES MODE 2 ROTATE INSTRUCTIONS.
: THE SAME PROCEDURE AS IN THE OTHER ROTATE TESTS ARE USED. RO
: IS USED AS THE ADDRESSING REGISTER AND IS CHECKED FOR PROPER
: INCREMENTING. BYTE INSTRUCTIONS ARE ALSO CHECKED.
:*****
  
```

```

:TEST 166 TEST ROTATE INSTRUCTIONS W/ MODE 2
:*****
TS166:
  
```

```

      CLR      RO      ;POINT TO LOC 0
      MOV      #173737,(RO) ;INITIALIZE DATA
      CLC      ;CLEAR C-BIT
      ROL      (RO)+     ;TRY ROL W/ MODE 2
      BCC      ROT2A     ;CHECK C-BIT
      CMP      #167676,@#0 ;CHECK DATA
      BNE      ROT2A     ;BRANCH IF RESULT INCORRECT
      DEC      RO      ;TEST RO
  
```

```
3741 011730 005300          DEC      R0
3742 011732 001401          BEQ     ROT2B
3743 011734
3744 011734 104000          ROT2A:
3745 011736 005000          EMT
3746 011740 012710 004040      MOV     #4040,(R0)      ;ROL W/ MODE 2 FAILED
3747 011744 000241          CLC
3748 011746 106120          ROLB   (R0)+          ;POINT TO LOC 0
3749 011750 103406          BCS    ROT2C          ;INITIALIZE DATA
3750 011752 022737 004100 000000  CMP     #4100,@#0     ;CLEAR C-BIT
3751 011760 001002          BNE    ROT2C          ;TRY ROLB W/ MODE 2 EVEN BYTE
3752 011762 005300          DEC     R0            ;CHECK C-BIT
3753 011764 001401          BEQ    ROT2D          ;CHECK DATA
3754 011766
3755 011766 104000          EMT
3756 011770 005000          ROT2D: CLR     R0            ;ROLB W/ MODE 2 EVEN BYTE FAILED
3757 011772 012710 004040      MOV     #4040,(R0)     ;POINT TO LOC 0
3758 011776 005200          INC     R0            ;INITIALIZE DATA
3759 012000 000261          SEC
3760 012002 106120          ROLB   (R0)+          ;POINT TO ODD BYTE OF DATA
3761 012004 103407          BCS    ROT2E          ;SET C-BIT
3762 012006 022737 010440 000000  CMP     #10440,@#0    ;TRY ROL W/ MODE 2 ODD BYTE
3763 012014 001003          BNE    ROT2E          ;CHECK C-BIT
3764 012016 005300          DEC     R0            ;CHECK DATA
3765 012020 005300          DEC     R0            ;BRANCH IF DATA INCORRECT
3766 012022 001401          BEQ    TS167          ;CHECK R0
3767 012024
3768 012024 104000          ROT2E: EMT
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3780 012026
3781 012026 012737 052525 000000  MOV     #52525,@#0    ;INITIALIZE DATA IN LOC 0
3782 012034 000261          SEC
3783 012036 006137 000000      ROL     @#0           ;SET C-BIT
3784 012042 103404          BCS    ROT3A          ;TRO ROL W/ MODE 3
3785 012044 022737 125253 000000  CMP     #125253,@#0   ;CHECK C-BIT
3786 012052 001401          BEQ    ROT3B          ;CHECK DATA
3787 012054          ROT3A:
3788 012054 104000          EMT
3789 012056 012737 125252 000000  ROT3B: MOV     #125252,@#0 ;ROL W/ MODE 3 FAILED
3790 012064 000241          CLC
3791 012066 106137 000000      ROLB   @#0           ;INITIALIZE DATA
3792 012072 103004          BCC    ROT3C          ;CLEAR C-BIT
3793 012074 023727 000000 125124 4$: CMP     @#0,#125124   ;TRY ROL W/ MODE 3 EVEN BYTE
3794 012102 001401          BEQ    ROT3D          ;CHECK C-BIT
3795 012104
3796 012104 104000          ROT3C: EMT
3797
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
3824
3825
3826
3827
3828
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3910
3911
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
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3999
```

```
*****
:
: THIS TEST VERIFIES MODE 3 ROTATE INSTRUCTIONS.
: THIS TEST USES THE SAME PROCEDURES AS IN THE OTHER ROTATE
: TESTS. THE DATA IS STORED IN LOC. 0 AND IS ADDRESSED USING
: MODE 37. BYTE ADDRESSING IS ALSO CHECKED FOR EVEN AND ODD BYTES.
:
: *****
: TEST 167 TEST ROTATE INSTRUCTIONS /W MODE 3
: *****
```

```
TS167:
MOV     #52525,@#0      ;INITIALIZE DATA IN LOC 0
SEC
SET C-BIT
TRO ROL W/ MODE 3
CHECK C-BIT
CHECK DATA
ROT3A:
EMT
ROT3B: MOV     #125252,@#0 ;ROL W/ MODE 3 FAILED
INITIALIZE DATA
CLEAR C-BIT
TRY ROL W/ MODE 3 EVEN BYTE
CHECK C-BIT
CHECK DATA
4$: CMP     @#0,#125124
CHECK DATA
ROT3D:
EMT
ROL W/ MODE 3 EVEN BYTE FAILED
```

3797 012106 012737 125252 000000
3798 012114 000261
3799 012116 106137 000001
3800 012122 103004
3801 012124 022737 052652 000000
3802 012132 001401
3803 012134
3804 012134 104000
3805
3806
3807
3808
3809
3810
3811
3812
3813
3814
3815
3816
3817 012136
3818 012136 012737 070707 000000
3819 012144 012700 000002
3820 012150 000261
3821 012152 006140
3822 012154 103406
3823 012156 022737 161617 000000
3824 012164 001002
3825 012166 005700
3826 012170 001401
3827 012172
3828 012172 104000
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3840
3841
3842
3843 012174
3844 012174 012737 012240 000000
3845 012202 012700 000002
3846 012206 012767 107070 000024
3847 012214 000241
3848 012216 006150
3849 012220 103006
3850 012222 022737 016160 012240
3851 012230 001002
3852 012232 005700

ROT3D: MOV #125252,@#0 ;INITIALIZE DATA IN LOC. 0
SEC ;SET C-BIT
ROLB @#1 ;TRY ROL W/ MODE 3 ODD BYTE
BCC ROT3E ;CHECK C-BIT
CMP #052652,@#0 ;CHECK DATA
BEQ TS170
ROT3E: EMT ;ROL W/ MODE 3 ODD BYTE FAILED

: THIS TEST VERIFIES MODE 4 ROTATE INSTRUCTIONS. THE DATA IS
: STORED IN LOC. 0. RO IS SET TO 2 AND THE CARRY IS SET. AN ROL MODE 4
: IS USED TO ROTATE LOCATION 0 USING RO. THE DATA IS CHECKED
: AND THE C AND V BITS ARE TESTED. THE PROPER DECREMENTING OF
: RO IS VERIFIED.

: TEST 170 TEST MODE 4 W/ ROTATE INSTRUCTIONS

TS170: MOV #070707,@#0 ;INITIALIZE DATA IN LOC. 0
MOV #2,RO ;INITIALIZE RO AS POINTER
SEC ;SET C-BIT
ROL -(RO) ;TRY ROL W/ MODE 4
BCS ROT4 ;CHECK C-BIT
CMP #161617,@#0 ;CHECK DATA
BNE ROT4 ;BRANCH IF DATA INCORRECT
TST RO ;CHECK MODE 4 REGISTER
BEQ TS171

ROT4: EMT ;ROL MODE 4 FAILED

: THIS TEST VERIFIES MODE 5 ROTATE INSTRUCTIONS.
: THE DATA IS STORED IN A WORK LOCATION (ROTX) AT THE END OF THE
: TEST CODE. LOC. 0 IS LOADED WITH THE ADDRESS OF THE DATA (ROTX).
: RO IS SET TO 2. THE CARRY IS CLEARED AND A MODE 5 ROL
: IS EXECUTED USING RO AS AN ADDRESSING REGISTER. THE DATA IS
: CHECKED, THE C AND V BITS TESTED, AND RO CHECKED FOR PROPER
: DECREMENTING.

: TEST 171 TEST MODE 5 W/ ROTATE INSTRUCTIONS

TS171: MOV #ROTX,@#0 ;MOVE POINTER TO LOC. 0
MOV #2,RO ;SET MODE 5 REG. TO LOC. 0
MOV #107070,ROTX ;INITIALIZE DATA
CLC ;CLEAR C-BIT
ROL @-(RO) ;TRY ROL W/ MODE 5
BCC ROT5 ;CHECK C-BIT
CMP #016160,@#ROTX ;CHECK DATA
BNE ROT5 ;BRANCH IF DATA INCORRECT
TST RO ;CHECK MODE 5 REGISTER

3853 012234 001402
3854 012236
3855 012236 104000
3856 012240 000000
3857
3858
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868 012242
3869 012242 012737 125252 012240
3870 012250 000261
3871 012252 006167 177762
3872 012256 103004
3873 012260 022737 052525 012240
3874 012266 001401
3875 012270
3876 012270 104000
3877
3878
3879
3880

ROTS: BEQ TS172
EMT ;ROL MODE 5 FAILED
ROTX: 0

: THIS TEST VERIFIES MODE 6 ROTATE INSTRUCTIONS.
: IT USES THE SAME PROCEDURE AS THE ABOVE TEST EXCEPT THE
: ROTATE INSTRUCTION USES MODE 6 ADDRESSING WITH REGISTER 7.
: THE DATA IS STILL OPERATED ON IN LOC. ROTX (SEE PREVIOUS TEST).

TEST 172 TEST MODE 6 W/ ROTATE INSTRUCTIONS

TS172:
MOV #125252,@#ROTX ;INITIALIZE DATA
SEC ;SET C-BIT
ROL ROTX ;TRY ROL W/ MODE 6
BCC ROT6 ;CHECK C-BIT
CMP #52525,@#ROTX ;CHECK DATA
BEQ TS173
ROTS: EMT ;ROL W/ MODE 6 FAILED

: THIS TEST VERIFIES MODE 7 ROTATE INSTRUCTIONS.

3881
3882
3883
3884
3885
3886
3887
3888 012272
3889 012272 012737 052525 012240
3890 012300 012737 012240 012330
3891 012306 000241
3892 012310 006177 000014
3893 012314 103404
3894 012316 023727 012240 125252
3895 012324 001402
3896 012326
3897 012326 104000
3898 012330 000000
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3910
3911 012332
3912 012332 012700 177400
3913 012336 000300
3914 012340 100401
3915 012342 104000
3916 012344 022700 000377
3917 012350 001401
3918 012352 104000
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3930 012354
3931 012354 012737 125652 000000
3932 012362 005000
3933 012364 000310
3934 012366 022737 125253 000000
3935 012374 001401
3936 012376 104000

:THE DATA IS SET IN LOC. ROTX, (SEE PREVIOUS TEST). THE ROL INSTRUCTION
:ADDRESSES IT INDIRECTLY USING MODE 7 AND INDIRECT ADDRESS LOCATION
:(ROTXAD) FOLLOWING THE TEST CODE.

:TEST 173 TEST MODE 7 W/ ROTATE INSTRUCTIONS

TS173:
MOV #52525,@ROTX ;INITIALIZE DATA
MOV #ROTX,@ROTXAD ;INITIALIZE ADDRESS POINTER
CLC ;CLEAR C-BIT
ROL @ROTXAD ;TRY ROL W/ MODE 7
BCS ROT7 ;CHECK C-BIT
CMP @ROTX,#125252 ;CHECK DATA
BEQ TS174

ROT7:
EMT ;ROL W/ MODE 7 FAILED
ROTXAD: 0

: THIS TEST VERIFIES MODE 0 SWAB INSTRUCTION. RO IS SET TO
:177400. A SWAB MODE 0 IS EXECUTED AND THE CONDITIONAL BRANCH
:IS USED TO CHECK THE SIGN OF THE RESULT. ALSO, A COMPARISON
:IS MADE TO CHECK THE DATA RESULTS.

:TEST 174 TEST MODE 0 W/ SWAB INST.

TS174:
MOV #177400,RO ;MOVE TEST PATTERN TO RO
SWAB RO ;TRY SWAB MODE 0
BMI SBO
EMT ;SWAB DID NOT SET CC'S CORRECT
SBO: CMP #377,RO ;CHECK RESULT
BEQ TS175
EMT ;RESULT OF SWAB MODE 0 FAILED

: THIS TEST VERIFIES MODE 1 SWAB INSTRUCTION. THE TEST
:PATTERN IS MOVED TO LOC 0. RO IS CLEARED AND USED AS THE ADDRESSING
:REGISTER IN THE MODE 1 SWAB. THE DATA RESULTS ARE CHECKED WITH
:A COMPARE.

:TEST 175 TEST MODE 1 W/ SWAB INST

TS175:
MOV #125652,@#0 ;MOVE TEST PATTERN TO LOC. 0
CLR RO ;RO=0
SWAB (RO) ;TRY SWAB MODE 1
CMP #125253,@#0 ;CHECK RESULT
BEQ TS176
EMT ;RESULT OF SWAB MODE 1 FAILED

3937
3938
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949 012400
3950 012400 012737 125152 000000
3951 012406 005000
3952 012410 000320
3953 012412 022737 065252 000000
3954 012420 001401
3955 012422 104000
3956 012424 162700 000002
3957 012430 001401
3958 012432 104000
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3970
3971 012434
3972 012434 012737 000377 00000
3973 012442 000337 000000
3974 012446 022737 177400 000000
3975 012454 001401
3976 012456 104000
3977
3978
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989 012460
3990 012460 012737 125652 000000
3991 012466 012700 000002
3992 012472 000340

: THIS TEST VERIFIES MODE 2 SWAB INSTRUCTION. THE TEST
: PATTERN IS MOVED TO LOC 0. RO IS CLEARED AND USED AS THE MODE
: 2 ADDRESSING REGISTER. THE RESULTS ARE CHECKED WITH A COMPARE.
: RO IS CHECKED FOR PROPER DECREMENTING.

: TEST 176 TEST MODE 2 W/ SWAB INST

TS176:
MOV #125152,@#0 ;MOVE TEST PATTERN TO LOC. 0
CLR RO ;RO=0
SWAB (RO)+ ;TRY SWAB MODE 2
CMP #65252,@#0 ;CHECK RESULT
BEQ SB2
EMT ;RESULT OF SWAB MODE 0 FAILED
SB2: SUB #2,RO ;CHECK EFFECT OF REG.
BEQ TS177
EMT ;REGISTER VALUE INCORRECT

: THIS TEST VERIFIES MODE 3 SWAB INSTRUCTION. THE TEST
: PATTERN IS MOVED TO LOC 0. A MODE 3 SWAB INSTRUCTION IS EXECUTED
: USING R7 AS THE ADDRESSING REGISTER. A COMPARE VERIFIES THE
: DATA RESULTS.

: TEST 177 TEST MODE 3 W/SWAB INST.

TS177:
MOV #377,@#0 ;MOVE TEST PATTERN TO LOC. 0
SWAB @#0 ;TRY SWAB W/ MODE 3
CMP #177400,@#0 ;CHECK RESULT
BEQ TS200
EMT ;RESULT OF SWAB INCORRECT

: THIS TEST VERIFIES MODE 4 SWAB INSTRUCTIONS. THE DATA
: IS MOVED TO LOC 0. RO IS SET TO 2 AND USED AS THE MODE 4 ADDRESSING
: REGISTER. THE DATA IS CHECKED WITH A COMPARE AND RO IS CHECKED
: FOR PROPER DECREMENTING.

: TEST 200 TEST MODE 4 W/ SWAB INST

TS200:
MOV #125652,@#0 ;MOVE TEST PATTERN TO LOC. 0
MOV #2,RO ;SET UP REGISTER POINTER
SWAB -(RO) ;TRY SWAB MODE 4

3993 012474 022737 125253 000000
3994 012502 001401
3995 012504 104000
3996 012506 005700
3997 012510 001401
3998 012512 104000
3999

CMP #125253,@#0 ;CHECK RESULT
BEQ SB4
EMT ;RESULT OF SWAB INCORRECT
SB4: TST R0 ;CHECK EFFECT ON REG.
BEQ TS201
EMT ;REGISTER VALUE INCORRECT

400
4001
4002
4003
4004
4005
4006
4007
4008
4009
4010
4011

: THIS TEST VERIFIES MODE 5 SWAB INSTRUCTION. THE TEST USES
: TWO LOCATIONS FOLLOWING THE TEST CODE. SB5X HOLDS THE DATA;
: SB5XAD IS A POINTER TO THE DATA LOCATION. THE DATA IS MOVED TO
: SB5X AND R0 IS SET TO TWO PLUS THE ADDRESS OF SB5XAD. FOLLOWING
: THE MODE 5 SWAB SB5X IS CHECKED FOR THE PROPER DATA. R0 IS
: CHECKED TO SEE THAT IT WAS DECREMENTED PROPERLY.

4012
4013 012514
4014 012514 012700 012556
4015 012520 012767 125125 000024
4016 012526 000350
4017 012530 022767 052652 000014
4018 012536 001401
4019 012540 104000
4020 012542 020027 012554
4021 012546 001403
4022 012550
4023 012550 104000
4024 012552 000000
4025 012554 012552

: TEST 201 TEST MODE 5 W/ SWAB INST.

TS201:
MOV #SB5XAD+2,R0 ;SET UP POINTER TO WORK LOCATION
MOV #125125,SB5X ;MOVE PATTERN TO WORK LOCATION
SWAB @-(R0) ;TRY SWAB MODE 5
CMP #52652,SB5X ;CHECK RESULT
BEQ SB5A
EMT ;RESULT OF SWAB INCORRECT
SB5A: CMP R0,#SB5XAD ;CHECK RESULT OF REG.
BEQ TS202
SB5:
EMT ;REGISTER VALUE INCORRECT
SB5X: 0 ;WORK LOCATION
SB5XAD: SB5X

4026
4027
4028
4029
4030
4031
4032
4033
4034
4035
4036
4037
4038
4039

: THIS TEST VERIFIES MODE 6 SWAB INSTRUCTION. THIS TEST
: USES A WORK LOCATION (SB6X) FOLLOWING THE TEST CODE. TEST DATA
: IS LOADED INTO THE WORK LOCATION. R0, THE ADDRESSING REGISTER
: IS LOADED WITH 6 LESS THEN THE ADDRESS OF THE WORK LOCATION.
: THE MODE 6 SWAB IS EXECUTED WITH A +6 OFFSET. THE DATA IS
: VERIFIED WITH A COMPARE.

4040 012556
4041 012556 012767 125125 000022
4042 012564 012700 012600
4043 012570 000360 000006
4044 012574 022760 052652 000006
4045 012602 001402
4046 012604
4047 012604 104000
4048 012606 000000

: TEST 202 TEST MODE 6 W/ SWAB INST.

TS202:
MOV #125125,SB6X ;MOVE PATTERN TO WORK LOCATION
MOV #SB6X-6,R0 ;MOVE OFFSET POINTER TO R0
SWAB 6(R0) ;TRY SWAB W/ MODE 6
CMP #52652,6(R0) ;CHECK RESULT
BEQ TS203
SB6:
EMT ;RESULT OF SWAB INCORRECT
SB6X: 0 ;WORK LOCATION

4049
4050
4051
4052
4053
4054
4055
4056
4057
4058
4059
4060
4061
4062
4063
4064 012610
4065 012610 012767 177400 000022
4066 012616 012700 012550
4067 012622 000370 000072
4068 012626 027027 000072 000377
4069 012634 001403
4070 012636
4071 012636 104000
4072 012640 000000
4073 012642 012640
4074
4075
4076
4077
4078
4079
4080
4081
4082
4083
4084
4085
4086
4087
4088
4089
4090
4091
4092
4093
4094
4095
4096
4097
4098
4099
4100
4101
4102
4103
4104

```

:*****
:
:      THIS TEST VERIFIES MODE 7 SWAB INSTRUCTION. THIS TEST
:      USES TWO LOCATIONS FOLLOWING THE TEST CODE: A WORK LOCATION
:      (SB7X) AND A POINTER TO THE WORK LOCATION (SB7XAD). DATA IS MOVED
:      TO THE WORK LOCATION. R0 IS LOADED WITH 72 LESS THAN THE ADDRESS
:      OF THE ADDRESS POINTER. THE DATA IS SWAB'ED USING A MODE 7
:      INSTRUCTION WITH AN OFFSET OF +72. THE DATA IS VERIFIED WITH A
:      COMPARE.
:
:*****

```

```

:*****
:TEST 203      TEST MODE 7 W/ SWAB INST.
:*****
TS203:
      MOV      #177400,SB7X      ;MOVE PATTERN TO WORK LOCATION
      MOV      #SB7XAD-72,R0    ;MOVE OFFSET POINTER TO R0
      SWAB     @72(R0)          ;TRY SWAB MODE 7
      CMP      @72(R0),#377     ;CHECK RESULTS
      BEQ      TS204
SB7:
      EMT
SB7X: 0
SB7XAD: SB7X      ;RESULT OF SWAB INCORRECT
                        ;WORK LOCATION
                        ;POINTER TO WORK LOCATION
:*****

```

```

:*****
:
:      THIS TEST VERIFIES ALL LEGAL MODES OF THE JMP INSTRUCTION.
:      BECAUSE OF THE NATURE OF THE INSTRUCTION UNDER TEST, THIS TEST
:      UTILIZES SEVERAL DIFFERENT TECHNIQUES. THE CODE IS NOT EXECUTED
:      IN A LINEAR FASHION. THE DIFFERENT MODES ARE EXECUTED IN ORDER
:      FROM 1-7; HOWEVER, THE CODE IS ARRANGED SO THAT CONTROL LEAP
:      FROGS THRU THE TEST CODE. THE ORDER OF APPEARANCE OF THE CODE
:      IS:
:
:      JMP MODE 1
:      JMP MODE 3
:      JMP MODE 2
:      JMP MODE 4
:      JMP MODE 6
:      JMP MODE 5
:      JMP MODE 7
:
:      AN INTERNAL SEQUENCE TEST (JMPSEQ) IS USED TO INSURE THAT THE
:      JUMPS ARE OCCURRING IN THE PROGRAMMED SEQUENCE.
:      THE TEST IS MADE UP OF SEVERAL BLOCKS OF CODE. EACH CODE
:      BEGINS WITH A LABEL WHICH INDICATES THE MODE BEING EXECUTED IN
:      THAT BLOCK. A SIMPLE PROCEDURE IS FOLLOWED IN EACH BLOCK. FOR
:      EXAMPLE THE CODE BEGINNING AT JMP3 WILL FIRST COMPARE THE RESULTS
:      OF THE PREVIOUS MODE 2 JUMP. (ANY REGISTER CHANGES ARE VERIFIED
:      AND THE SEQUENCE CHECK IS MADE). THEN THE REGISTERS ARE SETUP
:      FOR A MODE 3 JUMP TO THE NEXT TEST BLOCK (HERE, JMP4), THE SEQUENCE
:      CHECKER IS UPDATED AND THE JUMP IS EXECUTED.
:      IF A FAILURE OCCURS, THE SEQUENCE CHECKER WILL ASSIST IN
:      DETERMINING JUST WHICH MODE FAILED. IF THE SEQUENCE IS CORRECT
:      THEN THE ERROR DETECTED WAS A MODE FAILURE (E.G. FAILURE OF THE
:*****

```

```
4105 ;REGISTER TO BE INCREMENTED IN MODE 2 JUMP.)
4106 :
4107 :*****
4108 :TEST 204 TEST THE JMP INSTRUCTION IN ALL MODES
4109 :*****
4110 TS204:
4111 012644 005067 C00240 CLR JMPSEQ ;ESTABLISH A SEQUENCE CHECKER
4112 012650 012700 012714 MOV #JMP2,R0 ;SET R0=JUMP TARGET
4113 012654 000110 JMP (R0) ;TRY JMP MODE 1
4114 012656 022700 012660 JMP3: CMP #.+2,R0 ;CHECK RESULT OF MODE 2 JUMP
4115 012662 001401 BEQ JMP3A
4116 012664 104000 EMT ;REGISTER VALUE AFTER JMP MODE 2 INCORRECT
4117 012666 026727 000216 000001 JMP3A: CMP JMPSEQ,#1 ;MAKE SURE JMPS ARE IN SEQUENCE: JMPSEQ=1?
4118 012674 001401 BEQ JMP3B
4119 012676 104000 EMT ;SHOULD BE HERE FROM JMP MODE 2 ONLY
4120 012700 012700 012712 JMP3B: MOV #IJMP4,R0 ;POINT R0 TO INDIRECT JMP ADDR.
4121 012704 005267 000200 INC JMPSEQ ;UPDATE SEQUENCE CHECKER
4122 012710 000130 JMP @-(R0)+ ;TRY JMP MODE 3
4123 012712 012736 IJMP4: JMP4 ;ADDRESS INDIRECT JUMP
4124
4125 012714 005767 000170 JMP2: TST JMPSEQ ;CHECK THAT JMPS ARE IN SEQUENCE: JMPSEQ=0?
4126 012720 001401 BEQ JMP2A
4127 012722 104000 EMT ;SHOULD BE HERE FROM JMP MODE 1 ONLY
4128 012724 005267 000160 JMP2A: INC JMPSEQ ;UPDATE SEQUENCE CHECKER
4129 012730 012700 012656 MOV #JMP3,R0 ;SET R0=JUMP TARGET
4130 012734 000120 JMP (R0)+ ;TRY A JUMP MODE 2 TO "JMP3"
4131 012736 022700 012714 JMP4: CMP #IJMP4+2,R0 ;CHECK RESULT OF REGISTER IN MODE 3 JUMP
4132 012742 001401 BEQ JMP4A
4133 012744 104000 EMT ;REGISTER VALUE AFTER MODE 3 JUMP INCORRECT
4134 012746 022767 000002 000134 JMP4A: CMP #2,JMPSEQ ;CHECK JUMP SEQUENCE: JMPSEQ=2?
4135 012754 001401 BEQ JMP4B
4136 012756 104000 EMT ;SHOULD BE ONLY FROM MODE 3 JUMP
4137 012760 012700 013022 JMP4B: MOV #JMP5+2,R0 ;SET UP POINTER TO JUMP TARGET
4138 012764 005267 000120 INC JMPSEQ ;UPDATE SEQUENCE CHECKER
4139 012770 000140 JMP -(R0) ;TRY JUMP MODE 4 TO "JMP4"
4140
4141 012772 022767 000004 000110 JMP6: CMP #4,JMPSEQ ;CHECK THAT JUMPS ARE IN SEQUENCE: JMPSEQ=4?
4142 013000 001401 BEQ JMP6A
4143 013002 104000 EMT ;SHOULD BE HERE ONLY FROM MODE 5 JUMP
4144 013004 012700 013444 JMP6A: MOV #JMP7+376,R0 ;SET UP OFFSET POINTER TO JUMP TARGET
4145 013010 005267 000074 INC JMPSEQ ;UPDATE JUMP SEQUENCE
4146 013014 000160 177402 JMP -376(R0) ;TRY MODE 6 JUMP
4147
4148 013020 022767 000003 000062 JMP5: CMP #3,JMPSEQ ;CHECK THAT JUMPS ARE IN SEQUENCE: JMPSEQ=3?
4149 013026 001401 BEQ JMP5A
4150 013030 104000 EMT ;SHOULD ONLY BE HERE FROM MODE 4 JUMP
4151 013032 012700 013046 JMP5A: MOV #IJMP5+2,R0 ;SET UP POINTER TO INDIRECT JUMP ADDR.
4152 013036 005267 000046 INC JMPSEQ ;UPDATE JUMP SEQUENCE
4153 013042 000150 JMP @-(R0) ;TRY JUMP MODE 5 TO "JMP6"
4154 013044 012772 IJMP5: JMP6 ;INDIRECT ADDRESS POINTER
4155
4156 013046 022767 000005 000034 JMP7: CMP #5,JMPSEQ ;CHECK JUMPS IN SEQUENCE: JMPSEQ=5?
4157 013054 001401 BEQ JMP7A
4158 013056 104000 EMT ;SHOULD ONLY BE HERE FROM MODE 6 JUMP
4159 013060 012700 013104 JMP7A: MOV #IJMP+10,R0 ;SET UP OFFSET POINTER TO INDIRECT ADDR.
4160 013064 005267 000020 INC JMPSEQ ;UPDATE JUMP SEQUENCE
```

4161 013070 000170 177770
4162 013074 013076
4163
4164 013076 026727 000006 000006
4165 013104 001402
4166 013106 104000
4167 013110 000000
4168
4169
4170
4171
4172
4173
4174
4175
4176
4177
4178
4179
4180
4181
4182
4183
4184
4185
4186
4187
4188 013112
4189 013112 000402
4190 013114 000137 013476
4191
4192 013120 012706 001000
4193 013124 012700 013216
4194 013130 005037 013456
4195 013134 005001
4196 013136 005101
4197 013140 004110
4198
4199
4200 013142
4201 013142 104000
4202
4203 013144 022737 000001 013456
4204 013152 001014
4205 013154 020127 013272
4206 013160 001011
4207 013162 022706 000776
4208 013166 001006
4209 013170 022716 125252
4210 013174 001003
4211 013176 022700 013146
4212 013202 001401
4213 013204
4214 013204 104000
4215 013206 005237 013456
4216 013212 004137 013272

JMP @-10(R0) ;TRY MODE 7 JUMP
JMPCK : JMPCK ;INDIRECT ADDRESS
JMPCK: CMP JMPSEQ,#6 ;CHECK JUMPS IN SEQUENCE: JMPSEQ
BEQ TS205
EMT ;SHOULD ONLY BE HERE FROM MODE 6 JUMP
JMPSEQ: 0

: THIS TEST VERIFIES ALL LEGAL MODES OF THE JSR INSTRUCTION.
: THE CONCEPT OF LEAP FROGGING AND SEQUENCE CHECKING (JSRSEQ) IS
: IDENTICAL TO THAT USED IN JMP TEST (SEE PREVIOUS TEST). EACH
: BLOCK OF CODE VERIFIES THE PREVIOUS JSR BY CHECKING THE SEQUENCE,
: CHECKING THAT THE PC WAS SAVED IN THE SPECIFIED REGISTER, CHECKING
: THAT THE SP WAS DECREMENTED, CHECKING THAT THE REGISTER WAS
: SAVED ON THE STACK, AND FINALLY CHECKING THAT ANY MODE ADDRESS
: REGISTER ALTERATIONS (E.G. INCREMENT REGISTER IN MODE 2) WERE
: SUCCESSFUL. R1 IS USED AS THE REGISTER IN ALL JSR INSTRUCTIONS.
: IF A FAILURE OCCURS, THE SEQUENCE CHECKER WILL ASSIST IN
: DETERMINING JUST WHICH MODE FAILED. IF THE SEQUENCE IS CORRECT
: THEN THE ERROR DETECTED WAS A FUNCTIONAL FAILURE (E.G., INCORRECT
: REGISTER SAVED).

TEST 205 TEST JSR INSTRUCTION W/ ALL MODES

:TEST 205 TEST JSR INSTRUCTION W/ ALL MODES

TS205:
JSR0: BR JSR1
JMP @#JSRCK1
JSR1: MOV #STBOT,R6 ;SET STACK POINTER
MOV #JSR2,R0 ;SET TARGET ADDRESS
CLR @#JSRSEQ ;INITIALIZE SEQUENCE CHECKER
CLR R1 ;INITIALIZE R1
COM R1
JSR R1,(R0) ;TRY JSR MODE 1
; TO SCOPE: REPLACE THE MOVE INSTRUCTION <=
; FOLLOWING W/ 774 <=
JSR1A: EMT ;JSR MODE 1 FAILED
JSR3: CMP #1,@#JSRSEQ ;CHECK SEQUENCE: JSRSEQ=1?
BNE JSR3A ;BRANCH IF OUT OF SEQUENCE
CMP R1,#JSR4 ;PROPER PC SAVED?
BNE JSR3A ;BRANCH IF PC WRONG
CMP #STBOT-2,R6 ;STACK POINTER DECREMENTED?
BNE JSR3A ;BRANCH IF SP WRONG
CMP #125252,(R6) ;REG SAVED ON STACK?
BNE JSR3A ;BRANCH IF REG. NOT SAVED
CMP #JSR3+2,R0 ;MODE 2 INCREMENT CORRECT?
BEQ JSR3B
JSR3A: EMT ;JSR MODE 3 MALFUNCTIONED
JSR3B: INC @#JSRSEQ ;UPDATE SEQUENCE CHECKER
JSR R1,@#JSR4 ;TRY JSR MODE 4

4217									
4218	013216	005737	013456		JSR2:	TST @#JSRSEQ		:CHECK SEQUENCE: JSRSEQ=0?	
4219	013222	001011				BNE JSR2A		:BRANCH IF OUT OF SEQUENCE	
4220	013224	020127	013142			CMP R1,#JSR1A		:PROPER PC SAVED?	
4221	013230	001006				BNE JSR2A		:BRANCH IF PC WRONG	
4222	013232	022706	000776			CMP #STBOT-2,R6		:R6 DECREMENT?	
4223	013236	001003				BNE JSR2A		:BRANCH IF R6 IS INCORRECT	
4224	013240	021627	177777			CMP (R6), #-1		:REGISTER SAVED?	
4225	013244	001401				BEQ JSR2B			
4226	013246				JSR2A:				
4227	013246	104000				EMT		:JSR MODE 1 MALFUNCTIONED	
4228	013250	012706	001000		JSR2B:	MOV #STBOT,R6		:INITIALIZE R6	
4229	013254	012701	125252			MOV #125252,R1		:INITIALIZE R1	
4230	013260	005237	013456			INC @#JSRSEQ		:UPDATE SEQUENCE CHECKER	
4231	013264	012700	013144			MOV #JSR3,R0		:SET TARGET ADDRESS	
4232	013270	004120				JSR R1,(R0)+		:TRY JSR MODE 2	
4233									
4234	013272	022737	000002	013456	JSR4:	CMP #2,@#JSRSEQ		:CHECK SEQUENCE: JSRSEQ=2?	
4235	013300	001003				BNE JSR4A		:BRANCH IF OUT OF SEQUENCE	
4236	013302	022701	013216			CMP #JSR2,R1		:PROPER PC SAVED?	
4237	013306	001401				BEQ JSR4B			
4238	013310				JSR4A:				
4239	013310	104000				EMT		:JSR MODE 3 MALFUNCTIONED	
4240	013312	005237	013456		JSR4B:	INC @#JSRSEQ		:UPDATE SEQUENCE CHECKER	
4241	013316	012700	013364			MOV #JSR5+2,R0		:SET TARGET ADDRESS	
4242	013322	004140				JSR R1,-(R0)		:TRY JSR MODE 4	
4243									
4244	013324	022767	000004	000124	JSR6:	CMP #4,JSRSEQ		:CHECK SEQUENCE: JSRSEQ=4?	
4245	013332	001006				BNE JSR6A		:BRANCH IF OUT OF SEQUENCE	
4246	013334	022701	013422			CMP #JSR7,R1		:PROPER PC SAVED?	
4247	013340	001003				BNE JSR6A		:BRANCH IF PC WRONG	
4248	013342	022700	013452			CMP #JSR6AD,R0		:MODE 5 REGISTER CORRECT?	
4249	013346	001401				BEQ JSR6B			
4250	013350				JSR6A:				
4251	013350	104000				EMT		:JSR MODE 5 FAILED	
4252	013352	005237	013456		JSR6B:	INC @#JSRSEQ		:UPDATE SEQUENCE CHECKER	
4253	013356	004167	000040			JSR R1,JSR7		:TRY JSR MODE 6	
4254	013362	022767	000003	000066	JSR5:	CMP #3,JSRSEQ		:CHECK SEQUENCE: JSRSEQ=3?	
4255	013370	001006				BNE JSR5A		:BRANCH IF OUT OF SEQUENCE	
4256	013372	022701	013324			CMP #JSR6,R1		:PROPER PC SAVED?	
4257	013376	001003				BNE JSR5A		:BRANCH IF PC WRONG	
4258	013400	022700	013362			CMP #JSR5,R0		:CHECK MODE 4 REGISTER	
4259	013404	001401				BEQ JSR5B			
4260	013406				JSR5A:				
4261	013406	104000				EMT		:JSR MODE 4 MALFUNCTIONED	
4262	013410	005237	013456		JSR5B:	INC @#JSRSEQ		:UPDATE SEQUENCE CHECKER	
4263	013414	012700	013454			MOV #JSR6AD+2,R0		:POINT R0 TO TARGET ADDRESS	
4264	013420	004150				JSR R1,@-(R0)		:TRY JSR MODE 5	
4265									
4266	013422	022737	000005	013456	JSR7:	CMP #5,@#JSRSEQ		:CHECK SEQUENCE: JSRSEQ=5?	
4267	013430	001003				BNE JSR7A		:BRANCH IF OUT OF SEQUENCE	
4268	013432	022701	013362			CMP #JSR5,R1		:PROPER PC SAVED?	
4269	013436	001401				BEQ JSR7B			
4270	013440				JSR7A:				
4271	013440	104000				EMT		:JSR MODE 6 FAILED	
4272	013442	005237	013456		JSR7B:	INC @#JSRSEQ		:UPDATE SEQUENCE CHECKER	

4273 013446 004177 000002
4274
4275 013452 013324
4276 013454 013460
4277 013456 000000
4278
4279 013460 022767 000006 177770
4280 013466 001003
4281 013470 022701 013452
4282 013474 001401
4283 013476
4284 013476 104000
4285
4286
4287
4288
4289
4290
4291
4292
4293
4294
4295
4296
4297 013500
4298 013500 012706 001000
4299 013504 012746 052525
4300 013510 012700 013520
4301 013514 000200
4302
4303
4304 013516 104000
4305 013520 022700 052525
4306 013524 001401
4307 013526 104000
4308
4309
4310
4311
4312
4313
4314
4315
4316
4317
4318
4319
4320
4321
4322
4323
4324
4325 013530
4326 013530 000277
4327 013532 000251
4328 013534 012700 100000

JSR R1,@JSRCKAD ;TRY JSR MODE 7
JSR6AD: JSR6 ;MODE 5 TARGET ADDRESS
JSRCKAD:JSRCK ;MODE 7 TARGET ADDRESS
JSRSEQ: 0 ;SEQUENCE CHECKER
JSRCK: CMP #6,JSRSEQ ;CHECK SEQUENCE: JSRSEQ=6?
BNE JSRCK1 ;BRANCH IF OUT OF SEQUENCE
CMP #JSR6AD,R1 ;PROPER PC SAVED?
BEQ TS206
JSRCK1: EMT ;JSR MODE 7 MALFUNCTIONED

: THIS TEST VERIFIES THE RTS INSTRUCTION. THE STACK POINTER
: IS INITIALIZED AND A TEST PATTERN STORED ON STACK. R0 IS LOADED
: WITH RETURN ADDRESS. AN RTS IS EXECUTED, AND, AT THE TARGET
: ADDRESS, A CHECK IS MADE THAT R0 WAS PROPERLY RESTORED FROM THE
: STACK.
: *****
: TEST 206 TEST RTS INSTRUCTION
: *****

TS206:
MOV #STBOT,R6 ;INITIALIZE STACK POINTER
MOV #52525,-(R6) ;INITIALIZE TOP OF STACK
MOV #RTS1,R0 ;INITIALIZE RETURN REGISTER
RTS R0 ;TRY RTS THROUGH R0
: TO SCOPE: REPLACE THE MOVE INSTRUCTION <=====
: FOLLOWING W/ 770 <=====
EMT ;RTS FAILED
RTS1: CMP #52525,R0 ;CHECK THAT R0 RESTORED FROM STACK
BEQ TS207
EMT ;RTS MALFUNCTIONED

: THESE NEXT FOUR TESTS VERIFY THE FUNCTIONING OF A GROUP
: OF FOUR INSTRUCTIONS. THE GROUP CONSISTS OF THE INSTRUCTIONS:
: MOV, BIC, BIT, AND BIS. THESE INSTRUCTIONS ARE SIMILAR IN THE
: WAY THEY EFFECT THE C AND V BITS. THEY ALL LEAVE THE V-BIT
: CLEAR AND THE C-BIT UNAFFECTED.
: THE TEST PROCEDURE IS AS FOLLOWS: THE N, Z, AND V BITS
: ARE LOADED WITH THE COMPLEMENT OF THE EXPECTED RESULTS, THE C-BIT
: IS LOADED WITH THE DESIRED RESULT. THE INSTRUCTION IS EXECUTED
: WITH DIFFERENT DATA PATTERNS AND THE RESULTS ARE VERIFIED WITH
: A SERIES OF CONDITIONAL BRANCH INSTRUCTIONS. THE DATA IS CHOSEN
: TO PRODUCT ALL POSSIBLE COMBINATIONS OF THE C AND V BITS.
: *****

: TEST 207 TEST MOV INSTRUCTION
: *****
TS207:
SCC ;CC=0110
+CLN!CLC
MOV #100000,R0 ;CC=1000

4329	013540	101402	
4330	013542	102401	
4331	013544	100401	
4332	013546		
4333	013546	104000	
4334			
4335	013550	000277	
4336	013552	000244	
4337	013554	012700	000000
4338	013560	101002	
4339	013562	102401	
4340	013564	100001	
4341	013566		
4342	013566	104000	
4343			
4344			
4345			
4346	013570		
4347	013570	012700	100001
4348	013574	000277	
4349	013576	000251	
4350	013600	032700	100000
4351	013604	101402	
4352	013606	102401	
4353	013610	100401	
4354	013612		
4355	013612	104000	
4356			
4357	013614	000277	
4358	013616	000244	
4359	013620	032700	077776
4360	013624	101002	
4361	013626	102401	
4362	013630	100001	
4363	013632		
4364	013632	104000	
4365			
4366			
4367			
4368	013634		
4369	013634	012700	177777
4370	013640	000277	
4371	013642	000251	
4372	013644	042700	077777
4373	013650	101402	
4374	013652	102401	
4375	013654	100401	
4376	013656		
4377	013656	104000	
4378	013660	000277	
4379	013662	000244	
4380	013664	042700	100000
4381	013670	101002	
4382	013672	102401	
4383	013674	100001	
4384	013676		

```

      BLOS  MOV1
      BVS   MOV1
      BMI   MOV2
MOV1:    EMT                               ;MOV DID NOT SET CC'S CORRECTLY
MOV2:    SCC                               ;CC=1011
          CLZ
          MOV  #0,R0                       ;CC=0101
          BHI  MOV3                         ;C OR Z = 0?
          BVS  MOV3                         ;V=1?
          BPL  TS210
MOV3:    EMT                               ;MOV DID NOT SET CC'S CORRECTLY
:*****
:TEST 210 TEST BIT INSTRUCTION
:*****
TS210:   MOV  #100001,R0
          SCC                               ;CC=0110
          +CLN!CLC
          BIT  #100000,R0                  ;CC=1000
          BLOS BITST1
          BVS  BITST1
          BMI  BITST2
BITST1:  EMT                               ;BIT DID NOT SET CC'S CORRECTLY
BITST2:  SCC                               ;CC=1011
          CLZ
          BIT  #77776,R0                  ;CC=0101
          BHI  BITST3
          BVS  BITST3
          BPL  TS211
BITST3:  EMT                               ;BIT DID NOT SET CC'S CORRECTLY
:*****
:TEST 211 TEST BIC INSTRUCTION
:*****
TS211:   MOV  #177777,R0
          SCC                               ;CC=0110
          +CLN!CLC
          BIC  #77777,R0                  ;CC=1000
          BLOS BIC1
          BVS  BIC1
          BMI  BIC2
BIC1:    EMT                               ;BIC DID NOT SET CC'S CORRECTLY
BIC2:    SCC                               ;CC=1011
          CLZ
          BIC  #100000,R0                  ;CC=0101
          BHI  BIC3
          BVS  BIC3
          BPL  TS212
BIC3:
  
```

4385 013676 104000
4386
4387
4388
4389 013700
4390 013700 005000
4391 013702 000277
4392 013704 000251
4393 013706 052700 000000
4394 013712 103403
4395 013714 102402
4396 013716 100401
4397 013720 001401
4398 013722
4399 013722 104000
4400 013724 000277
4401 013726 000250
4402 013730 052700 177777
4403 013734 103003
4404 013736 102402
4405 013740 001401
4406 013742 100401
4407 013744
4408 013744 104000
4409
4410
4411
4412
4413
4414
4415
4416
4417
4418
4419
4420
4421
4422
4423
4424 013746
4425 013746 012700 077777
4426 013752 000257
4427 013754 000264
4428 013756 005200
4429 013760 101402
4430 013762 100001
4431 013764 102401
4432 013766
4433 013766 104000
4434 013770 052700 077777
4435 013774 000261
4436 013776 000244
4437 014000 005200
4438 014002 100403
4439 014004 102402
4440 014006 103001

```
EMT ;BIC DID NOT SET CC'S CORRECTLY
:*****
:TEST 212 TEST BIC INSTRUCTION
:*****
TS212:
CLR R0 ;R0=0
SCC ;CC=1010
+CLN!CLC
BIS #0,R0 ;CC=0100 R0=0
BCS BIS1
BVS BIS1
BMI BIS1
BEQ BIS2
BIS1:
EMT ;BIS DID NOT SET CC'S CORRECTLY
BIS2:
SCC ;CC=0111
CLN
BIS #177777,R0 ;CC=1001
BCC BIS3
BVS BIS3
BEQ BIS3
BMI TS213
BIS3:
EMT ;BIS DID NOT SET CC'S CORRECTLY
:*****
:
: THESE NEXT TWO TESTS VERIFY THE FUNCTIONING OF THE INC AND
: DEC INSTRUCTIONS. THESE INSTRUCTIONS BOTH EFFECT THE C AND V
: BITS THE SAME; THE C-BIT IS LEFT UNCHANGED AND THE V-BIT IS DEPENDENT
: UPON THE DATA RESULTS. THE SAME PROCEDURE IS USED. THE CONDITION
: CODE BITS ARE INITIALIZED, THE INSTRUCTION IS EXECUTED AND THE
: RESULTS ARE VERIFIED WITH A SERIES OF CONDITIONAL BRANCH INSTRUCTIONS.
: THIS PROCEDURE IS REPEATED WITH SEVERAL DATA PATTERNS TO PRODUCE
: DIFFERENT COMBINATIONS OF THE C AND V BITS.
:*****
:TEST 213 TEST INC INSTRUCTION
:*****
TS213:
MOV #077777,R0 ;R0=077777
CCC ;CC=0100
SEZ
INC R0 ;CC=1010 R0=10000
BLOS INC1
BPL INC1
BVS INC2
INC1:
EMT ;INC DID NOT SET CC'S CORRECTLY
INC2:
BIS #77777,R0 ;R0=177777
SEC ;CC=1011
CLZ
INC R0 ;CC=0101 R0=0
BMI INC3
BVS INC3
BCC INC3
```

4441 014010 001401
4442 014012
4443 014012 104000
4444
4445 014014 000277
4446 014016 000241
4447 014020 005200
4448 014022 101402
4449 014024 100401
4450 014026 100001
4451 014030
4452 014030 104000
4453
4454
4455
4456
4457 014032
4458 014032 012700 000002
4459 014036 000277
4460 014040 005300
4461 014042 100403
4462 014044 001402
4463 014046 102401
4464 014050 103401
4465 014052
4466 014052 104000
4467 014054 000261
4468 014056 000244
4469 014060 005300
4470 014062 101002
4471 014064 100401
4472 014066 102001
4473 014070
4474 014070 104000
4475 014072 000277
4476 014074 000251
4477 014076 005300
4478 014100 101402
4479 014102 102401
4480 014104 100401
4481 014106
4482 014106 104000
4483 014110 042700 077777
4484 014114 000277
4485 014116 000252
4486 014120 005300
4487 014122 100403
4488 014124 001402
4489 014126 102001
4490 014130 103401
4491 014132
4492 014132 104000
4493
4494
4495
4496

INC3: BEQ INC4
EMT ;INC DID NOT SET CC'S CORRECTLY
INC4: SCC ;CC=1110
CLC
INC R0 ;CC=0000 R0=1
BLOS INCS
BMI INCS
BPL TS214
INC5: EMT ;INC DID NOT SET CC'S CORRECTLY

:TEST 214 TEST DEC INSTRUCTION

TS214:
MOV #2,R0 ;R0=2
SCC ;CC=1111
DEC R0 ;CC=0001 R0=1
BMI DEC1
BEQ DEC1
BVS DEC1
BCS DEC2
DEC1: EMT ;DEC DID NOT SET CC'S CORRECTLY
DEC2: SEC ;CC=1011
CLZ
DEC R0 ;CC=0101 R0=0
BHI DEC3
BMI DEC3
BVC DEC4
DEC3: EMT ;DEC DID NOT SET CC'S CORRECTLY
DEC4: SCC ;CC=0110
+CLN!CLC
DEC R0 ;CC=1000 R0=177777
BLOS DECS
BVS DECS
BMI DEC6
DEC5: EMT ;DEC DID NOT SET CC'S CORRECTLY
DEC6: BIC #77777,R0 ;R0=100000
SCC ;CC=0101
+CLN!CLV
DEC R0 ;CC=1011 R0=77777
BMI DEC7
BEQ DEC7
BVC DEC7
BCS TS215
DEC7: EMT ;DEC DID NOT SET CC'S CORRECTLY

:

4497
4498
4499
4500
4501
4502
4503
4504
4505
4506
4507 014134
4508 014134 000277
4509 014136 000244
4510 014140 005000
4511 014142 100403
4512 014144 102402
4513 014146 103401
4514 014150 001401
4515 014152
4516 014152 104000
4517
4518
4519
4520
4521 014154
4522 014154 000277
4523 014156 000244
4524 014160 005700
4525 014162 100403
4526 014164 102402
4527 014166 103401
4528 014170 001401
4529 014172
4530 014172 104000
4531 014174 005300
4532 014176 000277
4533 014200 000250
4534 014202 005700
4535 014204 101402
4536 014206 102401
4537 014210 100401
4538 014212
4539 014212 104000
4540
4541
4542
4543 014214
4544 014214 012700 170000
4545 014220 000277
4546 014222 000250
4547 014224 000300
4548 014226 101402
4549 014230 102401
4550 014232 100401
4551 014234
4552 014234 104000

: THESE NEXT THREE TESTS VERIFY THE FUNCTIONING OF THE CLR,
:TST, AND SWAB INSTRUCTIONS. THESE THREE INSTRUCTIONS ALL LEAVE
:THE C AND V BITS CLEARED. AGAIN, THE CONDITION CODES ARE PRESET,
:THE INSTRUCTION EXECUTED AND THE RESULTS CHECKED WITH CONDITIONAL
:BRANCH INSTRUCTIONS. THE PROCEDURE IS REPEATED TO PRODUCE OTHER
:COMBINATIONS OF CONDITION CODES.

:TEST 215 TEST CLR INSTRUCTION

TS215:
SCC ;CC=1011
CLZ
CLR R0 ;CC=0100 R0=0
BMI CLR1
BVS CLR1
BCS CLR1
BEQ TS216

CLR1: EMT ;CLR DID NOT SET CC'S CORRECTLY

:TEST 216 TEST TST INSTRUCTION

TS216:
SCC ;CC=1011
CLZ
TST R0 ;CC=0100
BMI TEST1
BVS TEST1
BCS TEST1
BEQ TEST2

TEST1: EMT ;TEST DID NOT SET CC'S CORRECTLY
TEST2: DEC R0 ;MAKE R0 NEGATIVE
SCC ;CC=0111
CLN
TST R0 ;CC=1000
BLOS TEST3
BVS TEST3
BMI TS217

TEST3: EMT ;TEST DID NOT SET CC'S CORRECTLY

:TEST 217 TEST SWAB INSTRUCTION

TS217:
MOV #170000,R0 ;R0=170000
SCC ;CC=0111
CLN
SWAB R0 ;CC=1000 R0=360
BLOS SWB1
BVS SWB1
BMI SWB2

SWB1: EMT ;SWAB DID NOT SET CC'S CORRECTLY

4553 014236 000277
4554 014240 000244
4555 014242 000300
4556 014244 102403
4557 014246 103402
4558 014250 100401
4559 014252 001401
4560 014254
4561 014254 104000
4562
4563
4564
4565
4566
4567
4568
4569
4570
4571
4572
4573
4574
4575
4576 014256
4577 014256 012700 040000
4578 014262 000277
4579 014264 062700 030000
4580 014270 101402
4581 014272 102401
4582 014274 100001
4583 014276
4584 014276 104000
4585 014300 000264
4586
4587 014302 062700 010000
4588 014306 101402
4589 014310 102001
4590 014312 100401
4591 014314
4592 014314 104000
4593 014316 000257
4594 014320 000270
4595 014322 062700 100000
4596 014326 101002
4597 014330 102001
4598 014332 100001
4599 014334
4600 014334 104000
4601 014336 062700 177777
4602 014342 101402
4603 014344 102401
4604 014346 100401
4605 014350
4606 014350 104000
4607 014352 000277
4608 014354 000245

SWB2: SCC ;CC=1011
CLZ
SWAB RO ;CC=0100 RO=170000
BVS SWB3
BCS SWB3
BMI SWB3
BEQ TS220
SWB3: EMT ;

: THESE NEXT TWO TESTS VERIFY THE FUNCTIONING OF THE ADD AND
: ADC INSTRUCTIONS. BOTH OF THESE INSTRUCTIONS HANDLE THE C AND
: V BITS IDENTICALLY. THE PROCEDURE IS TO PRESET THE CONDITION
: CODES, EXECUTE THE INSTRUCTION WITH A PARTICULAR SET OF DATA, AND
: THEN CHECK THE RESULTS BY EXECUTING A SERIES OF CONDITIONAL
: BRANCHES. THIS PROCEDURE IS REPEATED SEVERAL TIMES WITH DIFFERENT
: DATA TO PRODUCE EVERY COMBINATION OF C AND V BITS.
: *****
: TEST 220 TEST ADD INSTRUCTION
: *****

TS220:
MOV #40000,RO ;RO=40000
SCC ;CC=1111
ADD #30000,RO ;CC=0000 RO=70000
BLOS ADD1
BVS ADD1
BPL ADD2
ADD1: EMT ;ADD DID NOT SET CC'S CORRECTLY
ADD2: SEZ ;CC=0100
ADD #10000,RO ;CC=1010 RO=100000
BLOS ADD3
BVC ADD3
BMI ADD4
ADD3: EMT ;ADD DID NOT SET CC'S CORRECTLY
ADD4: CCC ;CC=1000
SEN
ADD #100000,RO ;CC=0111 RO=0
BHI ADD5
BVC ADD5
BPL ADD6
ADD5: EMT ;ADD DID NOT SET CC'S CORRECTLY
ADD6: ADD #177777,RO ;CC=1000 RO=177777
BLOS ADD7
BVS ADD7
BMI ADD8
ADD7: EMT ;ADD DID NOT SET CC'S CORRECTLY
ADD8: SCC ;CC=1010
+CLC!CLZ

4609 014356 062700 000001
4610 014362 102403
4611 014364 103002
4612 014366 100401
4613 014370 001401
4614 014372
4615 014372 104000
4616
4617
4618
4619
4620 014374
4621 014374 012700 077777
4622 014400 000277
4623 014402 000252
4624 014404 005500
4625 014406 101402
4626 014410 102001
4627 014412 100401
4628 014414
4629 014414 104000
4630 014416 052700 077777
4631 014422 000277
4632 014424 000244
4633 014426 005500
4634 014430 101002
4635 014432 102401
4636 014434 100001
4637 014436
4638 014436 104000
4639 014440 000277
4640 014442 000245
4641 014444 005500
4642 014446 102403
4643 014450 103402
4644 014452 100401
4645 014454 001401
4646 014456
4647 014456 104000
4648
4649
4650
4651
4652
4653
4654
4655
4656
4657
4658
4659
4660
4661
4662 014460
4663 014460 012700 000001
4664 014464 000277

ADD #1,R0 ;CC=0101 R=0
BVS ADD9
BCC ADD9
BMI ADD9
BEQ TS221
ADD9:
EMT ;ADD DID NOT SET CC'S CORRECTLY

:TEST 221 TEST ADC INSTRUCTION

TS221:

MOV #077777,R0
SCC ;CC=0101
+CLN!CLV
ADC R0 ;CC=1010
BLOS ADC1
BVC ADC1
BMI ADC2
ADC1:
EMT ;ADC DID NOT SET CC'S CORRECTLY
ADC2:
BIS #77777,R0
SCC ;CC=1011
CLZ
ADC R0 ;CC=0101 R0=0
BHI ADC3
BVS ADC3
BPL ADC4
ADC3:
EMT ;ADC DID NOT SET CC'S CORRECTLY
ADC4:
SCC
+CLZ!CLC ;CC=1010
ADC R0 ;CC=0100
BVS ADC5
BCS ADC5
BMI ADC5
BEQ TS222
ADC5:
EMT ;ADC DID NOT SET CC'S CORRECTLY

: THESE NEXT THREE TESTS VERIFY THE FUNCTIONING OF THE NEG,
: CMP, AND COM INSTRUCTIONS. EACH OF THESE INSTRUCTIONS GENERATE
: THE C AND V BITS IDENTICALLY. THE CONDITION CODES ARE PRESET,
: THE INSTRUCTIONS EXECUTED, AND THE RESULTS CHECKED WITH A SERIES
: OF CONDITIONAL BRANCH INSTRUCTIONS. THIS PROCEDURE IS REPEATED
: SEVERAL TIMES WITH DIFFERENT DATA IN ORDER TO GENERATE DIFFERENT
: COMBINATIONS OF THE C AND V BITS.

:TEST 222 TEST NEG INSTRUCTION

TS222:

MOV #1,R0
SCC ;CC=0110

4665	014466	000251		+CLN!CLC		
4666	014470	005400		NEG	RO	:CC=1001 RO=17777
4667	014472	103003		BCC	NEG1	
4668	014474	102402		BVS	NEG1	
4669	014476	001401		BEQ	NEG1	
4670	014500	100401		BMI	NEG2	
4671	014502			NEG1:		
4672	014502	104000		EMT		:NEG DID NOT SET CC'S CORRECTLY
4673	014504	042700	077777	NEG2:	BIC #77777,RO	
4674	014510	000257		CCC		:CC=0100
4675	014512	000264		SEZ		
4676	014514	005400		NEG	RO	:CC=1011 RO=100000
4677	014516	102003		BVC	NEG3	
4678	014520	103002		BCC	NEG3	
4679	014522	001401		BEQ	NEG3	
4680	014524	100401		BMI	NEG4	
4681	014526			NEG3:		
4682	014526	104000		EMT		:NEG DID NOT SET CC'S CORRECTLY
4683	014530	005000		NEG4:	CLR RO	
4684	014532	000277		SCC		:CC=1011
4685	014534	000244		CLZ		
4686	014536	005400		NEG	RO	:CC=0100 RO=0
4687	014540	102403		BVS	NEG5	
4688	014542	103402		BCS	NEG5	
4689	014544	001001		BNE	NEG5	
4690	014546	100001		BPL	TS223	
4691	014550			NEG5:		
4692	014550	104000		EMT		:NEG DID NOT SET CC'S CORRECTLY

 :TEST 223 TEST CMP INSTRUCTION
 :*****
 TS223:

4697	014552			MOV	#5,RO	
4698	014552	012700	000005	CCC		:CC=1010
4699	014556	000257		+SEN!SEC		
4700	014560	000271		CMP	#5,RO	:CC=0101
4701	014562	022700	000005	BHI	CMP1	
4702	014566	101002		BVS	CMP1	
4703	014570	102401		BPL	CMP2	
4704	014572	100001		CMP1:		
4705	014574			EMT		:CMP DID NOT SET CC'S CORRECTLY
4706	014574	104000		CMP2:	MOV #100000,RO	
4707	014576	012700	100000	SCC		:CC=1101
4708	014602	000277		CLV		
4709	014604	000242		CMP	RO,#77777	:CC=0010
4710	014606	020027	077777	BLOS	CMP3	
4711	014612	101402		BVC	CMP3	
4712	014614	102001		BPL	CMP4	
4713	014616	100001		CMP3:		
4714	014620			EMT		:CMP DID NOT SET CC'S CORRECTLY
4715	014620	104000		CMP4:	BIS #40000,RO	:RO=140000
4716	014622	052700	040000	CCC		:CC=0100
4717	014626	000257		SEZ		
4718	014630	000264		CMP	#40000,RO	:CC=1011
4719	014632	022700	040000	BVC	CMP5	
4720	014636	102003				

4721 014640 103002
 4722 014642 001401
 4723 014644 100401
 4724 014646
 4725 014646 104000
 4726 014650 042700 040000
 4727 014654 000277
 4728 014656 022700 177777
 4729 014662 101402
 4730 014664 102401
 4731 014666 100001
 4732 014670
 4733 014670 104000
 4734
 4735
 4736
 4737
 4738 014672
 4739 014672 012700 177777
 4740 014676 000257
 4741 014700 000265
 4742 014702 005100
 4743 014704 101062
 4744 014706 102401
 4745 014710 100001
 4746 014712
 4747 014712 104000
 4748
 4749
 4750
 4751
 4752
 4753
 4754
 4755
 4756
 4757
 4758
 4759
 4760
 4761
 4762
 4763 014714
 4764 014714 012700 125252
 4765 014720 000257
 4766 014722 000271
 4767 014724 162700 125252
 4768 014730 101002
 4769 014732 102401
 4770 014734 100001
 4771 014736
 4772 014736 104000
 4773 014740 052700 100000
 4774 014744 000277
 4775 014746 000242
 4776 014750 162700 077777

BCC CMP5
 BEQ CMP5
 BMI CMP6
 CMP5: EMT ;CMP DID NOT SET CC'S CORRECTLY
 CMP6: BIC #40000,R0 ;CC=1111
 SCC ;CC=0000
 CMP #-1,R0
 BLOS CMP7
 BVS CMP7
 BPL TS224
 CMP7: EMT ;CMP DID NOT SET CC'S CORRECTLY

 ;TEST 224 TEST COM INSTRUCTION
 ;*****
 TS224:

MOV #-1,R0
 CCC ;CC=1010
 +SEC!SEZ
 COM RO ;CC=0101
 BHI COM1
 BVS COM1
 BPL TS225
 COM1: EMT ;COM DID NOT SET CC'S CORRECTLY

 ;
 ; THESE NEXT TWO TESTS VERIFY THE FUNCTIONING OF THE SUB
 ; AND SBC INSTRUCTIONS. BOTH OF THESE INSTRUCTIONS HANDLE THE
 ; C AND V BITS IDENTICALLY. THE PROCEDURE IS TO PRESET THE CONDITION
 ; CODES, EXECUTE THE INSTRUCTION WITH A PARTICULAR SET OF DATA, AND
 ; THEN CHECK THE RESULTS BY EXECUTING A SERIES OF CONDITIONAL
 ; BRANCHES. THIS PROCEDURE IS REPEATED SEVERAL TIMES WITH DIFFERENT
 ; DATA PATTERNS TO PROVIDE EVERY COMBINATION OF THE C AND V BITS.
 ;

 ;TEST 225 TEST SUB INSTRUCTION
 ;*****
 TS225:

MOV #125252,R0
 CCC ;CC=1010
 +SEN!SEC
 SUB #125252,R0 ;CC=0101 R0=0
 BHI SUB1
 BVS SUB1
 BPL SUB2
 SUB1: EMT ;SUB DID NOT SET CC'S CORRECTLY
 SUB2: BIS #100000,R0
 SCC ;CC=1101
 CLV
 SUB #77777,R0 ;CC=0010 R0=1

4777 014754 101402
4778 014756 102001
4779 014760 100001
4780 014762
4781 014762 104000
4782 014764 005100
4783 014766 000277
4784
4785 014770 162700 100000
4786 014774 101402
4787 014776 102401
4788 015000 100001
4789 015002
4790 015002 104000
4791 015004 000257
4792 015006 000264
4793 015010 162700 140000
4794 015014 102003
4795 015016 103002
4796 015020 001401
4797 015022 100401
4798 015024
4799 015024 104000

BLOS SUB3
BVC SUB3
BPL SUB4
SUB3:
EMT ;
SUB4: COM RO ;RO=17777
SCC ;CC=11111
SUB #100000,RO ;CC=0000 RO=77777
BLOS SUB5
BVS SUB5
BPL SUB6
SUB5:
EMT ;SUB DID NOT SET CC'S CORRECTLY
SUB6: CCC ;CC=0100
SEZ
SUB #140000,RO ;CC=1011
BVC SUB7
BCC SUB7
BEQ SUB7
BMI TS226
SUB7:
EMT ;

4800
4801
4802
4803

:TEST 226 TEST SBC INSTRUCTION

4804 015026
4805 015026 012700 000001
4806 015032 000277
4807 015034 000244
4808 015036 005600
4809 015040 103403
4810 015042 102402
4811 015044 100401
4812 015046 001401
4813 015050
4814 015050 104000
4815 015052 000277
4816 015054 000245
4817 015056 005600
4818 015060 103403
4819 015062 102402
4820 015064 100401
4821 015066 001401
4822 015070
4823 015070 104000
4824 015072 000277
4825 015074 000250
4826 015076 005600
4827 015100 103003
4828 015102 102402
4829 015104 001401
4830 015106 100401
4831 015110
4832 015110 104000

TS226:
MOV #1,RO
SCC ;CC=1011
CLZ
SBC RO ;CC=0100 R=0
BCS SBC1
BVS SBC1
BMI SBC1
BEQ SBC2
SBC1:
EMT ;SBC DID NOT SET CC'S CORRECTLY
SBC2: SCC ;CC=1010
+CLZ!CLC
SBC RO ;CC=0100 R=0
BCS SBC3
BVS SBC3
BMI SBC3
BEQ SBC4
SBC3:
EMT ;SBC DID NOT SET CC'S CORRECTLY
SBC4: SCC ;CC=0111
CLN
SBC RO ;CC=1001 RO=17777
BCC SBC5
BVS SBC5
BEQ SBC5
BMI SBC6
SBC5:
EMT ;SBC DID NOT SET CC'S CORRECTLY

4833 015112 042700 077777
4834 015116 000277
4835 015120 000242
4836 015122 005600
4837 015124 101402
4838 015126 102001
4839 015130 100001
4840 015132
4841 015132 104000
4842
4843
4844
4845
4846
4847
4848
4849
4850
4851
4852
4853
4854
4855
4856 015134
4857 015134 012700 144000
4858 015140 000257
4859 015142 000266
4860 015144 006100
4861 015146 103003
4862 015150 102402
4863 015152 001401
4864 015154 100401
4865 015156
4866 015156 104000
4867 015160 000277
4868 015162 000243
4869 015164 006100
4870 015166 103003
4871 015170 102002
4872 015172 001401
4873 015174 100001
4874 015176
4875 015176 104000
4876 015200 000277
4877 015202 000250
4878 015204 006100
4879 015206 101402
4880 015210 102401
4881 015212 100001
4882 015214
4883 015214 104000
4884 015216 000257
4885 015220 000265
4886 015222 006100
4887 015224 101405
4888 015226 102004

SBC6: BIC #77777,R0 ;RO=100000
SCC ;CC=1101
CLV
SBC R0 ;CC=0010
BLOS SBC7
BVC SBC7
BPL TS227

SBC7: EMT ;SBC DID NOT SET CC'S CORRECTLY

: THESE NEXT FOUR TESTS VERIFY THE FUNCTIONING OF THE ROL,
:ROR, ASL AND ASR INSTRUCTIONS. SPECIAL DATA PATTERNS ARE LOADED
:AND ROTATED SEVERAL TIMES FOR EACH TEST. THE CONDITION CODES
:ARE PRESET BEFORE EACH ROTATION AND THE CONDITION CODES ARE
:CHECKED AFTER EACH ROTATION. THE FINAL CHECK IN EACH TEST IS
:TO VERIFY THE COMMULATIVE DATA RESULT. THE DATA PATTERNS HAVE
:BEEN SELECTED TO PRODUCE ALL COMBINATIONS OF THE C AND V BITS.

:TEST 227 TEST ROL INSTRUCTION

TS227:

MOV #144000,R0 ;RO=144000
CCC ;CC=0110
+SEZ!SEV
ROL R0 ;CC=1001 RO=110000
BCC ROL1
BVS ROL1
BEQ ROL1
BMI ROL2

ROL1: EMT ;
ROL2: SCC ;CC=1100
+CLV!CLC
ROL R0 ;CC=0011 RO=020000
BCC ROL3
BVC ROL3
BEQ ROL3
BPL ROL4

ROL3: EMT ;ROL DID NOT SET CC'S CORRECTLY
ROL4: SCC ;CC=0111
CLN
ROL R0 ;CC=0000 RO=040001
BLOS ROL5
BVS ROL5
BPL ROL6

ROL5: EMT ;ROL DID NOT SET CC'S CORRECTLY
ROL6: CCC ;CC=0101
+SEZ!SEC
ROL R0 ;CC=1010 RO=100003
BLOS ROL7
BVC ROL7

4889 015230 100003
4890 015232 022700 100003
4891 015236 001401
4892 015240
4893 015240 104000
4894
4895
4896
4897 015242
4898 015242 012700 000023
4899 015246 000277
4900 015250 000250
4901 015252 006000
4902 015254 102403
4903 015256 103002
4904 015260 001401
4905 015262 100401
4906 015264
4907 015264 104000
4908 015266 000257
4909 015270 000274
4910 015272 006000
4911 015274 102003
4912 015276 103002
4913 015300 001401
4914 015302 100001
4915 015304
4916 015304 104000
4917 015306 000277
4918 015310 000241
4919 015312 006000
4920 015314 101403
4921 015316 102402
4922 015320 001401
4923 015322 100001
4924 015324
4925 015324 104000
4926 015326 000257
4927 015330 000265
4928 015332 006000
4929 015334 101402
4930 015336 102001
4931 015340 100401
4932 015342
4933 015342 104000
4934
4935
4936
4937 015344
4938 015344 012700 144000
4939 015350 000257
4940 015352 000271
4941 015354 006300
4942 015356 103003
4943 015360 102402
4944 015362 001401

BPL ROL7
CMP #100003,R0
BEQ TS230
ROL7:
EMT ;ROL MALFUNCTIONED
:*****
:TEST 230 TEST ROR INSTRUCTION
:*****
TS230:
MOV #23,R0 ;R0=23
SCC ;CC=0111
RLN
ROR R0 ;CC=1001 R0=100011
BVS ROR1
BCC ROR1
BEQ ROR1
BMI ROR2
ROR1:
EMT ;ROR DID NOT SET CC'S CORRECTLY
ROR2:
CCC ;CC=1100
+SEN!SEZ
ROR R0 ;CC=0011 R0=040004
BVC ROR3
BCC ROR3
BEQ ROR3
BPL ROR4
ROR3:
EMT ;ROR DID NOT SET CC'S CORRECTLY
ROR4:
SCC ;CC=1110
CLC
ROR R0 ;CC=0000 R0=020002
BLOS ROR5
BVS ROR5
BEQ ROR5
BPL ROR6
ROR5:
EMT ;ROR DID NOT SET CC'S CORRECTLY
ROR6:
CCC ;CC=0101
+SEC!SEZ
ROR R0 ;CC=1010 R0=110001
BLOS ROR7
BVC ROR7
BMI TS231
ROR7:
EMT ;ROR DID NOT PRODUCE CORRECT RESULTS
:*****
:TEST 231 TEST ASL INSTRUCTION
:*****
TS231:
MOV #144000,R0 ;R0=14000
CCC ;CC=0110
+SEN!SEC
ASL R0 ;CC=1001 R0=110000
BCC ASL1
BVS ASL1
BEQ ASL1

4945 015364 100401
4946 015366
4947 015366 104000
4948 015370 000277
4949 015372 000243
4950 015374 006300
4951 015376 103003
4952 015400 102002
4953 015402 001401
4954 015404 100001
4955 015406
4956 015406 104000
4957 015410 000277
4958 015412 000250
4959 015414 006300
4960 015416 101402
4961 015420 102401
4962 015422 100001
4963 015424
4964 015424 104000
4965 015426 000257
4966 015430 000265
4967 015432 006300
4968 015434 103406
4969 015436 001405
4970 015440 102004
4971 015442 100003
4972 015444 022700 100000
4973 015450 001401
4974 015452
4975 015452 104000
4976
4977
4978
4979 015454
4980 015454 012700 100023
4981 015460 000277
4982 015462 000250
4983 015464 006200
4984 015466 102403
4985 015470 103002
4986 015472 001401
4987 015474 100401
4988 015476
4989 015476 104000
4990 015500 042700 100000
4991 015504 000277
4992 015506 000243
4993 015510 006200
4994 015512 102003
4995 015514 103002
4996 015516 001401
4997 015520 100001
4998 015522
4999 015522 104000
5000 015524 000277

ASL1: BMI ASL2
ASL2: EMT :
SCC :CC=1100
+CLV!CLC
ASL RO :CC=0011 RO=020000
BCC ASL3
BVC ASL3
BEQ ASL3
BPL ASL4
ASL3: EMT ;ASL DID NOT SET CC'S CORRECTLY
ASL4: SCC ;CC=0111
CLN
ASL RO :CC=0000 RO=040000
BLOS ASL5
BVS ASL5
BPL ASL6
ASL5: EMT ;ASL DID NOT SET CC'S CORRECTLY
ASL6: CCC ;CC=0101
+SEZ!SEC
ASL RO :CC=1010 RO=100000
BCS ASL7
BEQ ASL7
BVC ASL7
BPL ASL7
CMP #100000,RO
BEQ TS232
ASL7: EMT ;ASL MALFUNCTIONED
:*****
:TEST 232 TEST ASR INSTRUCTION
:*****
TS232: MOV #100023,RO ;RO=100023
SCC ;CC=0110
CLN
ASR RO :CC=1001 RP=140011
BVS ASR1
BCC ASR1
BEQ ASR1
BMI ASR2
ASR1: EMT ;ASR DID NOT SET CC'S CORRECTLY
ASR2: BIC #100000,RO ;RO=40011
SCC ;CC=1100
+CLV!CLC
ASR RO :CC=0011 RO=020004
BVC ASR3
BCC ASR3
BEQ ASR3
BPL ASR4
ASR3: EMT ;ASR DID NOT SET CC'S CORRECTLY
ASR4: SCC ;CC=1111

5001					
5002	015526	006200		ASR	R0 ;CC=0000 R0=010002
5003	015530	101403		BLOS	ASR5
5004	015532	102402		BVS	ASR5
5005	015534	001401		BEQ	ASR5
5006	015536	100001		BPL	ASR6
5007	015540			ASR5:	
5008	015540	104000		EMT	;ASR DID NOT SET CC'S CORRECTLY
5009	015542	052700	100000	ASR6:	BIS #100000,R0 ;R0=110002
5010	015546	000257		CCC	;CC=0101
5011	015550	000265		+SEZ!SEC	
5012	015552	006200		ASR	R0 ;C=1010 R0=144001
5013	015554	101406		BLOS	ASR7
5014	015556	102005		BVC	ASR7
5015	015560	100004		BPL	ASR7
5016	015562	001403		BEQ	ASR7
5017	015564	022700	144001	ASR7:	CMP #144001,R0 ;CHECK RESULT OF ASR'S
5018	015570	001401		BEQ	TS233
5019	015572			ASR7:	
5020	015572	104000		EMT	;ASR DID NOT FUNCTION CORRECTLY
5021					
5022					
5023					
5024					
5025					
5026	015574				
5027	015574	112701	000004	TS233:	
5028	015600	000257		MOV	#4,R1 ;LOAD REGISTER
5029	015602	106001		CCC	;CLEAR ALL FLAGS
5030	015604	106001		RORB	R1 ;SHIFT BYTE RIGHT
5031	015606	122701	000001	RORB	R1 ;SHIFT BYTE RIGHT
5032	015612	001401		CMPB	#1,R1 ;CHECK RESULT
5033	015614	104000		BEQ	RORB1
5034	015616	106001		EMT	;RORB DID NOT FUNCTION CORRECTLY
5035	015620	100403		RORB1:	RORB R1 ;SHIFT BYTE RIGHT
5036	015622	001002		BMI	RORB2 ;CC=7?
5037	015624	102001		BNE	RORB2
5038	015626	103401		BVC	RORB2
5039	015630			BCS	RORB3
5040	015630	104000		RORB2:	
5041	015632	106001		EMT	;RORB DID NOT SET CC'S CORRECTLY
5042	015634	100002		RORB3:	RORB R1 ;SHIFT BYTE RIGHT
5043	015636	101401		BPL	RORB4 ;CC=12
5044	015640	102401		BLOS	RORB4
5045	015642			BVS	RORB5
5046	015642	104000		RORB4:	
5047	015644	122701	000200	EMT	;RORB DID NOT SET CC CORRECTLY
5048	015650	001401		RORB5:	CMPB #200,R1 ;CHECK RESULT
5049	015652	104000		BEQ	RORB7
5050	015654			EMT	;RORB DID NOT FUNCTION CORRECTLY
5051				RORB7:	
5052	015654	005000		:ROTATE	ODD BYTE
5053	015656	012710	025125	CLR	R0 ;MAKE R0 ZERO
5054	015662	005200		MOV	#025125,(R0) ;PUT STARTING VALUE IN LOC. 0
5055	015664	000257		INC	R0 ;MAKE R0 POINT TO ODD BYTE
5056	015666	000261		CCC	;CLEAR ALL CC
				SEC	;SEC CARRY BIT

```

5057 015670 106010          RORB (R0)          :SHIFT BYTE RIGHT
5058 015672 100002          BPL  RORB10        :CC=12?
5059 015674 101401          BLOS RORB10
5060 015676 102401          BVS  RORB11
5061 015700          RORB10:
5062 015700 104000          EMT
5063 015702 022737 112525 000000 RORB11: CMP #112525,@#0 :RORB DID NOT SET CC'S CORRECTLY
5064 015710 001401          BEQ  RORB12        :CHECK RESULT
5065 015712 104000          EMT
5066 015714 106010          RORB12: RORB (R0)          :RORB DID NOT FUNCTION CORRECTLY
5067 015716 100403          BMI  RORB13        :SHIFT BYTE RIGHT
5068 015720 001402          BEQ  RORB13        :CC=3?
5069 015722 102001          BVC  RORB13
5070 015724 103401          BCS  RORB14
5071 015726          RORB13:
5072 015726 104000          EMT
5073 015730 022737 045125 000000 RORB14: CMP #045125,@#0 :RORB DID NOT SET CC CORRECTLY
5074 015736 001401          BEQ  TS234         :CHECK RESULT
5075 015740 104000          EMT
5076
5077
5078
5079
5080

```

```

:*****
:TEST 234 TEST ASLB INSTRUCTION
:*****
TS234:

```

```

5081 015742          MOVB #40,R1        :LOAD REGISTER
5082 015742 112701 000040 CCC                :CLEAR ALL CONDITION CODES
5083 015746 000257          ASLB R1            :SHIFT BYTE LEFT
5084 015750 106301          ASLB R1            :SHIFT BYTE LEFT
5085 015752 106301          BPL  ASLB2         :CHECK CC=12
5086 015754 100002          BLOS ASLB2
5087 015756 101401          BVS  ASLB3
5088 015760 102401          ASLB2:
5089 015762          ASLB2:
5090 015762 104000          EMT
5091 015764 022701 000200 ASLB3: CMP #200,R1 :ASLB DID NOT SET CONDITION CODE CORRECTLY
5092 015770 001401          BEQ  ASLB1        :CHECK RESULT
5093 015772 104000          EMT
5094 015774 106301          ASLB1: ASLB R1            :ASLB DID NOT FUNCTION CORRECTLY
5095 015776 100403          BMI  ASLB4        :SHIFT BYTE LEFT
5096 016000 001002          BNE  ASLB4        :CHECK CC=7?
5097 016002 102001          BVC  ASLB4
5098 016004 103401          BCS  TS235
5099 016006          ASLB4:
5100 016006 104000          EMT
5101
5102
5103
5104
5105

```

```

:*****
:TEST 235 TEST ASRB INSTRUCTION
:*****
TS235:

```

```

5106 016010          MOVB #4,R1        :SET UP STARTING DATA
5107 016010 112701 000004 CCC                :CLEAR ALL CONDITION CODES
5108 016014 000257          ASRB R1            :SHIFT BYTE RIGHT
5109 016016 106201          ASRB R1            :SHIFT BYTE RIGHT
5110 016020 106201          CMPB #1,R1        :CHECK DATA
5111 016022 122701 000001 BEQ  ASRB1
5112 016026 001401

```

5113 016030 104000
 5114 016032 106201
 5115 016034 100403
 5116 016036 001002
 5117 016040 102001
 5118 016042 103401
 5119 016044
 5120 016044 104000
 5121 016046 106201
 5122 016050 103401
 5123 016052 001401
 5124 016054
 5125 016054 104000
 5126 016056 112701 000202
 5127 016062 106201
 5128 016064 106201
 5129 016066 100003
 5130 016070 001402
 5131 016072 102401
 5132 016074 103401
 5133 016076
 5134 016076 104000
 5135 016100 122701 000340
 5136 016104 001401
 5137 016106 104000

ASRB1: EMT ;ASRB DID NOT SHIFT DATA CORRECTLY
 ASRB ASRB R1 ;SHIFT BYTE RIGHT
 BMI ASRB2 ;CHECK CONDITION CODE = 7?
 BNE ASRB2
 BVC ASRB2
 BCS ASRB3
 ASRB2: EMT ;ASRB DID NOT SET CC'S CORRECTLY
 ASRB3: ASRB R1 ;SHIFT BYTE RIGHT
 BCS ASRB4 ;CHECK CC=4
 BEQ ASRB5
 ASRB4: EMT ;ASRB DID NOT SET CC'S CORRECTLY
 ASRB5: MOVB #202,R1 ;PUT STARTING DATA IN REGISTER
 ASRB R1 ;SHIFT BYTE RIGHT
 ASRB R1 ;SHIFT BYTE RIGHT
 BPL ASRB6 ;CHECK CC'S =11?
 BEQ ASRB6
 BVS ASRB6
 BCS ASRB7
 ASRB6: EMT ;ASRB DID NOT SET CC'S CORRECTLY
 ASRB7: CMPB #340,R1 ;CHECK RESULT
 BEQ TS236
 EMT ;ASRB DID NOT SHIFT DATA CORRECTLY

```

:*****
:
: THIS TEST VERIFIES THE SXT INSTRUCTION. CONDITION CODES
: ARE PRESET IN EACH OF THE TWO POSSIBLE CASES. WITH THE N-BIT SET,
: THE TEST CHECKS FOR ALL ONES IN THE DESTINATION. WITH THE N-BIT
: CLEAR, THE DESTINATION SHOULD CONTAIN ALL ZEROES. THE DATA
: IS VERIFIED BY CONDITIONAL BRANCHES.
:
:*****
:TEST 236 TEST THE SXT INSTRUCTION
:*****

```

5150 016110
 5151 016110 005000
 5152 016112 000277
 5153 016114 000244
 5154 016116 006700
 5155 016120 100006
 5156 016122 001405
 5157 016124 102404
 5158 016126 103003
 5159 016130 022700 177777
 5160 016134 001401
 5161 016136
 5162 016136 104000
 5163 016140 005000
 5164 016142 005010
 5165 016144 005110
 5166 016146 000257
 5167 016150 000266
 5168 016152 006710

TS236:
 CLR R0 ;SET CC=1011
 SCC
 CLZ
 SXT R0 ;TRY SXT
 BPL SXT0 ;TEST CC=1001
 BEQ SXT0
 BVS SXT0
 BCC SXT0
 CMP #-1,R0 ;CHECK DATA RESULT
 BEQ SXT1
 SXT0: EMT ;RESULTS OF SXT INCORRECT
 SXT1: CLR R0 ;R0=0
 CLR (R0) ;LOC. 0=0
 COM (R0) ;LOC. 0=177777
 CCC ;SET CC=0110
 +SEZ!SEV
 SXT (R0)

CJKDJBO 11/23-B CPU CLUSTER DIAG.
CJKDJB.P11 26-MAY-82 11:14

DNMAC X24.07-563 26-MAY-82 11:18 PAGE 98
T236 TEST THE SXT INSTRUCTION

SEQ 0097

5169 016154 001005
5170 016156 103404
5171 016160 102403
5172 016162 100402
5173 016164 005710
5174 016166 001401
5175 016170
5176 016170 104000
5177
5178

BNE SXT2 ;TEST CC=0100
BCS SXT2
BVS SXT2
BMI SXT2
TST (R0)
BEQ TS237

SXT2: EMT ;RESULTS OF SXT INCORRECT
:*****
;

5179
5180
5181
5182
5183
5184
5185
5186
5187
5188 016172
5189 016172 012700 007463
5190 016176 012701 031525
5191 016202 000277
5192 016204 000241
5193 016206 074100
5194 016210 101406
5195 016212 102405
5196 016214 001404
5197 016216 100403
5198 016220 022700 036146
5199 016224 001401
5200 016226
5201 016226 104000
5202 016230 010104
5203 016232 000261
5204 016234 000241
5205 016236 074400
5206 016240 101406
5207 016242 102405
5208 016244 001404
5209 016246 100403
5210 016250 022700 007463
5211 016254 001401
5212 016256
5213 016256 104000
5214
5215
5216
5217
5218
5219
5220
5221
5222
5223
5224 016260
5225 016260 012700 000525
5226 016264 010004
5227 016266 000277
5228 016270 101002
5229 016272 100001
5230 016274 102401
5231 016276
5232 016276 104000
5233 016300 005304
5234 016302 000277

: THIS TEST VERIFIES THE XOR INSTRUCTION. UNIQUE PATTERNS
: OF ONES AND ZEROES ARE MOVED TO DATA REGISTERS R0 AND R1.
: AFTER THE FIRST XOR INSTRUCTION R0=36146. AN XOR IS THEN
: EXECUTED WITH THIS NEW VALUE AND THE CONTENTS OF R1 TO
: REPRODUCE THE ORIGINAL VALUE IF R0=31525.

: TEST 237 TEST THE XOR INSTRUCTION

TS237:
MOV #7463,R0 ;SET UP R0
MOV #31525,R1 ;SET UP R1
SCC ;SET CC=1110
CLC
XOR R1,R0 ;TRY XOR
BLOS XOR1 ;CC=0000?
BVS XOR1
BEQ XOR1
BMI XOR1
CMP #36146,R0 ;DATA RESULT CORRECT?
BEQ XOR2
XOR1: EMT ;
XOR2: MOV R1,R4 ;
SEC ;CC=1110
CLC
XOR R4,R0 ;TRY XOR MODE 0.0
BLOS XOR3 ;CC=0000?
BVS XOR3
BEQ XOR3
BMI XOR3
CMP #7463,R0
BEQ TS240
XOR3: EMT ;RESULT OF XOR INCORRECT

: THIS TEST VERIFIES THE SOB INSTRUCTION. R4 IS USED AS A
: COUNTER WHILE R0 IS THE ADDRESS REGISTER. CONDITIONAL
: BRANCHES ARE USED TO VERIFY PROPER TRANSFER OF CONTROL
: WHILE R4 IS CHECKED TO INSURE PROPER DECREMENTING OF R0.

: TEST 240 TEST SOB INSTRUCTION

TS240:
MOV #525,R0
MOV R0,R4
SCC ;SET CC=1111
SOB1: BMI SOB2 ;CC=1111?
BPL SOB2
BVS SOB3
SOB2: EMT ;
SOB3: DEC R4 ;COUNT ITERATIONS
SCC ;CC=1111

5235 016304 077007
5236 016306 101004
5237 016310 100003
5238 016312 102002
5239 016314 005704
5240 016316 001401
5241 016320
5242 016320 104000
5243
5244
5245
5246
5247
5248
5249
5250
5251
5252
5253 016322
5254 016322 012706 001000
5255 016326 012746 125252
5256 016332 162706 000074
5257 016336 012705 016354
5258 016342 012746 006436
5259 016346 000277
5260 016350 000116
5261 016352 104000
5262 016354 101010
5263 016356 100007
5264 016360 102006
5265 016362 020527 125252
5266 016366 001003
5267 016370 022706 001000
5268 016374 001401
5269 016376
5270 016376 104000
5271 016400 012746 052525
5272 016404 012746 006400
5273 016410 010605
5274 016412 004737 016422
5275 016416 000137 016426
5276 016422 000205
5277 016424 104000
5278 016426 022706 001000
5279 016432 001003
5280 016434 022705 052525
5281 016440 001401
5282 016442
5283 016442 104000
5284 177776
5285
5286
5287
5288
5289
5290

SOB R0,SOB1 ;DO SOB W/ R0
BHI SOB4 ;CHECK CC=1111
BPL SOB4
BVC SOB4
TST R4 ;ITERATION COUNT OK?
BEQ TS241
SOB4: EMT ;INCORRECT # OF BRANCHES OR CC'S CHANGED
:*****
: THIS TEST VERIFIES THE MARK INSTRUCTION. THE EFFECTS
: OF THE MARK INSTRUCTION ARE SIMULATED BY THE PROGRAM INSTRUCTIONS.
: THE CONTENTS OF R5 AND THE STACK POINTER ARE CHECKED AFTER EACH
: OF THE TWO ROUTINES IN THE TEST.
:*****
:TEST 241 TEST MARK INSTRUCTION
:*****
TS241:
MOV #STBOT,SP
MOV #125252,-(SP) ;PUT R5 VALUE ON STACK
SUB #74,SP ;EFFECTIVELY PUT 36 ARGUMENTS ON STACK
MOV #MRK1,R5 ;SET NEW PC IN R5
MOV #6436,-(SP) ;PUT MARK 36 INST. ON STACK
SCC ;SET CC=1111
JMP (SP) ;XFER CONTRL TO MARK 36 INST. ON STACK
EMT ;MARK INST. SHOULD HAVE JUMPED TO MRK1
MRK1: BHI MRK2 ;TEST CC UNAFFECTED
BPL MRK2 ;IE. CC=1111
BVC MRK2
CMP R5,#125252 ;CHECK R5 RESTORED FROM STACK
BNE MRK2
CMP #STBOT,R6 ;CHECK STACK POINTER READJUSTED CORRECTLY.
BEQ MRK3
MRK2: EMT ;RESULTS OF MARK INCORRECT
MRK3: MOV #52525,-(SP)
MOV #6400,-(SP) ;PUT MARK 0 INST. ON STACK
MOV SP,R5 ;SET ADDR. OF MARK INST. IN R5
JSR PC,@MRK4 ;DO JSR
JMP @MRK5 ;
MRK4: RTS R5 ;DO RTS WITH R5 TO MARK INST ON STACK
EMT ;RTS,MARK SEQUENCE FAILED
MRK5: CMP #STBOT,R6 ;STACK ADJUSTED CORRECTLY
BNE MRK6 ;IF NOT: BR
CMP #52525,R5 ;CHECK IF R5 RESTORED FROM STACK
BEQ TS242
MRK6: EMT ;RESULTS OF MARK INCORRECT
PS=177776
:*****
: THESE NEXT SEVEN TESTS VERIFY THE MTPS INSTRUCTION IN ALL
: MODES. THE PSW IS DEFINED BY AN EQUATE STATEMENT BEFORE THE
: FIRST MTPS TEST. IN EACH TEST A PATTERN OF ONES AND
: ZEROES IS SET IN A DATA REGISTER AND MOVED TO THE PSW.
:

5291
5292
5293
5294
5295
5296
5297 016444
5298 016444 012700 000377
5299 016450 000257
5300 016452 106400
5301 016454 022767 000357 161314
5302 016462 001401
5303 016464 104000
5304 016466 005000
5305 016470 005010
5306 016472 000277
5307 016474 106410
5308 016476 100403
5309 016500 102402
5310 016502 103401
5311 016504 001001
5312 016506
5313 016506 104000
5314
5315
5316
5317
5318 016510
5319 016510 005000
5320 016512 012710 177777
5321 016516 005037 177776
5322 016522 106420
5323 016524 022737 000357 177776
5324 016532 001401
5325 016534 104000
5326 016536 022700 000001
5327 016542 001401
5328 016544 104000
5329
5330
5331
5332
5333 016546
5334 016546 012700 000402
5335 016552 005010
5336 016554 012737 052652 000000
5337 016562 005037 177776
5338 016566 106430
5339 016570 022737 000252 177776
5340 016576 001401
5341 016600 104000
5342 016602 022700 000404
5343 016606 001401
5344 016610 104000
5345
5346

```
THE DATA IN THE PSW, AND THE DATA REGISTER ADDRESS,  
ARE CHECKED TO VERIFY PROPER EXECUTION OF THE INSTRUCTION.  
*****  
:TEST 242 TEST MTPS INSTRUCTION  
*****  
TS242:  
MOV #377,R0  
CCC  
MTPS R0  
CMP #357,PS  
BEQ MTPS1  
EMT ;MTPS FAILED  
MTPS1: CLR R0  
CLR (R0)  
SCC ;CC=1111  
MTPS (R0) ;TRY MTPS MODE 1  
BMI MTPS1A ;CHECK PS  
BVS MTPS1A  
BCS MTPS1A  
BNE TS243  
MTPS1A: EMT ;MTPS FAILED  
*****  
:TEST 243 TEST MTPS MODE 2  
*****  
TS243:  
CLR R0 ;R0=0  
MOV #-1,(R0) ;LOC. 0=-1  
CLR @MPS ;PS=0  
MTPS (R0)+ ;TRY MTPS W/MODE 2  
CMP #357,@MPS ;CHECK DATA  
BEQ MTPS2  
EMT ;DEST. DATA INCORRECT  
MTPS2: CMP #1,R0 ;CHECK DEST. REGISTER.  
BEQ TS244  
EMT ;DEST REGISTER NOT INCREMENTED BY 1  
*****  
:TEST 244 TEST MTPS MODE 3  
*****  
TS244:  
MOV #402,R0 ;R0=402  
CLR (R0) ;LOC. 402=0  
MOV #52652,R0 ;LOC. 0=52652  
CLR @MPS ;PS=0  
MTPS @R0+ ;TRY MTPS W/MODE 3  
CMP #252,@MPS ;CHECK DEST. DATA  
BEQ MTPS3  
EMT ;DEST. DATA INCORRECT  
MTPS3: CMP #404,R0 ;CHECK MODE 3 REGISTER.  
BEQ TS245  
EMT ;MODE 3 REGISTER INCORRECT  
*****
```

5347
5348
5349 016612
5350 016612 012700 000001
5351 016616 012737 125125 000000
5352 016624 005037 177776
5353 016630 106440
5354 016632 022737 000105 177776
5355 016640 001401
5356 016642 104000
5357 016644 005700
5358 016646 001401
5359 016650 104000

```
:TEST 245 TEST MTPS MODE 4
:*****
TS245:
MOV #1,R0 ;R0=1
MOV #125125,@#0 ;LOC. 0 = 125125
CLR @#PS ;PS=0
MTPS -(R0) ;TRY MTPS W/MODE 4
CMP #105,@#PS ;CHECK DEST. DATA
BEQ MTPS4
MTPS4: TST R0 ;DEST. DATA INCORRECT
BEQ TS246 ;CHECK MODE 4 REGISTER
EMT ;MODE 4 REGISTER NOT DECREMENTED BY 1
```

5360
5361
5362
5363
5364 016652
5365 016652 012700 000404
5366 016656 012737 177400 000000
5367 016664 000277
5368 016666 106450
5369 016670 005737 177776
5370 016674 001401
5371 016676 104000
5372 016700 022700 000402
5373 016704 001401
5374 016706 104000

```
:*****
:TEST 246 TEST MTPS MODE 5
:*****
TS246:
MOV #404,R0 ;R0=404
MOV #177400,@#0 ;LOC. 0=177400
SCC ;SET ALL COND. CODES
MTPS @-(R0) ;TRY MTPS W/MODE 5
TST @#PS ;CHECK DEST. DATA.
BEQ MTPS5
MTPS5: EMT ;DESTINATION DATA INCORRECT
CMP #402,R0 ;CHECK MODE 5 REGISTER
BEQ TS247
EMT ;MODE 5 REGISTER NOT DECREMENTED BY 2
```

5375
5376
5377
5378
5379 016710
5380 016710 012737 052652 000000
5381 016716 012700 000406
5382 016722 005037 177776
5383 016726 106460 177372
5384 016732 022737 000252 177776
5385 016740 001401
5386 016742 104000
5387 016744 022700 000406
5388 016750 001401
5389 016752 104000

```
:*****
:TEST 247 TEST MTPS MODE 6
:*****
TS247:
MOV #52652,@#0 ;LOC. 0=52652
MOV #406,R0 ;R0=406
CLR @#PS ;PS=0
MTPS -406(R0) ;TRY MTPS W/MODE 6
CMP #252,@#PS ;CHECK DEST. DATA
BEQ MTPS6
MTPS6: EMT ;DEST. DATA INCORRECT
CMP #406,R0 ;CHECK MODE 6 REGISTER
BEQ TS250
EMT ;MODE 6 REGISTER MODIFIED
```

5390
5391
5392
5393
5394 016754
5395 016754 012737 052652 000000
5396 016762 012700 000410
5397 016766 005037 177776
5398 016772 106470 177776
5399 016776 022737 000105 177776
5400 017004 001401
5401 017006 104000
5402 017010 022700 000410

```
:*****
:TEST 250 TEST MTPS MODE 7
:*****
TS250:
MOV #52652,@#0 ;LOC. 0=52652
MOV #410,R0 ;R0=410
CLR @#PS ;PS=0
MTPS @-2(R0) ;TRY MTPS W/MODE 7
CMP #105,@#PS ;CHECK DEST. DATA
BEQ MTPS7
MTPS7: EMT ;DESTINATION DATA INCORRECT
CMP #410,R0 ;CHECK MODE 7 REGISTER
```

5403 017014 001401
5404 017016 104000
5405
5406
5407
5408
5409
5410
5411
5412
5413
5414
5415
5416
5417 017020
5418 017020 012737 000377 177776
5419 017026 106700
5420 017030 022700 177757
5421 017034 001401
5422 017036 104000
5423
5424 017040 005000
5425 017042 012737 177777 000000
5426 017050 005037 177776
5427 017054 106710
5428 017056 105737 000000
5429 017062 001401
5430 017064 104000
5431
5432
5433
5434
5435 017066
5436 017066 005000
5437 017070 005010
5438 017072 012737 000377 177776
5439 017100 106720
5440 017102 103003
5441 017104 102402
5442 017106 001401
5443 017110 100401
5444 017112
5445 017112 104000
5446 017114 022737 000357 000000
5447 017122 001401
5448 017124 104000
5449 017126 022700 000001
5450 017132 001401
5451 017134 104000
5452
5453
5454
5455
5456 017136
5457 017136 012700 000406
5458 017142 005037 000000

```
BEQ TS251
EMT ;MODE 7 REGISTER MODIFIED

:*****
: THESE NEXT SEVEN TESTS VERIFY THE MFPS INSTRUCTION IN ALL
: MODES. IN EACH TEST, A PATTERN OF ONES AND ZEROES IS MOVED TO THE
: PSW, AND AN MFPS INSTRUCTION MOVES THE DATA TO A LOCATION SETUP
: BY R0, EITHER DIRECTLY OR INDIRECTLY. CONDITIONAL BRANCHES ARE
: USED TO CHECK PROPER ADDRESSING AND DATA.
:*****
: TEST 251 TEST MFPS INSTRUCTION
:*****
TS251:
MOV #377,@#PS
MFPS R0
CMP #177757,R0
BEQ MFPS1
EMT ;MFPS FAILED

MFPS1: CLR R0
MOV #-1,@#0
CLR @#PS
MFPS (R0)
TSTB @#0
BEQ TS252
EMT ;MFPS FAILED

:*****
: TEST 252 TEST MFPS MODE 2
:*****
TS252:
CLR R0 ;R0=0
CLR (R0) ;LOC. 0=0
MOV #377,@#PS ;SET PS=357
MFPS (R0)+ ;TRY MFPS W/MODE 2
BCC MFPS2A ;BR TO ERROR IF C BIT CLEAR
BVS MFPS2A ;BR TO ERROR IF V BIT SET
BEQ MFPS2A ;BR TO ERROR IF Z BIT SET
BMI MFPS2B

MFPS2A: EMT ;COND. CODES INCORRECT
MFPS2B: CMP #357,@#0 ;CHECK DEST. DATA
BEQ MFPS2C
EMT ;DEST. DATA INCORRECT
MFPS2C: CMP #1,R0 ;CHECK MODE 2 REGISTER
BEQ TS253
EMT ;MODE 2 REGISTER NOT INCREMENTED 1

:*****
: TEST 253 TEST MFPS MODE 3
:*****
TS253:
MOV #406,R0 ;R0=406
CLR @#0 ;LOC. 0=0
```

```
5459 017146 012737 000252 177776      MOV    #252,@#PS      ;PS=252
5460 017154 106730                      MFPS   @-(R0)+        ;TRY MFPS WITH MODE 3
5461 017156 103403                      BCS   MFPS3A         ;BR TO ERROR IF C-BIT SET
5462 017160 102402                      BVS   MFPS3A         ;BR TO ERROR IF V-BIT SET
5463 017162 001401                      BEQ   MFPS3A         ;BR TO ERROR IF Z-BIT SET
5464 017164 100401                      BMI   MFPS3B
5465 017166                                MFPS3A:
5466 017166 104000                      EMT
5467 017170 022737 125000 000000 MFPS3B: CMP    #125000,@#0 ;CONDITION CODES INCORRECT
5468 017176 001401                      BEQ   MFPS3C         ;CHECK DEST. DATA
5469 017200 104000                      EMT
5470 017202 020027 000410 MFPS3C: CMP    R0,#410 ;DEST DATA INCORRECT
5471 017206 001401                      BEQ   TS254         ;CHECK MODE 3 REGISTER.
5472 017210 104000                      EMT ;MODE 3 REGISTER NOT INCREMENTED BY 2
```

```
:*****
:TEST 254      TEST MFPS MODE 4
:*****
TS254:
```

```
5477 017212
5478 017212 012700 000002      MOV    #2,R0          ;R0=2
5479 017216 005037 000000      CLR    @#0           ;LOC. 0=0
5480 017222 012737 000125 177776      MOV    #125,@#PS     ;PS=125
5481 017230 106740                      MFPS   -(R0)         ;TRY MFPS W/MODE 4
5482 017232 103003                      BCC   MFPS4A         ;BR TO ERROR IF C-BIT CLEAR
5483 017234 102402                      BVS   MFPS4A         ;BR TO ERROR IF V-BIT SET
5484 017236 001401                      BEQ   MFPS4A         ;BR TO ERROR IF Z-BIT SET
5485 017240 100001                      BPL   MFPS4B
5486 017242                                MFPS4A:
5487 017242 104000                      EMT
5488 017244 022737 042400 000000 MFPS4B: CMP    #42400,@#0 ;COND. CODES INCORRECT
5489 017252 001401                      BEQ   MFPS4C         ;CHECK DEST. DATA
5490 017254 104000                      EMT
5491 017256 020027 000001 MFPS4C: CMP    R0,#1   ;DEST. DATA INCORRECT
5492 017262 001401                      BEQ   TS255         ;CHECK MODE 4 REGISTER
5493 017264 104000                      EMT ;MODE 4 REGISTER NOT DECREMENTED BY 1
```

```
:*****
:TEST 255      TEST MFPS MODE 5
:*****
TS255:
```

```
5494
5495
5496
5497
5498 017266
5499 017266 012700 000410      MOV    #410,R0       ;R0=410
5500 017272 012737 177777 000000      MOV    #-1,@#0      ;LOC. 0=-1
5501 017300 005037 177776      CLR    @#PS         ;PS=0
5502 017304 106750                      MFPS   @-(R0)       ;TRY MFPS W/MODE 5
5503 017306 103403                      BCS   MFPS5A         ;BR TO ERROR IF C-BIT SET
5504 017310 102402                      BVS   MFPS5A         ;BR TO ERROR IF V-BIT SET
5505 017312 100401                      BMI   MFPS5A         ;BR TO ERROR IF N-BIT SET
5506 017314 001401                      BEQ   MFPS5B
5507 017316                                MFPS5A:
5508 017316 104000                      EMT
5509 017320 022737 000377 000000 MFPS5B: CMP    #377,@#0 ;COND. CODES INCORRECT
5510 017326 001401                      BEQ   MFPS5C         ;CHECK DEST. DATA
5511 017330 104000                      EMT
5512 017332 020027 000406 MFPS5C: CMP    R0,#406 ;DEST DATA INCORRECT
5513 017336 001401                      BEQ   TS256         ;CHECK MODE 5 REGISTER
5514 017340 104000                      EMT ;MODE 5 REGISTER NOT DECREMENTED BY 2
```

```
5515
5516
5517
5518
5519 017342
5520 017342 012700 000401
5521 017346 005037 000000
5522 017352 012737 000252 177776
5523 017360 106760 177377
5524 017364 102403
5525 017366 103402
5526 017370 001401
5527 017372 100401
5528 017374
5529 017374 104000
5530 017376 022737 000252 000000
5531 017404 001401
5532 017406 104000
5533 017410 022700 000401
5534 017414 001401
5535 017416 104000
5536
5537
5538
5539
5540 017420
5541 017420 012700 000777
5542 017424 005037 000000
5543 017430 012737 000125 177776
5544 017436 106770 177407
5545 017442 102403
5546 017444 103002
5547 017446 001401
5548 017450 100001
5549 017452
5550 017452 104000
5551 017454 022737 042400 000000
5552 017462 001401
5553 017464 104000
5554 017466 022700 000777
5555 017472 001401
5556 017474 104000
5557
5558
5559
5560
5561
5562
5563
5564
5565
5566
5567
5568
5569 017476
5570 017476 032737 000001 001020
```

```
*****
:TEST 256 TEST MFPS MODE 6
*****
TS256:
MOV #401,R0 ;R0=410
CLR @#0 ;LOC. 0=0
MOV #252,@#PS ;PS=252
MFPS -401(R0) ;TRY MFPS W/MODE 6
BVS MFPS6A ;BR TO ERROR IF V-BIT SET
BCS MFPS6A ;BR TO ERROR IF C-BIT SET
BEQ MFPS6A ;BR TO ERROR IF Z-BIT SET
BMI MFPS6B
MFPS6A:
EMT ;COND. CODES INCORRECT
MFPS6B: CMP #252,@#0 ;CHECK DEST. DATA
BEQ MFPS6C
EMT ;DEST. DATA INCORRECT
MFPS6C: CMP #401,R0 ;CHECK DEST. REGISTER
BEQ TS257
EMT ;DEST. DATA INCORRECT
*****
:TEST 257 TEST MFPS MODE 7
*****
TS257:
MOV #777,R0 ;R0=777
CLR @#0 ;LOC. 0=0
MOV #125,@#PS ;PS=125
MFPS @-371(R0) ;TRY MFPS W/MODE 7
BVS MFPS7A ;BR TO ERROR IF V-BIT SET
BCC MFPS7A ;BR TO ERROR IF C-BIT SET
BEQ MFPS7A ;BR TO ERROR IF Z-BIT SET
BPL MFPS7B
MFPS7A:
EMT ;CONDITION CODE INCORRECT
MFPS7B: CMP #42400,@#0 ;CHECK DESTINATION DATA
BEQ MFPS7C
EMT ;DEST. DATA INCORRECT
MFPS7C: CMP #777,R0 ;CHECK MODE 7 REGISTER
BEQ TS260
EMT ;MODE 7 REGISTER MODIFIED
*****
: THIS TEST VERIFIES THAT RESET DOES NOT CLEAR THE PSW.
: THE PSW IS LOADED WITH ONES, A RESET IS ISSUED, AND THE
: CONTENTS OF THE PSW ARE CHECKED TO VERIFY THAT THEY HAVE NOT
: CHANGED. THIS TEST IS EXECUTED ONLY ONCE EVERY 240 (DECIMAL)
: ITERATIONS OF PROGRAM.
*****
:TEST 260 TEST THAT RESET DOES NOT CLEAR PSW
*****
TS260:
BIT #1,@#SENV ;ARE WE RUNNING UNDER APT
```


5571 017504 001403
5572 017506 005737 001006
5573 017512 001011
5574 017514
5575 017514 012737 000357 177776
5576 017522 000005
5577 017524 022737 000357 177776
5578 017532 001401
5579 017534 104000
5580 017536

BEQ 70S ;IF NO THEN DO TEST
TST @#SPASS ;IS THIS FIRST PASS
BNE TS261 ;IF NO THEN SHIP TO NEXT TEST
70S: MOV #357,@#PS ;MOV ONES TO PSW
RESET ;
CMP #357,@#PS ;PSW CORRECT?
BEQ TS261
EMT ;RESET ALTERED PSW

REST:

5581
5582
5583
5584
5585
5586
5587
5588
5589
5590 017536
5591 017536 052767 140000 160232
5592 017544 012706 000001
5593 017550 000241
5594 017552 006106
5595 017554 103376
5596 017556 001404
5597 017560 042767 140000 160210
5598 017566 104000
5599 017570 042767 140000 160200
5600
5601
5602
5603
5604
5605
5606
5607
5608
5609
5610
5611
5612 017576
5613 017576 052767 140000 160172
5614 017604 012706 177777
5615 017610 022706 177777
5616 017614 001404
5617 017616 042767 140000 160152
5618 017624 104000
5619 017626 042767 140000 160142
5620 017634 022706 177777
5621 017640 001001
5622 017642 104000
5623 017644 005006
5624 017646 052767 140000 160122
5625 017654 022706 177777
5626 017660 042767 140000 160110

: THE FOLLOWING TEST CHECKS THE INDEPENDENT FUNCTIONING OF BASIC
: DATA PATH COMPONENTS WITH USER MODE SET.
:*****

TEST 261 TEST USER MODE R6 CAN HOLD A ONE IN EVERY POSITION

TS261: BIS #USRM,PS ;SET USER MODE
MOV #1,R6 ;SET BIT0
CLC ;CLEAR C-BIT
USP1: ROL R6 ;ROTATE 1 POSITION
BCC USP1 ;BR IF NOT ALL DONE
BEQ USP1A ;BR IF NO BITS PICKED
BIC #USRM,PS ;CLEAR USER MODE
EMT ;USER MODE R6 PICKED A BIT
USP1A: BIC #USRM,PS ;CLEAR USER MODE

: THIS TEST CHECKS THE INDEPENDENT FUNCTIONING OF THE USER
: AND KERNEL MODE R6'S. R6 IS SETUP AND ADDRESSED IN EACH
: OF THE TWO MODES TO VERIFY THAT THE TWO R6'S ARE INDEPENDENT
: OF EACH OTHER.
:*****

TEST 262 TEST INDEPENDENCE OF USER AND KERNEL MODE R6'S

TS262: BIS #USRM,PS ;SET USER MODE
MOV #-1,R6 ;SET USER R6 TO ALL ONES
CMP #-1,R6 ;READ AND CHECK USER R6
BEQ USP2 ;BR IF NO ERROR
BIC #USRM,PS ;CLEAR USER MODE
EMT ;USER R6 WILL NOT HOLD ALL ONES
USP2: BIC #USRM,PS ;SET KERNEL MODE
CMP #-1,R6 ;KERNEL MODE R6 ADDR. FROM USER MODE?>>
BNE USP3
EMT ;DUAL ADDRESSING ERROR USER/KERNEL R6
USP3: CLR R6 ;CLEAR KERNEL MODE SP
BIS #USRM,PS ;SET USER MODE
CMP #-1,R6 ;CHECK USER R6 NOT ADDR. FROM KERNEL MODE
BIC #USRM,PS ;CLEAR USER MODE

5627 017666 001401
5628 017670 104000
5629 017672 012706 001000
5630 017676 042767 140000 160072
5631 017704 012706 001000

BEQ USP4 ;BR IF NO ERROR
EMT ;DUAL ADDRESSING ERROR OR SEQUENCE ERROR
USP4: MOV #STBOT,R6 ;RESTORE SP USER
BIC #USRM,PS ;SET KERNEL MODE
MOV #STBOT,R6 ;RESTORE SP KERNEL

5632
5633
5634
5635
5636
5637
5638

: THESE NEXT TWO TESTS VERIFY MFPI AND MTPI INSTRUCTIONS
: WITH R6 IN MODE 0.

5639
5640
5641 017710

:TEST 263 TEST MFPI WITH R6 IN MODE 0

TS263:

5642 017710 012706 001000
5643 017714 012767 140000 160054
5644 017722 012706 000600
5645 017726 006506
5646 017730 022767 140000 160040
5647 017736 001404
5648 017740 042767 140000 160030
5649 017746 104000
5650 017750 042767 140000 160020
5651 017756 022767 001000 160612
5652 017764 001401
5653 017766 104000
5654 017770

MOV #STBOT,R6 ;INITIALIZE KERNEL STACK POINTER
MOV #USRM,PS ;SET USER MODE/PREVIOUS KERNEL
MOV #USESTK,R6 ;INITIALIZE USER STACK POINTER
MFPI R6 ;TRY MFPI WITH MODE 0
CMP #140000,PS ;CHECK PSW
BEQ MFPIO ;BR IF NO ERROR
BIC #USRM,PS ;CLEAR USER MODE
EMT ;INCORRECT PSW FROM MFPI
MFPIO: BIC #USRM,PS ;CLEAR USER MODE
CMP #STBOT,USESTK-2 ;CHECK DATA ON STACK
BEQ MFPIOA ;BR IF NO ERROR
EMT ;INCORRECT DATA FROM MFPI
MFPIOA:

5655
5656
5657
5658
5659 017770

:TEST 264 TEST MTPI WITH R6 IN MODE 0

TS264:

5660 017770 005067 160002
5661 017774 005006
5662 017776 012767 140000 157772
5663 020004 012706 000600
5664 020010 012746 001000
5665 020014 006606
5666 020016 022767 140000 157752
5667 020024 001404
5668 020026 042767 140000 157742
5669 020034 104000
5670 020036 005067 157734
5671 020042 020627 001000
5672 020046 001401
5673 020050 104000

CLR PS ;SET KERNEL MODE
CLR R6 ;INITIALIZE KERNEL R6
MOV #USRM,PS ;SET USER MODE/PREVIOUS KERNEL
MOV #USESTK,R6 ;INITIALIZE USER STACK POINTER
MOV #STBOT,-(R6) ;SET UP TARGET DATA
MTPI R6 ;TRY MODE 0 MTPI
CMP #USRM,PS ;CHECK PSW
BEQ MTPIO ;BR IF NO ERROR
BIC #USRM,PS ;CLEAR USER MODE
EMT ;PS INCORRECT FOLLOWING MTPI
MTPIO: CLR PS ;SET KERNEL MODE
CMP R6,#STBOT ;CHECK TARGET DATA
BEQ TS265
EMT ;DATA INCORRECT FOLLOWING MTPI

5674
5675
5676
5677
5678
5679
5680
5681
5682

: THE FOLLOWING TEST VERIFIES THAT NO DUAL ADDRESSING OF THE GENERAL
: REGISTERS OCCURS. ALL REGISTERS ARE CLEARED, AND A UNIQUE BIT IS SET
: IN EACH. CMP INSTRUCTIONS CHECK THAT ONLY ONE BIT IS SET IN EACH
: REGISTER.

5683
5684
5685
5686
5687 020052
5688 020052 005000
5689 020054 005001
5690 020056 005002
5691 020060 005003
5692 020062 005004
5693 020064 005005
5694 020066 005006
5695 020070 052700 000001
5696 020074 052701 000002
5697 020100 052702 000004
5698 020104 052703 000010
5699 020110 052704 000020
5700 020114 052705 000040
5701 020120 052706 000100
5702 020124 022706 000100
5703 020130 001022
5704 020132 022705 000040
5705 020136 001017
5706 020140 022704 000020
5707 020144 001014
5708 020146 022703 000010
5709 020152 001011
5710 020154 022702 000004
5711 020160 001006
5712 020162 022701 000002
5713 020166 001003
5714 020170 022700 000001
5715 020174 001401
5716 020176
5717 020176 104000
5718 020200 012702 001004
5719
5720
5721
5722
5723
5724
5725
5726
5727
5728
5729 020204
5730 020204 052737 170357 177776
5731 020212 105037 177776
5732 020216 013700 177776
5733 020222 032700 170000
5734 020226 001003
5735 020230 005037 177776
5736 020234 104000
5737 020236 005037 177776
5738

```
:
:*****
:TEST 265      DUAL REGISTER ADDRESSING TEST
:*****
TS265:
BITCLR: CLR      R0          ;INITIALIZE ALL REGISTERS
        CLR      R1
        CLR      R2
        CLR      R3
        CLR      R4
        CLR      R5
        CLR      R6
BITSET: BIS      #1,R0      ;SET R0=1
        BIS      #2,R1      ;R1=2
        BIS      #4,R2      ;R2=4
        BIS      #10,R3     ;R3=10
        BIS      #20,R4     ;R4=20
        BIS      #40,R5     ;R5=40
        BIS      #100,R6    ;R6=100
BITCHK: CMP      #100,R6    ;TEST THAT NO DUAL ADDRESSING OCCURRED
        BNE      DAERR      ;BR TO ERROR HALT IF ANY OTHER BITS ARE SET
        CMP      #40,R5
        BNE      DAERR
        CMP      #20,R4
        BNE      DAERR
        CMP      #10,R3
        BNE      DAERR
        CMP      #4,R2
        BNE      DAERR
        CMP      #2,R1
        BNE      DAERR
        CMP      #1,R0
        BEQ      BITCON
DAERR:  EMT
BITCON: MOV      #$TESTN,R2 ;DUAL ADDRESSING ERROR
        ;RESTORE POINTER
:*****
:      THIS TEST VERIFIES THAT THE UPPER BYTE OF THE PSW IS NOT AFFECTED
:WHEN THE PRIORITY LEVEL OR CC'S ARE CHANGED. ALL BITS ARE
:INITIALLY SET IN THE PSW, AND THE LOW BYTE IS CLEARED. A BIT
:INSTRUCTION VERIFIES THE DATA.
:*****
:TEST 266      TEST BYTE INSTRUCTION ON PSW
:*****
TS266:
        BIS      #170357,&#PSW ;SET ALL POSSIBLE BITS IN PSW
        CLR      &#PSW         ;CLR PR LEVEL AND CC'S
        MOV      &#PSW,R0      ;COPY CONTENTS OF PSW
        BIT      #170000,R0     ;TEST THAT UPPER BYTE IS UNAFFECTED
        BNE      BITCON        ;CONTINUE IF OK
BTERR:  CLR      &#PSW         ;RETURN TO KERNEL MODE
        EMT
BTCON:  CLR      &#PSW         ;BYTE INSTRUCTION ALTERED PSW
        ;RETURN TO KERNEL MODE
```

5739
5740
5741
5742
5743
5744
5745
5746
5747
5748 020242
5749 020242 000277
5750 020244 000252
5751 020246 000167 000000
5752 020252 100403
5753 020254 001002
5754 020256 102401
5755 020260 103401
5756 020262
5757 020262 104000
5758
5759
5760
5761
5762
5763
5764
5765
5766
5767
5768
5769
5770
5771
5772
5773
5774
5775 020264
5776 020264 012767 000240 000024
5777 020272 012767 000017 000032
5778 020300 012767 000261 000074
5779 020306 012767 000001 000102
5780 020314 000277
5781 020316 000000
5782 020320 013704 177776
5783 020324 042704 177760
5784 020330 022704
5785 020332 000000
5786 020334 001401
5787 020336 104000
5788 020340 005367 177766
5789 020344 005267 177746
5790 020350 026727 177742 000257
5791 020356 003756
5792 020360 026727 177732 000260
5793 020366 001004
5794 020370 012767 000017 177734

```
*****
:
: THIS TEST VERIFIES THAT A JMP INSTRUCTION DOES NOT ALTER THE
: CONDITION CODES IN THE PSW. THE CC'S ARE PRESET, THE JMP IS
: EXECUTED, AND CONDITIONAL BRANCHES VERIFY THE STATE OF THE CC'S.
:
: *****
: TEST 267 TEST THAT JMP INSTRUCTION DOES NOT AFFECT CONDITION CODES
: *****
TS267:
      SCC
      +CLN!CLV ;CC=0101
      JMP JMPT ;JUMP TO TEST PSW
JMPT:  BMI JMPERR ;BR TO ERROR HALT IF N-BIT IS SET
      BNE JMPERR ;BR TO ERROR HALT IF Z-BIT IS CLEAR
      BVS JMPERR ;BR TO ERROR HALT IF V-BIT IS SET
      BCS TS270
JMPPERR:
      EMT ;JMP INSTRUCTION AFFECTED CC'S
: *****
:
: THIS TEST VERIFIES THE SET AND CLEAR CONDITION CODE INSTRUCTIONS.
: THE TEST CONSISTS OF TWO ROUTINES, ONE TO TEST ALL CLEAR CC
: INSTRUCTIONS, AND THE SECOND TO TEST ALL SET CC INSTRUCTIONS. ALL
: POSSIBLE COMBINATIONS OF CONDITION CODES ARE TESTED, INCLUDING NOP'S.
: TO TEST THE CLEAR CC INSTRUCTIONS, ALL CONDITION CODES ARE
: INITIALLY SET. THE INSTRUCTION IS EXECUTED, AND THE PSW IS CHECKED
: TO VERIFY THE PROPER COMBINATION OF CONDITION CODES.
: TO TEST THE SET CC INSTRUCTIONS, THE CONDITION CODES ARE
: INITIALLY CLEARED, AND ONLY THE REQUIRED BITS ARE SET BY THE SET CC
: INSTRUCTION. THE CONTENTS OF THE PSW ARE CHECKED TO VERIFY THAT
: ONLY THE REQUIRED BITS WERE SET.
:
: *****
: TEST 270 TEST SET CC AND CLEAR CC INSTRUCTIONS
: *****
TS270:
      MOV #240,CC3 ;INITIALIZE CLR CC INSTRUCTION CODES
      MOV #17,CC2 ;INITIALIZE OCTAL MAP
      MOV #261,SC3 ;INITIALIZE SET CC INSTRUCTION CODES
      MOV #1,SC4 ;INITIALIZE OCTAL MAP
CLRCD:  SCC ;SET ALL CONDITION CODES
CC3:  0 ;CONDITION CODE INSTRUCTION
      MOV @MPS,R4 ;COPY THE PSW
      BIC #177760,R4 ;ISOLATE CONDITION CODES
      CMP (PC)+,R4 ;CHECK THAT PROPER CC'S WERE CLEARED
CC2:  0 ;OCTAL REPRESENTATION OF CC'S
      BEQ CON1
      EMT ;CLEAR CC INSTRUCTION FAILED
CON1:  DEC CC2 ;SET NEXT OCTAL MAP OF CC'S
      INC CC3 ;GET NEXT CLEAR CC INSTRUCTION
      CMP CC3,#257 ;TEST FOR CCC INSTRUCTION
      BLE CLRCD ;GO TEST NEXT INSTRUCTION IF NOT FOUND
      CMP CC3,#260 ;CHECK FOR NOP=260
      BNE SETCD ;GO TEST SET CC INSTRUCTIONS
      MOV #17,CC2 ;SET OCTAL MAP TO TEST NOP
```

5795	020376	000746		
5796	020400	000257		
5797	020402	000000		
5798	020404	013704	177776	
5799	020410	042704	177760	
5800	020414	022704		
5801	020416	000000		
5802	020420	001401		
5803	020422			
5804	020422	104000		
5805	020424	005267	177766	
5806	020430	005267	177746	
5807	020434	026727	177742	000277
5808	020442	003756		
5809	020444	000167	000006	

BR	CLRCD			:GO TEST NOP
SETCD:	CCC			:CLEAR ALL CONDITION CODES
SC3:	0			:CONDITION CODE INSTRUCTION
MOV	2#PS,R4			:COY PSW
BIC	#177760,R4			:CLEAR AWAY UNWANTED BITS
CMP	(PC)+,R4			:CHECK THAT PROPER CC'S WERE SET
SC4:	0			:OCTAL REPRESENTATION OF CC'S
BEQ	CON2			
CCERR:				
EMT				:SET CC FAILED OR SEQUENCE ERROR
CON2:	INC	SC4		:SET NEXT OCTAL MAP
	INC	SC3		:PREPARE NEXT SET CC INSTRUCTION
	CMP	SC3,#277		:FINISHED?
	BLE	SETCD		:BR IF NO
	JMP	MORO		:JUMP TO NEXT TESTS

5810
5811
5812
5813
5814
5815
5816
5817
5818
5819 020450 000000 000000 000000
5820 020456
5821
5822
5823
5824 020456
5825 020456 005037 020450
5826 020462 012700 020450
5827 020466 060020
5828
5829 020470 022700 020452
5830 020474 001401
5831 020476 104000
5832
5833 020500 022737 020452 020450
5834
5835
5836 020506 001401
5837 020510 104000
5838
5839
5840
5841
5842 020512
5843 020512 005037 020450
5844 020516 012700 020452
5845 020522 060040
5846
5847 020524 022700 020450
5848 020530 001401
5849 020532 104000
5850
5851 020534 022737 020450 020450
5852
5853
5854 020542 001401
5855 020544 104000
5856
5857
5858
5859
5860 020546
5861 020546 005037 020450
5862 020552 005037 020454
5863 020556 012737 020450 020452
5864 020564 012700 020452
5865 020570 060030

```
*****  
:SBTTL TEST INSTRUCTIONS USING SAME REGISTER FOR SOURCE & DESTINATION  
:  
:IN AUTO INCREMENT (DECREMENT) MODES AND  
:AUTO INCREMENT (DECREMENT) DEFERRED MODES,  
:CONTENTS OF THE REGISTER IN USED ARE  
:INCREMENTED (DECREMENTED) BY 2  
:BEFORE USED AS THE SOURCE OPERAND.  
:  
A: .WORD 0,0,0  
MORO:  
*****  
:TEST 271 TEST AUTO-INCREMENT MODE, USING RO  
*****  
TS271:  
CLR @#A ;CLEAR LOC A  
MOV #A,RO ;RO STORES ADDR OF A  
ADD RO,(RO)+ ;CHECK THAT RO IS INCR BY 2 BEFORE  
;BEING USED AS THE SOURCE OPERAND  
;RO INCR BY 2?  
CMP #A+2,RO  
BEQ MOR1  
EMT ;RO WAS NOT INCREMENTED BY 2  
:  
MOR1: CMP #A+2,@#A ;CHECK CONTENT OF RO WAS INCR BY 2 BEFORE  
;BEING USED IN THE "ADD" INSTR  
;LOC A CONTAINS (A+2)?  
BEQ TS272  
EMT ;WRONG SUM IN LOC A  
:  
*****  
:TEST 272 AUTO-DECREMENT MODE, USING RO  
*****  
TS272:  
CLR @#A ;CLEAR LOC A  
MOV #A+2,RO ;RO STORES ADDR OF A+2  
ADD RO,-(RO) ;CHECK THAT RO IS DECR BY 2 BEFORE  
;BEING USED AS THE SOURCE OPERAND  
;RO DECR BY 2?  
CMP #A,RO  
BEQ MOR2  
EMT ;RO WAS NOT DECREMENTED BY 2  
:  
MOR2: CMP #A,@#A ;CONTENT OF RO WAS DECR BY 2 BEFORE  
;BEING USED IN THE "ADD" INSTR  
;LOC A CONTAINS (RO)  
BEQ TS273  
EMT ;WRONG SUM IN LOC A  
:  
*****  
:TEST 273 TEST AUTO-INCREMENT DEFERRED MODE, USING RO  
*****  
TS273:  
CLR @#A ;CLEAR LOC A  
CLR @#A+4 ;CLEAR LOC A+4  
MOV #A,@#A+2 ;STORE ADDR A IN LOC A+2  
MOV #A+2,RO ;RO STORES ADDR A+2  
ADD RO,@(RO)+ ;CHECK THAT RO IS INCR BY 2 BEFORE
```


5912
5913
5914
5915
5916
5917
5918
5919
5920
5921
5922 020726
5923 020726 012700 177777
5924 020732 010700
5925 020734 022700 020734
5926 020740 001401
5927 020742 104000
5928
5929
5930
5931
5932 020744
5933 020744 012700 020450
5934 020750 010760 000004
5935 020754 022737 020754 020454
5936 020762 001401
5937 020764 104000
5938
5939
5940
5941
5942 020766
5943 020766 012737 020450 020454
5944 020774 012700 020450
5945 021000 010770 000004
5946 021004 022737 021004 020450
5947 021012 001401
5948 021014 104000
5949
5950
5951
5952
5953 021016
5954 021016 012737 020452 020450
5955 021024 010777 177420
5956 021030 022737 021030 020452
5957 021036 001401
5958 021040 104000
5959
5960
5961
5962
5963 021042
5964 021042 005037 020450
5965 021046 010767 177376
5966 021052 022737 021052 020450
5967 021060 001401

```
*****
:SBTTL INSTRUCTION USING PC AS SOURCE REGISTER
:
:IN INDEX, INDEX DEFERRED, RELATIVE, AND
:RELATIVE DEFERRED MODES, DESTINATION WILL CONTAIN
:THE PC COUNT OF THE CURRENT INSTRUCTION +4.
:
*****
:TEST 275 TEST PC AS SOURCE IN MODE 0, USING RO
*****
TS275:
PCNO1: MOV #-1,RO ;SET ALL 1 IN RO
MOV PC,RO ;STORES PC IN RO
CMP #PCNO1+2,RO ;RO STORES PC+2?
BEQ TS276
EMT ;RO STORED WRONG VALUE

*****
:TEST 276 TEST PC AS SOURCE IN MODE 6, USING RO
*****
TS276:
PCN2: MOV #A,RO ;RO STORES ADDR A
MOV PC,4(RO) ;EFFECTIVE ADDR IS A+4
CMP #PCN2+4,@#A+4 ;LOC A+4 STORES PC+4?
BEQ TS277
EMT ;LOC A+4 STORED WRONG VALUE

*****
:TEST 277 TEST PC AS SOURCE IN MODE 7, USING RO
*****
TS277:
PCN3: MOV #A,@#A+4 ;LOC A+4 STORES ADDR A
MOV #A,RO ;RO STORES ADDR A
MOV PC,@4(RO) ;EFFECTIVE ADDR IS A
CMP #PCN3+4,@#A ;LOC A STORES PC+4?
BEQ TS300
EMT ;LOC A STORED WRONG VALUE

*****
:TEST 300 TEST PC AS SOURCE IN RELATIVE DEFERRED MODE ,USING RO
*****
TS300:
PCN4: MOV #A+2,@#A ;LOC A STORES ADDR A+2
MOV PC,@A ;EFFECTIVE ADDR IS A+2
CMP #PCN4+4,@#A+2 ;LOC A+2 STORES PC+4?
BEQ TS301
EMT ;LOC A+2 STORED WRONG VALUE

*****
:TEST 301 TEST PC AS SOURCE IN RELATIVE MODE ,USING RO
*****
TS301:
PCN5: CLR @#A ;CLEAR A
MOV PC,A ;EFFECTIVE ADDR IS A
CMP #PCN5+4,@#A ;LOC A STORES PC+4?
BEQ TS302
```


5968	021062	104000		
5969				
5970				
5971				
5972				
5973				
5974				
5975				
5976				
5977	021064			
5978		000007		
5979	021064	012706	001000	
5980	021070	000007		
5981	021072	022700	000003	
5982	021076	001401		
5983	021100	104000		
5984				
5985				
5986				
5987				
5988				
5989				
5990				
5991				
5992	021102			
5993	021102	012737	052525	000000
5994	021110	012701	050505	
5995	021114	005000		
5996	021116	160120		
5997	021120	022737	002020	000000
5998	021126	001401		
5999	021130	104000		
6000				
6001				
6002				
6003				
6004	021132			
6005	021132	012737	052525	000000
6006	021140	005000		
6007	021142	012767	170000	156626
6008	021150	012706	000600	
6009	021154	106520		
6010	021156	005067	156614	
6011	021162	022767	052525	157406
6012	021170	001401		
6013	021172	104000		
6014				
6015				
6016				
6017				
6018	021174			
6019	021174	012767	170000	156574
6020	021202	012706	000600	
6021	021206	012746	125252	
6022	021212	012737	000000	000000
6023	021220	005000		

```

EMT ;LOCATION A STORED WRONG VALUE
:*****
:THIS TESTS THE MOVE FROM PROCESSOR TYPE INSTRUCTION(MFPT)
:UPON EXECUTION R0 WILL RECIEVE THE PROCESSOR MODEL CODE
:WHICH IS '000003' FOR THE DCF11-AA
:*****
:TEST 302 TEST MFPT
:*****
TS302:
MFPT=000007
MOV #STBOT,SP ;INITIALIZE STACK POINT IN CASE OF TRAP
MFPT ;GET MODEL CODE,IF THIS TRAPS AN ERROR WILL BE REPORTED
CMP #3,R0 ;CHECK IF CORRECT CODE RETURNED
BEQ TS303
EMT ;WRONG CODE RETURNED

:*****
:SBTTL THE NEXT THREE TESTS EXERCISE MASKING ACTION OF MICROCODES.
:*****
:TEST 303 TEST SUB INSTRUCTION, SM=0, DM=2
:*****
TS303:
MOV #052525,@#0 ;SET UP LOC 0
MOV #050505,R1 ;SET UP R1
CLR R0 ;CLEAR R0
SUB R1,(R0)+ ;SUBTRACTION, SM=0,DM=2
CMP #2020,@#0 ;CHECK DIFFERENCE AT LOC 0
BEQ TS304
EMT ;WRONG RESULT FROM SUBTRACTION

:*****
:TEST 304 TEST MFPD WITH R0, IN MODE 2
:*****
TS304:
MOV #052525,@#0 ;SET UP LOC 0
CLR R0 ;CLEAR R0
MOV #170000,PS ;SET USER MODE ON, CURRENT & PREVIOUS
MOV #USESTK,R6 ;SET USER STACK POINTER
MFPD (R0)+ ;MODE 2, MFPD
CLR PS ;SET KERNEL MODE
CMP #052525,USESTK-2 ;CHECK DATA ON STACK
BEQ TS305
EMT ;INCORRECT DATA FROM MFPD

:*****
:TEST 305 TEST MTPD WITH R0, IN MODE 2
:*****
TS305:
MOV #170000,PS ;SET USER MODE ON, CURRENT & PREVIOUS
MOV #USESTK,R6 ;SET USER STACK POINTER
MOV #125252,-(R6) ;PUSH DATA IN USER STACK
MOV #0,@#0 ;CLEAR LOC 0
CLR R0 ;CLEAR R0

```

```

6024 021222 106620
6025 021224 005067 156546
6026 021230 022737 125252 000000
6027 021236 001463
6028 021240 104000
6029
6030 021242 000402
6031 021244 001002
6032 021246 001402
6033 021250 002002
6034 021252 002402
6035 021254 003002
6036 021256 003402
6037 021260 100002
6038 021262 100402
6039 021264 101002
6040 021266 101402
6041 021270 102002
6042 021272 102402
6043 021274 103002
6044 021276 103402
6045
6046 000002
6047 021300 177777
6048 021302 170360
6049 021304 007417
6050 021306 146063
6051 021310 031714
6052 021312 140060
6053 021314 037717
6054
6055 021316 177400
6056 021320 000377
6057 021322 120240
6058 021324 057537
6059 021326 146314
6060 021330 031463
6061 021332 125252
6062 021334 052525
6063 000010
6064
6065
6066
6067
6068
6069
6070
6071 021336
6072 021336 104000
6073 021340
6074 021340 104000
6075 021342
6076 021342 104000
6077 021344
6078 021344 104000
6079 021346

```

```

MTPD (R0)+ ;MODE 2, MTPD
CLR PS ;SET KERNEL MODE
CMP #125252, @#0 ;CHECK DATA ON LOC 0
BEQ TESTN1
EMT ;INCORRECT DATA FROM MTPD

BRTAB: BR .+6
      BNE .+6
      BEQ .+6
      BGE .+6
      BLT .+6
      BGT .+6
      BLE .+6
      BPL .+6
      BMI .+6
      BHI .+6
      BLOS .+6
      BVC .+6
      BVS .+6
      BCC .+6 ;SAME AS BHIS
      BCS .+6 ;SAME AS BLO

.RADIX 2
YNTAB: 1111111111111111 ;BR
       1111000011110000 ;BNE: Z=0
       0000111100001111 ;BEQ: Z=1
       1100110000110011 ;BGE: N XOR V =0
       0011001111001100 ;BLT: N XOR V =1
       1100000000110000 ;BGT: Z+(N XOR V) =0
       0011111111001111 ;BLE: Z+(N XOR V) =1

       1111111100000000 ;BPL: N=0
       0000000011111111 ;BMI: N=1
       1010000010100000 ;BHI: C+Z=0
       0101111101011111 ;BLOS: C+Z=1
       1100110011001100 ;BVC: V=0
       0011001100110011 ;BVS: V=1
       1010101010101010 ;BCC: C=0
       0101010101010101 ;BCS: C=1

.RADIX 8

:*****
: THE FOLLOWING ARE SPECIAL CPU TRAP
: HANDLERS TO TRAP AND REPORT SPECIAL TRAPS.
:*****

T04: ;TRAPPED THRU LOC. 4
T010: EMT ;TRAPPED THRU LOC. 10
T014: EMT ;TRAPPED THRU LOC. 14
T020: EMT ;TRAPPED THRU LOC. 20
T030: EMT

```

6080	021346	104000		
6081	021350			
6082	021350	104000		
6083	021352			
6084	021352	104000		
6085	021354			
6086	021354	104000		
6087	021356			
6088	021356	104000		
6089				
6090				
6091				
6092		000000		
6093				
6094	021360	000000		
6095	021362	000000		
6096	021364	000000		
6097	021366	000000		
6098	021370	000000		
6099	021372	000000		
6100	021374	052525		
6101	021376	052400		
6102	021400	000000		
6103	021402	000000		
6104	021404	000176		
6105				
6106	021406	032737	000001	001020
6107	021414	001403		
6108	021416	012767	001022	177760
6109	021424			
6110				
6111				
6112				
6113	021424			
6114	021424	005006		
6115	021426	112667	156346	
6116	021432	020627	000002	
6117	021436	001401		
6118	021440	104000		
6119				
6120	021442	012706	001000	
6121	021446	114627	000000	
6122	021452	020627	000776	
6123	021456	001401		
6124	021460	104000		
6125				
6126	021462	005006		
6127	021464	112626		
6128	021466	020627	000004	
6129	021472	001401		
6130	021474	104000		
6131				
6132	021476	005006		
6133	021500	005004		
6134	021502	122624		
6135	021504	020627	000002	

```

T034: EMT ;TRAPPED THRU LOC. 30
T0114: EMT ;TRAPPED THRU LOC. 34
T0244: EMT ;TRAPPED THRU LOC. 114
T0250: EMT ;TRAPPED THRU LOC. 244
EMT ;TRAPPED THRU LOC. 250
.SBTTL ** STARTING OF TRAP TEST **

```

```

:SPECIAL CASE OF ODD;.EVEN .BYTE AND REGISTER 6
HERE=0

```

```

K1: 0
K2: 0
K3: 0
K4: 0
K5: 0
K6: 0
K7: 052525
K10: 052400
K11: 0
K12: 0
SWR: 176

```

```

TESTN1: BIT #1, @SENV
        BEQ 1$
        MOV #$$SWREG, SWR

```

1\$:

```

*****
:TEST 306 TEST AUTO INCREMENT AND DECREMENT OF R6 FOR WORD AND BYTES
*****

```

TS306:

```

        CLR %6
        MOVB (6)+, HERE ;SIX SHOULD INCREMENT BY TWO
        CMP %6, #2
        BEQ BR1
        EMT ;R6 DID NOT AUTO INCREMENT BY TWO

BR1:    MOV #1000, %6
        MOVB -(6), #HERE ;SHOULD DECREMENT BY TWO
        CMP %6, #776
        BEQ BR2
        EMT ;R6 DID NOT AUTO DECREMENT BY 2

BR2:    CLR %6
        MOVB (6)+, (6)+ ;DOUBLES AUTO INCREMENT OF R6
        CMP %6, #4
        BEQ BR3
        EMT ;WRONG AUTO INCREMENT OF R6

BR3:    CLR %6
        CLR %4
        MOVB (6)+, (4)+ ;TEST INCREMENT OF R6
        CMP %6, #2

```

```

6136 021510 001401      BEQ      BR4
6137 021512 104000      EMT
6138
6139 021514 005006      BR4:    CLR      %6
6140 021516 005004      CLR      %4
6141 021520 122426      CMPB     (4)+,(6)+      ;TEST INCREMENT OF R6
6142 021522 020627 000002      CMP      %6,#2
6143 021526 001401      BEQ      BR5
6144 021530 104000      EMT
6145
6146 021532 005006      BR5:    CLR      %6
6147 021534 005004      CLR      %4
6148 021536 122624      CMPB     (6)+,(4)+      ;TEST INCREMENT OF R4
6149 021540 020427 000001      CMP      %4,#1
6150 021544 001401      BEQ      BR6
6151 021546 104000      EMT
6152 021550 005006      BR6:    CLR      %6
6153 021552 005004      CLR      %4
6154 021554 122426      CMPB     (4)+,(6)+      ;TEST INCREMENT OF R6
6155 021556 020627 000002      CMP      %6,#2
6156 021562 001401      BEQ      BR7
6157 021564 104000      EMT
6158
6159 021566 005006      BR7:    CLR      %6
6160 021570 005004      CLR      %4
6161 021572 122426      CMPB     (4)+,(6)+      ;TEST INCREMENT OF R4
6162 021574 020427 000001      CMP      %4,#1
6163 021600 001401      BEQ      BR10
6164 021602 104000      EMT
6165
6166 021604 012706 001000      BR10:   MOV      #1000,%6
6167 021610 124627 000000      CMPB     -(6),#HERE      ;TEST DECREMENT OF R6
6168 021614 022706 000776      CMP      #776,%6
6169 021620 001401      BEQ      TS307
6170 021622 104000      EMT
6171
6172
6173
6174 021624
6175 021624 012767 123456 177536
6176 021632 012767 050505 177520
6177 021640 012705 021360
6178 021644 012706 021370
6179 021650 112625
6180 021652 022767 050456 177500
6181 021660 001401
6182 021662 104000
6183
6184 021664 012767 123456 177476
6185 021672 012767 050505 177460
6186 021700 012705 021360
6187 021704 012706 021372
6188 021710 114625
6189 021712 026727 177442 050456
6190 021720 001401
6191 021722 104000

;*****
;TEST 307      TEST TRANSFER OF .BYTE USING R6
;*****
TS307:
MOV      #123456,K5
MOV      #050505,K1
MOV      #K1,%5      ;%5=(050505)K1
MOV      #K5,%6      ;%6=(123456)K5
MOVB     (6)+,(5)+      ;LOW .BYTE OF R6 TO R5
CMP      #050456,K1
BEQ      BR11
EMT
;FALSE TRANSFER OF .BYTE

BR11:    MOV      #123456,K5
MOV      #050505,K1
MOV      #K1,%5      ;%5(050505)K1
MOV      #K6,%6      ;%6(123456)K5
MOVB     -(6),(5)+      ;LOW .BYTE OF R6 TO R5 (DECREMENT)
CMP      K1,#050456
BEQ      BR12
EMT
;FALSE R6 .BYTE TRANSFER
  
```

```
6192
6193 021724 012767 123456 177426 BR12: MOV #123456,K1
6194 021732 012767 050505 177430 MOV #050505,K5
6195 021740 012705 021360 MOV #K1,%5 ;(123456)
6196 021744 012706 021370 MOV #K5,%6 ;(050505)
6197 021750 112526 MOV#B (5)+,(6)+ ;LOW OF R5 TO LOW OF R6
6198 021752 022767 050456 177410 CMP #050456,K5
6199 021760 001401 BEQ BR13
6200 021762 104000 EMT ;FALSE R6 .BYTE TRANSFER
6201
6202 021764 012767 123456 177366 BR13: MOV #123456,K1
6203 021772 012767 050505 177370 MOV #050505,K5
6204 022000 012705 021361 MOV #K1+1,%5 ;123456
6205 022004 012706 021370 MOV #K5,%6 ;050505
6206 022010 112526 MOV#B (5)+,(6)+ ;HIGH OF R5 TO LOW OF R6
6207 022012 026727 177352 050647 CMP K5,#050647
6208 022020 001401 BEQ BR14
6209 022022 104000 EMT ;FALSE R6 .BYTE TRANSFER
6210
6211 022024 012767 123456 177326 BR14: MOV #123456,K1
6212 022032 012767 050505 177330 MOV #050505,K5
6213 022040 012705 021361 MOV #K1+1,%5 ;R5-123456-ODD ADDRESS
6214 022044 012706 021370 MOV #K5,%6 ;R6-050505--.EVEN ADDRESS
6215 022050 112625 MOV#B (6)+,(5)+ ;LOW OF R6 TO HIGH OF R5
6216 022052 022767 042456 177300 CMP #042456,K1
6217 022060 001401 BEQ TS310
6218 022062 104000 EMT ;FAILED LOW OF 6 TO HIGH OF 5,OR WRONG $STMP
6219
6220 ;*****
6221 ;TEST 310 TEST BYTE OPERATION WITH SEQUENTIAL ODD-EVEN ADDRESS
6222 ;*****
6223 022064 126767 177304 177303 TS310: CMPB K7,K7+1 ;SAME .WORD LOW TO HIGH
6224 022072 001401 BEQ BR15
6225 022074 104000 EMT ;SHOULD COMPARE LOW TO HIGH
6226
6227 022076 126767 177273 177270 BR15: CMPB K7+1,K7 ;COMPARE ODD TO .EVEN SAME .WORD
6228 022104 001401 BEQ BR16
6229 022106 104000 EMT ;ODD TO .EVEN .B..E FAILURE
6230
6231 022110 126767 177263 177256 BR16: CMPB K10+1,K7 ;SEQUENTIAL .BYTES
6232 022116 001401 BEQ BR17
6233 022120 104000 EMT ;ODD TO .EVEN FAILED
6234
6235 022122 126767 177250 177242 BR17: CMPB K10,K6
6236 022130 001401 BEQ BR20
6237 022132 104000 EMT ;.EVEN TO EVEN FAILED
6238 022134 126767 177235 177235 BR20: CMPB K7+1,K10+1
6239 022142 001401 BEQ BR21
6240 022144 104000 EMT ;ODD TO ODD FAILED
6241
6242 022146 126767 177224 177223 BR21: CMPB K10,K10+1
6243 022154 001001 BNE BR22
6244 022156 104000 EMT ;LOW TO HIGH IN SAME .WORD FAILED
6245
6246 022160 126767 177213 177211 BR22: CMPB K10+1,K10+1
6247 022166 001401 BEQ BR23
```

6248 022170 104000
6249
6250 022172 126767 177200 177175
6251 022200 001001
6252 022202 104000
6253
6254
6255
6256
6257
6258 022204
6259 022204 012706 000150
6260 022210 012767 022222 155566
6261 022216 005746
6262 022220 104000
6263 022222
6264
6265
6266
6267
6268 022222
6269 022222 012706 000150
6270 022226 012767 022236 155550
6271 022234 005746
6272 022236 020627 000142
6273 022242 001401
6274 022244 104000
6275
6276
6277
6278
6279 022246
6280 022246 012706 000150
6281 022252 005067 155670
6282 022256 012767 022266 155520
6283 022264 005246
6284 022266 005767 155654
6285 022272 001001
6286 022274 104000
6287 022276 012705 001000
6288 022302 012706 000400
6289 022306 012767 022320 155470
6290 022314 124645
6291 022316 104000
6292 022320 012706 000400
6293 022324 012767 022336 155452
6294 022332 134546
6295 022334
6296 022334 104000
6297 022336
6298
6299
6300
6301
6302 022336
6303 022336 012706 000400

EMT ;HIGH TO LOW IN SAME .WORD FAILED
BR23: CMPB K10,K7+1
BNE TS311
EMT ;.EVEN TO ODD FAILED,OR WRONG \$STNM
:*****
:TEST 311 TEST THAT DECREMENT R6 TO A VALUE LESS THAN 400 TRAPS
:*****
TS311:
MOV #150,%6 ;R6 = 150
MOV #TDEC1,4 ;STACK OVERFLOW TRAP POINTER
TST -(6) ;WITH R6 = 150 SHOULD TRAP
EMT ;SHOULD HAVE TRAPPED,OR WRONG \$STNM
TDEC1:
:*****
:TEST 312 TEST FOR DECREMENT OF R6 ON OVERFLOW TRAP
:*****
TS312:
MOV #150,%6 ;R6 = 150
MOV #TDEC2,4 ;TRAP POINTER
TST -(6) ;WITH R6 = 150 SHOULD TRAP
TDEC2: CMP %6,#142 ;DID R6 DECREMENT
BEQ TS313
EMT ;R6 NOT = 142,OR WRONG \$STNM
:*****
:TEST 313 TEST DIFFERENT TYPES OF OVERFLOW
:*****
TS313:
MOV #150,%6
CLR 146 ;STATUS WORD OF LOC 10
MOV #TDEC3,4 ;RETURN TO LOC 4
INC -(6)
TDEC3: TST 146
BNE 1\$
EMT ;INCREMENT OPERATION NOT INHIBITED
1\$: MOV #1000,%5
MOV #400,%6
MOV #TDEC4,4
CMPB -(6),-(5)
EMT ;STACK = 400 AND DECREMENTED, SHOULD TRAP
TDEC4: MOV #400,%6
MOV #TDEC7,4
BITB -(5),-(6)
TDEC6: EMT ;NO STACK OVERFLOW,OR WRONG \$STNM
TDEC7:
:*****
:TEST 314 TEST THAT AN 77 CAUSES AN OVERFLOW TRAP
:*****
TS314: MOV #400,%6 ;SET UP STACK TO OVERFLOW

```
6304 022342 012767 022360 155440      MOV      #VDEC2,10      ;SET UP 77 VECTOR
6305 022350 012767 022364 155426      MOV      #VDEC,4      ;SET UP OVERFLOW VECTOR
6306 022356 000077              77              ;THIS TRAP SHOULD CAUSE OVERFLOW
6307 022360 000167 157544              VDEC2: JMP      ERROR1      ;USE JUMP TO GET TO ERROR BECAUSE UNSURE WHAT EMT WILL D
6308 022364 012767 021340 155416      VDEC:  MOV      #T010,10      ;RESTORE VECTOR
6309                                     ;*****
6310                                     ;TEST 315      TEST THAT AN IOT CAUSES AN OVERFLOW TRAP
6311                                     ;*****
6312 022372                                     TS315:
6313 022372 012706 000400              MOV      #400,%6      ;SET UP STACK TO OVERFLOW
6314 022376 012767 022414 155414      MOV      #VDEC4,20      ;SET UP IOT VECTOR
6315 022404 012767 022420 155372      MOV      #VDEC3,4      ;SET UP OVERFLOW VECTOR
6316 022412 000004              IOT              ;THIS TRAP SHOULD CAUSE OVERFLOW
6317 022414 000167 157510              VDEC4: JMP      ERROR1      ;USE JUMP TO GET TO ERROR BECAUSE UNSURE WHAT EMT WILL D
6318 022420 012767 021344 155372      VDEC3: MOV      #T020,20      ;RESTORE VECTOR
6319                                     ;*****
6320                                     ;TEST 316      TEST THAT AN EMT CAUSES AN OVERFLOW TRAP (CHECK OF YELLOW ZONE)
6321                                     ;*****
6322                                     TS316:
6323 022426                                     MOV      #400,%6      ;SET UP STACK TO OVERFLOW
6324 022426 012706 000400              MOV      #VDEC6,30      ;SET UP INST VECTOR
6325 022432 012767 022450 155370      MOV      #VDEC5,4      ;SET UP OVERFLOW VECTOR
6326 022440 012767 022454 155336      EMT              ;THIS TRAP SHOULD CAUSE OVERFLOW
6327 022446 104000              EMT              ;USE JUMP TO GET TO ERROR BECAUSE UNSURE WHAT EMT WILL D
6328 022450 000167 157454              VDEC6: JMP      ERROR1
6329 022454 012767 002130 155346      VDEC5: MOV      #ERROR1,30      ;RESTORE VECTOR
6330                                     ;*****
6331                                     ;TEST 317      TEST THAT AN TRAP CAUSES AN OVERFLOW TRAP
6332                                     ;*****
6333                                     TS317:
6334 022462                                     MOV      #400,%6      ;SET UP STACK TO OVERFLOW
6335 022462 012706 000400              MOV      #VDEC8,34      ;SET UP TRAP VECTOR
6336 022466 012767 022504 155340      MOV      #VDEC7,4      ;SET UP OVERFLOW VECTOR
6337 022474 012767 022510 155302      TRAP              ;THIS TRAP SHOULD CAUSE OVERFLOW
6338 022502 104400              TRAP              ;USE JUMP TO GET TO ERROR BECAUSE UNSURE WHAT EMT WILL D
6339 022504 000167 157420              VDEC8: JMP      ERROR1
6340 022510 012767 021350 155316      VDEC7: MOV      #T034,34      ;RESTORE VECTOR
6341                                     ;*****
6342                                     ;TEST 320      TEST THAT AN TRT CAUSES AN OVERFLOW TRAP
6343                                     ;*****
6344 022516                                     TS320:
6345 022516 012706 000400              MOV      #400,%6      ;SET UP STACK TO OVERFLOW
6346 022522 012767 022540 155264      MOV      #VDEC10,14      ;SET UP TRT VECTOR
6347 022530 012767 022544 155246      MOV      #VDEC9,4      ;SET UP OVERFLOW VECTOR
6348 022536 000003              TRT              ;THIS TRAP SHOULD CAUSE OVERFLOW
6349 022540 000167 157364              VDEC10: JMP      ERROR1      ;USE JUMP TO GET TO ERROR BECAUSE UNSURE WHAT EMT WILL D
6350 022544 012767 021342 155242      VDEC9:  MOV      #T014,14      ;RESTORE VECTOR
6351                                     ;*****
6352                                     ;TEST 321      TEST THAT AN ILLA CAUSES AN OVERFLOW TRAP
6353                                     ;*****
6354 022552                                     TS321:
6355 022552 012706 000400              MOV      #400,%6      ;SET UP STACK TO OVERFLOW
6356 022556 012767 022574 155220      MOV      #VDEC11,4      ;SET UP ILLA VECTOR
6357 022564 012767 022600 155212      MOV      #VDEC12,4      ;SET UP OVERFLOW VECTOR
6358 022572 004700              ILLA              ;THIS TRAP SHOULD CAUSE OVERFLOW
6359 022574 000167 157330              VDEC11: JMP      ERROR1      ;USE JUMP TO GET TO ERROR BECAUSE UNSURE WHAT EMT WILL D
```

```
6360 022600 012767 021336 155176 VDEC12: MOV #T04,4 ;RESTORE VECTOR
6361 022606 020627 000370 CMP #6,#370 ;STACK PUSHED FOUR WORDS?
6362 022612 001401 BEQ TS322
6363 022614 104000 EMT ;TRAP OVERFLOW DID NOT OCCUR
6364 ;*****
6365 ;TEST 322 TEST THAT AN ILLB CAUSES AN OVERFLOW TRAP
6366 ;*****
6367 TS322:
6368 022616 012706 000400 MOV #400,%6 ;SET UP STACK TO OVERFLOW
6369 022622 012767 022640 155154 MOV #VDEC13,4 ;SET UP ILLB VECTOR
6370 022630 012767 022644 155146 MOV #VDEC14,4 ;SET UP OVERFLOW VECTOR
6371 022636 000100 ILLB ;THIS TRAP SHOULD CAUSE OVERFLOW
6372 022640 000167 157264 VDEC13: JMP ERROR1 ;USE JUMP TO GET TO ERROR BECAUSE UNSURE WHAT EMT WILL D
6373 022644 012767 021336 155132 VDEC14: MOV #T04,4 ;RESTORE VECTOR
6374 ;*****
6375 ;TEST 323 TEST FOR FALSE OVERFLOW TRAP
6376 ;*****
6377 TS323:
6378 022652
6379
6380 022652 012767 022720 155124 MOV #FOVER,4 ;SET UP OVERFLOW POINTER
6381 022660 012706 001002 MOV #1002,%6
6382 022664 005746 TST -(6) ;SHOULD NOT OVERFLOW
6383 022666 012706 002002 MOV #2002,%6
6384 022672 005746 TST -(6) ;SHOULD NOT OVERFLOW
6385 022674 012706 004002 MOV #4002,%6
6386 022700 005746 TST -(6) ;SHOULD NOT OVERFLOW
6387 022702 012706 010002 MOV #10002,%6
6388 022706 005746 TST -(6)
6389 022710 012706 020000 MOV #20000,%6 ;SHOULD NOT OVERFLOW
6390 022714 005746 TST -(6)
6391 022716 000401 BR STP
6392 022720
6393 022720 104000 FOVER: EMT ;IT OVERFLOWED,OR WRONG $STNM
6394 022722 012767 021336 155054 STP: MOV #T04,4
6395 022730 005067 155052 CLR 6
6396 ;*****
6397 ;TEST 324 TEST THAT BIT 4 PSW WILL CAUSE A TRAP TO 14
6398 ;*****
6399 TS324:
6400 022734 012706 001000 MOV #STBOT,SP
6401 022740 012767 022764 155046 MOV #RETAT,RTRAP4 ;SET UP TO TRAP TO 14
6402 022746 012746 000020 MOV #20,-(SP) ;PUSH T BIT
6403 022752 012746 022760 MOV #.+6,-(SP) ;PUSH PC
6404 022756 000002 RTI ;SET T BIT
6405 022760 000240 NOP ;TRAP HERE
6406 022762 104000 EMT ;TRACE BIT DID NOT TRAP!,OR WRONG $STSN
6407 022764
6408 RETAT:
6409 ;*****
6410 ;TEST 325 TEST STACK POINTER DECREMENTS
6411 ;*****
6412 TS325:
6413 022764 012706 001000 MOV #STBOT,SP
6414 022770 012767 023014 155016 MOV #RETBT,RTRAP4
6415 022776 012746 000020 MOV #20,-(SP) ;PUSH T BIT
6415 023002 012746 023010 MOV #.+6,-(SP) ;PUSH PC
```



```
6416 023006 000002          RTI          ;SET T BIT
6417 023010 000240          NOP          ;TRAP HERE
6418 023012 104000          EMT         ;TRACE BIT DID NOT TRAP!
6419 023014 020627 000774 RETBT: CMP      SP,#STBOT-4
6420 023020 001401          BEQ      TS326
6421 023022 104000          EMT         ;STACK POINTER WAS NOT PUSHED BY TRAP,OR WRONG $TESTM
6422          ;*****
6423          ;TEST 326      TEST FOR PROPER PC ON STACK
6424          ;*****
6425          TS326:
6426 023024 012706 001000          MOV      #STBOT,SP
6427 023030 012767 023050 154756          MOV      #RETCT,RTRAP4
6428 023036 012746 000020          MOV      #20,-(SP)      ;PUSH T BIT
6429 023042 012746 023050          MOV      #.+6,-(SP)     ;PUSH PC
6430 023046 000002          RTI         ;SET T BIT
6431          ;TRAP HERE
6432 023050 022767 023050 155716 RETCT: CMP      #.STBOT-4
6433 023056 001401          BEQ      TS327
6434 023060 104000          EMT         ;CORRECT PC WAS NOT SAVED ON STACK,OR WRONG $TESTM
6435          ;*****
6436          ;TEST 327      TEST THAT RTT POPS T- BIT
6437          ;*****
6438          TS327:
6439          ;*****
6440 023062          ;*****
6441          ;*****
6442 023062 012706 001000          MOV      #STBOT,SP
6443 023066 005001          CLR      R1      ;CLEAR R1
6444 023070 012746 000020          MOV      #20,-(SP)
6445 023074 012746 023110          MOV      #RTT1,-(SP)
6446 023100 012767 023116 154706          MOV      #RTT2,14
6447 023106 000006          RTT
6448 023110 000240          RTT1: NOP
6449 023112 001401          BEQ      TS330
6450 023114 104000          EMT         ;T-BIT DID NOT TRAP,OR WRONG $TESTM
6451          ;*****
6452 023116          RTT2:
6453          ;*****
6454          ;TEST 330      TEST THAT RTT ALLOWS ONE INST. BEFORE TRAP
6455          ;*****
6456          TS330:
6457 023116 012705 177777          MOV      #177777,%5
6458 023122 012706 001000          RTT5: MOV      #STBOT,SP
6459 023126 012746 000020          MOV      #20,-(SP)
6460 023132 012746 023150          MOV      #RTT3,-(SP)
6461 023136 012767 023160 154650          MOV      #RTT4,14
6462 023144 005001          CLR      R1      ;CLEAR R0
6463 023146 000006          RTT         ;SET T-BIT
6464 023150 005201          RTT3: INC      R1
6465 023152 005205          INC      %5
6466 023154 001762          BEQ      RTT5      ;DO THIS TEST NO MORE THAN 2 TIMES
6467 023156 104000          EMT         ;DID NOT TRAP
6468 023160 005301          RTT4: DEC      R1      ;SEE IF RTT ALLOWS 1 INST.
6469 023162 001403          BEQ      RTT6
6470 023164 005205          INC      %5
6471 023166 001755          BEQ      RTT5      ;DO THIS TEST NO MORE THAN TWO TIMES
```

6472 023170 104000
6473 023172
6474
6475
6476
6477 023172
6478 023172 012706 001000
6479 023176 012746 000020
6480 023202 012746 023220
6481 023206 012767 023224 154600
6482 023214 005001
6483 023216 000002
6484 023220 005201
6485 023222 104000
6486 023224 005701
6487
6488 023226 001401
6489 023230 104000
6490
6491
6492
6493
6494 023232
6495
6496 023232 012706 001000
6497 023236 012767 023276 154550
6498 023244 005027 000016
6499 023250 005027 000022
6500 023254 012767 023302 154536
6501 023262 012746 000020
6502 023266 012746 023274
6503 023272 000006
6504 023274 000004
6505 023276
6506 023276 104000
6507 023300
6508 023300 104000
6509 023302 012767 000016 154504
6510 023310 012767 000022 154502

EMT ;RTT DID NOT ALLOW 1 INST.,OR WRONG \$TESTM
RTT6:
:*****
:TEST 331 TEST THAT RTI DOES NOT ALLOW 1 INST.
:*****
TS331:
MOV #STBOT,SP
MOV #20,-(SP)
MOV #RTI1,-(SP)
MOV #RTI2,14
CLR R1
RTI ;SET T-BIT
RTI1: INC R1 ;RTI SHOULD NOT ALLOW THIS
EMT ;T- BIT DID NOT CAUSE TRAP
RTI2: TST R1
;RTI SHOULD NOT ALLOW 1 INST. BEFORE TRAP
BEQ TS332
EMT ;RTI DID ALLOW 1 INST. BEFORE TRAP,OR WRONG \$TESTM
:*****
:TEST 332 TEST TRAP ON TRAP THAT TRACE BIT TRAPS ARE INHIBITED ON TRAP INST
:*****
TS332:
MOV #STBOT,%6
MOV #TRACE,14 ;TRACE TRAP
CLR #16
CLR #22
MOV #TONT1,20 ;IOT TRAP
MOV #20,-(SP) ;PUSH T BIT
MOV #.+6,-(SP) ;PUSH PC
RTT
IOT ;TRAP, NEW CC HAVE TRACE RESET
TRACE: EMT ;TRACE TRAP WAS NOT INHIBITED
BR70:
EMT ;WRONG TSTNM,OR WRONG \$STNM
TONT1: MOV #16,14
MOV #22,20

CJKDJB0 11/23-B CPU CLUSTER DIAG.
CJKDJB.P11 26-MAY-82 11:14

G 10

DNMAC X24.07-563 26-MAY-82 11:18 PAGE 124
T332 TEST TRAP ON TRAP THAT TRACE BIT TRAPS ARE INHIBITED ON TRAP INST

SE

6511
6512
6513
6514 023316
6515 023316 012706 001000
6516 023322 012767 023346 154464
6517 023330 005067 154462
6518 023334 012746 000020
6519 023340 012746 023346
6520 023344 000002
6521 023346 036727 155424 000020 TRC1:
6522 023354 001001
6523 023356
6524 023356 104000
6525 023360 012767 021342 154426 STP3D:
6526

```
*****  
:TEST 333 TEST THAT THE TRACE BIT IS SAVED IN THE STACK  
*****  
TS333:  
MOV #STBOT,%6 ;SET UP STACK POINTER  
MOV #TRC1,14 ;TRACE TRAP RETURN  
CLR 16  
MOV #20,-(SP) ;SET THE T BIT  
MOV #TRC1,-(SP)  
RTI  
BIT STBOT-2,#20 ;CHECK FOR T BIT ON STACK  
BNE STP3D  
EMT  
MOV #T014,14 ;T BIT NOT SAVED ON THE STACK,OR WRONG $TSTMM
```

```

6527 :*****
6528 :THIS ROUTINE TESTS THAT NO LEGAL ADDRESS TRAPS AND THAT AN ILLEGAL
6529 :ADDRESS TRAPS TO LOCATION 4. THIS WILL RUN ON 30K SYSTEM. BUT IF
6530 :SWITCH REGISTER BIT 1=0, THEN THE MEMORY FROM 28K-30K IS NOT LOOKED
6531 :AT, SINCE IT MAY HAVE I/O DEVICES. IF SWR BIT 1=1, THEN THAT AREA IS
6532 :CHECKED. (IT SHOULD EITHER ALL TRAP OR ALL NOT TRAP). LOC 160000
6533 :IS NO LONGER GUARANTEED TO TRAP, SINCE IT MAY CONTAIN MEMORY. LOCATION
6534 :177700 (THE UNIBUS ADDRESS FOR RO ON OLDER SYSTEMS) IS USED FOR FORCING
6535 :A TIMEOUT IN THE EVENT THAT THERE WAS NO TIMEOUT FROM 0K-28K OR 30K.
6536 :THIS ROUTINE TESTS MEMORY UNTIL IT DOES A NXM STOP
6537 :*****
6538 :TEST 334 TEST NON-EXISTENT ADDRESS TRAPS
6539 :*****
6540 TS334:
6541 023366 005000 1$: CLR R0 ;
6542 023370 005067 154412 CLR 6 ;
6543 023374 012767 023460 154402 MOV #ATRAP,4 ;SET UP ADDRESS TRAP ENTRANCE
6544 023402 012706 001000 MOV #STBOT,SP ;SET STACK POINTER
6545 023406 105720 NOR: TSTB (0)+ ;IF OUTSIDE OF CORE, TRAP TO 4
6546 023410 020027 CMP RO,(PC)+ ;IS POINTER INSIDE 28K (30K) CORE
6547 023412 160000 HICORE: .WORD 160000 ;MAY BE CHANGED TO 170000 IF 30K
6548 023414 103774 BLO NOR ;TEST THE REST OF CORE
6549 023416 012737 023432 000004 MOV #ROTRAP,@#4 ;SET UP NEW VECTOR POINTER
6550 023424 105737 177700 TSTB @#177700 ;SHOULD CAUSE A TRAP
6551 023430 TRPADR:
6552 023430 104000 EMT ;SHOULD HAVE TRAPED
6553 ;TRAP TO HERE IF FORCING TRAP BY TESTING 177700
6554 023432 106767 154340 ROTRAP: MFPS STATUS ;
6555 023436 005767 154334 TST STATUS ;TEST PSW
6556 023442 001401 BEQ 1$ ;
6557 023444 104000 EMT ;NEW PSW SHOULD HAVE BEEN ZERO
6558 023446 026727 155322 023430 1$: CMP STBOT-4,#TRPADR ;TEST OLD PC AT STACK
6559 023454 001437 BEQ TRAPB ;
6560 023456 104000 EMT ;OLD PC WAS NOT SAVED
6561 ;RETURN HERE ON AN ADDRESS TRAP FROM MEMORY BELOW 28K (OR 30K)
6562 023460 005300 ATRAP: DEC R0 ;
6563 023462 010067 000032 MOV RO,CORH ;MOVE THE FIRST NXM LOCATION IN CORH
6564 ;THIS ROUTINE DOES NXM TRAPS UNTIL IT FINDS AN EXISTENT MEMORY LOCATION
6565 023466 013700 023412 MOV @#HICORE,RO ;SET UP THE HIGHEST MEM LOCATION
6566 023472 005300 DEC RO ;MAKE 1 LESS THAN THE HIGHEST CORE BOUNDARY
6567 023474 000402 BR NOSUB ;DON'T SUBTRACT 1K FIRST TIME
6568 023476 162700 001000 CTRAP: SUB #1000,RO ;SUBTRACT 1K OCTAL BYTE FROM ADDRESS
6569 ;TO SPEED UP TESTING
6570 023502 012767 023526 154274 NOSUB: MOV #BTRAP,4 ;SET UP THE VECTOR
6571 023510 012706 001000 MOV #STBOT,SP ;
6572 023514 005710 TST (RO) ;DOES THIS MEMORY EXIST?
6573 ;IF NXM, TRAP TO BTRAP
6574 023516 020027 DTRAP1: CMP RO,(PC)+ ;IF EXISTS, IS THIS THE SAME TRAP THAT CAUSED
6575 ;TRAP TO ATRAP
6576 023520 000000 CORH: .WORD 0 ;
6577 023522 101414 BLOS TRAPB ;CONTENTS OF RO SHOULD BE LESS THAN OR EQUAL TO CORH
6578 023524 104000 EMT ;IF THIS COMPARISON FAILS IT MEANS
6579 ;THAT SOME LEGAL ADDRESS TRAPPED, OR
6580 ;THAT AN ILLEGAL ADDRESS DID NOT TRAP
6581
6582 023526 106767 154244 BTRAP: MFPS STATUS

```

```
6583 023532 005767 154240          TST      STATUS
6584 023536 001401          BEQ      1$
6585 023540 104000          EMT
6586 023542 026727 155226 023516 1$:  CMP      STBOT-4, #DTRAP1 ;NEW PSW SHOULD HAVE BEEN ZERO
6587 023550 001752          BEQ      CTRAP ;CHECK IF TRAP PC IS OK
6588 023552          AUTO1:
6589 023552 104000          EMT ;OLD PC WAS NOT SAVED OR WRONG $TESTN
6590 023554 012767 021336 154222 TRAPB: MOV      #T04,4 ;RESET TRAP CATCHER
6591 023562 005067 154220          CLR      6 ;RESET TRAP CATCHER
6592
6593
6594 ;THIS ROUTINE WILL FIGURE OUT IF YOU HAVE A DL11W
6595 023566 012706 001000          MOV      #STBOT, SP ;SET UP THE STACK POINTER
6596 023572 012767 023606 154204          MOV      #NODL,4 ;SET UP THE TRAP VECTOR
6597 023600 005767 153760          TST      TTCSR ;TEST THE PUNCH STATUS REGISTER
6598 023604 000405          BR       DL11W
6599 023606 012767 021336 154170 NODL: MOV      #T04,4
6600 023614 000167 101036          JMP      SLU1ST ;IF NO SLU FIND OUT WHY IN SLU TEST
6601 023620 012767 021336 154156 DL11W: MOV      #T04,4
6602
6603 ;*****
6604 ;TEST 335 TEST THAT A TTY INTERRUPT CAUSES AN OVERFLOW TRAP
6605 ;*****
6606 TS335:
6607 023626 012767 000340 154142          MOV      #340, STATUS ;LOCK OUT INTERRUPT
6608 023634 012706 000400          MOV      #400, %6 ;SET UP STACK TO OVERFLOW
6609 023640 012767 023702 154136          MOV      #TDEC77,4 ;SET UP OVERFLOW TRAP
6610 023646 016767 154212 001542          MOV      64, TEMP1 ;SAVE CONTENTS OF INTERRUPT VECTOR
6611 023654 012767 023700 154202          MOV      #TDEC8,64 ;SET UP INTERRUPT VECTOR
6612 023662 012767 000100 153674          MOV      #100, TTCSR ;SET INTERRUPT ENABLE
6613 023670 005067 154102          CLR      STATUS ;ALLOW INTERRUPT TO OCCUR
6614 023674 000167 100756          JMP      SLU1ST ;NO INTERRUPT OCCURRED SO GO TO SLU TEST
6615 ;TO FIND OUT WHY ADD REPORT PROPER ERROR
6616 TDEC8:
6617 023700          EMT ;OVERFLOW TRAP DID NOT OCCUR
6618 023702 005067 153656          CLR      TTCSR ;CLEAR INTERRUPT ENABLE
6619 023706 012767 021336 154070          MOV      #T04,4
6620 023714 005067 154066          CLR      6
6621 023720 016767 001472 154136          MOV      TEMP1, 64 ;RESTORE CONTENTS OF INTERRUPT VECTOR
6622 ;*****
6623 ;TEST 336 TEST THAT A PENDING INTERRUPT OCCURS BEFORE TRAP
6624 ;*****
6625 TS336:
6626 023726          MOV      #STBOT, %6
6627 023732 012767 000340 154036          MOV      #340, STATUS ;SET TO A HIGH PRIORITY LEVEL
6628 023740 016767 154120 001450          MOV      64, TEMP1 ;SAVE CONTENTS OF INTERRUPT VECTOR
6629 023746 012767 024012 154110          MOV      #TRO,64
6630 023754 012767 000100 153602          MOV      #100, TTCSR ;INTERRUPT FOR TTY PUNCH/PRINTER
6631 023762 012767 024014 154044          MOV      #BR71,34 ;TRAP VECTOR
6632 023770 012767 024016 154066          MOV      #TR2,64 ;TTY VECTOR
6633 023776 012767 000340 154032          MOV      #340,36 ;IF TRAP TRAPS, MOVE 340 TO PRIORITY
6634 024004 005067 153766          CLR      STATUS ;SHOULD INTERRUPT AT END OF CLR INST
6635 024010 104400          TRAP ;TTY INTERRUPT SHOULD OVERRIDE TRAP
6636 024012
6637 024012 104000          TRO: EMT ;TTY SHOULDN'T HAVE INTERRUPTED
6638 024014          BR71:
```

6639 024014 104000
6640 024016 005067 154014
6641 024022 016767 001370 154034
6642 024030 042767 000100 153526
6643
6644
6645
6646 024036
6647 024036 012706 001000
6648 024042 012767 000340 153726
6649 024050 012767 000100 153506
6650 024056 012767 024124 153750
6651 024064 016767 153774 001324
6652 024072 012767 024130 153764
6653 024100 012767 000340 153760
6654 024106 012767 024126 153704
6655 024114 012767 000340 153700
6656 024122 104400
6657 024124 000004
6658
6659 024126 104000
6660 024130 005067 153666
6661 024134 005067 153726
6662 024140 012767 021350 153666
6663 024146 016767 001244 153710
6664 024154 012767 000022 153636
6665 024162 042767 000100 153374
6666
6667
6668
6669
6670 024170
6671 024170 032737 000001 001020
6672 024176 001403
6673 024200 005737 001006
6674 024204 001013
6675 024206
6676 024206 016700 153350
6677 024212 012767 000100 153340
6678 024220 000005
6679 024222 032767 000100 153330
6680 024230 001401
6681 024232 104000
6682 024234
6683
6684
6685
6686 024234
6687 024234 032737 000001 001020
6688 024242 001403
6689 024244 005737 001006
6690 024250 001024
6691 024252
6692 024252 012706 001000
6693 024256 012767 024304 153530
6694 024264 012746 000020

EMT ;TRAP OCCURRED FIRST
TR2: CLR 36
MOV TEMP1, 64 ;RESTORE CONTENTS OF INTERRUPT VECTOR
BIC #100,ITCSR
:*****
:TEST 337 TEST THAT A PENDING INTERRUPT, INTERRUPTS BETWEEN TRAPS
:*****
TS337:
MOV #STBOT,%6
MOV #340,STATUS
MOV #100,ITCSR
MOV #TR3,34 ;TRAP
MOV 64, TEMP1 ;SAVE CONTENTS OF INTERRUPT VECTOR
MOV #TR4,64 ;TTY OUTPUT
MOV #340,66 ;TTY OUTPUT PRIORITY
MOV #TR5,20 ;IOT
MOV #340,22 ;IOT PRIORITY
TRAP ;THE ACT OF TRAPPING LOWER PRIORITY
TR3: IOT ;INTERRUPT SHOULD OCCUR IN PLACE OF IOT TRAP
TR5:
EMT ;NO INTERRUPT BETWEEN TRAPS,OR WRONG \$STNM
TR4: CLR 22 ;CLR IOT PRIORITY
CLR 66
MOV #T034,34
MOV TEMP1, 64 ;RESTORE CONTENTS OF INTERRUPT VECTOR
MOV #22,20
BIC #100,ITCSR ;CLEAR IE BIT IN SLU1 XMIT CSR
:*****
:TEST 340 TEST THAT 'RESET' GOES TO OUTSIDE WORLD
:*****
TS340:
BIT #1, @#SENV ;ARE WE RUNNING UNDER APT
BEQ 70\$;IF NO THEN DO TEST
TST @#\$PASS ;IS THIS FIRST PASS
BNE TS341 ;IF NO THEN SHIP TO NEXT TEST
70\$:
MOV TKB,R0 ;MAKE SURE RECEIVER DONE IS CLEAR
MOV #100,TRCSR ;SET INTERRUPT ENABLE
RESET ;SHOULD CLEAR INTERRUPT ENABLE
BIT #100,TRCSR ;TEST FOR CLEAR
BEQ TS341
EMT ;RESET FAILED TO CLEAR TRCSR,OR WRONG \$STNM
NODL2:
:*****
:TEST 341 TEST THAT RESET HAS NO EFFECT ON THE TRACE TRAP
:*****
TS341:
BIT #1, @#SENV ;ARE WE RUNNING UNDER APT
BEQ 70\$;IF NO THEN DO TEST
TST @#\$PASS ;IS THIS FIRST PASS
BNE TS342 ;IF NO THEN SHIP TO NEXT TEST
70\$:
MOV #STBOT,%6 ;SET STACK
MOV #RESET,14 ;SET UP TRACE VECTOR
MOV #20,-(R6) ;SET THE T-BIT ON STACK

6695 024270 012746 024276
6696 024274 000006
6697 024276 000005
6698 024300 000005
6699 024302
6700 024302 104000
6701 024304 005067 153466
6702 024310 005067 153502
6703 024314 012767 021342 153472
6704 024322
6705
6706
6707
6708
6709 024322
6710 024322 122767 000001 154470
6711 024330 001003
6712 024332 005767 154450
6713 024336 001051
6714 024340
6715 024340 042767 000100 153216
6716 024346 012706 001000
6717 024352 016767 153506 000136
6718 024360 012767 024440 153476
6719 024366 005067 153474
6720 024372 105767 153166
6721 024376 100375
6722 024400 012767 000015 153160
6723 024406 105767 153152
6724 024412 100375
6725 024414 012767 000015 153144
6726 024422 052767 000100 153134
6727 024430 005067 153342
6728 024434 000001
6729 024436 104000
6730 024440 005767 153332
6731 024444 001401
6732 024446 104000
6733 024450 026727 154320 024436
6734 024456 001401
6735 024460
6736 024460 104000
6737 024462 016767 000730 153374
6738 024470 042767 000100 153066
6739
6740
6741
6742 024476
6743
6744
6745
6746
6747 024476 012706 001000
6748 024502 012767 024516 153274
6749 024510 005237 177700
6750 024514 104000

MOV #1\$,-(R6) ;MOVE NEW PC ON STACK
RTT
1\$: RESET ;SHOULD HAVE NO EFFECT
RESET RESET ;NO EFFECT
RESET3:
EMT ;TRACE TRAP FAILED,OR WRONG \$STNM
RESET2: CLR STATUS ;CLEAR TRACK
CLR 16 ;TRACE STATUS
MOV #T014,14
SKTST2:
:*****
:TEST 342 TEST THE 'WAIT' INSTRUCTION
:*****
TS342:
CMPB #APTENV,\$ENV ;RUNING IN APT MODE?
BNE 1\$;IF NOT, DO THIS TEST
TST \$PASS ;IS THIS THE FIRST PASS?
BNE STP4E ;IF NOT FIRST PASS, SKIP TEST
1\$:
BIC #100,TTCSR ;CLEAR INTERRUPT ENABLE
MOV #STBOT,SP ;SET UP THE STACK
MOV 64,TEMP1 ;SAVE CONTENTS OF INTERRUPT VECTOR
MOV #WATE,64 ;SET UP THE INTERRUPT VECTOR
CLR 66
WATE1: TSTB TTCSR ;WAIT FOR READY
BPL WATE1 ;TO BE UP
MOV #15,TPB ;DO A CARRIAGE RETURN
WATE2: TSTB TTCSR ;WAIT FOR READY TO COME UP
BPL WATE2
MOV #15,TPB ;DO ANOTHER CARRIAGE RETURN
BIS #100,TTCSR ;SET THE INTERRUPT ENABLE
CLR STATUS ;CLEAR THE PSW
WATE3: WAIT ;WAIT FOR THE INTERRUPT
EMT ;WAIT INSTRUCTION DID NOT LOOP
WATE: TST STATUS ;IS THE PSW CORRECT?
BEQ 1\$
1\$: EMT ;NEW PSW SHOULD HAVE BEEN ZERO
CMP STBOT-4,#WATE3+2 ;IS THE OLD PC SAVED
BEQ STP4:
EMT ;OLD PC WAS NOT SAVED OR WRONG \$STNM
STP4E: MOV TEMP1,64 ;RESTORE CONTENTS OF INTERRUPT VECTOR
BIC #100,TTCSR ;CLEAR IE BIT IN SLU1 XMIT CSR
:*****
:TEST 343 TEST THAT USING REGISTER ADDR (177700) CAUSES TIME OUT.
:*****
TS343:
:REGISTER ADDRESS (177700-177717) CAUSE TIME OUT WHEN USED
:AS PROGRAM ADDRESS BY THE CPU.
:
MOV #STBOT,SP ;SET STACK POINTER
MOV #RETR1,RTRAP5 ;SET TRAP RETURN ADDR
PCN1: INC @#177700 ;BAD ADDR REFERENCE, TRAP TO 4
EMT ;REFERENCING 177700 DID NOT CAUSE TIME OUT

6807 024650
6808 024650 104000
6809 024652 022737 024643 000000
6810 024660 001401
6811 024662 104000
6812 024664 022737 000340 000002
6813 024672 001401
6814 024674 104000
6815 024676 022706 000000
6816 024702 001401
6817 024704 104000
6818 024706 012706 001000
6819 024712 012767 021336 153064
6820 024720 012767 021340 153062

DBE2: EMT ; TRAP TO WRONG LOCATION
DBE1: CMP #DBE+2,@#0 ; OLD PC GOT SAVED?
BEQ DBE3
EMT ; OLD PC DID NOT GET SAVEDD
DBE3: CMP #340,@#2 ; CORRECT PS SAVED?
BEQ DBE4
EMT ; CORRECT PS DID NOT GET SAVE
DBE4: CMP #0,SP ; SP POINTS TO LOC 0?
BEQ DBE5
EMT ; SP IS NOT POINTING TO LOC 0
DBE5: MOV #STBOT,SP ; RESET SP
MOV #T04,4 ; RESET VECTOR 4
MOV #T010,10 ; RESET VECTOR 10

: THIS TEST WILL CHECK THE SERVICE ROUTINE FOR A CONTROL CHIP ERROR.
: THIS IS DONE BY EXECUTING INSTRUCTIONS WHICH JUMP TO NON-EXISTENT
: CONTROL-CHIP. THE TEST EXECUTES AN FIS INSTRUCTION WHICH
: IS ILLEGAL ON ALL PROCESSORS USING THE DCF11-A CHIP SET.
: A CTLERR TRAPS TO LOCATION 10.
: THE RESET LINE IS ALSO ASSERTED FOR 1 CYCLE.

: TEST 347 TEST CTLERR SERVICE ROUTINE

TS347:

6834 024726 012706 001000
6835 024732 012737 024752 000010
6836 024740 012737 000340 000012
6837 024746 075006
6838 024750 000000
6839 024752 012706 001000
6840 024756 012767 021340 153024

MOV #STBOT,R6 ; INIT STACK POINTER
MOV #1\$,@#10 ; SET UP RETURN ADDR FROM TRAP
MOV #340,@#12 ; SET TRAP PRIORITY=7
FADD R6 ; EXECUTE FIS INSTR..SHOULD CAUSE CTLERR
HALT ; DID NOT TRAP..CHECK CSEL LINE
1\$: MOV #STBOT,R6 ; RE-INIT STACK POINTER
MOV #T010,10 ; RESET VECTOR 10

: TEST 350 TEST THAT ALL RESERVED INSTRUCTIONS TRAP

TS350:

6845 024764
6846 024764 042767 000100 152572
6847
6848 024772 013767 000010 000042
6849 025000 012737 025044 000010
6850 025006 170127 000000
6851
6852 025012 013767 025366 000356
6853 025020 000411

BIC #100,TTCSR ; SET UP TO SEE IF
MOV @#10,TENSAVE ; THIS PROCESSOR HAS THE
MOV #TRAP10,@#10 ; FLOATING POINT OPTION
LDFPS #0 ; DO A FPP INSTRUCTION
; IF NO TRAP FPP INSTALLED
MOV @#FPP,FINISH ; SO RESET END OF TABLE POINTER
BR AROUND ; THE FOLLOWING

6854
6855 025022 042777 000040 174354
6856 025030 012716 025076
6857
6858 025034 000002
6859 025036 000000
6860 025040 000000
6861 025042 000000
6862 025044

;* IF NO CIS OPTION TRAP TO HERE
CISTRP: BIC #40,ASWR ; CLEAR CIS OPTION IN SWR
MOV #CONCIS,(SP) ; CHANGE RETURN ADDRESS TO CONCIS LOCATION
RTI ; RETURN
CISADR: .WORD 0 ; DATA FOR CIS
; INSTRUCTION
TENSAVE: .WORD 0 ; A PLACE TO STORE CONTENTS OF 10
TRAP10: 0 ; LEAVE THE TABLE ALONE

6863										
6864	025044					AROUND:	MOV	#246,@#244	:	CONTINUATION POINT
6865	025044	012737	000246	000244			MOV	#CISTRP,@#10	:	RESTORE THE TRAP VECTOR
6866	025052	012737	025022	000010			.WORD	076144	:	SET UP TO SEE IF THIS HAS THE CIS OPTION
6867	025060	076144					.WORD	CISADR	:	EXECUTE A CMPCI INSTRUCTION
6868	025062	025036					.WORD	CISADR	:	OPERANDS
6869	025064	025036					.WORD	0	:	FOR CIS
6870	025066	000000					BIS	#40,@SWR	:	INSTRUCTION
6871	025070	052777	000040	174306			MOV	TENSAVE,@#10	:	SET CIS PRESENT BIT
6872	025076	016737	177740	000010	CONCIS:		MOV	#TABLE,TAB	:	RESTORE THE ILLEGAL INST. VECTOR
6873	025104	012703	025256				MOV	(TAB)+,FIRST	:	TABLE POINTER
6874	025110	012305			GIN1:		MOV	(TAB)+,LAST	:	FIRST OR CURRENT INSTRUCTION
6875	025112	012301					CMP	FIRST,@#CIS	:	LAST INSTRUCTION OR GROUP
6876	025114	020537	025332				BNE	1\$		
6877	025120	001007					BIT	#40,@SWR		
6878	025122	032777	000040	174254			BEQ	1\$		
6879	025130	001403					MOV	#FPP,TAB		
6880	025132	012703	025366				BR	GIN1		
6881	025136	000764				1\$:	CMP	FIRST,FINISH	:	TESTED ALL
6882	025140	020567	000232				BEQ	GIN3	:	YES BRANCH
6883	025144	001415					MOV	FIRST,INST	:	SET UP INST
6884	025146	010567	000226			GIN2:	INC	INST		
6885	025152	005267	000222				MOV	#RET,10	:	SET UP RETURN FROM TRAP
6886	025156	012767	025212	152624			MOV	#STBOT,SP	:	SET UP STACK POINTER
6887	025164	012706	001000				CLR	CC	:	CLEAR PRIORITY
6888	025170	005067	152602				JMP	INST	:	EXECUTE RESERVED INSTRUCTION
6889	025174	000167	000200			GIN3:	MOV	#TO10,10	:	RESET VECTOR 10
6890	025200	012767	021340	152602			JMP	THRPR1	:	JUMP TO EIS TEST
6891	025206	000167	000252							
6892										
6893										
6894	025212	020627	000774							
6895	025216	001401				RET:	CMP	SP,#STBOT-4	:	TEST DECREMENT OF SP
6896	025220	104000					BEQ	RET1		
6897	025222	026727	153546	025402		RET1:	EMT		:	WRONG DECREMENT
6898	025230	001401					CMP	STBOT-4,#INST+2	:	LOC OF INST UNINCREMENTED
6899	025232	104000					BEQ	RET2		
6900	025234	005767	153536			RET2:	EMT		:	INST INC ON TRAP
6901	025240	001401					TST	STBOT-2		
6902	025242						BEQ	RET3		
6903	025242	104000				RET4:				
6904	025244	026701	000130				EMT		:	CONDITION CODES SET ON TRAP OR WRONG \$STNM
6905	025250	001717				RET3:	CMP	INST,LAST		
6906	025252	000167	177674				BEQ	GIN1	:	SET UP NEW GROUP
6907							JMP	GIN2	:	FINISH OLD GROUP
6908	025256	000007				TABLE:	7		:	END OF INSTRUCTION GROUP
6909	025260	000077					77		:	END OF OPERATE
6910	025262	000207					207			
6911	025264	000227					227		:	RTS,RT1,JMP
6912	025266	006777					6777			
6913	025270	007777					7777			
6914	025272	075037					075037			
6915	025274	076017					76017			
6916	025276	076032					76032			
6917	025300	076037					76037			
6918	025302	076045					76045			

6919 025304 076047
6920 025306 076077
6921 025310 076127
6922 025312 076132
6923 025314 076137
6924 025316 076145
6925 025320 076147
6926 025322 076157
6927 025324 076167
6928 025326 076177
6929 025330 076777
6930 025332 076017
6931 025334 076032
6932 025336 076037
6933 025340 076045
6934 025342 076047
6935 025344 076077
6936 025346 076127
6937 025350 076132
6938 025352 076137
6939 025354 076145
6940 025356 076147
6941 025360 076157
6942 025362 076167
6943 025364 076177
6944 025366 167777
6945 025370 177700
6946 025372 177716
6947 025374 177777
6948 025376 025376
6949 025400 000000
6950 025402 000000
6951 025404 000000
6952 025406 000000
6953 025410 000000
6954
6955
6956
6957 000000
6958 000051
6959 000176
6960
6961 025412
6962 025414
6963 025414 025416
6964 025416
6965 025416
6966 025420
6967 025420 025422
6968 025422
6969 025422
6970 025424
6971 025424
6972 025426
6973 025426 000000
6974 025430 000000

76047
76077
76127
76132
76137
76145
76147
76157
76167
76177
76777
CIS: 76017
76032
76037
76045
76047
76077
76127
76132
76137
76145
76147
76157
76167
76177
FPP: 167777
177700
177716
177777
FINISH: .
INST: HALT
HALT
HALT
HALT
HALT

; START OF THE FPP INSTRUCTIONS
;END FLAG
;WILL CONTINUE RESERVED INST
;SHOULD TRAP TO LOC 10
;LOC 10 SHOULD SEND YOU TO
;RET

.SBTTL ** STARTING OF EIS TEST **

DUMMY= 0
F= 51
N= 176
COUNT: . =COUNT+2
PSWORD: . =PSWORD+2
TEMP1: . =TEMP1+2
TEMP2: . =TEMP2+2
TEMP3: . =TEMP3+2
TEMP4: . =TEMP4+2
TEMP5: .WORD
TEMP6: .WORD

6975 025432 177771
6976 025434 025432
6977 025436 177772
6978 025440 177777
6979 025442 040000
6980 025444 025442
6981 025446 040000
6982 025450 177776
6983 025452 000002
6984 025454 025452
6985 025456 000002
6986 025460 177566
6987 025462 177564
6988
6989
6990
6991
6992
6993

S1: -7
S2: S1
S3: -6
S4: -1
S5: 40000
S6: S5
S7: 40000
S8: -2
S9: 2
S10: S9
S11: 2
\$TPB: 177566
\$TPS: 177564

6994 025464

THRPR:

6995
6996 025464 012705 001004
6997 025470 005037 025412
6998 025474 012715 000001
6999 025500 012706 001000
7000 025504 012737 000001 025416 2\$:
7001 025512 005037 025420
7002 025516 012737 000001 025422
7003 025524 005037 025424
7004 025530 106427 000000
7005
7006

```
MOV    #STESTN,R5      :MAKE R5 POINT TO WHERE TEST # IS SAVED
CLR    @COUNT         :CLEAR THE COUNTER
MOV    #1,(R5)         :INITIALIZE TEST NUMBER
MOV    #STBOT,SP       :** STACK AT STBOT **
MOV    #1,@TEMP1       :TEMP1=1
CLR    @TEMP2          :TEMP2=0
MOV    #1,@TEMP3       :TEMP3=1
CLR    @TEMP4          :TEMP4=0
MTPS   #0
```

```
7007
7008
7009
7010
7011
7012
7013
7014
7015
7016
7017
7018
7019
7020 025534 013700 025416 ASTART: MOV @#TEMP1,%0 ;LOAD R0 WITH THE CONTENTS OF TEMP1
7021 025540 032737 000001 001006 BIT #1,@#SPASS ;IS IT AN EVEN PASS ?
7022 025546 001004 2$ BNE 2$ ;IF NOT THEN GO TO 2$
7023 025550 013701 025420 MOV @#TEMP2,R1 ;OTHERWISE EXECUTE THE INSTRUCTION
7024 ;IN MODE 0 USING R1
7025 025554 072001 ASH R1,R0
7026 025556 000402 BR 4$
7027 025560 072067 177634 2$: ASH TEMP2,%0 ;SHIFT R0 BY THE NUMBER SPECIFIED BY TEMP2
7028 025564 106737 025414 4$: MFPS @#PSWORD ;SAVE PS
7029 025570 123737 025424 025414 CMPB @#TEMP4,@#PSWORD;IS THE PS = TEMP4 ?
7030 025576 001401 BEQ 11$
7031 025600 104000 EMT ;THE PS IS NOT EQUAL TO 0
7032 025602 005237 025412 11$: INC @#COUNT ;INCREMENT THE COUNTER
7033 025606 023700 025422 CMP @#TEMP3,%0 ;IS THE RESULT IN R0 EQUAL TO TEMP3?
7034 025612 001401 BEQ 12$
7035 025614 6$:
7036 025614 104000 EMT ;EITHER INCORRECT R0 OR INCORRECT SEQUENCE
7037 025616 021537 025412 12$: CMP (R5),@#COUNT ;IS THE TEST NUMBER EQUAL TO THE
7038 ;COUNTER?
7039 025622 001374 BNE 6$ ;IF NOT GO TO THE HLT ABOVE
7040 025624 005215 INC (R5)
7041 025626 021527 000037 CMP (R5),#37 ;HAS THE CONTENTS OF REGISTERS BEEN SHIFTED LEFT
7042 ;BY 14. AND RIGHT BY 14.?
7043 025632 002011 BGE 8$
7044 025634 005237 025420 INC @#TEMP2
7045 025640 006367 177556 ASL TEMP3 ;SHIFT TEMP3 LEFT.
7046 025644 021527 000020 CMP (R5),#20 ;HAS THE CONTENTS OF REGISTERS BEEN SHIFTED LEFT BY 14.?
7047 025650 001004 BNE REGR1
7048 025652 000167 000670 JMP NEGAT ;IF SO GO TO NEGAT AND INITIATE RIGHT SHIFT
7049 025656 004767 000712 8$: JSR PC,TST37 ;IF SO GO AND CONTINUE THE REST OF THE PROGRAM
7050 025662 013701 025416 REGR1: MOV @#TEMP1,%1 ;LOAD R1 WITH THE CONTENTS OF TEMP1
7051 025666 032737 000001 001006 BIT #1,@#SPASS ;IS IT AN EVEN PASS ?
7052 025674 001004 BNE 2$ ;IF NOT THEN GO TO 2$
7053 025676 013702 025420 MOV @#TEMP2,R2 ;OTHERWISE EXECUTE ASH INSTRUCTION IN MODE 0
7054 025702 072102 ASH R2,R1 ;USING R1
7055 025704 000402 BR 4$
7056 025706 072167 177506 2$: ASH TEMP2,%1 ;SHIFT R1 BY THE NUMBER SPECIFIED BY TEMP2
7057 025712 106737 025414 4$: MFPS @#PSWORD ;SAVE PS
7058 025716 123737 025424 025414 CMPB @#TEMP4,@#PSWORD;IS THE PS = TEMP4 ?
7059 025724 001401 BEQ 11$
7060 025726 104000 EMT ;THE PS IS NOT EQUAL TO 0
7061 025730 005237 025412 11$: INC @#COUNT ;INCREMENT THE COUNTER
7062 025734 023701 025422 CMP @#TEMP3,%1 ;IS THE RESULT IN R1 EQUAL TO TEMP3?
```

F 11

7063	025740	001401				
7064	025742					
7065	025742	104000				
7066	025744	021537	025412			
7067	025750	001374				
7068	025752	005215				
7069	025754	021527	000037			
7070						
7071	025760	002011				
7072	025762	005237	025420			
7073	025766	006367	177430			
7074	025772	021527	000020			
7075	025776	001004				
7076	026000	000167	000542			
7077	026004	004767	000564			
7078	026010	013702	025416			
7079	026014	032737	000001	001006		
7080	026022	001004				
7081	026024	013703	025420			
7082	026030	072203				
7083	026032	000402				
7084	026034	072267	177360			
7085	026040	106737	025414			
7086	026044	123737	025424	025414		
7087	026052	001401				
7088	026054	104000				
7089	026056	005237	025412			
7090	026062	023702	025422			
7091	026066	001401				
7092	026070					
7093	026070	104000				
7094	026072	021537	025412			
7095	026076	001374				
7096	026100	005215				
7097	026102	021527	000037			
7098						
7099	026106	002011				
7100	026110	005237	025420			
7101	026114	006367	177302			
7102	026120	021527	000020			
7103	026124	001004				
7104	026126	000167	000414			
7105	026132	004767	000436			
7106	026136	013703	025416			
7107	026142	032737	000001	001006		
7108	026150	001004				
7109	026152	013704	025420			
7110	026156	072304				
7111	026160	000402				
7112	026162	072367	177232			
7113	026166	106737	025414			
7114	026172	123737	025424	025414		
7115	026200	001401				
7116	026202	104000				
7117	026204	005237	025412			
7118	026210	023703	025422			

	6S:	BEQ	12S		
		EMT			: EITHER INCORRECT R1 OR INCORRECT SEQUENCE
	12S:	CMP	(R5),@#COUNT		: IS THE TEST NUMBER EQUAL TO THE COUNTER?
		BNE	6S		: IF NOT GO TO THE HLT ABOVE
		INC	(R5)		
		CMP	(R5),#37		: HAS THE CONTENTS OF REGISTERS BEEN SHIFTED LEFT
					: BY 14. AND RIGHT BY 14.?
		BGE	8S		
		INC	@#TEMP2		
		ASL	TEMP3		: SHIFT TEMP3 LEFT
		CMP	(R5),#20		: HAS THE CONTENTS OF REGISTERS BEEN SHIFTED LEFT BY 14.?
		BNE	REGR2		
		JMP	NEGAT		: IF SO GO TO NEGAT AND INITIATE RIGHT SHIFT
	8S:	JSR	PC,TST37		: IF SO GO AND CONTINUE THE REST OF THE PROGRAM
	REGR2:	MOV	@#TEMP1,%2		: LOAD R2 WITH THE CONTENTS OF TEMP1
		BIT	#1,@#SPASS		: IS IT AN EVEN PASS ?
		BNE	2S		: IF NOT THEN GO TO 2S
		MOV	@#TEMP2,R3		: OTHERWISE EXECUTE ASH INSTRUCTION IN MODE 0
		ASH	R3,R2		: USING R2
		BR	4S		
	2S:	ASH	TEMP2,%2		: SHIFT R2 BY THE NUMBER SPECIFIED BY TEMP2
	4S:	MFPS	@#PSWORD		: SAVE PS
		CMPB	@#TEMP4,@#PSWORD		: IS THE PS = TEMP4 ?
		BEQ	11S		
		EMT			: THE PS IS NOT EQUAL TO 0
	11S:	INC	@#COUNT		
		CMP	@#TEMP3,%2		: IS THE RESULT IN R2 EQUAL TO TEMP3?
		BEQ	12S		
	6S:	EMT			: EITHER INCORRECT R2 OR INCORRECT SEQUENCE
	12S:	CMP	(R5),@#COUNT		: IS THE TEST NUMBER EQUAL TO THE COUNTER?
		BNE	6S		: IF NOT GO TO THE HLT ABOVE
		INC	(R5)		
		CMP	(R5),#37		: HAS THE CONTENTS OF REGISTERS BEEN SHIFTED
					: LEFT BY 14, AND RIGHT BY 14.?
		BGE	8S		
		INC	@#TEMP2		
		ASL	TEMP3		: SHIFTED TEMP3 LEFT
		CMP	(R5),#20		: HAS THE CONTENTS OF REGISTERS BEEN SHIFTED LEFT BY 14.?
		BNE	REGR3		
		JMP	NEGAT		: IF SO GO TO NEGAT AND INITIATE RIGHT SHIFT
	8S:	JSR	PC,TST37		: IF SO GO AND CONTINUE THE REST OF THE PROGRAM
	REGR3:	MOV	@#TEMP1,%3		: LOAD R3 WITH THE CONTENTS OF TEMP1
		BIT	#1,@#SPASS		: IS IT AN EVEN PASS ?
		BNE	2S		: IF NOT THEN GO TO 2S
		MOV	@#TEMP2,R4		: OTHERWISE EXECUTE ASH INSTRUCTION IN MODE 0
		ASH	R4,R3		: USING R3
		BR	4S		
	2S:	ASH	TEMP2,%3		: SHIFT R3 BY THE NUMBER SPECIFIED BY TEMP2
	4S:	MFPS	@#PSWORD		: SAVE PS
		CMPB	@#TEMP4,@#PSWORD		: IS THE PS = TEMP4 ?
		BEQ	11S		
		EMT			: THE PS IS NOT EQUAL TO 0.
	11S:	INC	@#COUNT		
		CMP	@#TEMP3,%3		: IS THE RESULT IN R3 EQUAL TO TEMP3?

7119	026214	001401									
7120	026216				6\$:	BEQ	12\$				
7121	026216	104000				EMT					;EITHER INCORRECT R3 OR INCORRECT SEQUENCE
7122	026220	021537	025412		12\$:	CMP	(R5),@#COUNT				;IS THE TEST NUMBER EQUAL TO THE COUNTER?
7123	026224	001374				BNE	6\$;IF NOT GO TO THE HLT ABOVE
7124	026226	005215				INC	(R5)				
7125	026230	021527	000037			CMP	(R5),#37				;HAS THE CONTENTS OF REGISTERS BEEN SHIFTED
7126											;LEFT BY 14, AND RIGHT BY 14.?
7127	026234	002010				BGE	8\$				
7128	026236	005237	025420			INC	@#TEMP2				
7129	026242	006367	177154			ASL	TEMP3				;SHIFT TEMP3 LEFT?
7130	026246	021527	000020			CMP	(R5),#20				;HAS THE CONTENTS OF REGISTERS BEEN SHIFTED LEFT BY 14.?
7131	026252	001003				BNE	REGR4				
7132	026254	000534				BR	NEGAT				;IF SO GO TO NEGAT AND INITIATE RIGHT SHIFT
7133	026256	004767	000312		8\$:	JSR	PC,TST37				;IF SO GO AND CONTINUE THE REST OF THE PROGRAM
7134	026262	013704	025416		REGR4:	MOV	@#TEMP1,%4				;LOAD R4 WITH THE CONTENTS OF TEMP1
7135	026266	010501				MOV	R5,R1				;SAVE R5
7136	026270	032737	000001	001006		BIT	#1,@#SPASS				;IS IT AN EVEN PASS ?
7137	026276	001004				BNE	2\$;IF NOT THEN GO TO 2\$
7138	026300	013705	025420			MOV	@#TEMP2,R5				;OTHERWISE EXECUTE ASH INSTRUCTION IN MODE 0
7139	026304	072405				ASH	R5,R4				;USING R4
7140	026306	000402				BR	4\$				
7141	026310	072467	177104		2\$:	ASH	TEMP2,%4				;SHIFT R4 BY THE NUMBER SPECIFIED BY TEMP2
7142	026314	106737	025414		4\$:	MFPS	@#PSWORD				;SAVE PS
7143	026320	123737	025424	025414		CMPB	@#TEMP4,@#PSWORD				;IS PS = TEMP4 ?
7144	026326	001401				BEQ	11\$				
7145	026330	104000				EMT					;THE PS IS NOT EQUAL TO 0
7146	026332	005237	025412		11\$:	INC	@#COUNT				
7147	026336	023704	025422			CMP	@#TEMP3,%4				;IS THE RESULT IN R4 EQUAL TO TEMP3?
7148	026342	001401				BEQ	12\$				
7149	026344				6\$:						
7150	026344	104000				EMT					;EITHER INCORRECT R4 OR INCORRECT SEQUENCE
7151	026346	010105			12\$:	MOV	R1,R5				;RESTORE R5
7152	026350	021537	025412			CMP	(R5),@#COUNT				;IS THE TEST NUMBER EQUAL TO THE COUNTER?
7153	026354	001373				BNE	6\$;IF NOT GO TO THE HLT ABOVE
7154	026356	005215				INC	(R5)				
7155	026360	021527	000037			CMP	(R5),#37				;HAS THE CONTENTS OF REGISTERS BEEN
7156											;SHIFTED LEFT BY 14. AND RIGHT BY 14.?
7157	026364	002010				BGE	8\$				
7158	026366	005237	025420			INC	@#TEMP2				
7159	026372	006367	177024			ASL	TEMP3				;SHIFT TEMP3 LEFT
7160	026376	021527	000020			CMP	(R5),#20				;HAS THE CONTENTS OF REGISTER BEEN SHIFTED BY 14.?
7161	026402	001003				BNE	REGR5				
7162	026404	000460				BR	NEGAT				;IF SO GO TO NEGAT AND INITIATE RIGHT SHIFT
7163	026406	004767	000162		8\$:	JSR	PC,TST37				;IF SO GO AND CONTINUE THE REST OF THE PROGRAM
7164	026412	010501			REGR5:	MOV	R5,R1				;SAVE R5
7165	026414	013705	025416			MOV	@#TEMP1,%5				;LOAD R5 WITH THE CONTENTS OF TEMP1
7166	026420	032737	000001	001006		BIT	#1,@#SPASS				;IS IT AN EVEN PASS ?
7167	026426	001004				BNE	2\$;IF NOT THEN GO TO 2\$
7168	026430	013700	025420			MOV	@#TEMP2,R0				;OTHERWISE EXECUTE ASH INSTRUCTION IN MODE 0
7169	026434	072500				ASH	R0,R5				;USING R5
7170	026436	000402				BR	4\$				
7171	026440	072567	176754		2\$:	ASH	TEMP2,%5				;SHIFT R5 BY THE NUMBER SPECIFIED BY TEMP2
7172	026444	106737	025414		4\$:	MFPS	@#PSWORD				;SAVE PS
7173	026450	123737	025424	025414		CMPB	@#TEMP4,@#PSWORD				;IS PS = TEMP4 ?
7174	026456	001401				BEQ	11\$				

Address	OpCode	Operand 1	Operand 2	Operand 3	Instruction	Comment
7175	026460	104000			EMT	:THE PS IS NOT EQUAL TO 0.
7176	026462	005237	025412		11\$: INC @#COUNT	
7177	026466	023705	025422		CMP @#TEMP3,%5	:IS THE RESULT IN R5 EQUAL TO TEMP3?
7178	026472	001401			BEQ 12\$	
7179	026474				6\$: EMT	:EITHER INCORRECT R5 OR INCORRECT SEQUENCE
7180	026474	104000			CMP (R1),@#COUNT	:IS THE TEST NUMBER EQUAL TO THE COUNTER?
7181	026476	021137	025412		BNE 6\$:IF NOT GO TO THE HLT ABOVE
7182	026502	001374			MOV R1,R5	:RESTORE R5
7183	026504	010105			INC (R5)	
7184	026506	005215			CMP (R5),#37	:HAS THE CONTENTS OF REGISTERS BEEN SHIFTED
7185	026510	021527	000037		BGE 8\$:LEFT BY 14. AND RIGHT BY 14.?
7186					INC @#TEMP2	:IF SO GO AND CONTINUE THE REST OF THE PROGRAM
7187	026514	002010			ASL TEMP3	:SHIFT TEMP3 LEFT
7188	026516	005237	025420		CMP (R5),#20	:HAS THE CONTENTS OF REGISTERS BEEN SHIFTED LEFT BY 14.?
7189	026522	006367	176674		BEQ NEGAT	:IF SO GO TO NEGAT AND INITIATE RIGHT SHIFT
7190	026526	021527	000020		BR 10\$	
7191	026532	001405			8\$: JSR PC,TST37	
7192	026534	000402			10\$: JMP ASTART	:GO BACK TO START
7193	026536	004767	000032		MOV #40000,@#TEMP1	:TEMP1=40000
7194	026542	000167	176766		MOV #177762,@#TEMP2	:TEMP2=1,762
7195	026546	012737	040000	025416	MOV #1,@#TEMP3	:TEMP3=1
7196	026554	012737	177762	025420	JMP ASTART	
7197	026562	012737	000001	025422	TST37: CMP (R5),#37	:IS IT TEST 37?
7198	026570	000167	176740		BNE TST40	:IF NOT THEN TRY TEST 40
7199	026574	021527	000037		CLR @#TEMP1	:0
7200	026600	001013			MOV #16,@#TEMP2	:SHIFTED BY 16
7201	026602	005037	025416		CLR @#TEMP3	:IS=0
7202	026606	012737	000020	025420	MOV #4,@#TEMP4	:AND PS=4
7203	026614	005037	025422		RTS PC	
7204	026620	012737	000004	025424	TST40: CMP (R5),#40	:IS IT TEST 40?
7205	026626	000207			BNE TST41	:IF NOT THEN TRY TEST 41
7206	026630	021527	000040		CLR @#TEMP2	:0 SHIFTED BY 0=0 AND PS=4
7207	026634	001003			RTS PC	
7208	026636	005037	025420		TST41: CMP (R5),#41	:IS IT TEST 41?
7209	026642	000207			BNE TST42	:IF NOT THEN TRY TEST 42
7210	026644	021527	000041		MOV #-16,@#TEMP2	:0 SHIFTED BY -16.=0 AND PS=4
7211	026650	001004			RTS PC	
7212	026652	012737	177760	025420	TST42: CMP (R5),#42	:IS IT TEST 42?
7213	026660	000207			BNE TST43	:IF NOT THEN TRY TEST 43
7214	026662	021527	000042		MOV #100000,@#TEMP1	:100000
7215	026666	001013			INC @#TEMP2	:SHIFTED BY -15
7216	026670	012737	100000	025416	DEC @#TEMP3	:IS=-1
7217	026676	005237	025420		MOV #10,@#TEMP4	:AND PS=10
7218	026702	005337	025422		RTS PC	
7219	026706	012737	000010	025424	TST43: CMP (R5),#43	:IS IT TEST 43?
7220	026714	000207			BNE TST44	:IF NOT THEN IF NOT THEN TRY TEST 44
7221	026716	021527	000043		MOV #125252,@#TEMP1	:125252
7222	026722	001012			MOV #-1,@#TEMP2	:SHIFTED BY -1
7223	026724	012737	125252	025416	MOV #152525,@#TEMP3	:IS=152525 AND PS=10
7224	026732	012737	177777	025420	RTS PC	
7225	026740	012737	152525	025422	TST44: CMP (R5),#44	:IS IT TEST 44?
7226	026746	000207			BNE TST45	:IF NOT THEN TRY TEST 45
7227	026750	021527	000044		MOV #1,@#TEMP2	:125252 SHIFTED BY 1
7228	026754	001012			MOV #52524,@#TEMP3	:IS=52524
7229	026756	012737	000001	025420		
7230	026764	012737	052524	025422		

```
7231 026772 012737 000003 025424      MOV      #3,@#TEMP4      ;AND PS=3
7232 027000 000207                      RTS      PC
7233 027002 021527 000045      TST45:  CMP      (R5),#45      ;IS IT TEST 45?
7234 027006 001012                      BNE     TST46      ;IF NOT THEN TRY TEST 46
7235 027010 012737 177776 025420      MOV      #-2,@#TEMP2     ;125252 SHIFTED BY -2
7236 027016 012737 165252 025422      MOV      #165252,@#TEMP3 ;IS=165252
7237 027024 012737 000011 025424      MOV      #11,@#TEMP4     ;AND PS=11
7238 027032 000207                      RTS      PC
7239 027034 021527 000046      TST46:  CMP      (R5),#46      ;IS IT TEST 46?
7240 027040 001014                      BNE     TST47      ;IF NOT THEN TRY TEST 47
7241 027042 012737 177777 025416      MOV      #-1,@#TEMP1     ;-1
7242 027050 012737 000020 025420      MOV      #16,@#TEMP2     ;SHIFTED BY 15.
7243 027056 005037 025422                      CLR     @#TEMP3         ;IS=0
7244 027062 012737 000007 025424      MOV      #7,@#TEMP4     ;AND PS=7
7245 027070 000207                      RTS      PC
7246 027072 021527 000047      TST47:  CMP      (R5),#47      ;IS IT TEST 47?
7247 027076 001011                      BNE     TST50      ;IF NOT THEN TRY TEST 50
7248 027100 005337 025420      DEC     @#TEMP2         ;-1 SHIFTED BY 15
7249 027104 012737 100000 025422      MOV      #100000,@#TEMP3 ;IS=100000
7250 027112 012737 000011 025424      MOV      #11,@#TEMP4     ;AND PS=11
7251 027120 000207                      RTS      PC
7252 027122 021527 000050      TST50:  CMP      (R5),#50      ;IS IT TEST 50
7253 027126 001007                      BNE     ENT51      ;IF NOT THEN TRY TEST 51
7254 027130 012737 137777 025416      MOV      #137777,@#TEMP1 ;137777 SHIFTED BY 15. IS=100000
7255 027136 012737 000013 025424      MOV      #13,@#TEMP4     ;AND PS=13
7256 027144 000207                      RTS      PC
7257 027146 021527 000051      ENT51:  CMP      (R5),#51      ;IS IT ENTERING TEST 51?
7258 027152 001401                      BEQ     1$
7259 027154 104000                      EMT
7260                      ;TEST NUMBER GOOFED
7261 027156 005726      1$:      TST      (SP)+      ;RESTORE STACK POINTER
7262 027160 012704 177771      MOV      #-7,%4
7263 027164 012702 025432      MOV      #S1,%2
7264 027170 012703 025434      MOV      #S2,%3
7265                      ;*****
7266                      ;TEST:51 11/34 ASH 125252 SHIFTED BY #5 = 52500 PS = 3
7267                      ;*****
7268
7269 027174 012701 125252      TST51:  MOV      #125252,%1     ;LOAD R1 WITH 125252
7270 027200 072127 000005      ASH     #5,%1          ;SHIFT R1 BY #5
7271 027204 106737 025414      MFPS   @#PSWORD       ;SAVE PS
7272 027210 122737 000003 025414      CMPB   #3,@#PSWORD     ;IS THE PS 3?
7273 027216 001401                      BEQ     11$
7274 027220 104000                      EMT
7275 027222 022701 052500      11$:    CMP      #52500,%1     ;THE PS IS NOT EQUAL TO 3
7276 027226 001401                      BEQ     12$           ;IS THE RESULT 52500?
7277 027230 104000                      EMT
7278 027232 005215      12$:    INC      (R5)       ;R1 IS NOT EQUAL TO 52500 OR INCORRECT SEQUENCE
7279
7280
7281
7282                      ;*****
7283                      ;TEST:52 11/34 ASH 125252 SHIFTED BY @S2 = 177525 PS = 10
7284                      ;*****
7285
7286 027234 012700 125252      TST52:  MOV      #125252,%0     ;LOAD R0 WITH 125252
```


7343 027416 022700 177525
7344 027422 001401
7345 027424 104000
7346 027426 005215

11\$: CMP #177525,%0 ;IS THE RESULT 177525?
BEQ 12\$
EMT ;RO IS NOT EQUAL TO 177525 OR INCORRECT SEQUENCE
12\$: INC (R5)

7347
7348
7349
7350
7351
7352
7353

:TEST:56 11/34 ASH 125252 SHIFTED BY -(2) = 177525 PS = 10

7354 027430 012700 125252
7355 027434 072042
7356 027436 106737 025414
7357 027442 122737 000010 025414
7358 027450 001401
7359 027452 104000
7360 027454 022700 177525
7361 027460 001401
7362 027462 104000
7363 027464 005215

TST56: MOV #125252,%0 ;LOAD RO WITH 125252
ASH -(2),%0 ;SHIFT RO BY -(2)
MFPS @#PSWORD ;SAVE PS
CMPB #10,@#PSWORD ;IS THE PS 10?
BEQ 11\$
EMT ;THE PS IS NOT EQUAL TO 10
11\$: CMP #177525,%0 ;IS THE RESULT 177525?
BEQ 12\$
EMT ;RO IS NOT EQUAL TO 177525 OR INCORRECT SEQUENCE
12\$: INC (R5)

7364
7365
7366
7367
7368
7369
7370

:TEST:57 11/34 ASH 125252 SHIFTED BY 2(3) = 177252 PS = 11

7371 027466 012700 125252
7372 027472 072063 000002
7373 027476 106737 025414
7374 027502 122737 000011 025414
7375 027510 001401
7376 027512 104000
7377 027514 022700 177252
7378 027520 001401
7379 027522 104000
7380 027524 005215

TST57: MOV #125252,%0 ;LOAD RO WITH 125252
ASH 2(3),%0 ;SHIFT RO BY 2(3)
MFPS @#PSWORD ;SAVE PS
CMPB #11,@#PSWORD ;IS THE PS 11?
BEQ 11\$
EMT ;THE PS IS NOT EQUAL TO 11
11\$: CMP #177252,%0 ;IS THE RESULT 177252?
BEQ 12\$
EMT ;RO IS NOT EQUAL TO 177252 OR INCORRECT SEQUENCE
12\$: INC (R5)

7381
7382
7383
7384
7385
7386
7387

:TEST:60 11/34 ASH 125252 SHIFTED BY @ (3) = 177525 PS = 10

7388 027526 012700 125252
7389 027532 072073 000000
7390 027536 106737 025414
7391 027542 122737 000010 025414
7392 027550 001401
7393 027552 104000
7394 027554 022700 177525
7395 027560 001401
7396 027562 104000
7397 027564 005215

TST60: MOV #125252,%0 ;LOAD RO WITH 125252
ASH @ (3),%0 ;SHIFT RO BY @ (3)
MFPS @#PSWORD ;SAVE PS
CMPB #10,@#PSWORD ;IS THE PS 10?
BEQ 11\$
EMT ;THE PS IS NOT EQUAL TO 10
11\$: CMP #177525,%0 ;IS THE RESULT 177525?
BEQ 12\$
EMT ;RO IS NOT EQUAL TO 177525 OR INCORRECT SEQUENCE
12\$: INC (R5)

7398

7399
7400
7401
7402
7403
7404
7405
7406
7407
7408
7409
7410
7411
7412
7413
7414
7415
7416
7417
7418
7419
7420
7421
7422
7423
7424
7425
7426
7427
7428
7429
7430
7431
7432
7433
7434

027566 012700 125252
027572 072033
027574 106737 025414
027600 122737 000010 025414
027606 001401
027610 104000
027612 022700 177525
027616 001401
027620 104000
027622 005215

027624 012700 125252
027630 072053
027632 106737 025414
027636 122737 000010 025414
027644 001401
027646 104000
027650 022700 177525
027654 001401
027656 104000
027660 005215

```
*****  
:TEST:61 11/34 ASH 125252 SHIFTED BY @ (3)+ = 177525 PS = 10  
*****  
TST61: MOV #125252,%0 ;LOAD R0 WITH 125252  
ASH @ (3)+,%0 ;SHIFT R0 BY @ (3)+  
MFPS @#PSWORD ;SAVE PS  
CMPB #10,@#PSWORD ;IS THE PS 10?  
BEQ 11$  
EMT ;THE PS IS NOT EQUAL TO 10  
11$: CMP #177525,%0 ;IS THE RESULT 177525?  
BEQ 12$  
EMT ;R0 IS NOT EQUAL TO 177525 OR INCORRECT SEQUENCE  
12$: INC (R5)  
  
*****  
:TEST:62 11/34 ASH 125252 SHIFTED BY @-(3) = 177525 PS = 10  
*****  
TST62: MOV #125252,%0 ;LOAD R0 WITH 125252  
ASH @-(3),%0 ;SHIFT R0 BY @-(3)  
MFPS @#PSWORD ;SAVE PS  
CMPB #10,@#PSWORD ;IS THE PS 10?  
BEQ 11$  
EMT ;THE PS IS NOT EQUAL TO 10  
11$: CMP #177525,%0 ;IS THE RESULT 177525?  
BEQ 12$  
EMT ;R0 IS NOT EQUAL TO 177525 OR INCORRECT SEQUENCE  
12$: INC (R5)
```

 : ASHC INSTRUCTION TESTS
 :*****

 : TESTS 63-157
 :*****

7435									
7436									
7437									
7438									
7439									
7440									
7441									
7442									
7443									
7444									
7445									
7446									
7447	027662	012737	000062	025412	MOV	#62,@#COUNT			
7448	027670	005037	025416		CLR	@#TEMP1	:TEMP1=0		
7449	027674	012737	000001	025420	MOV	#1,@#TEMP2	:TEMP2=1		
7450	027702	005037	025422		CLR	@#TEMP3	:TEMP3=0		
7451	027706	005037	025424		CLR	@#TEMP4	:TEMP4=0		
7452	027712	012737	000001	025426	MOV	#1,@#TEMP5	:TEMP5=1		
7453	027720	005037	025430		CLR	@#TEMP6	:0 1 SHIFTED BY 0=0 1, PS=0		
7454									
7455	027724	010502			REG01:	MOV	R5,R2	:SAVE R5	
7456	027726	013700	025416		MOV	@#TEMP1,%0	:PLACE THE CONTENTS OF TEMP1 IN REGISTER 0		
7457	027732	013701	025420		MOV	@#TEMP2,%0:1	:PLACE THE CONTENTS OF TEMP2 IN REGISTER 1		
7458	027736	000241			CLC				
7459	027740	032737	000001	001006	BIT	#1,@#SPASS	:IS IT AN EVEN PASS ?		
7460	027746	001004			BNE	2\$:IF NOT THEN GO TO 2\$		
7461	027750	013705	025422		MOV	@#TEMP3,R5	:OTHERWISE EXECUTE ASHC INSTRUCTION IN MODE 0		
7462	027754	073005			ASHC	R5,R0	:USING R0		
7463	027756	000402			BR	4\$			
7464	027760	073067	175436		2\$:	ASHC	TEMP3,%0	:ASHC REGISTER 0 BY THE CONTENTS OF TEMP3	
7465	027764	106737	025414		4\$:	MFPS	@#PSWORD	:SAVE PS	
7466	027770	123737	025430	025414	CMPB	@#TEMP6,@#PSWORD	:COMPARE PS WITH THE CONTENTS OF TEMP6		
7467	027776	001401			BEQ	11\$			
7468	030000	104000			EMT		:WRONG PS		
7469	030002	005237	025412		11\$:	INC	@#COUNT		
7470	030006	023700	025424		CMP	@#TEMP4,%0	:IS THE RESULT IN R0 SAME AS TEMP4?		
7471	030012	001401			BEQ	12\$			
7472	030014	104000			EMT		:WRONG RESULT IN R0		
7473	030016	023701	025426		12\$:	CMP	@#TEMP5,%1	:IS THE RESULT IN R1 SAME AS TEMP5?	
7474							:TEMP1 TEMP2 SHIFTED BY TEMP3=TEMP4 TEMPS		
7475							:AND PS=TEMP6		
7476	030022	001401			BEQ	13\$			
7477	030024	104000			EMT		:WRONG RESULT IN R1		
7478	030026	010205			13\$:	MOV	R2,R5	:RESTORE R5	
7479	030030	021537	025412		CMP	(R5),@#COUNT	:IS TEST NUMBER=COUNTER?		
7480	030034	001401			BEQ	14\$			
7481	030036	104000			EMT		:NO		
7482	030040	005215			14\$:	INC	(R5)		
7483	030042	021527	000160		CMP	(R5),#160	:HAVE THE FIRST 159 TEST BEEN EXECUTED?		
7484	030046	002014			BGE	6\$:YES		
7485	030050	005237	025422		INC	@#TEMP3			
7486	030054	000241			CLC				
7487	030056	006137	025426		ROL	@#TEMP5	:ROTATE TEMPS LEFT BY 1 PLACE		
7488	030062	006137	025424		ROL	@#TEMP4	:INTRODUCE CARRY FROM TEMP4 IN TEMPS		
7489	030066	021527	000121		CMP	(R5),#121	:IS IT TEST 121?		
7490	030072	001004			BNE	REGR23			

7491	030074	004467	000344			JSR	R4,RITSH	:IF SO THEN GO AND INITIATE RIGHT SHIFT
7492	030100	004767	000374		6\$:	JSR	X7,TST160	
7493	030104	013702	025416		REGR23:	MOV	@TEMP1,X2	:PLACE THE CONTENTS OF TEMP1 IN REGISTER 2
7494	030110	013703	025420			MOV	@TEMP2,X2!1	:PLACE THE CONTENTS OF TEMP2 IN REGISTER 3
7495	030114	000241				CLC		
7496	030116	032737	000001	001006		BIT	#1,@\$PASS	:IS IT AN EVEN PASS ?
7497	030124	001004				BNE	2\$:IF NOT THEN GO TO 2\$
7498	030126	013704	025422			MOV	@TEMP3,R4	:OTHERWISE EXECUTE ASHC INSTRUCTION IN MODE 0
7499	030132	073204				ASHC	R4,R2	:USING R2
7500	030134	000402				BR	4\$	
7501	030136	073267	175260		2\$:	ASHC	TEMP3,X2	:ASHC REGISTER 2 BY THE CONTENTS OF TEMP3
7502	030142	106737	025414		4\$:	MFPS	@\$PSWORD	:SAVE PS
7503	030146	123737	025430	025414		CMPB	@TEMP6,@\$PSWORD	:COMPARE PS WITH THE CONTENTS OF TEMP6
7504	030154	001401				BEQ	11\$	
7505	030156	104000				EMT		:WRONG PS
7506	030160	005237	025412		11\$:	INC	@\$COUNT	
7507	030164	023702	025424			CMP	@TEMP4,X2	:IS THE RESULT IN R2 SAME AS TEMP4?
7508	030170	001401				BEQ	12\$	
7509	030172	104000				EMT		:WRONG RESULT IN R2
7510	030174	023703	025426		12\$:	CMP	@TEMP5,X3	:IS THE RESULT IN R3 SAME AS TEMP5?
7511								:TEMP1 TEMP2 SHIFTED BY TEMP3=TEMP4 TEMP5
7512								:AND PS=TEMP6
7513	030200	001401				BEQ	13\$	
7514	030202	104000				EMT		:WRONG RESULT IN R1
7515	030204	021537	025412		13\$:	CMP	(R5),@\$COUNT	:IS TEST NUMBER=COUNTER?
7516	030210	001401				BEQ	14\$	
7517	030212	104000				EMT		:NO
7518	030214	005215			14\$:	INC	(R5)	
7519	030216	021527	000160			CMP	(R5),#160	:HAVE THE FIRST 159 TEST BEEN EXECUTED?
7520	030222	002014				BGE	6\$:YES
7521	030224	005237	025422			INC	@TEMP3	
7522	030230	000241				CLC		
7523	030232	006137	025426			ROL	@TEMP5	:ROTATE TEMP5 LEFT BY 1 PLACE
7524	030236	006137	025424			ROL	@TEMP4	:INTRODUCE CARRY FROM TEMP5 IN TEMP4
7525	030242	021527	000121			CMP	(R5),#121	:IS IT TEST 121?
7526	030246	001004				BNE	REG45	
7527	030250	004467	000170			JSR	R4,RITSH	:IF SO THEN GO AND INITIATE RIGHT SHIFT
7528	030254	004767	000220		6\$:	JSR	X7,TST160	
7529	030260	010501			REG45:	MOV	R5,R1	:SAVE R5
7530	030262	013704	025416			MOV	@TEMP1,X4	:PLACE THE CONTENTS OF TEMP1 IN REGISTER 4
7531	030266	013705	025420			MOV	@TEMP2,X4!1	:PLACE THE CONTENTS OF TEMP2 IN REGISTER 5
7532	030272	000241				CLC		
7533	030274	032737	000001	001006		BIT	#1,@\$PASS	:IS IT AN EVEN PASS ?
7534	030302	001004				BNE	2\$:IF NOT THEN GO TO 2\$
7535	030304	013700	025422			MOV	@TEMP3,R0	:OTHERWISE EXECUTE ASHC INSTRUCTION IN MODE 0
7536	030310	073400				ASHC	R0,R4	:USING R4
7537	030312	000402				BR	4\$	
7538	030314	073467	175102		2\$:	ASHC	TEMP3,X4	:ASHC REGISTER 4 BY THE CONTENTS OF TEMP3
7539	030320	106737	025414		4\$:	MFPS	@\$PSWORD	:SAVE PS
7540	030324	123737	025430	025414		CMPB	@TEMP6,@\$PSWORD	:COMPARE PS WITH THE CONTENTS OF TEMP6
7541	030332	001401				BEQ	11\$	
7542	030334	104000				EMT		:WRONG PS
7543	030336	005237	025412		11\$:	INC	@\$COUNT	
7544	030342	023704	025424			CMP	@TEMP4,X4	:IS THE RESULT IN R4 SAME AS TEMP4?
7545	030346	001401				BEQ	12\$	
7546	030350	104000				EMT		:WRONG RESULT IN R4

7547	030352	023705	025426	12\$:	CMP	@#TEMP5,25		:IS THE RESULT IN R5 SAME AS TEMP5? :TEMP1 TEMP2 SHIFTED BY TEMP3=TEMP4 TEMP5 :AND PS=TEMP6
7548								
7549								
7550	030356	001401			BEQ	13\$		
7551	030360	104000			EMT			:WRONG RESULT IN R5
7552	030362	021137	025412	13\$:	CMP	(R1),@#COUNT		:IS TEST NUMBER=COUNTER?
7553	030366	001401			BEQ	14\$		
7554	030370	104000			EMT			:NO
7555	030372	010105		14\$:	MOV	R1,R5		:RESTORE R5
7556	030374	005215			INC	(R5)		
7557	030376	021527	000160		CMP	(R5),#160		:HAVE THE FIRST 159 TEST BEEN EXECUTED?
7558	030402	002014			BGE	6\$:YES
7559	030404	005237	025422		INC	@#TEMP3		
7560	030410	000241			CLC			
7561	030412	006137	025426		ROL	@#TEMP5		:ROTATE TEMP5 LEFT BY 1 PLACE
7562	030416	006137	025424		ROL	@#TEMP4		:INTRODUCE CARRY FROM TEMP5 IN TEMP4
7563	030422	021527	000121		CMP	(R5),#121		:IS IT TEST 121?
7564	030426	001004			BNE	8\$		
7565	030430	004467	000010		JSR	R4,RITSH		:IF SO THEN GO AND INITIATE RIGHT SHIFT
7566	030434	004767	000C40	6\$:	JSR	%7,TST160		
7567	030440	000167	17:260	8\$:	JMP	REG01		
7568	030444	022424		RITSH:	CMP	(R4)+,(R4)+		:MAKE R4 POINT TO THE NEXT REG TAG
7569	030446	012737	040000	025416	MOV	#40000,@#TEMP1		:TEMP1=4000
7570	030454	005037	025420		CLR	@#TEMP2		:TEMP2=0
7571	030460	012737	177742	025422	MOV	#-30,@#TEMP3		:TEMP3=-30
7572	030466	005037	025424		CLR	@#TEMP4		:TEMP4=0
7573	030472	005237	025426		INC	@#TEMP5		:TEMP5=1
7574	030476	000204			RTS	R4		
7575	030500	021527	000160	TST160:	CMP	(R5),#160		:IS IT TEST 160
7576	030504	001010			BNE	TST161		:IF NOT THEN TRY TEST 161
7577	030506	005037	025416		CLR	@#TEMP1		:0 0 SHIFTED BY 0
7578	030512	005037	025424		CLR	@#TEMP4		:IS EQUAL TO 0 0
7579	030516	012737	000004	025430	MOV	#4,@#TEMP6		:AND PS=4
7580	030524	000207			RTS	%7		
7581	030526	021527	000161	TST161:	CMP	(R5),#161		:IS IT TEST 161
7582	030532	001004			BNE	TST162		
7583	030534	012737	177746	025422	MOV	#-32,@#TEMP3		:0 0 SHIFTED BY -32=0 0, PS=4
7584	030542	000207			RTS	%7		
7585	030544	021527	0C0162	TST162:	CMP	(R5),#162		:IS IT TEST 162
7586	030550	001004			BNE	TST163		:IF NOT THEN TRY TEST 163
7587	030552	012737	000052	025422	MOV	#32,@#TEMP3		:0 0 SHIFTED BY 32=0 0, PS=4
7588	030560	000207			PTS	%7		
7589	030562	021527	000163	TST163:	CMP	(R5),#163		:IS IT TEST 163?
7590	030566	001016			BNE	TST164		:IF NOT THEN TRY TEST 164
7591	030570	012737	052525	025416	MOV	#52525,@#TEMP1		:52525 0
7592	030576	012737	177760	025422	MOV	#-16,@#TEMP3		:SHIFTED BY -16.
7593	030604	005037	025424		CLR	@#TEMP4		
7594	030610	012737	052525	025426	MOV	#52525,@#TEMP5		:IS EQUAL TO 0 52525
7595	030616	005037	025430		CLR	@#TEMP6		:AND PS = 0
7596	030622	000207			RTS	%7		
7597	030624	021527	000164	TST164:	CMP	(R5),#164		:IS IT TEST 164?
7598	030630	001014			BNE	TST165		:IF NOT THEN TRY TEST 165
7599	030632	012737	125252	025416	MOV	#125252,@#TEMP1		:125252 0 SHIFTED BY -16.
7600	030640	005337	025424		DEC	@#TEMP4		
7601	030644	012737	125252	025426	MOV	#125252,@#TEMP5		:IS EQUAL TO -1 125252
7602	030652	012737	000010	025430	MOV	#10,@#TEMP6		:AND PS=10

7603 030660 000207
7604 030662 021527 000165
7605 030666 001007
7606 030670 012737 177777 025416
7607 030676 012737 177777 025426
7608 030704 000207
7609 030706 021527 000166
7610 030712 001011
7611 030714 012737 100000 025416
7612 030722 012737 177740 025422
7613 030730 005237 025430
7614 030734 000207
7615 030736 021527 000167
7616 030742 001014

TST165: RTS X7
CMP (R5),#165 ;IS IT TEST 165?
BNE TST166 ;IF NOT THEN TRY TEST 166
MOV #-1,@TEMP1 ;-1 0 SHIFTED BY -16
MOV #-1,@TEMP5 ;IS EQUAL TO -1 -1, AND PS=10
RTS X7
TST166: CMP (R5),#166 ;IS IT TEST 166?
BNE TST167 ;IF NOT THEN TRY TEST 167
MOV #100000,@TEMP1 ;100000 0
MOV #-32,@TEMP3 ;SHIFTED BY -32 IS EQUAL TO -1 -1
INC @TEMP6 ;AND PS=11
RTS X7
TST167: CMP (R5),#167 ;IS IT TEST 167?
BNE TST170 ;IF NOT THEN TRY TEST 170

CJKDJB0 11/23-B CPU CLUSTER DIAG.
CJKDJB.P11 26-MAY-82 11:14

DNMAC X24.07-563 26-MAY-82^{D 12} 11:18 PAGE 147
ASHC INSTRUCTION TESTS

SEQ 0146

7617 030744 005037 025416
7618 030750 005337 025420

CLR @TEMP1
DEC @TEMP2 ;0 -1

CJKDJBO 11/23-B CPU CLUSTER DIAG.
CJKDJB.P11 26-MAY-82 11:14

DNMAC X24.07-563 26-MAY-82 11:18 PAGE 148
ASHC INSTRUCTION TESTS

E 12

SEQ 0147

7619	030754	012737	000020	025422	MOV	#16.,@TEMP3	:SHIFTED BY 16.
7620	030762	005037	025426		CLR	@TEMP5	:IS EQUAL TO -1 0
7621	030766	005237	025430		INC	@TEMP6	:AND PS=12
7622	030772	000207			RTS	#7	
7623	030774	021527	000170		TST170: CMP	(R5),#170	:IS IT TEST 170?

CJKDJBO 11/23-B CPU CLUSTER DIAG.
CJKDJB.P11 26-MAY-82 11:14

DNMAC X24.07-563 26-MAY-82^{F 12} 11:18 PAGE 149
ASHC INSTRUCTION TESTS

SEQ 0148

7624 031000 001007

BNE TST171

;IF NOT THEN TRY TEST 171

7625	031002	012737	125252	025420	MOV	#125252,@#TEMP2	:0 125252 SHIFTED BY 16
7626	031010	012737	125252	025424	MOV	#125252,@#TEMP4	:IS EQUAL TO 125252 0, AND PS=12
7627	031016	000207			RTS	%7	
7628	031020	021527	000171		TST171: CMP	(R5),#171	:IS IT TEST 171?
7629	031024	001010			BNE	TST172	:IF NOT THEN TRY TEST 172
7630	031026	005337	025422		DEC	@#TEMP3	:0 125252 SHIFTED BY 15
7631	031032	012737	052525	025424	MOV	#52525,@#TEMP4	:IS EQUAL TO 52525 0
7632	031040	005037	025430		CLR	@#TEMP6	:AND PS=0
7633	031044	000207			RTS	%7	
7634	031046	021527	000172		TST172: CMP	(R5),#172	:IS IT TEST 172?
7635	031052	001006			BNE	TST173	:IF NOT THEN TRY TEST 173
7636	031054	012737	052525	025420	MOV	#52525,@#TEMP2	:0 52525
7637	031062	005237	025422		INC	@#TEMP3	:SHIFTED BY 16. IS EQUAL TO 52525 0, AND PS=0
7638	031066	000207			RTS	%7	
7639	031070	021527	000173		TST173: CMP	(R5),#173	:IS IT TEST 173?
7640	031074	001014			BNE	TST174	:IF NOT THEN TRY TEST 174
7641	031076	012737	177777	025420	MOV	#-1,@#TEMP2	:0 -1
7642	031104	005337	025422		DEC	@#TEMP3	:SHIFTED BY 15.
7643	031110	012737	077777	025424	MOV	#77777,@#TEMP4	
7644	031116	012737	100000	025426	MOV	#100000,@#TEMP5	:IS EQUAL TO 77777 100000, AND PS=0
7645	031124	000207			RTS	%7	
7646	031126	021527	000174		TST174: CMP	(R5),#174	:IS IT TEST 174?
7647	031132	001013			BNE	TST175	:IF NOT THEN TRY TEST 175
7648	031134	012737	100000	025416	MOV	#100000,@#TEMP1	
7649	031142	005337	025420		DEC	@#TEMP2	:100000 -2 SHIFTED BY 15.
7650	031146	005037	025426		CLR	@#TEMP5	:IS EQUAL TO 77777 0
7651	031152	012737	000002	025430	MOV	#2,@#TEMP6	:AND PS=2
7652	031160	000207			RTS	%7	
7653	031162	021527	000175		TST175: CMP	(R5),#175	:IS IT TEST 175?
7654	031166	001015			BNE	ENT176	:IF NOT THEN TRY TEST 176
7655	031170	012737	177777	025416	MOV	#-1,@#TEMP1	
7656	031176	005037	025420		CLR	@#TEMP2	:-1 0
7657	031202	005237	025422		INC	@#TEMP3	:SHIFTED BY 16.
7658	031206	005037	025424		CLR	@#TEMP4	:IS EQUAL TO 0 0
7659	031212	012737	000007	025430	MOV	#7,@#TEMP6	:AND PS=7

7660 031220 000207
7661 031222 021527 000176
7662 031226 001401
7663 031230 104000
7664
7665 031232 005726
7666
7667
7668
7669
7670
7671 031234
7672 031234 012701 000000
7673 031240 012701 000001
7674 031244 000241
7675 031246 073127 000010
7676 031252 106737 025414
7677 031256 122737 000000 025414
7678 031264 001401
7679 031266 104000
7680 031270 022701 000400
7681 031274 001401
7682 031276 104000
7683 031300
7684 031300 005215
7685
7686
7687
7688
7689
7690
7691 031302
7692 031302 012703 000000
7693 031306 012703 177777
7694 031312 000241
7695 031314 073327 000017
7696 031320 106737 025414
7697 031324 122737 000011 025414
7698 031332 001401
7699 031334 104000
7700 031336 022703 100000
7701 031342 001401
7702 031344 104000
7703 031346
7704 031346 005215
7705
7706
7707
7708
7709
7710
7711 031350
7712 031350 010501
7713 031352 012705 000000
7714 031356 012705 052525
7715 031362 000241

ENT176: RTS X7
CMP (R5),#176 ;IS THE PROGRAM ENTERING TEST 176?
BEQ 1\$
EMT ;TEST NUMBER GOOFED

1\$: TST (SP)+ ;RESTORE STACK POINTER

;TEST:176 1 SHIFTED BY 8. = 400 PS = 0

TST176:

MOV #DUMMY,X1 ;LOAD R1 WITH DUMMY
MOV #1,X1!1 ;LOAD R1!1 WITH 1
CLC
ASHC #8,X1 ;SHIFT R1,R1!1 BY 8.
MFPS @#PSWORD ;SAVE PS
CMPB #0,@#PSWORD ;IS THE PS 0?
BEQ 11\$
EMT ;THE PS IS NOT EQUAL TO 0
11\$: CMP #400,X1 ;IS THE RESULT 400?
BEQ 13\$
EMT ;R1 IS NOT EQUAL TO 400
13\$: INC (R5)

;TEST:177 -1 SHIFTED BY 15. = 100000 PS = 11

TST177:

MOV #DUMMY,X3 ;LOAD R3 WITH DUMMY
MOV #-1,X3!1 ;LOAD R3!1 WITH -1
CLC
ASHC #15,X3 ;SHIFT R3,R3!1 BY 15.
MFPS @#PSWORD ;SAVE PS
CMPB #11,@#PSWORD ;IS THE PS 11?
BEQ 11\$
EMT ;THE PS IS NOT EQUAL TO 11
11\$: CMP #100000,X3 ;IS THE RESULT 100000?
BEQ 13\$
EMT ;R3 IS NOT EQUAL TO 100000
13\$: INC (R5)

;TEST:200 52525 SHIFTED BY 0 = 52525 PS = 0

TST200:

MOV R5,R1 ;SAVE R5
MOV #DUMMY,X5 ;LOAD R5 WITH DUMMY
MOV #52525,X5!1 ;LOAD R5!1 WITH 52525
CLC

```
7716 031364 073527 000000 ASHC #0,X5 ;SHIFT R5,R5!1 BY 0
7717 031370 106737 025414 MFPS @#PSWORD ;SAVE PS
7718 031374 122737 000000 025414 CMPB #0,@#PSWORD ;IS THE PS 0?
7719 031402 001401 BEQ 11$
7720 031404 104000 EMT ;THE PS IS NOT EQUAL TO 0
7721 031406 022705 052525 11$: CMP #52525,X5 ;IS THE RESULT 52525?
7722 031412 001401 BEQ 13$
7723 031414 104000 EMT ;R5 IS NOT EQUAL TO 52525
7724 031416 13$:
7725 031416 010105 MOV R1,R5 ;RESTORE R5
7726 031420 005215 INC (R5)
7727
7728
7729 ;*****
7730 ;TEST:201 20010 SHIFTED BY -13. = 101 PS = 0
7731 ;*****
7732
7733 TST201:
7734 031422 012701 000000 MOV #DUMMY,X1 ;LOAD R1 WITH DUMMY
7735 031426 012701 020010 MOV #20010,X1!1 ;LOAD R1!1 WITH 20010
7736 031432 000241 CLC
7737 031434 073127 177763 ASHC #-13.,X1 ;SHIFT R1,R1!1 BY -13.
7738 031440 106737 025414 MFPS @#PSWORD ;SAVE PS
7739 031444 122737 000000 025414 CMPB #0,@#PSWORD ;IS THE PS 0?
7740 031452 001401 BEQ 11$
7741 031454 104000 EMT ;THE PS IS NOT EQUAL TO 0
7742 031456 022701 000101 11$: CMP #101,X1 ;IS THE RESULT 101?
7743 031462 001401 BEQ 13$
7744 031464 104000 EMT ;R1 IS NOT EQUAL TO 101
7745 031466 13$:
7746 031466 005215 INC (R5)
7747
7748
7749 ;*****
7750 ;TEST:202 -1 SHIFTED BY 16. = 0 PS = 11
7751 ;*****
7752
7753 TST202:
7754 031470 012703 000000 MOV #DUMMY,X3 ;LOAD R3 WITH DUMMY
7755 031474 012703 177777 MOV #-1,X3!1 ;LOAD R3!1 WITH -1
7756 031500 000241 CLC
7757 031502 073327 000020 ASHC #16.,X3 ;SHIFT R3,R3!1 BY 16.
7758 031506 106737 025414 MFPS @#PSWORD ;SAVE PS
7759 031512 122737 000011 025414 CMPB #11,@#PSWORD ;IS THE PS 11?
7760 031520 001401 BEQ 11$
7761 031522 104000 EMT ;THE PS IS NOT EQUAL TO 11
7762 031524 022703 000000 11$: CMP #0,X3 ;IS THE RESULT 0?
7763 031530 001401 BEQ 13$
7764 031532 104000 EMT ;R3 IS NOT EQUAL TO 0
7765 031534 13$:
7766 031534 005215 INC (R5)
7767
7768
7769 ;*****
7770 ;TEST:203 1 SHIFTED BY -1 = 100000 PS = 1
7771 ;*****
```

7772
7773 031536
7774 031536 010501
7775 031540 012705 000000
7776 031544 012705 000001
7777 031550 000241
7778 031552 073527 177777
7779 031556 106737 025414
7780 031562 122737 000001 025414
7781 031570 001401
7782 031572 104000
7783 031574 022705 100000
7784 031600 001401
7785 031602 104000
7786 031604
7787 031604 010105
7788 031606 005215

TST203:
MOV R5,R1 ;SAVE R5
MOV #DUMMY,%5 ;LOAD R5 WITH DUMMY
MOV #1,%5!1 ;LOAD R5!1 WITH 1
CLC
ASHC #-1,%5 ;SHIFT R5,R5!1 BY -1
MFPS @#PSWORD ;SAVE PS
CMPB #1,@#PSWORD ;IS THE PS 1?
BEQ 11\$
EMT ;THE PS IS NOT EQUAL TO 1
11\$: CMP #100000,%5 ;IS THE RESULT 100000?
BEQ 13\$
EMT ;R5 IS NOT EQUAL TO 100000
13\$: MOV R1,R5 ;RESTORE R5
INC (R5)

:TEST:204 125252 SHIFTED BY -16. = 125252 PS = 11

7794
7795 031610
7796 031610 012701 000000
7797 031614 012701 125252
7798 031620 000241
7799 031622 073127 177760
7800 031626 106737 025414
7801 031632 122737 000011 025414
7802 031640 001401
7803 031642 104000
7804 031644 022701 125252
7805 031650 001401
7806 031652 104000
7807 031654
7808 031654 005215

TST204:
MOV #DUMMY,%1 ;LOAD R1 WITH DUMMY
MOV #125252,%1!1 ;LOAD R1!1 WITH 125252
CLC
ASHC #-16.,%1 ;SHIFT R1,R1!1 BY -16.
MFPS @#PSWORD ;SAVE PS
CMPB #11,@#PSWORD ;IS THE PS 11?
BEQ 11\$
EMT ;THE PS IS NOT EQUAL TO 11
11\$: CMP #125252,%1 ;IS THE RESULT 125252?
BEQ 13\$
EMT ;R1 IS NOT EQUAL TO 125252
13\$: INC (R5)

:TEST:205 125252 125252 SHIFTED BY 21. = 52500 000000 PS = 3

7814
7815 031656
7816 031656 012702 125252
7817 031662 012703 125252
7818 031666 000241
7819 031670 073227 000025
7820 031674 106737 025414
7821 031700 122737 000003 025414
7822 031706 001401
7823 031710 104000
7824 031712 022702 052500
7825 031716 001401
7826 031720 104000
7827 031722 022703 000000

TST205:
MOV #125252,%2 ;LOAD R2 WITH 125252
MOV #125252,%2!1 ;LOAD R2!1 WITH 125252
CLC
ASHC #21.,%2 ;SHIFT R2,R2!1 BY 21.
MFPS @#PSWORD ;SAVE PS
CMPB #3,@#PSWORD ;IS THE PS 3?
BEQ 11\$
EMT ;THE PS IS NOT EQUAL TO 3
11\$: CMP #52500,%2 ;IS THE RESULT 52500?
BEQ 12\$
EMT ;R2 IS NOT EQUAL TO 52500
12\$: CMP #000000,%2!1 ;IS THE RESULT 000000?

7828 031726 001401
7829 031730 104000
7830 031732
7831 031732 005215
7832
7833
7834
7835 031734 012702 177771
7836 031740 012703 025432
7837 031744 012704 025434
7838
7839

BEQ 13\$
EMT ;R2!1 IS NOT EQUAL TO 000000
13\$:
INC (R5)
MOV #-7,%2
MOV #S1,%3
MOV #S2,%4

7840
7841
7842

:*****
:TEST:206 125252 125252 SHIFTED BY S1 = 177525 52525 PS = 10
:*****

7843 031750
7844 031750 012700 125252
7845 031754 012701 125252
7846 031760 000241
7847 031762 073067 173444
7848 031766 106737 025414
7849 031772 122737 000010 025414
7850 032000 001401
7851 032002 104000
7852 032004 022700 177525
7853 032010 001401
7854 032012 104000
7855 032014 022701 052525
7856 032020 001401
7857 032022 104000
7858 032024
7859 032024 005215
7860
7861

TST206:
MOV #125252,%0 ;LOAD R0 WITH 125252
MOV #125252,%0!1 ;LOAD R0!1 WITH 125252
CLC
ASHC S1,%0 ;SHIFT R0,R0!1 BY S1
MFPS @#PSWORD ;SAVE PS
CMPB #10,@#PSWORD ;IS THE PS 10?
BEQ 11\$
EMT ;THE PS IS NOT EQUAL TO 10
CMP #177525,%0 ;IS THE RESULT 177525?
BEQ 12\$
EMT ;R0 IS NOT EQUAL TO 177525
CMP #52525,%0!1 ;IS THE RESULT 52525?
BEQ 13\$
EMT ;R0!1 IS NOT EQUAL TO 52525 OR INCORRECT SEQUENCE
13\$:
INC (R5)

7862
7863
7864
7865

:*****
:TEST:207 125252 125252 SHIFTED BY @S2 = 177525 52525 PS = 10
:*****

7866 032026
7867 032026 012700 125252
7868 032032 012701 125252
7869 032036 000241
7870 032040 073077 173370
7871 032044 106737 025414
7872 032050 122737 000010 025414
7873 032056 001401
7874 032060 104000
7875 032062 022700 177525
7876 032066 001401
7877 032070 104000
7878 032072 022701 052525
7879 032076 001401
7880 032100 104000
7881 032102
7882 032102 005215
7883

TST207:
MOV #125252,%0 ;LOAD R0 WITH 125252
MOV #125252,%0!1 ;LOAD R0!1 WITH 125252
CLC
ASHC @S2,%0 ;SHIFT R0,R0!1 BY @S2
MFPS @#PSWORD ;SAVE PS
CMPB #10,@#PSWORD ;IS THE PS 10?
BEQ 11\$
EMT ;THE PS IS NOT EQUAL TO 10
CMP #177525,%0 ;IS THE RESULT 177525?
BEQ 12\$
EMT ;R0 IS NOT EQUAL TO 177525
CMP #52525,%0!1 ;IS THE RESULT 52525?
BEQ 13\$
EMT ;R0!1 IS NOT EQUAL TO 52525 OR INCORRECT SEQUENCE
13\$:
INC (R5)

7884
7885
7886
7887
7888
7889 032104
7890 032104 012700 125252
7891 032110 012701 125252
7892 032114 000241
7893 032116 073037 025432
7894 032122 106737 025414
7895 032126 122737 000010 025414
7896 032134 001401
7897 032136 104000
7898 032140 022700 177525
7899 032144 001401
7900 032146 104000
7901 032150 022701 052525
7902 032154 001401
7903 032156 104000
7904 032160
7905 032160 005215
7906
7907
7908
7909
7910
7911
7912 032162
7913 032162 012700 125252
7914 032166 012701 125252
7915 032172 000241
7916 032174 073013
7917 032176 106737 025414 025414
7918 032202 122737 000010 025414
7919 032210 001401
7920 032212 104000
7921 032214 022700 177525
7922 032220 001401
7923 032222 104000
7924 032224 022701 052525
7925 032230 001401
7926 032232 104000
7927 032234
7928 032234 005215
7929
7930
7931
7932
7933
7934
7935 032236
7936 032236 012700 125252
7937 032242 012701 125252
7938 032246 000241
7939 032250 073023

:TEST:210 125252 125252 SHIFTED BY @#S1 = 177525 52525 PS = 10

TST210:
MOV #125252,%0 ;LOAD R0 WITH 125252
MOV #125252,%0!1 ;LOAD R0!1 WITH 125252
CLC
ASHC @#S1,%0 ;SHIFT R0,R0!1 BY @#S1
MFPS @#PSWORD ;SAVE PS
CMPB #10,@#PSWORD ;IS THE PS 10?
BEQ 11\$
EMT ;THE PS IS NOT EQUAL TO 10
11\$: CMP #177525,%0 ;IS THE RESULT 177525?
BEQ 12\$
EMT ;R0 IS NOT EQUAL TO 177525
12\$: CMP #52525,%0!1 ;IS THE RESULT 52525?
BEQ 13\$
EMT ;R0!1 IS NOT EQUAL TO 52525 OR INCORRECT SEQUENCE
13\$: INC (R5)

:TEST:211 125252 125252 SHIFTED BY (3) = 177525 52525 PS = 10

TST211:
MOV #125252,%0 ;LOAD R0 WITH 125252
MOV #125252,%0!1 ;LOAD R0!1 WITH 125252
CLC
ASHC (3),%0 ;SHIFT R0,R0!1 BY (3)
MFPS @#PSWORD ;SAVE PS
CMPB #10,@#PSWORD ;IS THE PS 10?
BEQ 11\$
EMT ;THE PS IS NOT EQUAL TO 10
11\$: CMP #177525,%0 ;IS THE RESULT 177525?
BEQ 12\$
EMT ;R0 IS NOT EQUAL TO 177525
12\$: CMP #52525,%0!1 ;IS THE RESULT 52525?
BEQ 13\$
EMT ;R0!1 IS NOT EQUAL TO 52525 OR INCORRECT SEQUENCE
13\$: INC (R5)

:TEST:212 125252 125252 SHIFTED BY (3)+ = 177525 52525 PS = 10

TST212:
MOV #125252,%0 ;LOAD R0 WITH 125252
MOV #125252,%0!1 ;LOAD R0!1 WITH 125252
CLC
ASHC (3)+,%0 ;SHIFT R0,R0!1 BY (3)+

7940	032252	106737	025414			MFPS	@#PSWORD	:SAVE PS
7941	032256	122737	000010	025414		CMPB	#10,@#PSWORD	:IS THE PS 10?
7942	032264	001401				BEQ	11\$	
7943	032266	104000				EMT		:THE PS IS NOT EQUAL TO 10
7944	032270	022700	177525		11\$:	CMP	#177525,%0	:IS THE RESULT 177525?
7945	032274	001401				BEQ	12\$	
7946	032276	104000				EMT		:RO IS NOT EQUAL TO 177525
7947	032300	022701	052525		12\$:	CMP	#52525,%0!1	:IS THE RESULT 52525?
7948	032304	001401				BEQ	13\$	
7949	032306	104000				EMT		:RO!1 IS NOT EQUAL TO 52525 OR INCORRECT SEQUENCE
7950	032310				13\$:			
7951	032310	005215				INC	(R5)	

:TEST:213 125252 125252 SHIFTED BY -(3) = 177525 52525 PS = 10

7958	032312					TST213:		
7959	032312	012700	125252			MOV	#125252,%0	:LOAD R0 WITH 125252
7960	032316	012701	125252			MOV	#125252,%0!1	:LOAD R0!1 WITH 125252
7961	032322	000241				CLC		
7962	032324	073043				ASHC	-(3),%0	:SHIFT R0,R0!1 BY -(3)
7963	032326	106737	025414			MFPS	@#PSWORD	:SAVE PS
7964	032332	122737	000010	025414		CMPB	#10,@#PSWORD	:IS THE PS 10?
7965	032340	001401				BEQ	11\$	
7966	032342	104000				EMT		:THE PS IS NOT EQUAL TO 10
7967	032344	022700	177525		11\$:	CMP	#177525,%0	:IS THE RESULT 177525?
7968	032350	001401				BEQ	12\$	
7969	032352	104000				EMT		:RO IS NOT EQUAL TO 177525
7970	032354	022701	052525		12\$:	CMP	#52525,%0!1	:IS THE RESULT 52525?
7971	032360	001401				BEQ	13\$	
7972	032362	104000				EMT		:RO!1 IS NOT EQUAL TO 52525 OR INCORRECT SEQUENCE
7973	032364				13\$:			
7974	032364	005215				INC	(R5)	

:TEST:214 125252 125252 SHIFTED BY 2(4) = 177252 125252 PS = 11

7981	032366					TST214:		
7982	032366	012700	125252			MOV	#125252,%0	:LOAD R0 WITH 125252
7983	032372	012701	125252			MOV	#125252,%0!1	:LOAD R0!1 WITH 125252
7984	032376	000241				CLC		
7985	032400	073064	000002			ASHC	2(4),%0	:SHIFT R0,R0!1 BY 2(4)
7986	032404	106737	025414			MFPS	@#PSWORD	:SAVE PS
7987	032410	122737	000011	025414		CMPB	#11,@#PSWORD	:IS THE PS 11?
7988	032416	001401				BEQ	11\$	
7989	032420	104000				EMT		:THE PS IS NOT EQUAL TO 11
7990	032422	022700	177252		11\$:	CMP	#177252,%0	:IS THE RESULT 177252?
7991	032426	001401				BEQ	12\$	
7992	032430	104000				EMT		:RO IS NOT EQUAL TO 177252
7993	032432	022701	125252		12\$:	CMP	#125252,%0!1	:IS THE RESULT 125252?
7994	032436	001401				BEQ	13\$	
7995	032440	104000				EMT		:RO!1 IS NOT EQUAL TO 125252 OR INCORRECT SEQUENCE

7996 032442
7997 032442 005215
7998
7999
8000
8001
8002
8003
8004 032444
8005 032444 012700 125252
8006 032450 012701 125252
8007 032454 000241
8008 032456 073074 000000
8009 032462 106737 025414
8010 032466 122737 000010 025414
8011 032474 001401
8012 032476 104000
8013 032500 022700 177525
8014 032504 001401
8015 032506 104000
8016 032510 022701 052525
8017 032514 001401
8018 032516 104000
8019 032520
8020 032520 005215
8021
8022
8023
8024
8025
8026
8027 032522
8028 032522 012700 125252
8029 032526 012701 125252
8030 032532 000241
8031 032534 073034
8032 032536 106737 025414
8033 032542 122737 000010 025414
8034 032550 001401
8035 032552 104000
8036 032554 022700 177525
8037 032560 001401
8038 032562 104000
8039 032564 022701 052525
8040 032570 001401
8041 032572 104000
8042 032574
8043 032574 005215
8044
8045
8046
8047
8048
8049
8050 032576
8051 032576 012700 125252

13\$: INC (R5)
:*****
:TEST:215 125252 125252 SHIFTED BY @ (4) = 177525 52525 PS = 10
:*****
TST215:
MOV #125252,X0 ;LOAD R0 WITH 125252
MOV #125252,X0!1 ;LOAD R0!1 WITH 125252
CLC
ASHC @ (4),X0 ;SHIFT R0,R0!1 BY @ (4)
MFPS @#PSWORD ;SAVE PS
CMPB #10,@#PSWORD ;IS THE PS 10?
BEQ 11\$
EMT ;THE PS IS NOT EQUAL TO 10
11\$: CMP #177525,X0 ;IS THE RESULT 177525?
BEQ 12\$
EMT ;R0 IS NOT EQUAL TO 177525
12\$: CMP #52525,X0!1 ;IS THE RESULT 52525?
BEQ 13\$
EMT ;R0!1 IS NOT EQUAL TO 52525 OR INCORRECT SEQUENCE
13\$: INC (R5)

:*****
:TEST:216 125252 125252 SHIFTED BY @ (4)+ = 177525 52525 PS = 10
:*****
TST216:
MOV #125252,X0 ;LOAD R0 WITH 125252
MOV #125252,X0!1 ;LOAD R0!1 WITH 125252
CLC
ASHC @ (4)+,X0 ;SHIFT R0,R0!1 BY @ (4)+
MFPS @#PSWORD ;SAVE PS
CMPB #10,@#PSWORD ;IS THE PS 10?
BEQ 11\$
EMT ;THE PS IS NOT EQUAL TO 10
11\$: CMP #177525,X0 ;IS THE RESULT 177525?
BEQ 12\$
EMT ;R0 IS NOT EQUAL TO 177525
12\$: CMP #52525,X0!1 ;IS THE RESULT 52525?
BEQ 13\$
EMT ;R0!1 IS NOT EQUAL TO 52525 OR INCORRECT SEQUENCE
13\$: INC (R5)

:*****
:TEST:217 125252 125252 SHIFTED BY @-(4) = 177525 52525 PS = 10
:*****
TST217:
MOV #125252,X0 ;LOAD R0 WITH 125252

8052	032602	012701	125252		MOV	#125252,X0!1	;LOAD R0!1 WITH 125252
8053	032606	000241			CLC		
8054	032610	073054			ASHC	a-(4),X0	;SHIFT R0,R0!1 BY a-(4)
8055	032612	106737	025414		MFPS	a#PSWORD	;SAVE PS
8056	032616	122737	000010	025414	CMPB	#10,a#PSWORD	;IS THE PS 10?
8057	032624	001401			BEQ	11\$	
8058	032626	104000			EMT		;THE PS IS NOT EQUAL TO 10
8059	032630	022700	177525	11\$:	CMP	#177525,X0	;IS THE RESULT 177525?
8060	032634	001401			BEQ	12\$	
8061	032636	104000			EMT		;R0 IS NOT EQUAL TO 177525
8062	032640	022701	052525	12\$:	CMP	#52525,X0!1	;IS THE RESULT 52525?
8063	032644	001401			BEQ	13\$	
8064	032646	104000			EMT		;R0!1 IS NOT EQUAL TO 52525 OR INCORRECT SEQUENCE
8065	032650			13\$:			
8066	032650	005215			INC	(R5)	
8067							
8068							
8069							
8070							
8071							
8072							
8073							
8074							
8075							

8076
8077
8078
8079
8080
8081
8082
8083
8084
8085
8086
8087
8088
8089
8090
8091
8092
8093
8094
8095
8096
8097
8098
8099
8100
8101
8102
8103
8104
8105
8106
8107
8108
8109
8110
8111
8112
8113
8114
8115
8116
8117
8118
8119
8120
8121
8122
8123
8124
8125
8126
8127
8128
8129
8130
8131

032652
032652 012700 000001
032656 070027 000000
032662 106737 025414
032666 122737 000004 025414
032674 001401
032676 104000
032700 022700 000000
032704 001401
032706 104000
032710 022701 000000
032714 001401
032716 104000
032720
032720 005215

032722
032722 012700 177777
032726 070027 000001
032732 106737 025414
032736 122737 000010 025414
032744 001401
032746 104000
032750 022700 177777
032754 001401
032756 104000
032760 022701 177777
032764 001401
032766 104000
032770
032770 005215

032772
032772 012702 000002
032776 070227 000002

: MUL INSTRUCTION TESTS

:TEST:220 MUL 1 * #0 = 0 0 PS = 4

TST220:
MOV #1,%0 ;LOAD MULTIPLICAND WITH 1
MUL #0,%0 ;MULTIPLY 1 * #0
MFPS @#PSWORD ;SAVE PS
CMPB #4,@#PSWORD ;IS PS = 4
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #0,%0 ;IS HIGH ORDER = 0
BEQ 12\$
EMT ;HIGH ORDER IS WRONG
12\$: CMP #0,%0!1 ;IS LOW ORDER = 0
BEQ 13\$
EMT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
13\$: INC (R5)

:TEST:221 MUL -1 * #1 = -1 -1 PS = 10

TST221:
MOV #-1,%0 ;LOAD MULTIPLICAND WITH -1
MUL #1,%0 ;MULTIPLY -1 * #1
MFPS @#PSWORD ;SAVE PS
CMPB #10,@#PSWORD ;IS PS = 10
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #-1,%0 ;IS HIGH ORDER = -1
BEQ 12\$
EMT ;HIGH ORDER IS WRONG
12\$: CMP #-1,%0!1 ;IS LOW ORDER = -1
BEQ 13\$
EMT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
13\$: INC (R5)

:TEST:222 MUL 2 * #2 = 0 4 PS = 0

TST222:
MOV #2,%2 ;LOAD MULTIPLICAND WITH 2
MUL #2,%2 ;MULTIPLY 2 * #2

D 13

```
8132 033002 106737 025414 MFPS @#PSWORD ;SAVE PS
8133 033006 122737 000000 025414 CMPB #0,@#PSWORD ;IS PS = 0
8134 033014 001401 BEQ 11$
8135 033016 104000 EMT ;PS IS WRONG
8136 033020 022702 000000 11$: CMP #0,%2 ;IS HIGH ORDER = 0
8137 033024 001401 BEQ 12$
8138 033026 104000 EMT ;HIGH ORDER IS WRONG
8139 033030 022703 000004 12$: CMP #4,%2!1 ;IS LOW ORDER = 4
8140 033034 001401 BEQ 13$
8141 033036 104000 EMT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
8142 033040 13$:
8143 033040 005215 INC (R5)
```

```
*****
:TEST:223 MUL 1000 * #200 = 1 0 PS = 1
*****
```

```
TST223:
8150 033042 MOV R5,R1 ;SAVE R5
8151 033042 010501 MOV #1000,%4 ;LOAD MULTIPLICAND WITH 1000
8152 033044 012704 001000 MUL #200,%4 ;MULTIPLY 1000 * #200
8153 033050 070427 000200 MFPS @#PSWORD ;SAVE PS
8154 033054 106737 025414 025414 CMPB #1,@#PSWORD ;IS PS = 1
8155 033060 122737 000001 BEQ 11$
8156 033066 001401 EMT ;PS IS WRONG
8157 033070 104000 EMT ;IS HIGH ORDER = 1
8158 033072 022704 000001 11$: CMP #1,%4
8159 033076 001401 BEQ 12$
8160 033100 104000 EMT ;HIGH ORDER IS WRONG
8161 033102 022705 000000 12$: CMP #0,%4!1 ;IS LOW ORDER = 0
8162 033106 001401 BEQ 13$
8163 033110 104000 EMT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
8164 033112 13$:
8165 033112 010105 MOV R1,R5 ;RESTORE R5
8166 033114 005215 INC (R5)
```

```
*****
:TEST:224 MUL 2 * #77777 = 0 177776 PS = 1
*****
```

```
TST224:
8173 033116 MOV #2,%0 ;LOAD MULTIPLICAND WITH 2
8174 033116 012700 000002 MUL #77777,%0 ;MULTIPLY 2 * #77777
8175 033122 070027 077777 MFPS @#PSWORD ;SAVE PS
8176 033126 106737 025414 025414 CMPB #1,@#PSWORD ;IS PS = 1
8177 033132 122737 000001 BEQ 11$
8178 033140 001401 EMT ;PS IS WRONG
8179 033142 104000 EMT ;IS HIGH ORDER = 0
8180 033144 022700 000000 11$: CMP #0,%0
8181 033150 001401 BEQ 12$
8182 033152 104000 EMT ;HIGH ORDER IS WRONG
8183 033154 022701 177776 177776 12$: CMP #177776,%0!1 ;IS LOW ORDER = 177776
8184 033160 001401 BEQ 13$
8185 033162 104000 EMT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
8186 033164 13$:
8187 033164 005215 INC (R5)
```

8188
8189
8190
8191
8192
8193
8194 033166
8195 033166 012702 007777
8196 033172 070227 000010
8197 033176 106737 025414
8198 033202 122737 000000 025414
8199 033210 001401
8200 033212 104000
8201 033214 022702 000000
8202 033220 001401
8203 033222 104000
8204 033224 022703 077770
8205 033230 001401
8206 033232 104000
8207 033234
8208 033234 005215
8209
8210
8211
8212
8213
8214
8215 033236
8216 033236 010501
8217 033240 012704 077777
8218 033244 070427 077777
8219 033250 106737 025414
8220 033254 122737 000001 025414
8221 033262 001401
8222 033264 104000
8223 033266 022704 037777
8224 033272 001401
8225 033274 104000
8226 033276 022705 000001
8227 033302 001401
8228 033304 104000
8229 033306
8230 033306 010105
8231 033310 005215
8232
8233
8234
8235
8236
8237
8238 033312
8239 033312 012702 177777
8240 033316 070227 077777
8241 033322 106737 025414
8242 033326 122737 000010 025414
8243 033334 001401

:TEST:225 MUL 7777 * #10 = 0 77770 PS = 0

TST225:
MOV #7777,%2 ;LOAD MULTIPLICAND WITH 7777
MUL #10,%2 ;MULTIPLY 7777 * #10
MFPS @#PSWORD ;SAVE PS
CMPB #0,@#PSWORD ;IS PS = 0
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #0,%2 ;IS HIGH ORDER = 0
BEQ 12\$
EMT ;HIGH ORDER IS WRONG
12\$: CMP #77770,%2!1 ;IS LOW ORDER = 77770
BEQ 13\$
EMT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
13\$: INC (R5)

:TEST:226 MUL 77777 * #77777 = 37777 1 PS = 1

TST226:
MOV R5,R1 ;SAVE R5
MOV #77777,%4 ;LOAD MULTIPLICAND WITH 77777
MUL #77777,%4 ;MULTIPLY 77777 * #77777
MFPS @#PSWORD ;SAVE PS
CMPB #1,@#PSWORD ;IS PS = 1
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #37777,%4 ;IS HIGH ORDER = 37777
BEQ 12\$
EMT ;HIGH ORDER IS WRONG
12\$: CMP #1,%4!1 ;IS LOW ORDER = 1
BEQ 13\$
EMT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
13\$: MOV R1,R5 ;RESTORE R5
INC (R5)

:TEST:227 MUL -1 * #77777 = -1 100001 PS = 10

TST227:
MOV #-1,%2 ;LOAD MULTIPLICAND WITH -1
MUL #77777,%2 ;MULTIPLY -1 * #77777
MFPS @#PSWORD ;SAVE PS
CMPB #10,@#PSWORD ;IS PS = 10
BEQ 11\$

8244 033336 104000
8245 033340 022702 177777
8246 033344 001401
8247 033346 104000
8248 033350 022703 100001
8249 033354 001401
8250 033356 104000
8251 033360
8252 033360 005215

11\$: EMT ;PS IS WRONG
CMP # -1,%2 ;IS HIGH ORDER = -1
BEQ 12\$
12\$: EMT ;HIGH ORDER IS WRONG
CMP #100001,%2!1 ;IS LOW ORDER = 100001
BEQ 13\$
EMT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
13\$: INC (R5)

8253
8254
8255
8256
8257

:*****
:TEST:230 MUL -2 * #77777 = -1 2 PS = 11
:*****

8258
8259 033362
8260 033362 012700 177776
8261 033366 070027 077777
8262 033372 106737 025414
8263 033376 122737 000011 025414
8264 033404 001401
8265 033406 104000
8266 033410 022700 177777
8267 033414 001401
8268 033416 104000
8269 033420 022701 000002
8270 033424 001401
8271 033426 104000
8272 033430
8273 033430 005215

TST230:
MOV #-2,%0 ;LOAD MULTIPLICAND WITH -2
MUL #77777,%0 ;MULTIPLY -2 * #77777
MFPS @#PSWORD ;SAVE PS
CMPB #11,@#PSWORD ;IS PS = 11
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP # -1,%0 ;IS HIGH ORDER = -1
BEQ 12\$
EMT ;HIGH ORDER IS WRONG
12\$: CMP #2,%0!1 ;IS LOW ORDER = 2
BEQ 13\$
EMT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
13\$: INC (R5)

8274
8275
8276
8277
8278
8279

:*****
:TEST:231 MUL 125252 * #2 = -1 52524 PS = 11
:*****

8280 033432
8281 033432 012702 125252
8282 033436 070227 000002
8283 033442 106737 025414
8284 033446 122737 000011 025414
8285 033454 001401
8286 033456 104000
8287 033460 022702 177777
8288 033464 001401
8289 033466 104000
8290 033470 022703 052524
8291 033474 001401
8292 033476 104000
8293 033500
8294 033500 005215

TST231:
MOV #125252,%2 ;LOAD MULTIPLICAND WITH 125252
MUL #2,%2 ;MULTIPLY 125252 * #2
MFPS @#PSWORD ;SAVE PS
CMPB #11,@#PSWORD ;IS PS = 11
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP # -1,%2 ;IS HIGH ORDER = -1
BEQ 12\$
EMT ;HIGH ORDER IS WRONG
12\$: CMP #52524,%2!1 ;IS LOW ORDER = 52524
BEQ 13\$
EMT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
13\$: INC (R5)

8295
8296
8297
8298
8299

:*****
:TEST:232 MUL 125252 * #40000 = 165252 100000 PS = 11
:*****

8300
8301 033502
8302 033502 010501
8303 033504 012704 125252
8304 033510 070427 040000
8305 033514 106737 025414
8306 033520 122737 000011 025414
8307 033526 001401
8308 033530 104000
8309 033532 022704 165252
8310 033536 001401
8311 033540 104000
8312 033542 022705 100000
8313 033546 001401
8314 033550 104000
8315 033552
8316 033552 010105
8317 033554 005215
8318
8319
8320

TST232:
MOV R5,R1 ;SAVE R5
MOV #125252,%4 ;LOAD MULTIPLICAND WITH 125252
MUL #40000,%4 ;MULTIPLY 125252 * #40000
MFPS @#PSWORD ;SAVE PS
CMPB #11,@#PSWORD ;IS PS = 11
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #165252,%4 ;IS HIGH ORDER = 165252
BEQ 12\$
EMT ;HIGH ORDER IS WRONG
12\$: CMP #100000,%4!1 ;IS LOW ORDER = 100000
BEQ 13\$
EMT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
13\$:
MOV R1,R5 ;RESTORE R5
INC (R5)

:TEST:233 MUL 107070 * #107070 = 31222 26100 PS = 1

8323
8324 033556
8325 033556 012700 107070
8326 033562 070027 107070
8327 033566 106737 025414
8328 033572 122737 000001 025414
8329 033600 001401
8330 033602 104000
8331 033604 022700 031222
8332 033610 001401
8333 033612 104000
8334 033614 022701 026100
8335 033620 001401
8336 033622 104000
8337 033624
8338 033624 005215
8339
8340

TST233:
MOV #107070,%0 ;LOAD MULTIPLICAND WITH 107070
MUL #107070,%0 ;MULTIPLY 107070 * #107070
MFPS @#PSWORD ;SAVE PS
CMPB #1,@#PSWORD ;IS PS = 1
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #31222,%0 ;IS HIGH ORDER = 31222
BEQ 12\$
EMT ;HIGH ORDER IS WRONG
12\$: CMP #26100,%0!1 ;IS LOW ORDER = 26100
BEQ 13\$
EMT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
13\$:
INC (R5)

:TEST:234 MUL -1 * #1 = -1 -1 PS = 10

8341
8342
8343
8344
8345 033626
8346 033626 012701 177777
8347 033632 070127 000001
8348 033636 106737 025414
8349 033642 122737 000010 025414
8350 033650 001401
8351 033652 104000
8352 033654 022701 177777
8353 033660 001401
8354 033662 104000
8355 033664 022701 177777

TST234:
MOV #-1,%1 ;LOAD MULTIPLICAND WITH -1
MUL #1,%1 ;MULTIPLY -1 * #1
MFPS @#PSWORD ;SAVE PS
CMPB #10,@#PSWORD ;IS PS = 10
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #-1,%1 ;IS HIGH ORDER = -1
BEQ 12\$
EMT ;HIGH ORDER IS WRONG
12\$: CMP #-1,%1!1 ;IS LOW ORDER = -1

8356 033670 001401
8357 033672 104000
8358 033674
8359 033674 005215
8360
8361
8362
8363
8364
8365
8366 033676
8367 033676 012703 177777
8368 033702 070327 000000
8369 033706 106737 025414
8370 033712 122737 000004 025414
8371 033720 001401
8372 033722 104000
8373 033724 022703 000000
8374 033730 001401
8375 033732 104000
8376 033734 022703 000000
8377 033740 001401
8378 033742 104000
8379 033744
8380 033744 005215
8381
8382
8383
8384
8385
8386
8387 033746
8388 033746 010501
8389 033750 012705 077777
8390 033754 070527 100000
8391 033760 106737 025414
8392 033764 122737 000011 025414
8393 033772 001401
8394 033774 104000
8395 033776 022705 100000
8396 034002 001401
8397 034004 104000
8398 034006 022705 100000
8399 034012 001401
8400 034014 104000
8401 034016
8402 034016 010105
8403 034020 005215
8404
8405
8406
8407
8408
8409
8410 034022
8411 034022 012701 177777

BEQ 13\$
EMT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
13\$: INC (R5)

:TEST:235 MUL -1 * #0 = 0 0 PS = 4

TST235:
MOV #-1,%3 ;LOAD MULTIPLICAND WITH -1
MUL #0,%3 ;MULTIPLY -1 * #0
MFPS @#PSWORD ;SAVE PS
CMPB #4,@#PSWORD ;IS PS = 4
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #0,%3 ;IS HIGH ORDER = 0
BEQ 12\$
EMT ;HIGH ORDER IS WRONG
12\$: CMP #0,%3!1 ;IS LOW ORDER = 0
BEQ 13\$
EMT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
13\$: INC (R5)

:TEST:236 MUL 77777 * #100000 = 100000 100000 PS = 11

TST236:
MOV R5,R1 ;SAVE R5
MOV #77777,%5 ;LOAD MULTIPLICAND WITH 77777
MUL #100000,%5 ;MULTIPLY 77777 * #100000
MFPS @#PSWORD ;SAVE PS
CMPB #11,@#PSWORD ;IS PS = 11
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #100000,%5 ;IS HIGH ORDER = 100000
BEQ 12\$
EMT ;HIGH ORDER IS WRONG
12\$: CMP #100000,%5!1 ;IS LOW ORDER = 100000
BEQ 13\$
EMT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
13\$: MOV R1,R5 ;RESTORE R5
INC (R5)

:TEST:237 MUL -1 * #77777 = 100001 100001 PS = 10

TST237:
MOV #-1,%1 ;LOAD MULTIPLICAND WITH -1

8412 034026 070127 077777
8413 034032 106737 025414
8414 034036 122737 000010 025414
8415 034044 001401
8416 034046 104000
8417 034050 022701 100001
8418 034054 001401
8419 034056 104000
8420 034060 022701 100001
8421 034064 001401
8422 034066 104000
8423 034070
8424 034070 005215

MUL #77777,X1 ;MULTIPLY -1 * #77777
MFPS @#PSWORD ;SAVE PS
CMPB #10,@#PSWORD ;IS PS = 10
BEQ 11\$
EMT ;PS IS WRONG
CMP #100001,X1 ;IS HIGH ORDER = 100001
BEQ 12\$
EMT ;HIGH ORDER IS WRONG
CMP #100001,X1!1 ;IS LOW ORDER = 100001
BEQ 13\$
EMT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
13\$: INC (R5)

:TEST:240 MUL 77777 * #77777 = 1 1 PS = 1

8431 034072
8432 034072 012703 077777
8433 034076 070327 077777
8434 034102 106737 025414
8435 034106 122737 000001 025414
8436 034114 001401
8437 034116 104000
8438 034120 022703 000001
8439 034124 001401
8440 034126 104000
8441 034130 022703 000001
8442 034134 001401
8443 034136 104000
8444 034140
8445 034140 005215

TST240:
MOV #77777,X3 ;LOAD MULTIPLICAND WITH 77777
MUL #77777,X3 ;MULTIPLY 77777 * #77777
MFPS @#PSWORD ;SAVE PS
CMPB #1,@#PSWORD ;IS PS = 1
BEQ 11\$
EMT ;PS IS WRONG
CMP #1,X3 ;IS HIGH ORDER = 1
BEQ 12\$
EMT ;HIGH ORDER IS WRONG
CMP #1,X3!1 ;IS LOW ORDER = 1
BEQ 13\$
EMT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
13\$: INC (R5)

:TEST:241 MUL 2 * #2 = 4 4 PS = 0

8452 034142
8453 034142 010501
8454 034144 012705 000002
8455 034150 070527 000002
8456 034154 106737 025414
8457 034160 122737 000000 025414
8458 034166 001401
8459 034170 104000
8460 034172 022705 000004
8461 034176 001401
8462 034200 104000
8463 034202 022705 000004
8464 034206 001401
8465 034210 104000
8466 034212
8467 034212 010105

TST241:
MOV R5,R1 ;SAVE R5
MOV #2,X5 ;LOAD MULTIPLICAND WITH 2
MUL #2,X5 ;MULTIPLY 2 * #2
MFPS @#PSWORD ;SAVE PS
CMPB #0,@#PSWORD ;IS PS = 0
BEQ 11\$
EMT ;PS IS WRONG
CMP #4,X5 ;IS HIGH ORDER = 4
BEQ 12\$
EMT ;HIGH ORDER IS WRONG
CMP #4,X5!1 ;IS LOW ORDER = 4
BEQ 13\$
EMT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
13\$: MOV R1,R5 ;RESTORE R5

8468 034214 005215
8469
8470
8471 034216 012702 040000
8472 034222 012703 025442
8473 034226 012704 025444
8474

INC (R5)

MOV #40000,%2
MOV #S5,%3
MOV #S6,%4

:TEST:242 MUL 125252 * S5 = 165252 100000 PS = 11

8478
8479 034232
8480 034232 012700 125252
8481 034236 070067 171200
8482 034242 106737 025414
8483 034246 122737 000011 025414
8484 034254 001401
8485 034256 104000
8486 034260 022700 165252
8487 034264 001401
8488 034266 104000
8489 034270 022701 100000
8490 034274 001401
8491 034276 104000
8492 034300
8493 034300 005215
8494
8495
8496
8497
8498
8499

TST242:
MOV #125252,%0 ;LOAD MULTIPLICAND WITH 125252
MUL S5,%0 ;MULTIPLY 125252 * S5
MFPS @#PSWORD ;SAVE PS
CMPB #11,@#PSWORD ;IS PS = 11
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #165252,%0 ;IS HIGH ORDER = 165252
BEQ 12\$
EMT ;HIGH ORDER IS WRONG
12\$: CMP #100000,%0!1 ;IS LOW ORDER = 100000
BEQ 13\$
EMT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
13\$: INC (R5)

:TEST:243 MUL 125252 * @S6 = 165252 100000 PS = 11

8500 034302
8501 034302 012700 125252
8502 034306 070077 171132
8503 034312 106737 025414
8504 034316 122737 000011 025414
8505 034324 001401
8506 034326 104000
8507 034330 022700 165252
8508 034334 001401
8509 034336 104000
8510 034340 022701 100000
8511 034344 001401
8512 034346 104000
8513 034350
8514 034350 005215
8515
8516
8517

TST243:
MOV #125252,%0 ;LOAD MULTIPLICAND WITH 125252
MUL @S6,%0 ;MULTIPLY 125252 * @S6
MFPS @#PSWORD ;SAVE PS
CMPB #11,@#PSWORD ;IS PS = 11
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #165252,%0 ;IS HIGH ORDER = 165252
BEQ 12\$
EMT ;HIGH ORDER IS WRONG
12\$: CMP #100000,%0!1 ;IS LOW ORDER = 100000
BEQ 13\$
EMT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
13\$: INC (R5)

:TEST:244 MUL 125252 * @#S5 = 165252 100000 PS = 11

8520
8521 034352
8522 034352 012700 125252
8523 034356 070037 025442

TST244:
MOV #125252,%0 ;LOAD MULTIPLICAND WITH 125252
MUL @#S5,%0 ;MULTIPLY 125252 * @#S5

8524 034362 106737 025414
8525 034366 122737 000011 025414
8526 034374 001401
8527 034376 104000
8528 034400 022700 165252
8529 034404 001401
8530 034406 104000
8531 034410 022701 100000
8532 034414 001401
8533 034416 104000
8534 034420
8535 034420 005215

MFPS @#PSWORD :SAVE PS
CMPB #11,@#PSWORD :IS PS = 11
BEQ 11\$
EMT :PS IS WRONG
CMP #165252,%0 :IS HIGH ORDER = 165252
BEQ 12\$
EMT :HIGH ORDER IS WRONG
CMP #100000,%0!1 :IS LOW ORDER = 100000
BEQ 13\$
EMT :LOW ORDER IS WRONG OR WRONG SEQUENCE
INC (R5)

8536
8537
8538
8539
8540
8541

:TEST:245 MUL 125252 * %2 = 165252 100000 PS = 11

8542 034422
8543 034422 012700 125252
8544 034426 070002
8545 034430 106737 025414
8546 034434 122737 000011 025414
8547 034442 001401
8548 034444 104000
8549 034446 022700 165252
8550 034452 001401
8551 034454 104000
8552 034456 022701 100000
8553 034462 001401
8554 034464 104000
8555 034466
8556 034466 005215

TST245:
MOV #125252,%0 :LOAD MULTIPLICAND WITH 125252
MUL %2,%0 :MULTIPLY 125252 * %2
MFPS @#PSWORD :SAVE PS
CMPB #11,@#PSWORD :IS PS = 11
BEQ 11\$
EMT :PS IS WRONG
CMP #165252,%0 :IS HIGH ORDER = 165252
BEQ 12\$
EMT :HIGH ORDER IS WRONG
CMP #100000,%0!1 :IS LOW ORDER = 100000
BEQ 13\$
EMT :LOW ORDER IS WRONG OR WRONG SEQUENCE
INC (R5)

8557
8558
8559
8560
8561
8562

:TEST:246 MUL 125252 * (3)+ = 165252 100000 PS = 11

8563 034470
8564 034470 012700 125252
8565 034474 070023
8566 034476 106737 025414
8567 034502 122737 000011 025414
8568 034510 001401
8569 034512 104000
8570 034514 022700 165252
8571 034520 001401
8572 034522 104000
8573 034524 022701 100000
8574 034530 001401
8575 034532 104000
8576 034534
8577 034534 005215

TST246:
MOV #125252,%0 :LOAD MULTIPLICAND WITH 125252
MUL (3)+,%0 :MULTIPLY 125252 * (3)+
MFPS @#PSWORD :SAVE PS
CMPB #11,@#PSWORD :IS PS = 11
BEQ 11\$
EMT :PS IS WRONG
CMP #165252,%0 :IS HIGH ORDER = 165252
BEQ 12\$
EMT :HIGH ORDER IS WRONG
CMP #100000,%0!1 :IS LOW ORDER = 100000
BEQ 13\$
EMT :LOW ORDER IS WRONG OR WRONG SEQUENCE
INC (R5)

8578
8579

8580
8581
8582
8583
8584 034536
8585 034536 012700 125252
8586 034542 070043
8587 034544 106737 025414
8588 034550 122737 000011 025414
8589 034556 001401
8590 034560 104000
8591 034562 022700 165252
8592 034566 001401
8593 034570 104000
8594 034572 022701 100000
8595 034576 001401
8596 034600 104000
8597 034602
8598 034602 005215
8599
8600
8601
8602
8603
8604
8605 034604
8606 034604 012700 125252
8607 034610 070064 000002
8608 034614 106737 025414
8609 034620 122737 000011 025414
8610 034626 001401
8611 034630 104000
8612 034632 022700 165252
8613 034636 001401
8614 034640 104000
8615 034642 022701 100000
8616 034646 001401
8617 034650 104000
8618 034652
8619 034652 005215
8620
8621
8622
8623
8624
8625
8626 034654
8627 034654 012700 125252
8628 034660 070074 000000
8629 034664 106737 025414
8630 034670 122737 000011 025414
8631 034676 001401
8632 034700 104000
8633 034702 022700 165252
8634 034706 001401
8635 034710 104000

:TEST:247 MUL 125252 * -(3) = 165252 100000 PS = 11

TST247:
MOV #125252,%0 ;LOAD MULTIPLICAND WITH 125252
MUL -(3),%0 ;MULTIPLY 125252 * -(3)
MFPS @#PSWORD ;SAVE PS
CMPB #11,@#PSWORD ;IS PS = 11
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #165252,%0 ;IS HIGH ORDER = 165252
BEQ 12\$
EMT ;HIGH ORDER IS WRONG
12\$: CMP #100000,%0!1 ;IS LOW ORDER = 100000
BEQ 13\$
EMT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
13\$: INC (R5)

:TEST:250 MUL 125252 * 2(4) = 165252 100000 PS = 11

TST250:
MOV #125252,%0 ;LOAD MULTIPLICAND WITH 125252
MUL 2(4),%0 ;MULTIPLY 125252 * 2(4)
MFPS @#PSWORD ;SAVE PS
CMPB #11,@#PSWORD ;IS PS = 11
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #165252,%0 ;IS HIGH ORDER = 165252
BEQ 12\$
EMT ;HIGH ORDER IS WRONG
12\$: CMP #100000,%0!1 ;IS LOW ORDER = 100000
BEQ 13\$
EMT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
13\$: INC (R5)

:TEST:251 MUL 125252 * @ (4) = 165252 100000 PS = 11

TST251:
MOV #125252,%0 ;LOAD MULTIPLICAND WITH 125252
MUL @ (4),%0 ;MULTIPLY 125252 * @ (4)
MFPS @#PSWORD ;SAVE PS
CMPB #11,@#PSWORD ;IS PS = 11
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #165252,%0 ;IS HIGH ORDER = 165252
BEQ 12\$
EMT ;HIGH ORDER IS WRONG

8636 034712 022701 100000
8637 034716 001401
8638 034720 104000
8639 034722
8640 034722 005215
8641
8642
8643
8644
8645
8646
8647 034724
8648 034724 012700 125252
8649 034730 070034
8650 034732 106737 025414
8651 034736 122737 000011 025414
8652 034744 001401
8653 034746 104000
8654 034750 022700 165252
8655 034754 001401
8656 034756 104000
8657 034760 022701 100000
8658 034764 001401
8659 034766 104000
8660 034770
8661 034770 005215
8662
8663
8664
8665
8666
8667
8668 034772
8669 034772 012700 125252
8670 034776 070054
8671 035000 106737 025414
8672 035004 122737 000011 025414
8673 035012 001401
8674 035014 104000
8675 035016 022700 165252
8676 035022 001401
8677 035024 104000
8678 035026 022701 100000
8679 035032 001401
8680 035034 104000
8681 035036
8682 035036 005215
8683
8684

12\$: CMP #100000,%0!1 ;IS LOW ORDER = 100000
BEQ 13\$
EMT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
13\$: INC (R5)

:TEST:252 MUL 125252 * @ (4)+ = 165252 100000 PS = 11

TST252:
MOV #125252,%0 ;LOAD MULTIPLICAND WITH 125252
MUL @ (4)+,%0 ;MULTIPLY 125252 * @ (4)+
MFPS @#PSWORD ;SAVE PS
CMPB #11,@#PSWORD ;IS PS = 11
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #165252,%0 ;IS HIGH ORDER = 165252
BEQ 12\$
EMT ;HIGH ORDER IS WRONG
12\$: CMP #100000,%0!1 ;IS LOW ORDER = 100000
BEQ 13\$
EMT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
13\$: INC (R5)

:TEST:253 MUL 125252 * @-(4) = 165252 100000 PS = 11

TST253:
MOV #125252,%0 ;LOAD MULTIPLICAND WITH 125252
MUL @-(4),%0 ;MULTIPLY 125252 * @-(4)
MFPS @#PSWORD ;SAVE PS
CMPB #11,@#PSWORD ;IS PS = 11
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #165252,%0 ;IS HIGH ORDER = 165252
BEQ 12\$
EMT ;HIGH ORDER IS WRONG
12\$: CMP #100000,%0!1 ;IS LOW ORDER = 100000
BEQ 13\$
EMT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
13\$: INC (R5)

: DIV INSTRUCTION TESTS

: TEST:254 DIV 0 4 / #2 = 2 REM = 0 PS = 0

TST254:
MOV #0,%0 ;LOAD HIGH ORDER WITH 0
MOV #4,%0+1 ;LOAD LOW ORDER WITH 4
DIV #2,%0 ;DIVIDE BY #2
MFPS @#PSWORD ;SAVE PS
CMPB #0,@#PSWORD ;IS PS = 0
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #2,%0 ;IS QUOTIENT = 2
BEQ 12\$
EMT ;QUOTIENT IS WRONG
12\$: CMP #0,%0+1 ;IS REMAINDER = 0
BEQ 13\$
EMT ;WRONG REMAINDER
13\$: INC (R5)

: TEST:255 DIV -1 -9. / #3 = -3 REM = 0 PS = 10

TST255:
MOV #-1,%2 ;LOAD HIGH ORDER WITH -1
MOV #-9,%2+1 ;LOAD LOW ORDER WITH -9.
DIV #3,%2 ;DIVIDE BY #3
MFPS @#PSWORD ;SAVE PS
CMPB #10,@#PSWORD ;IS PS = 10
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #-3,%2 ;IS QUOTIENT = -3
BEQ 12\$
EMT ;QUOTIENT IS WRONG
12\$: CMP #0,%2+1 ;IS REMAINDER = 0
BEQ 13\$
EMT ;WRONG REMAINDER
13\$: INC (R5)

: TEST:256 DIV 0 9. / #2 = 4 REM = 1 PS = 0

TST256:
MOV R5,R1 ;SAVE R5
MOV #0,%4 ;LOAD HIGH ORDER WITH 0
MOV #9,%4+1 ;LOAD LOW ORDER WITH 9.

8685
8686
8687
8688
8689
8690
8691
8692
8693
8694
8695 035040
8696 035040 012700 000000
8697 035044 012701 000004
8698 035050 071027 000002
8699 035054 106737 025414
8700 035060 122737 000000 025414
8701 035066 001401
8702 035070 104000
8703 035072 022700 000002
8704 035076 001401
8705 035100 104000
8706 035102 022701 000000
8707 035106 001401
8708 035110 104000
8709 035112
8710 035112 005215
8711
8712
8713
8714
8715
8716 035114
8717 035114 012702 177777
8718 035120 012703 177767
8719 035124 071227 000003
8720 035130 106737 025414
8721 035134 122737 000010 025414
8722 035142 001401
8723 035144 104000
8724 035146 022702 177775
8725 035152 001401
8726 035154 104000
8727 035156 022703 000000
8728 035162 001401
8729 035164 104000
8730 035166
8731 035166 005215
8732
8733
8734
8735
8736
8737 035170
8738 035170 010501
8739 035172 012704 000000
8740 035176 012705 000011

8741 035202 071427 000002
8742 035206 106737 025414
8743 035212 122737 000000 025414
8744 035220 001401
8745 035222 104000
8746 035224 022704 000004
8747 035230 001401
8748 035232 104000
8749 035234 022705 000001
8750 035240 001401
8751 035242 104000
8752 035244
8753 035244 010105
8754 035246 005215

DIV #2,%4 ;DIVIDE BY #2
MFPS @#PSWORD ;SAVE PS
CMPB #0,@#PSWORD ;IS PS = 0
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #4,%4 ;IS QUOTIENT = 4
BEQ 12\$
EMT ;QUOTIENT IS WRONG
12\$: CMP #1,%4+1 ;IS REMAINDER = 1
BEQ 13\$
EMT ;WRONG REMAINDER
13\$: MOV R1,R5 ;RESTORE R5
INC (R5)

:TEST:257 DIV -1 -9. / #2 = -4 REM = -1 PS = 10

8760 035250
8761 035250 012700 177777
8762 035254 012701 177767
8763 035260 071027 000002
8764 035264 106737 025414
8765 035270 122737 000010 025414
8766 035276 001401
8767 035300 104000
8768 035302 022700 177774
8769 035306 001401
8770 035310 104000
8771 035312 022701 177777
8772 035316 001401
8773 035320 104000
8774 035322
8775 035322 005215

TST257:
MOV #-1,%0 ;LOAD HIGH ORDER WITH -1
MOV #-9,%0+1 ;LOAD LOW ORDER WITH -9.
DIV #2,%0 ;DIVIDE BY #2
MFPS @#PSWORD ;SAVE PS
CMPB #10,@#PSWORD ;IS PS = 10
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #-4,%0 ;IS QUOTIENT = -4
BEQ 12\$
EMT ;QUOTIENT IS WRONG
12\$: CMP #-1,%0+1 ;IS REMAINDER = -1
BEQ 13\$
EMT ;WRONG REMAINDER
13\$: INC (R5)

:TEST:260 DIV 0 2 / #-3 = 0 REM = 2 PS = 4

8781 035324
8782 035324 012702 000000
8783 035330 012703 000002
8784 035334 071227 177775
8785 035340 106737 025414
8786 035344 122737 000004 025414
8787 035352 001401
8788 035354 104000
8789 035356 022702 000000
8790 035362 001401
8791 035364 104000
8792 035366 022703 000002
8793 035372 001401
8794 035374 104000
8795 035376
8796 035376 005215

TST260:
MOV #0,%2 ;LOAD HIGH ORDER WITH 0
MOV #2,%2+1 ;LOAD LOW ORDER WITH 2
DIV #-3,%2 ;DIVIDE BY #-3
MFPS @#PSWORD ;SAVE PS
CMPB #4,@#PSWORD ;IS PS = 4
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #0,%2 ;IS QUOTIENT = 0
BEQ 12\$
EMT ;QUOTIENT IS WRONG
12\$: CMP #2,%2+1 ;IS REMAINDER = 2
BEQ 13\$
EMT ;WRONG REMAINDER
13\$: INC (R5)

8797
8798
8799
8800
8801
8802 035400
8803 035400 010501
8804 035402 012704 177777
8805 035406 012705 177776
8806 035412 071427 000003
8807 035416 106737 025414
8808 035422 122737 000004 025414
8809 035430 001401
8810 035432 104000
8811 035434 022704 000000
8812 035440 001401
8813 035442 104000
8814 035444 022705 177776
8815 035450 001401
8816 035452 104000
8817 035454
8818 035454 010105
8819 035456 005215
8820
8821
8822
8823
8824
8825 035460
8826 035460 012700 177777
8827 035464 012701 177777
8828 035470 071027 000001
8829 035474 106737 025414
8830 035500 122737 000010 025414
8831 035506 001401
8832 035510 104000
8833 035512 022700 177777
8834 035516 001401
8835 035520 104000
8836 035522 022701 000000
8837 035526 001401
8838 035530 104000
8839 035532
8840 035532 005215
8841
8842
8843
8844
8845
8846 035534
8847 035534 012700 000000
8848 035540 012701 000000
8849 035544 071027 000001
8850 035550 106737 025414
8851 035554 122737 000004 025414
8852 035562 001401

:TEST:261 DIV -1 -2 / #3 = 0 REM = -2 PS = 4

TST261:
MOV R5,R1 ;SAVE R5
MOV #-1,%4 ;LOAD HIGH ORDER WITH -1
MOV #-2,%4+1 ;LOAD LOW ORDER WITH -2
DIV #3,%4 ;DIVIDE BY #3
MFPS @#PSWORD ;SAVE PS
CMPB #4,@#PSWORD ;IS PS = 4
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #0,%4 ;IS QUOTIENT = 0
BEQ 12\$
EMT ;QUOTIENT IS WRONG
12\$: CMP #-2,%4+1 ;IS REMAINDER = -2
BEQ 13\$
EMT ;WRONG REMAINDER
13\$: MOV R1,R5 ;RESTORE R5
INC (R5)

:TEST:262 DIV -1 -1 / #1 = -1 REM = 0 PS = 10

TST262:
MOV #-1,%0 ;LOAD HIGH ORDER WITH -1
MOV #-1,%0+1 ;LOAD LOW ORDER WITH -1
DIV #1,%0 ;DIVIDE BY #1
MFPS @#PSWORD ;SAVE PS
CMPB #10,@#PSWORD ;IS PS = 10
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #-1,%0 ;IS QUOTIENT = -1
BEQ 12\$
EMT ;QUOTIENT IS WRONG
12\$: CMP #0,%0+1 ;IS REMAINDER = 0
BEQ 13\$
EMT ;WRONG REMAINDER
13\$: INC (R5)

:TEST:263 DIV 0 0 / #1 = 0 REM = 0 PS = 4

TST263:
MOV #0,%0 ;LOAD HIGH ORDER WITH 0
MOV #0,%0+1 ;LOAD LOW ORDER WITH 0
DIV #1,%0 ;DIVIDE BY #1
MFPS @#PSWORD ;SAVE PS
CMPB #4,@#PSWORD ;IS PS = 4
BEQ 11\$

8853 035564 104000
8854 035566 022700 000000
8855 035572 001401
8856 035574 104000
8857 035576 022701 000000
8858 035602 001401
8859 035604 104000
8860 035606
8861 035606 005215

11\$: EMT ;PS IS WRONG
CMP #0,X0 ;IS QUOTIENT = 0
BEQ 12\$
12\$: EMT ;QUOTIENT IS WRONG
CMP #0,X0+1 ;IS REMAINDER = 0
BEQ 13\$
EMT ;WRONG REMAINDER
13\$: INC (R5)

8862
8863 ;*****
8864 ;TEST:264 DIV -1 125252 / #2 = 152525 REM = 0 PS = 10
8865 ;*****
8866

8867 035610
8868 035610 012702 177777
8869 035614 012703 125252
8870 035620 071227 000002
8871 035624 106737 025414
8872 035630 122737 000010 025414
8873 035636 001401
8874 035640 104000
8875 035642 022702 152525
8876 035646 001401
8877 035650 104000
8878 035652 022703 000000
8879 035656 001401
8880 035660 104000
8881 035662
8882 035662 005215

TST264:
MOV #-1,X2 ;LOAD HIGH ORDER WITH -1
MOV #125252,X2+1 ;LOAD LOW ORDER WITH 125252
DIV #2,X2 ;DIVIDE BY #2
MFPS @#PSWORD ;SAVE PS
CMPB #10,@#PSWORD ;IS PS = 10
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #152525,X2 ;IS QUOTIENT = 152525
BEQ 12\$
EMT ;QUOTIENT IS WRONG
12\$: CMP #0,X2+1 ;IS REMAINDER = 0
BEQ 13\$
EMT ;WRONG REMAINDER
13\$: INC (R5)

8883
8884 ;*****
8885 ;TEST:265 DIV -1 -1 / #-1 = 1 REM = 0 PS = 0
8886 ;*****
8887

8888 035664
8889 035664 010501
8890 035666 012704 177777
8891 035672 012705 177777
8892 035676 071427 177777
8893 035702 106737 025414
8894 035706 122737 000000 025414
8895 035714 001401
8896 035716 104000
8897 035720 022704 000001
8898 035724 001401
8899 035726 104000
8900 035730 022705 000000
8901 035734 001401
8902 035736 104000
8903 035740
8904 035740 010105
8905 035742 005215

TST265:
MOV R5,R1 ;SAVE R5
MOV #-1,X4 ;LOAD HIGH ORDER WITH -1
MOV #-1,X4+1 ;LOAD LOW ORDER WITH -1
DIV #-1,X4 ;DIVIDE BY #-1
MFPS @#PSWORD ;SAVE PS
CMPB #0,@#PSWORD ;IS PS = 0
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #1,X4 ;IS QUOTIENT = 1
BEQ 12\$
EMT ;QUOTIENT IS WRONG
12\$: CMP #0,X4+1 ;IS REMAINDER = 0
BEQ 13\$
EMT ;WRONG REMAINDER
13\$: MOV R1,R5 ;RESTORE R5
INC (R5)

8906
8907 ;*****
8908 ;TEST:266 DIV 25253 1 / #125252 = 100000 REM = 1 PS = 10

8909
8910
8911 035744
8912 035744 012700 025253
8913 035750 012701 000001
8914 035754 071027 125252
8915 035760 106737 025414
8916 035764 122737 000010 025414
8917 035772 001401
8918 035774 104000
8919 035776 022700 100000
8920 036002 001401
8921 036004 104000
8922 036006 022701 000001
8923 036012 001401
8924 036014 104000
8925 036016
8926 036016 005215
8927
8928
8929
8930
8931
8932 036020
8933 036020 012702 037777
8934 036024 012703 077777
8935 036030 071227 077777
8936 036034 106737 025414
8937 036040 122737 000000 025414
8938 036046 001401
8939 036050 104000
8940 036052 022702 077777
8941 036056 001401
8942 036060 104000
8943 036062 022703 077776
8944 036066 001401
8945 036070 104000
8946 036072
8947 036072 005215
8948
8949
8950
8951
8952
8953 036074
8954 036074 010501
8955 036076 012704 000000
8956 036102 012705 100000
8957 036106 071427 000002
8958 036112 106737 025414
8959 036116 122737 000000 025414
8960 036124 001401
8961 036126 104000
8962 036130 022704 040000
8963 036134 001401
8964 036136 104000

:*****

TST266:
MOV #25253,%0 ;LOAD HIGH ORDER WITH 25253
MOV #1,%0+1 ;LOAD LOW ORDER WITH 1
DIV #125252,%0 ;DIVIDE BY #125252
MFPS @#PSWORD ;SAVE PS
CMPB #10,@#PSWORD ;IS PS = 10
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #100000,%0 ;IS QUOTIENT = 100000
BEQ 12\$
EMT ;QUOTIENT IS WRONG
12\$: CMP #1,%0+1 ;IS REMAINDER = 1
BEQ 13\$
EMT ;WRONG REMAINDER
13\$: INC (R5)

:*****
:TEST:267 DIV 37777 77777 / #77777 = 77777 REM = 77776 PS = 0
:*****

TST267:
MOV #37777,%2 ;LOAD HIGH ORDER WITH 37777
MOV #77777,%2+1 ;LOAD LOW ORDER WITH 77777
DIV #77777,%2 ;DIVIDE BY #77777
MFPS @#PSWORD ;SAVE PS
CMPB #0,@#PSWORD ;IS PS = 0
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #77777,%2 ;IS QUOTIENT = 77777
BEQ 12\$
EMT ;QUOTIENT IS WRONG
12\$: CMP #77776,%2+1 ;IS REMAINDER = 77776
BEQ 13\$
EMT ;WRONG REMAINDER
13\$: INC (R5)

:*****
:TEST:270 DIV 0 100000 / #2 = 40000 REM = 0 PS = 0
:*****

TST270:
MOV R5,R1 ;SAVE R5
MOV #0,%4 ;LOAD HIGH ORDER WITH 0
MOV #100000,%4+1 ;LOAD LOW ORDER WITH 100000
DIV #2,%4 ;DIVIDE BY #2
MFPS @#PSWORD ;SAVE PS
CMPB #0,@#PSWORD ;IS PS = 0
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #40000,%4 ;IS QUOTIENT = 40000
BEQ 12\$
EMT ;QUOTIENT IS WRONG

8965 036140 022705 000000
8966 036144 001401
8967 036146 104000
8968 036150
8969 036150 010105
8970 036152 005215

12\$: CMP #0,%4+1 ;IS REMAINDER = 0
BEQ 13\$
EMT ;WRONG REMAINDER
13\$: MOV R1,R5 ;RESTORE R5
INC (R5)

8971
8972
8973
8974
8975

:TEST:271 DIV 177777 77777 / #177776 = 40000 REM = 177777 PS = 0

8976 036154
8977 036154 012700 177777
8978 036160 012701 077777
8979 036164 071027 177776
8980 036170 106737 025414
8981 036174 122737 000000 025414
8982 036202 001401
8983 036204 104000
8984 036206 022700 040000
8985 036212 001401
8986 036214 104000
8987 036216 022701 177777
8988 036222 001401
8989 036224 104000
8990 036226
8991 036226 005215

TST271:
MOV #177777,%0 ;LOAD HIGH ORDER WITH 177777
MOV #77777,%0+1 ;LOAD LOW ORDER WITH 77777
DIV #177776,%0 ;DIVIDE BY #177776
MFPS @MPSWORD ;SAVE PS
CMPB #0,@MPSWORD ;IS PS = 0
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #40000,%0 ;IS QUOTIENT = 40000
BEQ 12\$
EMT ;QUOTIENT IS WRONG
12\$: CMP #177777,%0+1 ;IS REMAINDER = 177777
BEQ 13\$
EMT ;WRONG REMAINDER
13\$: INC (R5)

8992
8993
8994
8995
8996

:TEST:272 DIV 0 52525 / #52525 = 1 REM = 0 PS = 0

8997 036230
8998 036230 012702 000000
8999 036234 012703 052525
9000 036240 071227 052525
9001 036244 106737 025414
9002 036250 122737 000000 025414
9003 036256 001401
9004 036260 104000
9005 036262 022702 000001
9006 036266 001401
9007 036270 104000
9008 036272 022703 000000
9009 036276 001401
9010 036300 104000
9011 036302
9012 036302 005215

TST272:
MOV #0,%2 ;LOAD HIGH ORDER WITH 0
MOV #52525,%2+1 ;LOAD LOW ORDER WITH 52525
DIV #52525,%2 ;DIVIDE BY #52525
MFPS @MPSWORD ;SAVE PS
CMPB #0,@MPSWORD ;IS PS = 0
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #1,%2 ;IS QUOTIENT = 1
BEQ 12\$
EMT ;QUOTIENT IS WRONG
12\$: CMP #0,%2+1 ;IS REMAINDER = 0
BEQ 13\$
EMT ;WRONG REMAINDER
13\$: INC (R5)

9013
9014
9015
9016
9017

:TEST:273 DIV 0 77777 / #0 = DUMMY REM = DUMMY PS = 3

9018 036304
9019 036304 010501
9020 036306 012704 000000

TST273:
MOV R5,R1 ;SAVE R5
MOV #0,%4 ;LOAD HIGH ORDER WITH 0

9021 036312 012705 077777
9022 036316 071427 000000
9023 036322 106737 025414
9024 036326 042737 000014 025414
9025 036334 122737 000003 025414
9026 036342 001401
9027 036344 104000
9028 036346
9029 036346 010105
9030 036350 C05215

MOV #77777,%4+1 ;LOAD LOW ORDER WITH 77777
DIV #0,%4 ;DIVIDE BY #0
MFPS @#PSWORD ;SAVE PS
BIC #14,@#PSWORD
CMPB #3,@#PSWORD ;IS PS = 3
BEQ 13\$
EMT ;PS IS WRONG
13\$: MOV R1,R5 ;RESTORE R5
INC (R5)

:TEST:274 DIV 77777 177777 / #2 = DUMMY REM = DUMMY PS = 2

9031
9032
9033
9034
9035
9036 036352
9037 036352 012700 077777
9038 036356 012701 177777
9039 036362 071027 000002
9040 036366 106737 025414
9041 036372 042737 000014 025414
9042 036400 122737 000002 025414
9043 036406 001401
9044 036410 104000
9045 036412
9046 036412 005215
9047
9048 036414 012702 000002
9049 036420 012703 025452
9050 036424 012704 025454
9051
9052
9053
9054
9055
9056 036430
9057 036430 012700 000000
9058 036434 012701 052525
9059 036440 071067 167006
9060 036444 106737 025414
9061 036450 122737 000000 025414
9062 036456 001401
9063 036460 104000
9064 036462 022700 025252
9065 036466 001401
9066 036470 104000
9067 036472 022701 000001
9068 036476 001401
9069 036500 104000
9070 036502
9071 036502 005215
9072
9073
9074
9075
9076

TST274:
MOV #77777,%0 ;LOAD HIGH ORDER WITH 77777
MOV #177777,%0+1 ;LOAD LOW ORDER WITH 177777
DIV #2,%0 ;DIVIDE BY #2
MFPS @#PSWORD ;SAVE PS
BIC #14,@#PSWORD
CMPB #2,@#PSWORD ;IS PS = 2
BEQ 13\$
EMT ;PS IS WRONG
13\$: INC (R5)

MOV #2,%2
MOV #S9,%3
MOV #S10,%4

:TEST:275 DIV 0 52525 / S9 = 25252 REM = 1 PS = 0

TST275:
MOV #0,%0 ;LOAD HIGH ORDER WITH 0
MOV #52525,%0+1 ;LOAD LOW ORDER WITH 52525
DIV S9,%0 ;DIVIDE BY S9
MFPS @#PSWORD ;SAVE PS
CMPB #0,@#PSWORD ;IS PS = 0
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #25252,%0 ;IS QUOTIENT = 25252
BEQ 12\$
EMT ;QUOTIENT IS WRONG
12\$: CMP #1,%0+1 ;IS REMAINDER = 1
BEQ 13\$
EMT ;WRONG REMAINDER
13\$: INC (R5)

:TEST:276 DIV 0 52525 / @S10 = 25252 REM = 1 PS = 0

9077 036504
9078 036504 012700 000000
9079 036510 012701 052525
9080 036514 071077 166734
9081 036520 106737 025414
9082 036524 122737 000000 025414
9083 036532 001401
9084 036534 104000
9085 036536 022700 025252
9086 036542 001401
9087 036544 104000
9088 036546 022701 000001
9089 036552 001401
9090 036554 104000
9091 036556
9092 036556 005215
9093
9094
9095
9096
9097

TST276:
MOV #0,%0 ;LOAD HIGH ORDER WITH 0
MOV #52525,%0+1 ;LOAD LOW ORDER WITH 52525
DIV @S10,%0 ;DIVIDE BY @S10
MFPS @MPSWORD ;SAVE PS
CMPB #0,@MPSWORD ;IS PS = 0
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #25252,%0 ;IS QUOTIENT = 25252
BEQ 12\$
EMT ;QUOTIENT IS WRONG
12\$: CMP #1,%0+1 ;IS REMAINDER = 1
BEQ 13\$
EMT ;WRONG REMAINDER
13\$: INC (R5)

:TEST:277 DIV 0 52525 / @MPS9 = 25252 REM = 1 PS = 0

9098 036560
9099 036560 012700 000000
9100 036564 012701 052525
9101 036570 071037 025452
9102 036574 106737 025414
9103 036600 122737 000000 025414
9104 036606 001401
9105 036610 104000
9106 036612 022700 025252
9107 036616 001401
9108 036620 104000
9109 036622 022701 000001
9110 036626 001401
9111 036630 104000
9112 036632
9113 036632 005215
9114
9115
9116
9117
9118

TST277:
MOV #0,%0 ;LOAD HIGH ORDER WITH 0
MOV #52525,%0+1 ;LOAD LOW ORDER WITH 52525
DIV @MPS9,%0 ;DIVIDE BY @MPS9
MFPS @MPSWORD ;SAVE PS
CMPB #0,@MPSWORD ;IS PS = 0
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #25252,%0 ;IS QUOTIENT = 25252
BEQ 12\$
EMT ;QUOTIENT IS WRONG
12\$: CMP #1,%0+1 ;IS REMAINDER = 1
BEQ 13\$
EMT ;WRONG REMAINDER
13\$: INC (R5)

:TEST:300 DIV 0 52525 / %2 = 25252 REM = 1 PS = 0

9119 036634
9120 036634 012700 000000
9121 036640 012701 052525
9122 036644 071002
9123 036646 106737 025414
9124 036652 122737 000000 025414
9125 036660 001401
9126 036662 104000
9127 036664 022700 025252
9128 036670 001401
9129 036672 104000
9130 036674 022701 000001
9131 036700 001401
9132 036702 104000

TST300:
MOV #0,%0 ;LOAD HIGH ORDER WITH 0
MOV #52525,%0+1 ;LOAD LOW ORDER WITH 52525
DIV %2,%0 ;DIVIDE BY %2
MFPS @MPSWORD ;SAVE PS
CMPB #0,@MPSWORD ;IS PS = 0
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #25252,%0 ;IS QUOTIENT = 25252
BEQ 12\$
EMT ;QUOTIENT IS WRONG
12\$: CMP #1,%0+1 ;IS REMAINDER = 1
BEQ 13\$
EMT ;WRONG REMAINDER

9133 036704
9134 036704 005215
9135
9136
9137
9138
9139
9140 036706
9141 036706 012700 000000
9142 036712 012701 052525
9143 036716 071023
9144 036720 106737 025414
9145 036724 122737 000000 025414
9146 036732 001401
9147 036734 104000
9148 036736 022700 025252
9149 036742 001401
9150 036744 104000
9151 036746 022701 000001
9152 036752 001401
9153 036754 104000
9154 036756
9155 036756 005215
9156
9157
9158
9159
9160
9161 036760
9162 036760 012700 000000
9163 036764 012701 052525
9164 036770 071043
9165 036772 106737 025414
9166 036776 122737 000000 025414
9167 037004 001401
9168 037006 104000
9169 037010 022700 025252
9170 037014 001401
9171 037016 104000
9172 037020 022701 000001
9173 037024 001401
9174 037026 104000
9175 037030
9176 037030 005215
9177
9178
9179
9180
9181
9182 037032
9183 037032 012700 000000
9184 037036 012701 052525
9185 037042 071064 000002
9186 037046 106737 025414
9187 037052 122737 000000 025414
9188 037060 001401

13\$: INC (R5)
:*****
:TEST:301 DIV 0 52525 / (3)+ = 25252 REM = 1 PS = 0
:*****
TST301:
MOV #0,%0 ;LOAD HIGH ORDER WITH 0
MOV #52525,%0+1 ;LOAD LOW ORDER WITH 52525
DIV (3)+,%0 ;DIVIDE BY (3)+
MFPS @PSWORD ;SAVE PS
CMPB #0,@PSWORD ;IS PS = 0
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #25252,%0 ;IS QUOTIENT = 25252
BEQ 12\$
EMT ;QUOTIENT IS WRONG
12\$: CMP #1,%0+1 ;IS REMAINDER = 1
BEQ 13\$
EMT ;WRONG REMAINDER
13\$: INC (R5)
:*****
:TEST:302 DIV 0 52525 / -(3) = 25252 REM = 1 PS = 0
:*****
TST302:
MOV #0,%0 ;LOAD HIGH ORDER WITH 0
MOV #52525,%0+1 ;LOAD LOW ORDER WITH 52525
DIV -(3),%0 ;DIVIDE BY -(3)
MFPS @PSWORD ;SAVE PS
CMPB #0,@PSWORD ;IS PS = 0
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #25252,%0 ;IS QUOTIENT = 25252
BEQ 12\$
EMT ;QUOTIENT IS WRONG
12\$: CMP #1,%0+1 ;IS REMAINDER = 1
BEQ 13\$
EMT ;WRONG REMAINDER
13\$: INC (R5)
:*****
:TEST:303 DIV 0 52525 / 2(4) = 25252 REM = 1 PS = 0
:*****
TST303:
MOV #0,%0 ;LOAD HIGH ORDER WITH 0
MOV #52525,%0+1 ;LOAD LOW ORDER WITH 52525
DIV 2(4),%0 ;DIVIDE BY 2(4)
MFPS @PSWORD ;SAVE PS
CMPB #0,@PSWORD ;IS PS = 0
BEQ 11\$

9189 037062 104000
9190 037064 022700 025252
9191 037070 001401
9192 037072 104000
9193 037074 022701 000001
9194 037100 001401
9195 037102 104000
9196 037104
9197 037104 005215
9198
9199

11\$: EMT ;PS IS WRONG
CMP #25252,X0 ;IS QUOTIENT = 25252
BEQ 12\$
12\$: EMT ;QUOTIENT IS WRONG
CMP #1,X0+1 ;IS REMAINDER = 1
BEQ 13\$
EMT ;WRONG REMAINDER
13\$: INC (R5)

:TEST:304 DIV 0 52525 / @ (4) = 25252 REM = 1 PS = 0

9200
9201
9202
9203 037106
9204 037106 012700 000000
9205 037112 012701 052525
9206 037116 071074 000000
9207 037122 106737 025414
9208 037126 122737 000000 025414
9209 037134 001401
9210 037136 104000
9211 037140 022700 025252
9212 037144 001401
9213 037146 104000
9214 037150 022701 000001
9215 037154 001401
9216 037156 104000
9217 037160
9218 037160 005215
9219

TST304:
MOV #0,X0 ;LOAD HIGH ORDER WITH 0
MOV #52525,X0+1 ;LOAD LOW ORDER WITH 52525
DIV @ (4),X0 ;DIVIDE BY @ (4)
MFPS @#PSWORD ;SAVE PS
CMPB #0,@#PSWORD ;IS PS = 0
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #25252,X0 ;IS QUOTIENT = 25252
BEQ 12\$
EMT ;QUOTIENT IS WRONG
12\$: CMP #1,X0+1 ;IS REMAINDER = 1
BEQ 13\$
EMT ;WRONG REMAINDER
13\$: INC (R5)

:TEST:305 DIV 0 52525 / @ (4)+ = 25252 REM = 1 PS = 0

9220
9221
9222
9223
9224 037162
9225 037162 012700 000000
9226 037166 012701 052525
9227 037172 071034
9228 037174 106737 025414
9229 037200 122737 000000 025414
9230 037206 001401
9231 037210 104000
9232 037212 022700 025252
9233 037216 001401
9234 037220 104000
9235 037222 022701 000001
9236 037226 001401
9237 037230 104000
9238 037232
9239 037232 005215
9240

TST305:
MOV #0,X0 ;LOAD HIGH ORDER WITH 0
MOV #52525,X0+1 ;LOAD LOW ORDER WITH 52525
DIV @ (4)+,X0 ;DIVIDE BY @ (4)+
MFPS @#PSWORD ;SAVE PS
CMPB #0,@#PSWORD ;IS PS = 0
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #25252,X0 ;IS QUOTIENT = 25252
BEQ 12\$
EMT ;QUOTIENT IS WRONG
12\$: CMP #1,X0+1 ;IS REMAINDER = 1
BEQ 13\$
EMT ;WRONG REMAINDER
13\$: INC (R5)

:TEST:306 DIV 0 52525 / @ - (4) = 25252 REM = 1 PS = 0

9241
9242
9243
9244

```

9245 037234
9246 037234 012700 000000
9247 037240 012701 052525
9248 037244 071054
9249 037246 106737 025414
9250 037252 122737 000000 025414
9251 037260 001401
9252 037262 104000
9253 037264 022700 025252
9254 037270 001401
9255 037272 104000
9256 037274 022701 000001
9257 037300 001401
9258 037302 104000
9259 037304
9260 037304 005215
9261
9262
9263
9264
9265
9266 037306 012701 177777
9267 037312 012700 077700
9268 037316 070027 000001
9269 037322 022701 077700
9270 037326 001401
9271 037330 104000
9272 037332 005700
9273 037334 001401
9274 037336 104000
9275 037340 000167 000026
  
```

TST306:

```

MOV #0,%0 ;LOAD HIGH ORDER WITH 0
MOV #52525,%0+1 ;LOAD LOW ORDER WITH 52525
DIV @-(4),%0 ;DIVIDE BY @-(4)
MFPS @#PSWORD ;SAVE PS
CMPB #0,@#PSWORD ;IS PS = 0
BEQ 11$
EMT ;PS IS WRONG
11$: CMP #25252,%0 ;IS QUOTIENT = 25252
BEQ 12$
EMT ;QUOTIENT IS WRONG
12$: CMP #1,%0+1 ;IS REMAINDER = 1
BEQ 13$
EMT ;WRONG REMAINDER
13$: INC (R5)
  
```

:SPECIAL MULTIPLY DATA PATTERN TEST

```

TSTSPC: MOV #-1,R1 ;MAKE R1 -1 SO WE KNOW INSTR. WAS MODIFIER
MOV #77700,R0 ;SET UP TEST DATA
MUL #1,R0 ;DO MULTIPLY INSTRUCTION
CMP #77700,R1 ;CHECK LOW ORDER WORD
BEQ 1$
EMT ;LOW ORDER PRODUCT ERROR
1$: TST R0 ;CHECK HIGH ORDER WORD
BEQ EISEND
EMT ;HIGH ORDER PRODUCT ERROR
EISEND: JMP MMUTST ;JMP OVER GARBAGE AND GET TO MMU TEST
  
```

```
9276  
9277      .SBTTL MEMORY MANAGEMENT DEFINITIONS  
9278  
9279      ;*KT11 VECTOR ADDRESS  
9280  
9281      MMVEC= 250  
9282  
9283      ;*KT11 STATUS REGISTER ADDRESSES  
9284  
9285      SR0= 177572  
9286      SR1= 177574  
9287      SR2= 177576  
9288      SR3= 172516  
9289  
9290      ;*USER 'I' PAGE DESCRIPTOR REGISTERS  
9291  
9292      UIPDR0= 177600  
9293      UIPDR1= 177602  
9294      UIPDR2= 177604  
9295      UIPDR3= 177606  
9296      UIPDR4= 177610  
9297      UIPDR5= 177612  
9298      UIPDR6= 177614  
9299      UIPDR7= 177616  
9300  
9301      ;*USER 'I' PAGE ADDRESS REGISTERS  
9302  
9303      UIPAR0= 177640  
9304      UIPAR1= 177642  
9305      UIPAR2= 177644  
9306      UIPAR3= 177646  
9307      UIPAR4= 177650  
9308      UIPAR5= 177652  
9309      UIPAR6= 177654  
9310      UIPAR7= 177656  
9311  
9312      ;*KERNEL 'I' PAGE DESCRIPTOR REGISTERS  
9313  
9314      KIPDR0= 172300  
9315      KIPDR1= 172302  
9316      KIPDR2= 172304  
9317      KIPDR3= 172306  
9318      KIPDR4= 172310  
9319      KIPDR5= 172312  
9320      KIPDR6= 172314  
9321      KIPDR7= 172316  
9322  
9323      ;*KERNEL 'I' PAGE ADDRESS REGISTERS  
9324  
9325      KIPAR0= 172340  
9326      KIPAR1= 172342  
9327      KIPAR2= 172344  
9328      KIPAR3= 172346  
9329      KIPAR4= 172350  
9330      KIPAR5= 172352  
9331      KIPAR6= 172354
```

9332 172356
9333
9334 000006
9335 000006
9336 177776
9337 000020
9338 000100
9339 000001
9340 000004
9341
9342
9343
9344
9345
9346
9347 037344 000000
9348 037346 000000
9349 037350 000000
9350 037352 000000
9351 037354 000000
9352 037356 000000
9353 037360 000000
9354 037362 000000
9355 037364 000000
9356 037366 000000
9357 037370 000000
9358
9359

KIPAR7= 172356

KSP= SP
USP= SP
PSW= PS
TBIT= 20
WBIT= 100
BIT0= 1
ERRVEC= 4

;*ADDITIONAL DEFINITIONS
;*

WASR6: .WORD 0 ;USED TO STORE THE STACK POINTER AFTER A TRAP
TRAPPC: .WORD 0 ;USED TO STORE THE PC OF A TRAP OR ABORT
TRAPPS: .WORD 0 ;USED TO STORE THE PS OF A TRAP OR ABORT
WASSR0: .WORD 0 ;USED TO STORE CONTENTS OF SRO
WASSR2: .WORD 0 ;USED TO STORE CONTENTS OF SR2
TBITPS: .WORD 0 ;SAVES THE PSW THAT MAY HAVE ITS T-BIT ON
\$TMP0: .WORD 0 ;TEMPORARY STORAGE LOCATION
\$TMP1: .WORD 0 ;TEMPORARY STORAGE LOCATION
\$TMP2: .WORD 0 ;TEMPORARY STORAGE LOCATION
\$TMP3: .WORD 0 ;TEMPORARY STORAGE LOCATION
\$TMP4: .WORD 0 ;TEMPORARY STORAGE LOCATION

```
9360  
9361 037372 012706 001000 MMUTST: MOV #STBOT,KSP ;INITIALIZE THE STACK POINTER  
9362 037376 012767 021356 140644 MOV #T0250,MMVEC ;LOAD MEMORY MANAGENT ROUTINE INTO VECTOR  
9363 037404 012767 000340 140640 MOV #340,MMVEC+2 ;SET NEW PS TO PRIORITY LEVEL 7-KERNEL  
9364 037412 012767 000340 177736 MOV #340,TBITPS ;INITIALIZE LOG THAT HOLDS T-BIT PSW  
9365 037420 005067 140146 CLR SRO ;BE SURE MEM. MGMT IS OFF TO START WITH  
9366 037424 000244 CLZ ;CLR THE Z BIT  
9367 037426 032777 000001 161750 BIT #1,@SWR  
9368 037434 001402 BEQ 1$  
9369 037436 000167 011076 JMP EXATST  
9370 037442 012737 050434 000030 1$: MOV #ERROR3,@#30 ;SET UP EMT VECTOR TO GO TO RIGHT ERROR CALL  
9371 037450 012737 000002 001004 MOV #2,@$TESTN ;INCREMENT TEST NUMBER  
9372
```

9373
9374
9375
9376
9377
9378 037456
9379 037456 005000
9380 037460 005001
9381 037462 106400
9382 037464 106701
9383 037466 042701 177437
9384 037472 020001
9385 037474 001401
9386 037476 104000
9387
9388
9389
9390 037500 062700 000040
9391 037504 022700 000400
9392 037510 001363
9393
9394
9395
9396
9397 037512
9398 037512 005000
9399 037514 005067 140256
9400 037520 050067 140252
9401 037524 016701 140246
9402 037530 042701 007777
9403 037534 020001
9404 037536 001401
9405 037540 104000
9406
9407
9408
9409 037542 062700 010000
9410 037546 001362
9411 037550 005067 140222
9412
9413
9414
9415
9416 037554
9417 037554 005067 140216
9418 037560 012700 000360
9419 037564 110067 140207
9420 037570 016701 140202
9421 037574 042701 007437
9422 037600 000300
9423 037602 020001
9424 037604 001401
9425 037606 104000
9426
9427
9428

```
*****
:TEST 351      PSW PRIORITY BIT TEST
*****
TS351:
2$:   CLR      R0      ;INITIALIZE R0 WITH PRIORITY=0 DATA
      CLR      R1      ;PREPARE R1 TO ACCEPT DATA READ
      MTPS     R0      ;WRITE PRIORITY BITS IN THE PSW
      MFPS     R1      ;READ BACK THE LOW BYTE OF PSW
      BIC      #177437,R1 ;MASK OFF EVERYTHING EXCEPT PRIORITY BITS
      CMP      R0,R1   ;WAS CORRECT PRIORITY SET IN THE PSW?
      BEQ      3$
      EMT
      ;PRIORITY BITS SET WRONG IN PSW
      ;FOR TIGHTER SCOPE LOOP
      ;REPLACE ERROR CALL WITH
      ;'BR 2$' = 000770
3$:   ADD      #40,R0   ;CHANGE DATA TO NEXT PRIORITY
      CMP      #400,R0 ;HAVE PRIORITIES 0-7 ALL BEEN CHECKED?
      BNE      2$      ;BRANCH IF NO
*****
:TEST 352      PSW MODE BIT TEST
*****
TS352:
2$:   CLR      R0      ;INITIALIZE R0 WITH MODE BITS = 0000
      CLR      PSW     ;INITIALIZE PSW
      BIS      R0,PSW  ;BIT SET THE PSW MODE BITS WITH R0
      MOV      PSW,R1  ;READ BACK THE CONTENTS OF THE PSW
      BIC      #007777,R1 ;MASK OFF EVERYTHING EXCEPT THE MODE BITS
      CMP      R0,R1   ;WERE THE MODE BITS SET CORRECTLY?
      BEQ      3$
      EMT
      ;MODE BITS SET WRONG IN PSW
      ;FOR TIGHTER SCOPE LOOP
      ;REPLACE ERROR CALL WITH
      ;'BR 2$' = 000763
3$:   ADD      #10000,R0 ;CHANGE MODE BIT DATA
      BNE      2$      ;BRANCH IF STILL MORE COMBINATIONS
      CLR      PSW     ;RESET PSW BEFORE LEAVING
*****
:TEST 353      BYTE ADDRESSING TEST FOR PSW
*****
TS353:
2$:   CLR      PSW     ;CLEAR THE PSW
      MOV      #360,R0 ;PUT THE HIGH BYTE DATA INTO R0
      MOVB     R0,PSW+1 ;WRITE THE HIGH BYTE OF THE PSW
      MOV      PSW,R1  ;READ BACK THE ENTIRE PSW
      BIC      #007437,R1 ;MASK OFF THE T & CC BITS
      SWAB     R0      ;GET DATA WRITTEN IN HIGH BYTE OF R0
      CMP      R0,R1   ;WAS THE PSW WRITTEN TO CORRECTLY
      BEQ      4$
      EMT
      ;LOW BYTE EFFECTED BY WRITE TO HIGH BYTE OF PSW
      ;FOR TIGHTER SCOPE LOOP
      ;REPLACE ERROR CALL WITH
      ;'BR 2$' = 000760
```

9429	037610	005067	140162
9430	037614	012700	000340
9431	037620	110067	140152
9432	037624	016701	140146
9433	037630	042701	007437
9434	037634	020001	
9435	037636	001401	
9436	037640	104000	
9437			
9438			
9439			
9440			
9441			
9442			
9443			
9444	037642		
9445	037642	005067	140130
9446	037646	012706	001000
9447	037652	012767	140000
9448	037660	012706	000600
9449	037664	005067	140106
9450	037670	022706	001000
9451	037674	001401	
9452	037676	104000	
9453			
9454			
9455			
9456			
9457			
9458			
9459			
9460			
9461			
9462			
9463			
9464			
9465			
9466			
9467			
9468			
9469			
9470			
9471			
9472			
9473			
9474	037700		
9475	037700	012700	177572
9476	037704	012701	000003
9477	037710	005710	
9478			
9479	037712	062700	000002
9480	037716	077104	
9481	037720	005737	172516
9482			
9483			
9484			

140116

```

4$: CLR PSW ;CLEAR THE PSW
MOV #340,R0 ;PUT THE LOW BYTE DATA INTO R0
MOVB R0,PSW ;WRITE THE LOW BYTE OF THE PSW
MOV PSW,R1 ;READ BACK THE ENTIRE PSW
BIC #007437,R1 ;MASK OFF THE T&CC BITS
CMP R0,R1 ;WAS PSW WRITTEN TO CORRECTLY
BEQ TS354
EMT ;HIGH BYTE EFFECTED BY WRITE TO LOW BYTE OF PSW
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;'BR 2$' = 000736

```

```

;*****
;TEST 354 TEST AND SETUP OF STACK POINTERS
;*****
TS354:

```

```

CLR PSW ;GO TO KERNEL MODE
MOV #KERSTK,KSP ;SET KERNEL STACK POINTER TO 1100
MOV #140000,PSW ;GO TO USER MODE
MOV #USESTK,USP ;SET USER STACK POINTER TO 700
CLR PSW ;BACK TO KERNEL MODE
CMP #KERSTK,KSP ;IS KERNEL R6 STILL 1100?
BEQ TS355
EMT ;KERNEL R6 CHANGED BY WRITING USER R6
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;000756

```

```

;*****
;*
;* THE NEXT FIVE (5) TESTS WILL TRY TO ADDRESS ALL OF THE
;* MEMORY MANAGEMENT REGISTERS (SR0,SR1,SR2,KERNEL & USER PAR/PDR'S).
;* EVERY TIME A REGISTER TIMES OUT ITS ADDRESS WILL BE REPORTED.
;* AT THE END OF EACH TEST A SUMMARY OF THE ADDRESSES THAT TIMED
;* OUT DURING THAT TEST IS GIVEN. THE RESULTS OF 'AND-ING' AND 'OR-ING'
;* THEIR ADDRESSES IS GIVEN TO SHOW WHICH ADDRESS LINES MAY BE
;* STUCK AT 0 OR 1. THE PAR/PDR ADDRESS AND KT MUX'S ARE THE
;* THINGS BEING CHECKED.
;*
;*****

```

```

;*****
;TEST 355 SR0,SR1,SR2,SR3 TIMEOUT TEST
;*****
TS355:

```

```

MOV #SR0,R0 ;LOAD R0 WITH ADDRESS OF FIRST REG.
MOV #3,R1 ;LOAD R1 WITH THE LOOP COUNT
2$: TST (R0) ;TRY ADDRESSING A STATUS REGISTER
;IF IT TIMES OUT GO TO 5$
3$: ADD #2,R0 ;PUT NEXT ADDRESS IN R0
SOB R1,2$ ;LOOP BACK TO 2$ UNTIL ALL TESTED
TST #172516 ;CHECK SR3 FOR RESPONSE

```

```

;*****
;TEST 356 KERNEL PAR'S TIMEOUT TEST

```


9485
9486 037724
9487
9488 037724 012700 172340
9489 037730 012701 000010
9490 037734 005710
9491
9492 037736 062700 000002
9493 037742 077104
9494
9495
9496
9497
9498 037744
9499
9500 037744 012700 172300
9501 037750 012701 000010
9502 037754 005710
9503
9504 037756 062700 000002
9505 037762 077104
9506
9507
9508
9509
9510 037764
9511
9512 037764 012700 177640
9513 037770 012701 000010
9514 037774 005710
9515
9516 037776 062700 000002
9517 040002 077104
9518
9519
9520
9521
9522 040004
9523
9524 040004 012700 177600
9525 040010 012701 000010
9526 040014 005710
9527
9528 040016 062700 000002
9529 040022 077104
9530
9531
9532
9533
9534 040024
9535
9536 040024 012700 177572
9537 040030 012710 160000
9538 040034 000005
9539 040036 011001
9540 040040 001401

```
*****
TS356:
      MOV    #KIPAR0,R0      ;LOAD R0 WITH ADDRESS OF FIRST REG.
      MOV    #10,R1         ;LOAD R1 WITH LOOP COUNT (8)
2$:   TST    (R0)           ;TRY ADDRESSING A KIPAR
      ;IF IT TIMES OUT, WILL GO TO 5$
3$:   ADD    #2,R0         ;PUT NEXT KIPAR ADDRESS IN R0
      SOB   R1,2$         ;LOOP BACK TO 2$ UNTIL ALL TESTED
*****
;TEST 357      KERNEL PDR'S TIMEOUT TEST
*****
TS357:
      MOV    #KIPDR0,R0     ;LOAD R0 WITH ADDRESS OF FIRST REG.
      MOV    #10,R1         ;LOAD R1 WITH LOOP COUNT (8)
2$:   TST    (R0)           ;TRY ADDRESSING A KIPDR
      ;IF IT TIMES OUT, WILL GO TO 5$
3$:   ADD    #2,R0         ;PUT NEXT KIPDR ADDRESS IN R0
      SOB   R1,2$         ;LOOP BACK TO 2$ UNTIL ALL TESTED
*****
;TEST 360      USER PAR'S TIMEOUT TEST
*****
TS360:
      MOV    #UIPAR0,R0     ;LOAD R0 WITH ADDRESS OF FIRST REG.
      MOV    #10,R1         ;LOAD R1 WITH LOOP COUNT (8)
2$:   TST    (R0)           ;TRY ADDRESSING A UIPAR
      ;IF IT TIMES OUT, WILL GO TO 5$
3$:   ADD    #2,R0         ;PUT NEXT UIPAR ADDRESS IN R0
      SOB   R1,2$         ;LOOP BACK TO 2$ UNTIL ALL TESTED
*****
;TEST 361      USER PDR'S TIMEOUT TEST
*****
TS361:
      MOV    #UIPDR0,R0     ;LOAD R0 WITH ADDRESS OF FIRST REG.
      MOV    #10,R1         ;LOAD R1 WITH LOOP COUNT (8)
2$:   TST    (R0)           ;TRY ADDRESSING A UIPDR
      ;IF IT TIMES OUT, WILL GO TO 5$
3$:   ADD    #2,R0         ;PUT NEXT UIPDR ADDRESS IN R0
      SOB   R1,2$         ;LOOP BACK TO 2$ UNTIL ALL TESTED
*****
;TEST 362      SRO(15:13) BIT TEST & SR2 TEST
*****
TS362:
1$:   MOV    #SRO,R0        ;LOAD ADDRESS OF SRO INTO R0
      MOV    #160000,(R0)   ;SET BITS <15:13> IN SRO (ERROR BITS)
      RESET                          ;ISSUE AND 'INIT' SIGNAL
      MOV    (R0),R1        ;READ SRO INTO R1 TO SEE IF CLEAR
2$:   BEQ   2$
```

```
9541 040042 104000          EMT          ;SRO<15:13> NOT CLEARED BY A 'RESET'  
9542                          ;FOR TIGHTER SCOPE LOOP  
9543                          ;REPLACE ERROR CALL WITH  
9544                          ;'BR 1$' = 000770  
9545 040044 016767 137526 177302 2$:  MOV      SR2,WASSR2 ;READ CONTENTS OF SR2  
9546 040052 012701 040044      MOV      #2$,R1   ;LOAD EXPECTED CONTENTS INTO R1  
9547 040056 020167 177272      CMP      R1,WASSR2 ;IS SR2 TRACKING?  
9548 040062 001401      BEQ      3$  
9549 040064 104000          EMT          ;SR2 NOT 'TRACKING' VIRTUAL ADDRESSES  
9550                          ;FOR TIGHTER SCOPE LOOP  
9551                          ;REPLACE ERROR CALL WITH  
9552                          ;'BR 2$' = 000767  
9553 040066 012701 100000      3$:  MOV      #100000,R1 ;PUT DATA TO BE WRITTEN IN R1  
9554 040072 012703 000003      MOV      #3,R3    ;SETUP R3 AS A LOOP COUNTER  
9555 040076 005010      4$:  CLR      (R0)    ;CLEAR SRO  
9556 040100 050110      5$:  BIS      R1,(R0)  ;SET ONE OF THE ERROR BITS IN SRO  
9557 040102 011002      MOV      (R0),R2 ;READ SRO INTO R2  
9558 040104 020102      CMP      R1,R2   ;DID RIGHT ERROR BIT GET SET?  
9559 040106 001401      BEQ      6$  
9560 040110 104000          EMT          ;BITS WERE SET WRONG IN SRO  
9561                          ;FOR TIGHTER SCOPE LOOP  
9562                          ;REPLACE ERROR CALL WITH  
9563                          ;'BR 4$' = 000772  
9564 040112 012704 040100      6$:  MOV      #5$,R4   ;LOAD EXPECTED CONTENTS OF SR2 IN R4  
9565 040116 016767 137454 177230  MOV      SR2,WASSR2 ;READ SR2  
9566 040124 020467 177224      CMP      R4,WASSR2 ;DID SR2 LOCK UP WHEN ERROR  
9567                          ;BIT SET IN SR1?  
9568 040130 001401      BEQ      7$  
9569 040132 104000          EMT          ;SR2 DID NOT LOCK UP  
9570                          ;FOR TIGHTER SCOPE LOOP  
9571                          ;REPLACE ERROR CALL WITH  
9572                          ;'BR 4$' = 000761  
9573 040134 006001      7$:  ROR      R1    ;CHANGE DATA TO CHECK NEXT ERROR BIT  
9574 040136 077321      SOB      R3,4$  ;LOOP BACK UNTIL <15:13> ALL TESTED  
9575 040140 005010      CLR      (R0)   ;CLEAR SRO BEFORE LEAVING  
9576  
9577 ;*****  
9578 ;TEST 363      SRO & PSW DUAL ADDRESSING TEST  
9579 ;*****  
9580 040142      TS363:  
9581  
9582 040142 005067 137630      1$:  CLR      PSW    ;CLEAR THE PSW  
9583 040146 005067 137420      CLR      SRO   ;CLEAR STATUS REGISTER 0  
9584 040152 106427 000340      MTPS   #340  ;SET PRIORITY 7 IN LOW BYTE OF PSW  
9585 040156 016700 137410      MOV      SRO,R0 ;READ STATUS REGISTER 0  
9586 040162 001401      BEQ      2$  
9587 040164 104000          EMT          ;SRO EFFECTED BY A WRITE TO THE PSW  
9588                          ;FOR TIGHTER SCOPE LOOP  
9589                          ;REPLACE ERROR CALL WITH  
9590                          ;'BR 1$' = 000767  
9591 040166 005067 137400      2$:  CLR      SRO   ;BE SURE SRO IS 0 BEFORE LEAVING  
9592 040172 005067 137600      CLR      PSW   ;BE SURE PSW IS 0 BEFORE LEAVING  
9593  
9594 ;*****  
9595 ;TEST 364      TEST THAT SR1 READS ALL ZEROS  
9596 ;*****
```

9597 040176
9598 040176 012700 177777
9599 040202 016700 137366
9600 040206 001401
9601 040210 104000
9602
9603
9604
9605 040212 012767 177777 132276
9606 040220 022767 000060 132270
9607 040226 001401
9608 040230 104000
9609 040232 004567 010152
9610 040236 000402
9611 040240 000005
9612 040242 000402
9613 040244 005067 132246
9614 040250 005767 132242
9615 040254
9616 040254 001401
9617 040256 104000
9618
9619
9620
9621
9622
9623
9624
9625
9626 040260
9627
9628 040260 012700 172340
9629 040264 012703 000010
9630 040270 005010
9631 040272 011001
9632 040274 001401
9633 040276 104000
9634
9635
9636
9637 040300 012704 077777
9638 040304 005010
9639 040306 050410
9640 040310 011002
9641 040312 020402
9642 040314 001401
9643 040316 104000
9644
9645
9646
9647 040320 000261
9648 040322 006004
9649 040324 103767
9650 040326 062700 000002
9651 040332 077322
9652 040334 022700 177660

TS364:
1\$: MOV #-1,R0 ;FILL R0 WITH ALL ONES
MOV SR1,R0 ;READ SR1 INTO R0
BEQ 2\$
EMT ;SR1 DID NOT READ ALL ZEROS
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;000772
2\$: MOV #-1,SR3 ;TRY TO WRITE ONES TO SR3
CMP #60,SR3 ;ONLY BITS <5:4> SHOULD BE ONES
BEQ 3\$
EMT ;DIDN'T READ BACK A '60'
3\$: JSR R5,CHKAPT
BR 90\$
RESET ;CLEARS SR3
BR 91\$
90\$: CLR SR3
91\$: TST SR3 ;VERIFY THAT IT WAS CLEARED
4\$: BEQ TS365 ;SR3 DIDN'T READ ALL ZEROS
EMT
;NOTE F11 CHANGES INCLUDED CHECKING ALL BITS<15:0> OF PARS
; INSTEAD OF ONLY BITS<11:0>.
;*****
;TEST 365 BIT TEST OF KERNEL & USER PAR'S
;*****
TS365:
1\$: MOV #KIPAR0,R0 ;LOAD ADDRESS OF FIRST PAR IN R0
2\$: MOV #10,R3 ;SETUP R3 TO COUNT 8 PAR'S
3\$: CLR (R0) ;CLEAR THE PAR
MOV (R0),R1 ;READ THE PAR INTO R1
BEQ 4\$
EMT ;PAR WOULD NOT CLEAR
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;'BR 3\$' = 000774
4\$: MOV #077777,R4 ;LOAD 'WALKING 0' TEST PATTERN IN R4
5\$: CLR (R0) ;CLEAR THE PAR BEFORE LOADING DATA
BIS R4,(R0) ;BIT SET THE TEST PATTERN INTO THE PAR
MOV (R0),R2 ;READ THE PAR INTO R2
CMP R4,R2 ;DOES DATA WRITTEN=DATA READ?
BEQ 6\$
EMT ;PAR BITS DID NOT SET CORRECTLY
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;'BR 5\$' = 000767
6\$: SEC ;SET THE C-BIT FOR THE ROTATE INST.
ROR R4 ;ROTATE THE TEST PATTERN IN R4
BCS 5\$;BRANCH BACK IF MORE BITS TO TEST
ADD #2,R0 ;GET NEXT PAR ADDRESS IN R0
SOB R3,3\$;BRANCH BACK UNTIL ALL PAR'S TESTED
CMP #UIPAR7+2,R0 ;HAVE USER PAR'S BEEN TESTED

9653 040340 103003
9654 040342 012700 177640
9655 040346 000746
9656
9657
9658
9659
9660 040350
9661
9662 040350 012700 172300
9663 040354 012703 000010
9664 040360 005010
9665 040362 011001
9666 040364 001401
9667 040366 104000
9668
9669
9670
9671 040370 012704 077777
9672 040374 005010
9673 040376 010401
9674 040400 042701 100361
9675 040404 050110
9676 040406 011002
9677 040410 020102
9678 040412 001401
9679 040414 104000
9680
9681
9682
9683 040416 000261
9684 040420 006004
9685 040422 103764
9686 040424 062700 000002
9687 040430 077325
9688 040432 022700 177620
9689 040436 103003
9690 040440 012700 177600
9691 040444 000743
9692
9693
9694
9695
9696
9697
9698 040446
9699
9700 040446 012700 172340
9701 040452 012703 000010
9702 040456 012701 177777
9703 040462 005010
9704 040464 110110
9705 040466 011002
9706 040470 042701 177400
9707 040474 020102
9708 040476 001401

```
BHIS TS366 ;GET TO NEXT TEST
MOV #UIPARO,R0 ;LOAD FIRST USER PAR ADDR. IN R0
BR 2$ ;BRANCH BACK TO TEST USER PAR'S
;LEAVE TEST WITH BITS <11:1>=1 IN ALL PAR'S
:*****
:TEST 366 BIT TEST OF KERNEL & USER PDR'S
:*****
TS366:
1$: MOV #KIPDRO,R0 ;LOAD ADDRESS OF FIRST PDR IN R0
2$: MOV #10,R3 ;SETUP R3 TO COUNT 8 PDR'S
3$: CLR (R0) ;CLEAR THE PDR
MOV (R0),R1 ;READ THE PDR INTO R1
BEQ 4$
EMT ;PDR WOULD NOT CLEAR
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;'BR 3$' = 000774
4$: MOV #077777,R4 ;LOAD 'WALKING 0' TEST PATTERN IN R4
5$: CLR (R0) ;CLEAR THE PDR BEFORE LOADING DATA
MOV R4,R1 ;LOAD DATA INTO R1
BIC #100361,R1 ;MASK UNUSED BITS OUT OF THE DATA
BIS R1,(R0) ;BIT SET THE TEST PATTERN INTO THE PDR
MOV (R0),R2 ;READ THE PDR INTO R2
CMP R1,R2 ;DOES DATA WRITTEN=DATA READ?
BEQ 6$
EMT ;PDR BITS DID NOT SET CORRECTLY
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;'BR 5$' = 000767
6$: SEC ;SET THE C-BIT FOR THE ROTATE INST.
ROR R4 ;ROTATE THE TEST PATTERN IN R4
BCS 5$ ;BRANCH BACK IF MORE BITS TO TEST
ADD #2,R0 ;GET NEXT PDR ADDRESS IN R0
SOB R3,3$ ;BRANCH BACK UNTIL ALL PDR'S TESTED
CMP #UIPDR7+2,R0 ;HAVE USER PDR'S BEEN TESTED?
BHIS TS367 ;GET TO NEXT TEST
MOV #UIPDRO,R0 ;LOAD FIRST USER PDR ADDR. IN R0
BR 2$ ;BRANCH BACK TO TEST USER PDR'S
;LEAVE TEST WITH ALL WRITEABLE BITS IN
;ALL PDR'S = 1
:*****
:TEST 367 TEST FOR DUAL BYTE ADDRESSING OF KERNEL & USER PAR'S
:*****
TS367:
1$: MOV #KIPARO,R0 ;LOAD ADDRESS OF FIRST PAR INTO R0
MOV #10,R3 ;LOAD LOOP COUNTER TO DO 8 PAR'S
3$: MOV #-1,R1 ;LOAD TEST PATTERN INTO R1
CLR (R0) ;CLEAR THE PAR
MOVB R1,(R0) ;WRITE 1'S TO THE LOW BYTE OF THE PAR
MOV (R0),R2 ;READ THE ENTIRE PAR INTO R2
BIC #177400,R1 ;MASK HIGH BYTE & UNUSED BITS OUT OF THE DATA
CMP R1,R2 ;WAS ONLY THE LOW BYTE WRITTEN TO
BEQ 5$
```

9709	040500	104000		EMT		:HIGH BYTE EFFECTED BY WRITING LOW BYTE IN PAR
9710						:FOR TIGHTER SCOPE LOOP
9711						:REPLACE ERROR CALL WITH
9712						: 'BR 3\$' = 000766
9713	040502	005010		5\$: CLR (R0)		:CLEAR THE PAR
9714	040504	012701	177777	MOV #-1,R1		:LOAD TEST, PATTERN INTO R1
9715	040510	110160	000001	MOVB R1,1(R0)		:WRITE 1'S TO THE HIGH BYTE OF THE PAR
9716	040514	011002		MOV (R0),R2		:READ THE ENTIRE PAR INTO R2
9717						:F11 CHANGE WAS #170377
9718	040516	042701	000377	BIC #000377,R1		:MASK LOW BYTE & UNUSED BITS OUT OF DATA
9719	040522	020102		CMP R1,R2		:WAS ONLY THE HIGH BYTE WRITTEN TO?
9720	040524	001401		BEQ 6\$		
9721	040526	104000		EMT		:LOW BYTE EFFECTED BY WRITING HIGH BYTE IN PAR
9722						:FOR TIGHTER SCOPE LOOP
9723						:REPLACE ERROR CALL WITH
9724						: 'BR 5' = 000765
9725	040530	062700	000002	6\$: ADD #2,R0		:PUT ADDRESS OF NEXT PAR IN R0
9726	040534	077330		SOB R3,3\$:BRANCH BACK UNTIL 8 PAR'S TESTED
9727	040536	022700	177660	CMP #UIPAR7+2,R0		:HAVE USER PAR'S BEEN TESTED
9728	040542	103003		BHIS TS370		:GET TO NEXT TEST
9729	040544	012700	177640	MOV #UIPAR0,R0		:LOAD ADDRESS OF FIRST USER PAR IN R0
9730	040550	000742		BR 3\$:BRANCH BACK TO TEST USER PAR'S
9731						
9732						:*****
9733						:TEST 370 TEST FOR DUAL BYTE ADDRESSING OF KERNEL & USER PDR'S
9734						:*****
9735	040552					TS370:
9736						
9737	040552	012700	172300	1\$: MOV #KIPDR0,R0		:LOAD ADDRESS OF FIRST PDR INTO R0
9738	040556	012703	000010	MOV #10,R3		:LOAD LOOP COUNTER TO DO 8 PDR'S
9739	040562	012701	177777	3\$: MOV #-1,R1		:LOAD TEST PATTERN INTO R1
9740	040566	005010		CLR (R0)		:CLEAR THE PDR
9741	040570	110110		MOVB R1,(R0)		:WRITE 1'S TO THE LOW BYTE OF THE PDR
9742	040572	011002		MOV (R0),R2		:READ THE ENTIRE PDR INTO R2
9743	040574	042701	177761	BIC #177761,R1		:MASK HIGH BYTE & UNUSED BITS OUT OF DATA
9744	040600	020102		CMP R1,R2		:WAS ONLY THE LOW BYTE WRITTEN TO?
9745	040602	001401		BEQ 5\$		
9746	040604	104000		EMT		:HIGH BYTE EFFECTED BY WRITING LOW BYTE IN PDR
9747						:FOR TIGHTER SCOPE LOOP
9748						:REPLACE ERROR CALL WITH
9749						: 'BR 3\$' = 000766
9750	040606	005010		5\$: CLR (R0)		:CLEAR THE PDR
9751	040610	012701	177777	MOV #-1,R1		:LOAD TEST PATTERN INTO R1
9752	040614	110160	000001	MOVB R1,1(R0)		:WRITE 1'S TO THE HIGH BYTE OF THE PDR
9753	040620	011002		MOV (R0),R2		:READ THE ENTIRE PDR INTO R2
9754	040622	042701	100377	BIC #100377,R1		:MASK LOW BYTE & UNUSED BITS OUT OF DATA
9755	040626	020102		CMP R1,R2		:WAS ONLY THE HIGH BYTE WRITTEN TO?
9756	040630	001401		BEQ 6\$		
9757	040632	104000		EMT		:LOW BYTE EFFECTED BY WRITING HIGH BYTE IN PDR
9758						:FOR TIGHTER SCOPE LOOP
9759						:REPLACE ERROR CALL WITH
9760						: 'BR 5\$' = 000765
9761	040634	062700	000002	6\$: ADD #2,R0		:PUT ADDRESS OF NEXT PDR IN R0
9762	040640	077330		SOB R3,3\$:BRANCH BACK UNTIL 8 PDR'S TESTED
9763	040642	022700	177620	CMP #UIPDR7+2,R0		:HAVE USER PDR'S BEEN TESTED?
9764	040646	103003		BHIS TS371		:GET TO NEXT TEST

9765 040650 012700 177600
9766 040654 000742
9767
9768
9769
9770
9771 040656
9772

MOV #UIPDR0,R0 ;LOAD ADDRESS OF FIRST USER PDR IN R0
BR 3\$;BRANCH BACK TO TEST USER PDR'S

:TEST 371 PAR-PDR DUAL ADDRESSING TEST

TS371:

9773 040656 012703 000010
9774 040662 012700 172300
9775 040666 004767 007242
9776 040672 012706 001000
9777 040676 005010
9778 040700 004767 007322
9779 040704 012720 177777
9780 040710 077310
9781 040712 012703 000010
9782 040716 012700 172340
9783 040722 012706 001000
9784 040726 005010
9785 040730 004767 007272
9786 040734 012720 177777
9787 040740 077310
9788 040742 012703 000010
9789 040746 012700 177600
9790 040752 012706 001000
9791 040756 005010
9792 040760 004767 007242
9793 040764 012720 177777
9794 040770 077310
9795 040772 012703 000010
9796 040776 012700 177640
9797 041002 012706 001000
9798 041006 005010
9799 041010 004767 007212
9800 041014 012720 177777
9801 041020 077310
9802
9803
9804
9805
9806 041022
9807
9808

MOV #10,R3 ;LOAD LOOP COUNTER WITH AN 8
MOV #KIPDR0,R0 ;LOAD ADDRESS OF FIRST KERNEL PDR AND R0
JSR PC,SETREG ;SET ALL BITS IN ALL PAR'S IN PDR'S
2\$: MOV #KERSTK,KSP ;SETUP STACK POINTER
CLR (R0) ;CLEAR ONE OF THE KERNEL PDR'S
JSR PC,COMPREG ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
MOV #-1,(R0)+ ;RESTORE ALL ONES, AND SETUP FOR NEXT PDR
SOB R3,2\$;LOOP TO 2\$ UNTIL ALL KERNEL PDR'S CHECKED
MOV #10,R3 ;LOAD LOOP COUNTER WITH AN 8
MOV #KIPAR0,R0 ;LOAD ADDRESS OF FIRST KERNEL PAR IN R0
3\$: MOV #KERSTK,KSP ;SETUP STACK POINTER
CLR (R0) ;CLEAR ONE OF THE KERNEL PAR'S
JSR PC,COMPREG ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
MOV #-1,(R0)+ ;RESTORE ALL ONES, AND SETUP FOR NEXT PAR
SOB R3,3\$;LOOP TO 3\$ UNTIL ALL KERNEL PAR'S CHECKED
MOV #10,R3 ;LOAD LOOP COUNTER WITH AN 8
MOV #UIPDR0,R0 ;LOAD ADDRESS OF FIRST USER PDR IN R0
4\$: MOV #KERSTK,KSP ;SETUP STACK POINTER
CLR (R0) ;CLEAR ONE OF THE USER PDR'S
JSR PC,COMPREG ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
MOV #-1,(R0)+ ;RESTORE ALL ONES, AND SETUP FOR NEXT UPDR
SOB R3,4\$;LOOP TO 4\$ UNTIL ALL USER PDR'S CHECKED
MOV #10,R3 ;LOAD LOOP COUNTER WITH AN 8
MOV #UIPAR0,R0 ;LOAD ADDRESS OF FIRST USER PAR IN R0
5\$: MOV #KERSTK,KSP ;SETUP STACK POINTER
CLR (R0) ;CLEAR ONE OF THE USER PAR'S
JSR PC,COMPREG ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
MOV #-1,(R0)+ ;RESTORE ALL ONES, AND SETUP FOR NEXT UPAR
SOB R3,5\$;LOOP TO 5\$ UNTIL ALL USER PAR'S CHECKED

:TEST 372 TEST THAT PAR-PDR'S NOT AFFECTED BY RESET

TS372:

9809 041022 032737 000001 001020
9810 041030 001403
9811 041032 005737 001006
9812 041036 001063
9813 041040
9814 041040 004767 007070
9815 041044 000005
9816 041046 012700 172300
9817 041052 012704 000010
9818 041056 011001
9819 041060 022701 077416
9820 041064 001401

BIT #1,@\$ENV ;ARE WE RUNNING UNDER APT
BEQ 70\$;IF NO THEN DO TEST
TST @\$SPASS ;IS THIS FIRST PASS
BNE TS373 ;IF NO THEN SHIP TO NEXT TEST
70\$: JSR PC,SETREG ;SET ALL BITS IN ALL PAR'S AND PDR'S
1\$: RESET ;ISSUE AN "INIT" BY EXECUTING A RESET
10\$: MOV #KIPDR0,R0 ;LOAD ADDRESS OF FIRST KERNEL PDR IN R0
MOV #10,R4 ;LOAD LOOP COUNTER WITH AN 8
2\$: MOV (R0),R1 ;READ A KERNEL PDR INTO R1
CMP #77416,R1 ;ARE ALL THE BITS STILL SET?
BEQ 3\$


```
9877
9878
9879
9880
9881
9882
9883
9884
9885
9886
9887
9888
9889 041206
9890
9891 041206 012700 172340
9892 041212 005001
9893 041214 012702 000007
9894 041220 010120
9895 041222 062701 000200
9896 041226 077204
9897 041230 012710 177600
9898 041234 012700 172300
9899 041240 012701 077406
9900 041244 012702 000010
9901 041250 0120
9902 041252 07202
9903
9904 041254 012700 067776
9905 041260 012701 107776
9906 041264 012702 125250
9907 041270 012704 000600
9908 041274 010467 131050
9909 041300 011067 176054
9910 041304 005067 131206
9911 041310 052767 000001 136254
9912 041316 010211
9913 041320 005067 136246
9914 041324 011003
9915 041326 016710 176026
9916 041332 020203
9917
9918 041334 001401
9919 041336 104000
9920
9921
9922
9923
9924
9925
9926
9927 041340
9928 041340 012700 067776
9929 041344 012701 102576
9930 041350 012702 125251
9931 041354 012704 000652
9932 041360 010467 130764

:*****
:TEST 373 RELOCATION & ADDER TEST (NO CARRIES)
:*****
TS373:
1$: MOV #KIPAR0,R0 ;LOAD ADDRESS OF FIRST KERNEL PAR IN R0
   CLR R1 ;CLEAR R1
2$: MOV #7,R2 ;LOAD LOOP COUNTER WITH A 7
   MOV R1,(R0)+ ;MAP KERNEL PAR'S TO PAGES 0-6 (4K EACH)
   ADD #200,R1
   SOB R2,2$ ;LOOP UNTIL KIPAR0 - KIPAR6 ARE LOADED
   MOV #177600,(R0) ;MAP KIPAR7 TO THE I/O PAGE
   MOV #KIPDRO,R0 ;LOAD ADDRESS OF FIRST KERNEL PDR IN R0
   MOV #77406,R1 ;LOAD PDR DATA INTO R1
   MOV #10,R2 ;LOAD LOOP COUNTER WITH AN 8
3$: MOV R1,(R0)+ ;MAP ALL 8 PAGES 128 BLOCKS, UPWARD
   SOB R2,3$ ; EXPANDABLE, READ/WRITE

4$: MOV #67776,R0 ;LOAD PHYSICAL ADDR. PBA INTO R0
   MOV #107776,R1 ;LOAD VIRTUAL ADDR. VBA INTO R1
   MOV #125250,R2 ;LOAD TEST PATTERN INTO R2
   MOV #600,R4 ;LOAD R4 WITH PAR VALUE
   MOV R4,KIPAR4 ;LOAD KERNEL PAR 4 BITS <11:00>
   MOV (R0),$TMP0 ;SAVE CONTENTS AT TEST LOCATION
   CLR SR3 ;SET UP FOR 18-BIT ADDRESSING
   BIS #BIT0,SRO ;TURN ON 'RELOCATION'
   MOV R2,(R1) ;LOAD 125250 USING ADDER (PAR4 + VIRT ADDR.)
   CLR SRO ;TURN OFF MEMORY MGMT.
   MOV (R0),R3 ;READ 125250 BACK WITHOUT USING MEM. MGMT.
   MOV $TMP0,(R0) ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
   CMP R2,R3 ;WAS SAME PATTERN READ BACK THAT WAS
   ;WRITTEN USING 'DEST-ONLY-RELOC.'?
   BEQ 5$
   EMT ;TEST LOCATION DID NOT HAVE PATTERN
   ;THAT SHOULD HAVE BEEN WRITTEN TO IT.
   ;APPARENTLY PHYSICAL ADDR. WAS
   ;FORMED WRONG BY ADDERS USING
   ;THE VIRTUAL ADDR. AND KIPAR4
   ;FOR TIGHTER SCOPE LOOP
   ;REPLACE ERROR CALL WITH
   ;'BR 4$' = 000742
5$:
6$: MOV #67776,R0 ;LOAD PHYSICAL ADDR. PBA INTO R0
   MOV #102576,R1 ;LOAD VIRTUAL ADDR. VBA INTO R1
   MOV #125251,R2 ;LOAD TEST PATTERN INTO R2
   MOV #652,R4 ;LOAD R4 WITH PAR VALUE
   MOV R4,KIPAR4 ;LOAD KERNEL PAR 4 BITS <11:00>
```


9933	041364	011067	175770		MOV	(R0), \$TMP0	:SAVE CONTENTS AT TEST LOCATION
9934	041370	005067	131122		CLR	SR3	:SET UP FOR 18-BIT ADDRESSING
9935	041374	052767	000001	136170	BIS	#BIT0, SR0	:TURN ON 'RELOCATION'
9936	041402	010211			MOV	R2, (R1)	:LOAD 125251 USING ADDER (PAR4 + VIRT ADDR.)
9937	041404	005067	136162		CLR	SR0	:TURN OFF MEMORY MGMT.
9938	041410	011003			MOV	(R0), R3	:READ 125251 BACK WITHOUT USING MEM. MGMT.
9939	041412	016710	175742		MOV	\$TMP0, (R0)	:RESTORE ORIGINAL CONTENTS TO TEST LOC.
9940	041416	020203			CMP	R2, R3	:WAS SAME PATTERN READ BACK THAT WAS
9941							:WRITTEN USING 'DEST-ONLY-RELOC.'?
9942	041420	001401			BEQ	7\$	
9943	041422	104000			EMT		:TEST LOCATION DID NOT HAVE PATTERN
9944							:THAT SHOULD HAVE BEEN WRITTEN TO IT.
9945							:APPARENTLY PHYSICAL ADDR. WAS
9946							:FORMED WRONG BY ADDERS USING
9947							:THE VIRTUAL ADDR. AND KIPAR4
9948							:FOR TIGHTER SCOPE LOOP
9949							:REPLACE ERROR CALL WITH
9950							: 'BR 6\$' = 000742
9951	041424			7\$:			
9952	041424	012700	067776	8\$:	MOV	#67776, R0	:LOAD PHYSICAL ADDR. PBA INTO R0
9953	041430	012701	105276		MOV	#105276, R1	:LOAD VIRTUAL ADDR. VBA INTO R1
9954	041434	012702	125252		MOV	#125252, R2	:LOAD TEST PATTERN INTO R2
9955	041440	012704	000625		MOV	#625, R4	:LOAD R4 WITH PAR VALUE
9956	041444	010467	130700		MOV	R4, KIPAR4	:LOAD KERNEL PAR 4 BITS <11:00>
9957	041450	011067	175704		MOV	(R0), \$TMP0	:SAVE CONTENTS AT TEST LOCATION
9958	041454	052767	000020	131034	BIS	#BIT4, SR3	:SET UP FOR 22-BIT ADDRESSING
9959	041462	052767	000001	136102	BIS	#BIT0, SR0	:TURN ON 'RELOCATION'
9960	041470	010211			MOV	R2, (R1)	:LOAD 125252 USING ADDER (PAR4 + VIRT ADDR.)
9961	041472	005067	136074		CLR	SR0	:TURN OFF MEMORY MGMT.
9962	041476	011003			MOV	(R0), R3	:READ 125252 BACK WITHOUT USING MEM. MGMT.
9963	041500	016710	175654		MOV	\$TMP0, (R0)	:RESTORE ORIGINAL CONTENTS TO TEST LOC.
9964	041504	020203			CMP	R2, R3	:WAS SAME PATTERN READ BACK THAT WAS
9965							:WRITTEN USING 'DEST-ONLY-RELOC.'?
9966	041506	001401			BEQ	9\$	
9967	041510	104000			EMT		:TEST LOCATION DID NOT HAVE PATTERN
9968							:THAT SHOULD HAVE BEEN WRITTEN TO IT.
9969							:APPARENTLY PHYSICAL ADDR. WAS
9970							:FORMED WRONG BY ADDERS USING
9971							:THE VIRTUAL ADDR. AND KIPAR4
9972							:FOR TIGHTER SCOPE LOOP
9973							:REPLACE ERROR CALL WITH
9974							: 'BR 8\$' = 000742
9975	041512			9\$:			
9976							
9977	041512	012700	177776		MOV	#PSW, R0	:LOAD PHYS. ADDR. OF PSW INTO R0
9978	041516	012701	100076		MOV	#100076, R1	:LOAD VIRTUAL ADDR. FOR PSW INTO R1
9979	041522	012702	030340		MOV	#030340, R2	:LOAD DATA FOR PSW IN R2
9980	041526	012704	007777		MOV	#7777, R4	:LOAD R4 WITH PAR VALUE
9981	041532	010467	130612		MOV	R4, KIPAR4	:LOAD KERNEL PAR 4 BITS <11:00>
9982	041536	005010			CLR	(R0)	:CLEAR THE PSW
9983	041540	005067	130752		CLR	SR3	:SET UP FOR 18-BIT ADDRESSING
9984	041544	052767	000001	136020	BIS	#BIT0, SR0	:TURN ON 'MEMORY MANAGEMENT'
9985	041552	010211			MOV	R2, (R1)	:LOAD PSW USING ADDER (PAR4 + VIRT ADDR.)
9986	041554	005067	136012		CLR	SR0	:TURN OFF MEM. MGMT (SR0=0)
9987	041560	011003			MOV	(R0), R3	:READ PSW BACK WITHOUT USING MEM. MGMT.
9988	041562	005010			CLR	(R0)	:CLEAR THE PSW

9989	041564	042703	000037		BIC	#37,R3		:MASK T-BIT & CC BITS OUT OF DATA READ
9990	041570	020203			CMP	R2,R3		:WAS PSW WRITTEN?
9991	041572	001401			BEQ	11\$		
9992	041574	104000			EMT			:PSW DID NOT HAVE DATA THAT IT SHOULD HAVE,
9993								:APPARENTLY PHYS. ADDR. OF PSW WAS
9994								:NOT FORMED BY ADDERS USING THE
9995								:VIRTUAL ADDR. AND KIPAR4
9996								:FOR TIGHTER SCOPE LOOP
9997								:REPLACE ERROR CALL WITH
9998								: 'BR 10\$' = 000742
9999	041576	012700	177776	11\$:	MOV	#PSW,R0		:LOAD PHYS. ADDR. OF PSW INTO R0
10000	041602	012701	117776		MOV	#117776,R1		:LOAD VIRTUAL ADDR. FOR PSW INTO R1
10001	041606	012702	030240		MOV	#030240,R2		:LOAD DATA FOR PSW IN R2
10002	041612	012704	177600		MOV	#177600,R4		:LOAD R4 WITH PAR VALUE
10003	041616	010467	130526		MOV	R4,KIPAR4		:LOAD KERNEL PAR 4 BITS <11:00>
10004	041622	052767	000020	130666	BIS	#BIT4,SR3		:SET UP FOR 22-BIT ADDRESSING
10005	041630	052767	000001	135734	BIS	#BIT0,SR0		:TURN ON 'MEMORY MANAGEMENT'
10006	041636	010211			MOV	R2,(R1)		:LOAD PSW USING ADDER (PAR4 + VIRT. ADDR.)
10007	041640	005067	135726		CLR	SR0		:TURN OFF MEM. MGMT (SR0=0)
10008	041644	011003			MOV	(R0),R3		:READ PSW BACK WITHOUT USING MEM. MGMT.
10009	041646	005010			CLR	(R0)		:CLEAR THE PSW
10010	041650	042703	000037		BIC	#37,R3		:MASK T-BIT & CC BITS OUT OF DATA READ
011	041654	020203			CMP	R2,R3		:WAS PSW WRITTEN WHILE IN MAINT. MODE?
0012	041656	001401			BEQ	TS374		
10013	041660	104000			EMT			:PSW DID NOT HAVE DATA THAT IT SHOULD
10014								:HAVE, APPARENTLY PHYS. ADDR. OF PSW WAS
10015								:NOT FORMED BY ADDERS USING THE
10016								:VIRTUAL ADDR. AND KIPAR4
10017								:FOR TIGHTER SCOPE LOOP
10018								:REPLACE ERROR CALL WITH
10019								: 'BR 11\$' = 000743
10020								
10021								:*****
10022								:TEST 374 RELOCATION & ADDER TEST (WITH CARRIES)
10023								:*****
10024	041662							TS374:
10025								
10026	041662			1\$:				:KERNEL PAR'S AND PDR'S HAVE BEEN
10027								:SETUP BY THE PREVIOUS TEST
10028	041662	012700	066476	2\$:	MOV	#66476,R0		:LOAD PHYSICAL ADDR. PBA INTO R0
10029	041666	012701	114376		MOV	#114376,R1		:LOAD VIRTUAL ADDR. VBA INTO R1
10030	041672	012702	125253		MOV	#125253,R2		:LOAD TEST PATTERN INTO R2
10031	041676	012704	000521		MOV	#521,R4	:LOAD R4	:LOAD R4 WITH PAR VALUE
10032	041702	010467	130442		MOV	R4,KIPAR4		:LOAD KERNEL PAR 4 BITS <11:00>
10033	041706	011067	175446		MOV	(R0),STMP0		:SAVE CONTENTS AT TEST LOCATION
10034	041712	052767	000020	130576	BIS	#BIT4,SR3		:SET UP FOR 22-BIT ADDRESSING
10035	041720	052767	000001	135644	BIS	#BIT0,SR0		:TURN ON 'RELOCATION'
10036	041726	010211			MOV	R2,(R1)		:LOAD 125253 USING ADDER (PAR4 + VIRT ADDR.)
10037	041730	005067	135636		CLR	SR0		:TURN OFF MEMORY MGMT.
10038	041734	011003			MOV	(R0),R3		:READ 125253 BACK WITHOUT USING MEM. MGMT.
10039	041736	016710	175416		MOV	STMP0,(R0)		:RESTORE ORIGINAL CONTENTS TO TEST LOC.
10040	041742	020203			CMP	R2,R3		:WAS SAME PATTERN READ BACK THAT WAS
10041								:WRITTEN USING 'DEST-ONLY-RELOC.'?
10042	041744	001401			BEQ	3\$		
10043	041746	104000			EMT			:TEST LOCATION DID NOT HAVE PATTERN
10044								:THAT SHOULD HAVE BEEN WRITTEN TO IT.

Address	Instruction	Comments
10045		: APPARENTLY PHYSICAL ADDR. WAS
10046		: FORMED WRONG BY ADDERS USING
10047		: THE VIRTUAL ADDR. AND KIPAR4
10048		: FOR TIGHTER SCOPE LOOP
10049		: REPLACE ERROR CALL WITH
10050		: 'BR 2\$' = 000742
10051	041750	3\$:
10052	041750 012700 062276	4\$: MOV #62276,R0 ;LOAD PHYSICAL ADDR. PBA INTO R0
10053	041754 012701 107376	MOV #107376,R1 ;LOAD VIRTUAL ADDR. VBA INTO R1
10054	041760 012702 125254	MOV #125254,R2 ;LOAD TEST PATTERN INTO R2
10055	041764 012704 000527	MOV #527,R4 ;LOAD R4 WITH PAR VALUE
10056	041770 010467 130354	MOV R4,KIPAR4 ;LOAD KERNEL PAR 4 BITS <11:00>
10057	041774 011067 175360	MOV (R0),\$TMP0 ;SAVE CONTENTS AT TEST LOCATION
10058	042000 052767 000020 130510	BIS #BIT4,SR3 ;SET UP FOR 22-BIT ADDRESSING
10059	042006 052767 000001 135556	BIS #BIT0,SRO ;TURN ON 'RELOCATION'
10060	042014 010211	MOV R2,(R1) ;LOAD 125254 USING ADDER (PAR4 + VIRT ADDR.)
10061	042016 005067 135550	CLR SRO ;TURN OFF MEMORY MGMT.
10062	042022 011003	MOV (R0),R3 ;READ 125254 BACK WITHOUT USING MEM. MGMT.
10063	042024 016710 175330	MOV \$TMP0,(R0) ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
10064	042030 020203	CMP R2,R3 ;WAS SAME PATTERN READ BACK THAT WAS
10065		: WRITTEN USING 'DEST-ONLY-RELOC.'?
10066	042032 001401	BEQ 5\$
10067	042034 104000	EMT ;TEST LOCATION DID NOT HAVE PATTERN
10068		: THAT SHOULD HAVE BEEN WRITTEN TO IT.
10069		: APPARENTLY PHYSICAL ADDR. WAS
10070		: FORMED WRONG BY ADDERS USING
10071		: THE VIRTUAL ADDR. AND KIPAR4
10072		: FOR TIGHTER SCOPE LOOP
10073		: REPLACE ERROR CALL WITH
10074		: 'BR 4\$' = 000742
10075	042036	5\$:
10076	042036 012700 062076	6\$: MOV #62076,R0 ;LOAD PHYSICAL ADDR. PBA INTO R0
10077	042042 012701 104576	MOV #104576,R1 ;LOAD VIRTUAL ADDR. VBA INTO R1
10078	042046 012702 125255	MOV #125255,R2 ;LOAD TEST PATTERN INTO R2
10079	042052 012704 000553	MOV #553,R4 ;LOAD R4 WITH PAR VALUE
10080	042056 010467 130266	MOV R4,KIPAR4 ;LOAD KERNEL PAR 4 BITS <11:00>
10081	042062 011067 175272	MOV (R0),\$TMP0 ;SAVE CONTENTS AT TEST LOCATION
10082	042066 052767 000020 130422	BIS #BIT4,SR3 ;SET UP FOR 22-BIT ADDRESSING
10083	042074 052767 000001 135470	BIS #BIT0,SRO ;TURN ON 'RELOCATION'
10084	042102 010211	MOV R2,(R1) ;LOAD 125255 USING ADDER (PAR4 + VIRT ADDR.)
10085	042104 005067 135462	CLR SRO ;TURN OFF MEMORY MGMT.
10086	042110 011003	MOV (R0),R3 ;READ 125255 BACK WITHOUT USING MEM. MGMT.
10087	042112 016710 175242	MOV \$TMP0,(R0) ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
10088	042116 020203	CMP R2,R3 ;WAS SAME PATTERN READ BACK THAT WAS
10089		: WRITTEN USING 'DEST-ONLY-RELOC.'?
10090	042120 001401	BEQ 7\$
10091	042122 104000	EMT ;TEST LOCATION DID NOT HAVE PATTERN
10092		: THAT SHOULD HAVE BEEN WRITTEN TO IT.
10093		: APPARENTLY PHYSICAL ADDR. WAS
10094		: FORMED WRONG BY ADDERS USING
10095		: THE VIRTUAL ADDR. AND KIPAR4
10096		: FOR TIGHTER SCOPE LOOP
10097		: REPLACE ERROR CALL WITH
10098		: 'BR 6\$' = 000742
10099	042124	7\$:
10100	042124 012700 000000	8\$: MOV #000000,R0 ;LOAD PHYSICAL ADDR. PBA INTO R0

10101	042130	012701	111400	MOV	#111400,R1	:LOAD	UAL ADDR. VBA INTO R1		
10102	042134	012702	125256	MOV	#125256,R2	:LOAD	TEST PATTERN INTO R2		
10103	042140	012704	177664	MOV	#177664,R4	:LOAD	R4 WITH PAR VALUE		
10104	042144	010467	130200	MOV	R4,KIPAR4	:LOAD	KERNEL PAR 4 BITS <11:00>		
10105	042150	011067	175204	MOV	(R0), \$TMP0	:SAVE	CONTENTS AT TEST LOCATION		
10106	042154	052767	000020	BIS	#BIT4,SR3	:SET	UP FOR 22-BIT ADDRESSING		
10107	042162	052767	000001	BIS	#BIT0,SRO	:TURN	ON 'RELOCATION'		
10108	042170	010211		MOV	R2,(R1)	:LOAD	125256 USING ADDER (PAR4 + VIRT ADDR.)		
10109	042172	005067	135374	CLR	SRO	:TURN	OFF MEMORY MGMT.		
10110	042176	011003		MOV	(R0),R3	:READ	125256 BACK WITHOUT USING MEM. MGMT.		
10111	042200	016710	175154	MOV	\$TMP0,(R0)	:RESTORE	ORIGINAL CONTENTS TO TEST LOC.		
10112	042204	020203		CMP	R2,R3	:WAS	SAME PATTERN READ BACK THAT WAS		
10113						:WRITTEN	USING 'DEST-ONLY-RELOC.'?		
10114	042206	001401		BEQ	9\$				
10115	042210	104000		EMT		:TEST	LOCATION DID NOT HAVE PATTERN		
10116						:THAT	SHOULD HAVE BEEN WRITTEN TO IT.		
10117						:APPARENTLY	PHYSICAL ADDR. WAS		
10118						:FORMED	WRONG BY ADDERS USING		
10119						:THE	VIRTUAL ADDR. AND KIPAR4		
10120						:FOR	TIGHTER SCOPE LOOP		
10121						:REPLACE	ERROR CALL WITH		
10122						: 'BR	8\$' = 000742		
10123	042212			9\$:					
10124						:*****			
10125						:TEST	375 READ AND WRITE WHILE IN RELOCATE MODE		
10126						:*****			
10127						Ts375:			
10128	042212								
10129									
10130	042212	005067	135560	1\$:	CLR	PSW	:START	IN KERNEL MODE	
10131	042216	012704	001377		MOV	#1377,R4	:LOAD	R4 WITH VALUE FOR PAR4	
10132	042222	012705	001400		MOV	#1400,R5	:LOAD	R5 WITH VALUE FOR PAR5	
10133	042226	010467	130116		MOV	R4,KIPAR4	:LOAD	KERNEL PAR4	
10134	042232	010567	130114		MOV	R5,KIPAR5	:LOAD	KERNEL PAR5	
10135	042236	012700	177640		MOV	#UIPAR0,R0	:LOAD	ADDRESS OF FIRST USER PAR IN R0	
10136	042242	005001			CLR	R1	:CLEAR	R1	
10137	042244	012702	000007		MOV	#7,R2	:LOAD	LOOP COUNTER WITH A 7	
10138	042250	010120		2\$:	MOV	R1,(R0)+	:MAP	USER PAR'S TO PAGES 0-6 (4K EACH)	
10139	042252	062701	000200		ADD	#200,R1			
10140	042256	077204			SOB	R2,2\$:LOOP	UNTIL UIPAR0-UIPAR6 ARE LOADED	
10141	042260	012710	177600		MOV	#177600,(R0)	:MAP	USER PAR7 TO THE I/O PAGE	
10142	042264	012700	177600		MOV	#UIPDR0,R0	:LOAD	ADDRESS OF FIRST USER PDR IN R0	
10143	042270	012701	077406		MOV	#77406,R1	:LOAD	PDR DATA INTO R1	
10144	042274	012702	000010		MOV	#10,R2	:LOAD	LOOP COUNTER WITH AN 8	
10145	042300	010120		3\$:	MOV	R1,(R0)+	:MAP	ALL 8 PAGES 128 BLOCKS, UPWARD	
10146	042302	077202			SOB	R2,3\$:EXPANDABLE,	READ/WRITE	
10147	042304	012767	042550		MOV	#8\$,MMVEC	:SET	M. M. TRAP VECTOR TO 8\$	
10148	042312	052767	000020		BIS	#BIT4,SR3	:SET	UP FOR 22-BIT ADDRESSING	
10149	042320	012767	000001		MOV	#BIT0,SRO	:TURN	ON MEMORY MANAGEMENT	
10150	042326	105067	135256		CLRB	UIPDR4	:MAP	USER SPACE NON-RESIDENT WHILE	
10151	042332	105067	135254		CLRB	UIPDR5	:TESTING	KERNEL SPACE	
10152	042336	010567	135306		MOV	R5,UIPAR4	:MAP	USER PAR'S OPPOSITE OF KIPAR'S	
10153	042342	010467	135304		MOV	R4,UIPAR5			
10154	042346	016767	135424	175004	4\$:	MOV	PSW,\$TMP0	:SAVE	PSW IN CASE OF ERROR
10155	042354	012700	100100		MOV	#100100,R0	:PUT	VIRTUAL ADDR. THAT USES PAR4 IN R0	
10156	042360	012701	120000		MOV	#120000,R1	:PUT	VIRTUAL ADDR. THAT USES PAR5 IN R1	

```

10157 042364 010010      5$:  MOV    R0,(R0)      ;WRITE TO TEST LOC. USING PAR4
10158 042366 011102      MOV    (R1),R2      ;READ THE SAME LOC., BUT USING PARS
10159 042370 020002      CMP    R0,R2       ;DID WE READ WHAT WE WROTE?
10160 042372 001401      BEQ   6$           ;READING LOC. USING PARS AND A VIRT.
10161 042374 104000      EMT                    ;ADDR. DID NOT FIND DATA WRITTEN WHEN USING
10162                                     ;PAR4 AND VIRT. ADDRESS.
10163                                     ;FOR TIGHTER SCOPE LOOP
10164                                     ;REPLACE ERROR CALL WITH
10165                                     ;'BR 5$' = 000765
10166                                     ;CHANGE VIRTUAL ADDRS. TO POINT TO NEXT BLOCK
10167 042376 062700 000100 6$:  ADD    #100,R0
10168 042402 062701 000100      ADD    #100,R1
10169 042406 020127 127700      CMP    R1,#127700   ;WERE BLOCKS FROM 60000-676000 ALL TRIED?
10170 042412 001364      BNE   5$           ;BRANCH IF NO
10171 042414 032767 140000 135354 BIT    #140000,PSW   ;HAVE WE DONE TEST IN USER MODE YET?
10172 042422 001026      BNE   7$           ;BRANCH IF YES
10173 042424 010467 135220      MOV    R4,UIPAR4   ;LOAD USER PAR4
10174 042430 010567 135216      MOV    R5,UIPAR5   ;LOAD USER PARS
10175 042434 112767 000006 135146 MOVB   #6,UIPDR4    ;MAP USER SPACE R/W TO TEST IT
10176 042442 112767 000006 135142 MOVB   #6,UIPDR5
10177 042450 105067 127634      CLRB  KIPDR4       ;MAP KERNEL SPACE NON-RESIDENT WHILE
10178 042454 105067 127632      CLRB  KIPDR5       ; TESTING USER SPACE
10179 042460 010567 127664      MOV    R5,KIPAR4   ;MAP KERNEL PAR'S OPPOSITE UIPAR'S
10180 042464 010467 127662      MOV    R4,KIPAR5
10181 042470 012767 140000 135300 MOV    #140000,PSW ;GO TO USER MODE
10182 042476 000723      BR    4$           ;GO BACK AND READ/WRITE IN USER MODE
10183 042500 005067 135272      CLR   PSW          ;GO BACK TO KERNEL MODE BEFORE LEAVING
10184 042504 012767 077406 127576 7$:  MOV    #77406,KIPDR4 ;REMAP KERNEL PAGES READ/WRITE
10185 042512 012767 077406 127572 MOV    #77406,KIPDR5
10186 042520 010567 127624      MOV    R5,KIPAR4   ;MAP KERNEL AND USER PAR'S 4 & 5
10187 042524 010567 127622      MOV    R5,KIPAR5   ; BACK TO 12-16K
10188 042530 010567 135114      MOV    R5,UIPAR4
10189 042534 010567 135112      MOV    R5,UIPAR5
10190 042540 012767 021356 135502 MOV    #T0250,MMVEC ;RESTORE ADDR. OF NORMAL M.M. TRAP ROUTINE
10191 042546 000404      BR    TS376        ;GET TO NEXT TEST
10192 042550 042767 160000 135014 8$:  BIC   #160000,SRO  ;CLEAR ERROR BITS IN SRO
10193 042556 104000      EMT                    ;M.M. TRAP WHILE IN RELOCATE MODE -
10194                                     ;REFERENCED WRONG SET OF PDR'S
10195                                     ;FOR TIGHTER SCOPE LOOP
10196                                     ;REPLACE ERROR CALL WITH
10197                                     ;A 'NOP' = 000240
10198
10199
10200
10201
10202
10203
10204 042560 004767 005262      1$:  JSR    PC,TOFF     ;TURN T-BIT TRAPPING OFF FOR THIS TEST
10205 042564 012702 000004      MOV    #4,R2       ;SET LOOP COUNTER TO 4
10206 042570 012700 172346      MOV    #KIPAR3,R0  ;LOAD ADDRESS OF PAR3 INTO R0
10207 042574 012701 001400      MOV    #1400,R1    ;LOAD '24-28K' PAR VALUE INTO R1
10208 042600 010120      2$:  MOV    R1,(R0)+    ;MAP PARS 3-6 TO 12-16K
10209 042602 077202      SOB   R2,2$        ;LOOP TIL ALL 4 OF THEM LOADED
10210 042604 012705 172300      MOV    #KIPDR0,R5  ;LOAD ADDRESS OF FIRST PDR TO BE TESTED IN R5
10211 042610 012704 000010      MOV    #10,R4      ;SET LOOP COUNTER TO 8
10212 042614 012703 017776      MOV    #17776,R3   ;INITIALIZE VIRTUAL ADDRESS TO BE IN R3

```

```

:*****
:TEST 376      W-BIT LOGIC TEST, KERNEL PDR'S
:*****

```

TS376:

```

1$:
      JSR    PC,TOFF     ;TURN T-BIT TRAPPING OFF FOR THIS TEST
      MOV    #4,R2       ;SET LOOP COUNTER TO 4
      MOV    #KIPAR3,R0  ;LOAD ADDRESS OF PAR3 INTO R0
      MOV    #1400,R1    ;LOAD '24-28K' PAR VALUE INTO R1
2$:  MOV    R1,(R0)+    ;MAP PARS 3-6 TO 12-16K
      SOB   R2,2$        ;LOOP TIL ALL 4 OF THEM LOADED
      MOV    #KIPDR0,R5  ;LOAD ADDRESS OF FIRST PDR TO BE TESTED IN R5
      MOV    #10,R4      ;SET LOOP COUNTER TO 8
      MOV    #17776,R3   ;INITIALIZE VIRTUAL ADDRESS TO BE IN R3

```

10213	042620	012700	172300	3\$:	MOV	#KIPDR0,R0	:LOAD ADDR. OF FIRST PDR TO BE SETUP IN R0	
10214	042624	012702	000010		MOV	#10,R2	:SET LOOP COUNTER TO 8	
10215	042630	012701	077406		MOV	#77406,R1	:PUT 'W-BIT OFF DATA' INTO R1	
10216	042634	010120		4\$:	MOV	R1,(R0)+	:CLEAR ALL W-BITS BY WRITING TO ALL PDRS	
10217	042636	077202			SOB	R2,4\$:LOOP UNTIL ALL OF THEM SETUP	
10218	042640	011313			MOV	(R3),(R3)	:DO 'DATO' TO VIRTUAL ADDR.-SETTING A W-BIT	
10219	042642	031527	000100		BIT	(R5),#WBIT	:DID THAT CAUSE W-BIT TO BE SET?	
10220	042646	001001			BNE	5\$		
10221	042650	104000			EMT		:W-BIT DID NOT GET SET IN PDR	
10222							:FOR TIGHTER SCOPE LOOP	
10223							:REPLACE ERROR CALL WITH	
10224							: 'BR 3\$' = 000763	
10225	042652	012702	000010	5\$:	MOV	#10,R2	:SET LOOP COUNTER TO 8	
10226	042656	012700	172300		MOV	#KIPDR0,R0	:LOAD ADDR. OF FIRST PDR TO BE CHECKED IN R0	
10227	042662	031027	000100	6\$:	BIT	(R0),#WBIT	:DID W-BIT IN OTHER PDRS REMAIN CLEAR?	
10228	042666	001403			BEQ	7\$:BRANCH IF YES	
10229	042670	020500			CMP	R5,R0	:IF W-BIT SET, THEN WAS IT PDR UNDER TEST?	
10230	042672	001401			BEQ	7\$		
10231	042674	104000			EMT		:W-BIT GOT SET IN MORE THAN ONE PDR	
10232							:FOR TIGHTER SCOPE LOOP	
10233							:REPLACE ERROR CALL WITH	
10234							: 'BR 3\$' = 000750	
10235	042676	062700	000002	7\$:	ADD	#2,R0	:POINT R0 TO NEXT PDR TO BE CHECKED	
10236	042702	077211			SOB	R2,6\$:LOOP UNTIL ALL 8 CHECKED FOR CLEAR W-BIT	
10237	042704	010115			MOV	R1,(R5)	:WRITE TO THE PDR TESTED TO CLEAR W-BIT	
10238	042706	031527	000100		BIT	(R5),#WBIT	:DID WRITING PDR CLEAR THE W-BIT?	
10239	042712	001401			BEQ	8\$		
10240	042714	104000			EMT		:W-BIT DID NOT CLEAR BY WRITING THE PDR	
10241							:FOR TIGHTER SCOPE LOOP	
10242							:REPLACE ERROR CALL WITH	
10243							: 'BR 3\$' = 000740	
10244	042716	062705	000002	8\$:	ADD	#2,R5	:POINT R5 TO THE NEXT PDR TO BE TESTED	
10245	042722	062703	020000		ADD	#20000,R3	:CHANGE VIRT. ADDR TO REF. NEXT PDR	
10246	042726	077444			SOB	R4,3\$:LOOP BACK TO 3\$ UNTIL ALL 8 PDR'S TESTED	
10247	042730	004767	005146		JSR	PC,TON	:TURN T-BIT BACK ON FOR NEXT TEST	
10248								
10249								
10250							:*****	
10251							:TEST 377 W-BIT LOGIC TEST, USER PDR'S	
10252							:*****	
10253	042734	012767	140000	135034	TS377:			
10254	042742	004767	005100		1\$:	MOV	#140000,PSW	:GO TO USER MODE FOR THIS TEST
10255	042746	012702	000004			JSR	PC,TOFF	:TURN T-BIT TRAPPING OFF FOR THIS TEST
10256	042752	012700	177646			MOV	#4,R2	:SET LOOP COUNTER TO 4
10257	042756	012701	001400			MOV	#UIPAR3,R0	:LOAD ADDRESS OF PAR3 INTO R0
10258	042762	010120			2\$:	MOV	#1400,R1	:LOAD '24-28K' PAR VALUE INTO R1
10259	042764	077202				MOV	R1,(R0)+	:MAP PARS 3-6 TO 12-16K
10260	042766	012705	177600			SOB	R2,2\$:LOOP TIL ALL 4 OF THEM LOADED
10261	042772	012704	000010			MOV	#UIPDR0,R5	:LOAD ADDRESS OF FIRST PDR TO BE TESTED IN R5
10262	042776	012703	017776			MOV	#10,R4	:SET LOOP COUNTER TO 8
10263	043002	012700	177600		3\$:	MOV	#17776,R3	:INITIALIZE VIRTUAL ADDRESS TO BE IN R3
10264	043006	012702	000010			MOV	#UIPDR0,R0	:LOAD ADDR. OF FIRST PDR TO BE SETUP IN R0
10265	043012	012701	077406			MOV	#10,R2	:SET LOOP COUNTER TO 8
10266	043016	010120			4\$:	MOV	#77406,R1	:PUT 'W-BIT OFF DATA' INTO R1
10267	043020	077202				MOV	R1,(R0)+	:CLEAR ALL W-BITS BY WRITING TO ALL PDRS
10268	043022	011313				SOB	R2,4\$:LOOP UNTIL ALL OF THEM SETUP
						MOV	(R3),(R3)	:DO 'DATO' TO VIRTUAL ADDR.-SETTING A W-BIT

10269	043024	031527	000100		BIT (R5),#WBIT	:DID THAT CAUSE W-BIT TO BE SET?
10270	043030	001001			BNE 5\$	
10271	043032	104000			EMT	:W-BIT DID NOT GET SET IN PDR
10272						:FOR TIGHTER SCOPE LOOP
10273						:REPLACE ERROR CALL WITH
10274						: 'BR 3\$' = 000763
10275	043034	012702	000010	5\$:	MOV #10,R2	:SET LOOP COUNTER TO 8
10276	043040	012700	177600		MOV #UIPDR0,R0	:LOAD ADDR. OF FIRST PDR TO BE CHECKED IN R0
10277	043044	031027	000100	6\$:	BIT (R0),#WBIT	:DID W-BIT IN OTHER PDRS REMAIN CLEAR?
10278	043050	001403			BEQ 7\$:BRANCH IF YES
10279	043052	020500			CMP R5,R0	:IF W-BIT SET, THEN WAS IT PDR UNDER TEST?
10280	043054	001401			BEQ 7\$	
10281	043056	104000			EMT	:W-BIT GOT SET IN MORE THAN ONE PDR
10282						:FOR TIGHTER SCOPE LOOP
10283						:REPLACE ERROR CALL WITH
10284						: 'BR 3\$' = 000750
10285	043060	062700	000002	7\$:	ADD #2,R0	:POINT R0 TO NEXT PDR TO BE CHECKED
10286	043064	077211			SOB R2,6\$:LOOP UNTIL ALL 8 CHECKED FOR CLEAR W-BIT
10287	043066	010115			MOV R1,(R5)	:WRITE TO THE PDR TESTED TO CLEAR W-BIT
10288	043070	031527	000100		BIT (R5),#WBIT	:DID WRITING PDR CLEAR THE W-BIT?
10289	043074	001401			BEQ 8\$	
10290	043076	104000			EMT	:W-BIT DID NOT CLEAR BY WRITING THE PDR
10291						:FOR TIGHTER SCOPE LOOP
10292						:REPLACE ERROR CALL WITH
10293						: 'BR 3\$' = 000740
10294	043100	062705	000002	8\$:	ADD #2,R5	:POINT R5 TO THE NEXT PDR TO BE TESTED
10295	043104	062703	020000		ADD #20000,R3	:CHANGE VIRT. ADDR TO REF. NEXT PDR
10296	043110	077444			SOB R4,3\$:LOOP BACK TO 3\$ UNTIL ALL 8 PDR'S TESTED
10297	043112	004767	004764		JSR PC,TON	:TURN T-BIT BACK ON FOR NEXT TEST
10298	043116	005067	134654		CLR PSW	:BACK TO KERNEL MODE BEFORE LEAVING
10299						
10300						:*****
10301						:TEST 400 TEST 'W-BIT' SPECIAL CASES
10302						:*****
10303	043122					TS400:
10304						
10305	043122	004767	004720	1\$:	JSR PC,TOFF	:TURN OFF T-BIT TRAPPING FOR THIS TEST
10306	043126	012701	077406		MOV #77406,R1	:PUT 'W-BIT OFF' VALUE FOR PDR IN R1
10307	043132	010167	127160	2\$:	MOV R1,KIPDR7	:LOAD KERNEL PDR 7 TO CLEAR W-BIT
10308	043136	016700	134430		MOV SR0,R0	:READ PRESENT CONTENTS OF STATUS REG. 0
10309	043142	010067	134424		MOV R0,SR0	:WRITE PRESENT CONTENTS OF SR0 BACK TO ITSELF
10310	043146	016702	127144		MOV KIPDR7,R2	:READ CONTENTS OF KIPDR7 INTO R2
10311	043152	020102			CMP R1,R2	:WAS W-BIT LEFT CLEARED?
10312	043154	001401			BEQ 3\$	
10313	043156	104000			EMT	:W-BIT IN KIPDR7 SET WHEN SR0 WAS WRITTEN TO
10314						:FOR TIGHTER SCOPE LOOP
10315						:REPLACE ERROR CALL WITH
10316						: 'BR 2\$' = 000765
10317	043160	010167	127130	3\$:	MOV R1,KIPDR6	:LOAD KERNEL PDR6 WITH 77406 TO CLEAR W-BIT
10318	043164	012767	043176		MOV #4\$,ERRVEC	:SET UP LOC. 4 TO 4\$ FOR ODD ADDR. ABORT
10319	043172	005037	140000		CLR #140000	:CAUSE TIMEOUT ABORT THRU LOC. 4
10320	043176	012706	001000	4\$:	MOV #KERSTK,KSP	:RESTORE THE STACK POINTER
10321	043202	016702	127106		MOV KIPDR6,R2	:READ KIPDR6 INTO R2
10322	043206	052701	000100		BIS #100,R1	:R1-77506
10323	043212	020102			CMP R1,R2	:WAS W-BIT SET?
10324	043214	001401			BEQ 5\$	

```
10325 043216 104000          EMT          ;W-BIT WAS NOT SET DURING A TIMEOUT ABORT
10326                          ;FOR TIGHTER SCOPE LOOP
10327                          ;REPLACE ERROR CALL WITH
10328                          ;'BR 3$' = 000757
10329 043220 010167 127070    5$:  MOV     R1,KIPDR6      ;RESTORE KIPDR6 TO 77406
10330 043224 012767 001400 127122  MOV     #1400,KIPAR6    ;RESTORE KIPAR6 TO 1400
10331 043232 012767 021336 134544  MOV     #T04,ERRVEC     ;RESTORE NORMAL CPU TRAP ROUTINE TO LOC.4
10332 043240 004767 004636          JSR     PC,T0N          ;TURN T-BIT TRAPPING BACK ON
10333
10334                          ;*****
10335                          ;*
10336                          ;* THE NEXT THREE (3) TESTS CAUSE MEMORY MANAGEMENT ERRORS
10337                          ;* TO CHECK THE ABILITY OF STATUS REGISTER 0 TO RECORD KT
10338                          ;* ERRORS AND THE ABILITY OF STATUS REGISTER 2 TO LOCK UP THE
10339                          ;* VIRTUAL ADDR. OF THE INSTRUCTION THAT CAUSED THE ERROR.
10340                          ;* THE BITS OF SR? ARE CHECKED AND BITS <15:13>, <6:5>, AND <3:0>
10341                          ;* ARE CHECKED IN SRO. SO THE SRO AND SR2 LOGIC AND THE
10342                          ;* KT ERROR LOGIC ARE CHECKED.
10343                          ;*
10344                          ;*****
10345
10346                          ;*****
10347                          ;TEST 401      NON-RESIDENT ABORT TEST (ACF=0B4)
10348                          ;*****
10349 043244          TS401:
10350
10351 043244 012700 001400    1$:  MOV     #1400,R0          ;LOAD DATA FOR PAR'S INTO R0
10352 043250 010067 127072    MOV     R0,KIPAR3      ;MAP KERNEL PAR'S 3B4 TO 24-28K
10353 043254 010067 127070    MOV     R0,KIPAR4
10354 043260 010067 134362    MOV     R0,UIPAR3      ;MAP USER PAR'S 3B4 TO 24-28K
10355 043264 010067 134360    MOV     R0,UIPAR4
10356 043270 012767 077406 127010  MOV     #77406,KIPDR3  ;MAP KERNEL PDR 3 128 BLKS, READ-WRITE
10357 043276 012767 077406 134302  MOV     #77406,UIPDR3  ;MAP USER PDR 3 128 BLKS, READ-WRITE
10358 043304 012700 060000    MOV     #60000,R0      ;LOAD VIRTUAL ADDR. TO REFERENCE PDR3 INTO R0
10359 043310 012701 100000    MOV     #100000,R1     ;LOAD VIRTUAL ADDR. TO REFERENCE PDR4 INTO R1
10360 043314 012703 100011    MOV     #100011,R3     ;LOAD R3 WITH WHAT SRO SHOULD READ - N.R., KERNEL, PG.4
10361 043320 012702 077400    MOV     #77400,R2     ;LOAD ACF=0 (NON-RESIDENT) PDR VALUE IN R2
10362 043324 012767 043360 134716  2$:  MOV     #5$,MMVEC      ;POINT MEM. MGMT. TRAP VECTOR TO 5$ BELOW
10363 043332 010267 126752    MOV     R2,KIPDR4     ;LOAD ACF TEST VALUE INTO KIPDR4
10364 043336 010267 134246    MOV     R2,UIPDR4     ;LOAD ACF TEST VALUE INTO UIPDR4
10365 043342 005010          3$:  CLR     (R0)          ;CLEAR PHYS. LOC. 140000 USING PDR3
10366 043344 016767 134426 174006  MOV     PSW,$TMP0     ;SAVE PSW IN CASE OF ERROR
10367 043352 005211          4$:  INC     (R1)          ;TRY TO REF. IT USING PDR4 - SHOULD TRAP TO 5$
10368 043354 001462          BEQ    TS402
10369 043356 104000          EMT
10370                          ;MEM. MGMT. ABORT DID NOT OCCUR
10371                          ;FOR TIGHTER SCOPE LOOP
10372                          ;REPLACE ERROR CALL WITH
10373                          ;'BR 3$' = 000772
10374 043360 062706 000004    5$:  ADD     #4,SP
10375 043364 005710          TST    (R0)
10376 043366 001401          BEQ    6$
10377 043370 104000          EMT
10378                          ;INSTRUCTION WAS NOT ABORTED, LOC. GOT CHANGED
10379                          ;FOR TIGHTER SCOPE LOOP
10380 043372 016767 134174 173752  6$:  MOV     SRO,WASSRO   ;READ STATUS REGISTER 0
```


10437											;'BR 3\$' = 000764
10438	043606	016767	133760	173536	6\$:	MOV	SRO,WASSRO				;READ STATUS REG. 0
10439	043614	016767	133756	173532		MOV	SR2,WASSR2				;READ STATUS REG. 2
10440	043622	020367	173524			CMP	R3,WASSRO				;DID SRO REPORT READ-ONLY ERROR CORRECTLY?
10441	043626	001401				BEQ	7\$				
10442	043630	104000				EMT					;SRO DID NOT REPORT R/O ERROR CORRECTLY
10443											;FOR TIGHTER SCOPE LOOP
10444											;REPLACE ERROR CALL WITH
10445											;'BR 3\$' = 000752
10446	043632	012704	043570		7\$:	MOV	#4\$,R4				;LOAD R4 WITH WHAT SR2 SHOULD READ
10447	043636	020467	173512			CMP	R4,WASSR2				;DID SR2 LOCKUP RIGHT VIRTUAL ADDR. (=4\$)?
10448	043642	001401				BEQ	8\$				
10449	043644	104000				EMT					;SR2 DID NOT LOCKUP VIRTUAL ADDR. OF R/O ERROR
10450											;FOR TIGHTER SCOPE LOOP
10451											;REPLACE ERROR CALL WITH
10452											;'BR 3\$' = 000744
10453	043646	042767	160000	133716	8\$:	BIC	#160000,SRO				;CLEAR THE ERROR BITS IN SRO
10454	043654	032767	140000	173476		BIT	#140000,\$TMP0				;HAS ACF=2 BEEN TESTED IN USER MODE?
10455	043662	001006				BNE	9\$;BRANCH IF YES
10456	043664	012703	020151			MOV	#20151,R3				;LOAD R3 WITH WHAT SRO SHOULD READ-R/O, USER, PG.4
10457	043670	012767	140000	134100		MOV	#140000,PSW				;GO TO USER MODE
10458	043676	000721				BR	2\$;REPEAT TEST IN USER MODE
10459	043700	005067	134072		9\$:	CLR	PSW				;GO BACK TO KERNEL MODE BEFORE LEAVING
10460	043704	012767	021356	134336		MOV	#T0250,MMVEC				;RESTGRE ADDRESS OF NORMAL MEMORY
10461											;MANAGEMENT ERROR ROUTINE TO MMVEC.

;NOTE: MACRO MSG31A WAS DELETED AS IT DIDN'T APPLY TO F11.

```

*****
;*
;* THE NEXT TWO (2) TESTS WILL BE CHECKING THE PAGE LENGTH
;* COMPARATORS AND SOME MORE OF THE KT ERROR DETECTION
;* AND STATUS LOGIC. THE PAGE LENGTH FIELD (PLF) IN KERNEL
;* PDR 4 IS VARIED AND FOR EVERY PLF, THREE (3) VIRTUAL
;* ADDRESSES ARE READ. WHILE USING BOTH UPWARD & DOWNWARD PAGE
;* EXPANSION, ONE OF THOSE THREE VIRTUAL ADDRESSES WILL CAUSE A
;* 'PAGE LENGTH ABORT' WHILE THE OTHER TWO WON'T.
;*
;* STATUS REGISTER 0 & 2 ARE CHECKED WHEN THE PAGE LENGTH
;* ABORT DOES OCCUR TO SEE THAT THE ABORT IS REPORTED AND THAT
;* THE VIRTUAL ADDRESS OF THE INSTRUCTION THAT CAUSED THE ABORT
;* IS LOCKED UP.
;*
*****

```

10437
 10438
 10439
 10440
 10441
 10442
 10443
 10444
 10445
 10446
 10447
 10448
 10449
 10450
 10451
 10452
 10453
 10454
 10455
 10456
 10457
 10458
 10459
 10460
 10461
 10462
 10463
 10464
 10465
 10466
 10467
 10468
 10469
 10470
 10471
 10472
 10473
 10474
 10475
 10476
 10477
 10478
 10479
 10480
 10481
 10482
 10483
 10484
 10485
 10486
 10487
 10488
 10489
 10490
 10491
 10492

```
10493 ;*****  
10494 ;TEST 403 PAGE LENGTH FAULTS-UPWARD EXPANSION  
10495 ;*****  
10496 043712  
10497 043712 012767 077406 126366 TS403:  
10498 043720 012767 077406 126364 1$: MOV #77406,KIPDR3 ;MAKE SURE PDR3 IS DESCRIBED AS R/W  
10499 043726 012700 044126 MOV #77406,KIPDR5 ;MAKE SURE PDR5 IS DESCRIBED AS R/W  
10500 043732 012704 044144 MOV #DALTB1,R0 ;DAL TABLE FOR VIRTUAL ADDR'S. TO SELECT PDR4.  
10501 043736 012701 000006 MOV #PDRTB1,R4 ;PDR TAB. E FOR PDR4 (COINCIDES WITH DAL TABLE).  
10502 043742 012767 044104 134300 MOV #6,R1 ;SET UP LOOP COUNTER.  
10503 043750 012706 001000 MOV #9$,MMVEC ;SETUP M.M. TRAP VECTOR FOR UNEXPECTED ABORTS  
10504 MOV #KERSTK,KSP ;MAKE SURE STACK POINTER IS ALL SET UP  
10505 ;TEST NON-ABORT CASES (VBA < OR = PLF)  
10506 043754 012467 126330 2$: MOV (R4)+,KIPDR4 ;LOAD KIPDR4 WITH PAGE LENGTH VALUE  
10507 043760 005730 TST @ (R0)+ ;ACCESS VIRTUAL ADDR. (VBA < OR = PLF)  
10508 ;NO ABORT SHOULD OCCUR!!!  
10509 043762 077104 SOB R1,2$ ;DONE?...NO- TEST NEXT COMBINATION OF DAL & PDR.  
10510 ;TEST ABORT CASES (VBA > PLF)  
10511 3$: MOV #5,R1 ;SET UP LOOP COUNTER.  
10512 043764 012701 000005 MOV #DALTB2,R0 ;DAL TABLE  
10513 043770 012700 044162 MOV #PDRTB2,R4 ;PDR TABLE  
10514 043774 012704 044176 MOV #6$,MMVEC ;SETUP M.M. TRAP VECTOR FOR EXPECTED ABORT  
10515 044000 012767 044020 134242 4$: MOV (R4)+,KIPDR4 ;LOAD KIPDR4 WITH PAGE LENGTH VALUE  
10516 5$: TST @ (R0)+ ;ACCESS VIRTUAL ADDR. (VBA > PLF - ABORT TO 6$)  
10517 044006 012467 126276 BEQ TS404 ;EXPECTED PAGE LENGTH ABORT DID NOT OCCUR  
10518 044012 005730 EMT ;FOR TIGHTER SCOPE LOOP  
10519 044014 001476 ;REPLACE ERROR CALL WITH  
10520 044016 104000 ;'BR 5$' = 000776  
10521 6$: MOV #KERSTK,KSP ;RESTORE STACK POINTER FOLLOWING ABORT  
10522 044020 012706 001000 173320 MOV SRO,WASSRO ;READ M.M. STATUS REG. 0  
10523 044024 016767 133542 173314 MOV SR2,WASSR2 ;READ M.M. STATUS REG. 2  
10524 044032 016767 133540 MOV #40011,R2 ;PUT EXPECTED SRO CONTENTS IN R2  
10525 044040 012702 040011 CMP R2,WASSRO ;DID SRO REPORT PG. LENGTH ABORT, PAGE 4, KERNEL?  
10526 044044 020267 173302 BEQ 7$ ;SRO DID NOT REPORT PG. LENGTH ABORT CORRECTLY  
10527 044050 001401 ;FOR TIGHTER SCOPE LOOP  
10528 044052 104000 ;REPLACE ERROR CALL WITH  
10529 7$: MOV #5$,R3 ;'BR 5$' = 000757  
10530 044060 020367 173270 CMP R3,WASSR2 ;PUT EXPECTED SR2 CONTENTS IN R3  
10531 044064 001401 BEQ 8$ ;DID SR2 LOCKUP VIRT. ADDR. OF ABORTED INSTRUCTION?  
10532 044066 104000 EMT ;SR2 DID NOT LOCKUP VIRT. ADDR. OF ABORT CORRECTLY  
10533 8$: BIC #16000,SRO ;FOR TIGHTER SCOPE LOOP  
10534 044070 042767 160000 133474 SOB R1,4$ ;REPLACE ERROR CALL WITH  
10535 044076 077135 JMP 10$ ;'BR 5$' = 000751  
10536 044100 000167 000010 133460 9$: BIC #16000,SRO ;CLEAR ERROR BITS IN SRO  
10537 044104 042767 160000 133460 ;DONE?...NO -- GET NEXT DAL & PDR PAIR  
10538 044112 104000 EMT ;YES...  
10539 ;CLEAR ERROR BITS IN SRO  
10540 8$: BIC #16000,SRO ;GOT PG. LENGTH ABORT BEFORE IT WAS EXPECTED  
10541 044070 042767 160000 133474 8$: BIC #16000,SRO ;FOR TIGHTER SCOPE LOOP  
10542 044076 077135 SOB R1,4$ ;REPLACE ERROR CALL WITH  
10543 044100 000167 000010 133460 9$: BIC #16000,SRO ;'BR 5$' = 000751  
10544 044104 042767 160000 133460 9$: BIC #16000,SRO ;CLEAR ERROR BITS IN SRO  
10545 044112 104000 EMT ;DONE?...NO -- GET NEXT DAL & PDR PAIR  
10546 ;YES...  
10547 ;CLEAR ERROR BITS IN SRO  
10548 ;GOT PG. LENGTH ABORT BEFORE IT WAS EXPECTED  
10548 ;FOR TIGHTER SCOPE LOOP  
10548 ;REPLACE ERROR CALL WITH  
10548 ;A 'NOP' = 240
```

```
10549
10550 044114 012767 021356 134126 10$: MOV #T0250,MMVEC ;RESTORE NORMAL M.M. TRAP HANDLER
10551 ;ADDRESS TO M.M. TRAP VECTOR
10552 044122 000167 000064 JMP TS404 ;GET TO NEXT TEST
10553
10554 ;DAL TABLE FOR UPWARD EXPANSION (NON-ABORT CASES)
10555 044126 100000 DALTB1: 100000
10556 044130 106100 106100
10557 044132 102300 102300
10558 044134 102500 102500
10559 044136 113700 113700
10560 044140 104600 104600
10561 044142 117700 117700
10562
10563 ;PDR TABLE FOR KPDR4 (NON-ABORT CASES)
10564 044144 000006 PDRTB1: 000006
10565 044146 052006 052006
10566 044150 045006 045006
10567 044152 052006 052006
10568 044154 074406 074406
10569 044156 025006 025006
10570 044160 077406 077406
10571
10572 ;DAL TABLE (ABORT CASES)
10573 044162 100100 DALTB2: 100100
10574 044164 110100 110100
10575 044166 116600 116600
10576 044170 112700 112700
10577 044172 117000 117000
10578 044174 117700 117700
10579
10580 ;PDR TABLE (ABORT CASES)
10581 044176 000006 PDRTB2: 000006
10582 044200 030406 030406
10583 044202 046406 046406
10584 044204 042006 042006
10585 044206 073406 073406
10586 044210 077006 077006
10587
10588
10589 ;*****
10590 ;TEST 404 PAGE LENGTH FAULTS-DOWNWARD EXPANSION
10591 ;*****
10592 044212 TS404:
10593 044212 012700 044412 1$: MOV #DALTB3,R0 ;DAL TABLE FOR VIRTUAL ADDR'S. TO SELECT PDR4.
10594 044216 012704 044430 MOV #PDRTB3,R4 ;PDR TABLE FOR PDR4 (COINCIDES WITH DAL TABLE).
10595 044222 012701 000006 MOV #6,R1 ;SET UP LOOP COUNTER.
10596 044226 012767 044370 134014 MOV #9$,MMVEC ;SETUP M.M. TRAP VECTOR FOR UNEXPECTED ABORTS
10597 044234 012706 001000 MOV #KERSTK,KSP ;MAKE SURE STACK POINTER IS ALL SET UP
10598
10599 ;TEST NON-ABORT CASES (VBA > OR = PLF)
10600 044240 012467 126044 2$: MOV (R4)+,KIPDR4 ;LOAD KIPDR4 WITH PAGE LENGTH VALUE
10601 044244 005730 TST @ (R0)+ ;ACCESS VIRTUAL ADDR. (VBA > OR = PLF)
10602 ;NO ABORT SHOULD OCCUR!!!
10603 044246 077104 SOB R1,2$ ;DONE?...NO- TEST NEXT COMBINATION OF DAL & PDR.
10604
```

```
10605 :TEST ABORT CASES (VBA < PLF)
10606 044250 012701 000005 3$: MOV #5,R1 ;SET UP LOOP COUNTER.
10607 044254 012700 044446 MOV #DALTB4,R0 ;DAL TABLE
10608 044260 012704 044462 MOV #PDRTB4,R4 ;PDR TABLE
10609 044264 012767 044304 133756 MOV #6$,MMVEC ;SETUP M.M. TRAP VECTOR FOR EXPECTED ABORT
10610
10611 044272 012467 126012 4$: MOV (R4)+,KIPDR4 ;LOAD KIPDR4 WITH PAGE LENGHT VALUE
10612 044276 005730 5$: TST @ (R0)+ ;ACCESS VIRTUAL ADDR. (VBA < PLF - ABORT TO 6$)
10613 044300 001476 BEQ TS405
10614 044302 104000 EMT ;EXPECTED PAGE LENGTH ABORT DID NOT OCCUR
10615 ;FOR TIGHTER SCOPE LOOP
10616 ;REPLACE ERROR CALL WITH
10617 ;'BR 5$' = 000776
10618 044304 012706 001000 6$: MOV #KERSTK,KSP ;RESTORE STACK POINTER FOLLOWING ABORT
10619 044310 016767 133256 173034 MOV SRO,WASSRO ;READ M.M. STATUS REG. 0
10620 044316 016767 133254 173030 MOV SR2,WASSR2 ;READ M.M. STATUS REG. 2
10621 044324 012702 040011 MOV #40011,R2 ;PUT EXPECTED SRO CONTENTS IN R2
10622 044330 020267 173016 CMP R2,WASSRO ;DID SRO REPORT PG. LENGTH ABORT, PAGE 4, KERNEL?
10623 044334 001401 BEQ 7$
10624 044336 104000 EMT ;SRO DID NOT REPORT PG. LENGTH ABORT CORRECTLY
10625 ;FOR TIGHTER SCOPE LOOP
10626 ;REPLACE ERROR CALL WITH
10627 ;'BR 5$' = 000757
10628 044340 012703 044276 7$: MOV #5$,R3 ;PUT EXPECTED SR2 CONTENTS IN R3
10629 044344 020367 173004 CMP R3,WASSR2 ;DID SR2 LOCKUP VIRT. ADDR. OF ABORTED INSTRUCTION?
10630 044350 001401 BEQ 8$
10631 044352 104000 EMT ;SR2 DID NOT LOCKUP VIRT. ADDR. OF ABORT CORRECTLY
10632 ;FOR TIGHTER SCOPE LOOP
10633 ;REPLACE ERROR CALL WITH
10634 ;'BR 5$' = 000751
10635 044354 042767 160000 133210 8$: BIC #160000,SRO ;CLEAR ERROR BITS IN SRO
10636 044362 077135 SOB R1,4$ ;DONE?...NO - GET NEXT DAL & PDR PAIR
10637 044364 000167 000010 JMP 10$ ;YES...
10638 044370 042767 160000 133174 9$: BIC #160000,SRO ;CLEAR ERROR BITS IN SRO
10639 044376 104000 EMT ;GOT PG. LENGTH ABORT BEFORE IT WAS EXPECTED
10640 ;FOR TIGHTER SCOPE LOOP
10641 ;REPLACE ERROR CALL WITH
10642 ;A 'NOP' = 000240
10643
10644 044400 012767 021356 133642 10$: MOV #T0250,MMVEC ;RESTORE NORMAL M.M. TRAP HANDLER
10645 ;ADDRESS TO M.M. TRAP VECTOR
10646 044406 000167 000064 JMP TS405 ;GET TO NEXT TEST
10647
10648 ;DAL TABLE FOR DOWNWARD EXPANSION (NON-ABORT CASES)
10649 044412 117700 DALTB3: 117700
10650 044414 111600 111600
10651 044416 115400 115400
10652 044420 115200 115200
10653 044422 104000 104000
10654 044424 113100 113100
10655 044426 100000 100000
10656
10657 ;PDR TABLE (NON-ABORT CASES)
10658 044430 077416 PDRTB3: 77416
10659 044432 025416 25416
10660 044434 032416 32416
```

10661 044436 025416
10662 044440 003016
10663 044442 052416
10664 044444 000016

25416
03016
52416
00016

:DAL TABLE (ABORT CASES)

10666 044446 117600
10667 044450 107600
10668 044452 101100
10669 044454 105000
10670 044456 100700
10671 044460 100000

DALTB4: 117600
107600
101100
105000
100700
100000

:PDR TABLE (ABORT CASES)

10672 044462 077416
10673 044464 047016
10674 044466 031016
10675 044470 035416
10676 044472 004016
10677 044474 000416

PDRTB4: 77416
47016
31016
35416
04016
00416

10681
10682
10683

:TEST 405 SR2 BIT TEST

10684
10685
10686

TS405:

10687 044476
10688 044476 012767 001400 125642
10689 044504 012767 001400 125636
10690 044512 012767 077406 125566
10691 044520 012767 077402 125562
10692 044526 012700 060002
10693 044532 012701 100002
10694 044536 012767 044564 133504
10695 044544 012720 010727
10696 044550 005020
10697 044552 012720 000137
10698 044556 012710 044564
10699 044562 010107
10700 044564 012706 001000
10701 044570 016767 133002 172556
10702 044576 020167 172552
10703 044602 001401
10704 044604 104000

1\$: MOV #1400,KIPAR3 ;BE SURE PAR3 IS MAPPED TO 24-28K
MOV #1400,KIPAR4 ;BE SURE PAR4 IS MAPPED TO 24-28K
MOV #77406,KIPDR3 ;MAP PAGE 3 128 BLOCKS, R/W
MOV #77402,KIPDR4 ;MAP PAGE 4 128 BLOCKS, READ-ONLY
MOV #60002,R0 ;LOAD R0 WITH VIRTUAL ADDR. WHICH USES PDR3
MOV #100002,R1 ;LOAD R1 WITH VIRTUAL ADDR. WHICH USES PDR4
MOV #3\$,MMVEC ;SET M.M. TRAP VECTOR TO 3\$
2\$: MOV #010727,(R0)+ ;LOAD 'MOV PC,(PC)+' INSTRUCTION AT ADDR.
CLR (R0)+ ; REACHED THRU PDR/PAR 4.
MOV #000137,(R0)+ ;LOAD 'JMP @#3\$' INSTRUCTION AT VIRT. ADDR.
MOV #3\$,(R0) ; IN CASE R/O VIOL. DOES NOT ABORT
3\$: MOV R1,PC ;TRANSFER PROGRAM EXECUTION TO 'PAGE 4 INSTRUCTIONS'
MOV #KERSTK,KSP ;RESTORE STACK POINTER
MOV SR2,WASSR2 ;READ CONTENTS OF STATUS REG 2
CMP R1,WASSR2 ;WAS ADDR. OF 'RELOCATED - R/O ABORT' LOCKED UP?
BEQ 4\$
EMT ;SR2 DID NOT LOCK UP VIRTUAL ADDR. OF R/O VIOL.
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;'BR 2\$' = 000757
4\$: BIC #100000,SRO ;CLEAR THE ERROR BITS IN SRO
ADD R1,R1 ;SETUP TO FORM NEXT VIRTUAL ADDRESS
MOV R1,R0 ;SETUP R0 TO FORM NEXT VIRT. ADDR. TO LOAD
BIS #100000,R1 ;FORM VIRTUAL ADDR. THAT SHOULD BE LOCKED UP NEXT
BIS #60000,R0 ;POINT R0 TO NEXT VIRT. ADDR. TO LOAD
CMP R1,#110000 ;HAVE ALL VBA'S 100000-110000 BEEN TESTED?
BLOS 2\$;BRANCH IF NO
MOV #77406,KIPDR4 ;RESTORE PDR4 TO R/W ACCESS

10705
10706
10707
10708 044606 042767 160000 132756
10709 044614 060101
10710 044616 010100
10711 044620 052701 100000
10712 044624 052700 060000
10713 044630 020127 110000
10714 044634 101743
10715
10716 044636 012767 077406 125444

```
10717 044644 012767 021356 133376      MOV      #T0250,MMVEC      ;RESTORE ADDRESS OF NORMAL M.M.  
10718                                     ;TRAP HANDLER TO M.M. VECTOR  
10719  
10720  
10721 :*****  
10722 :TEST 406      MORE CHECKS OF SRO & SR2  
10723 :*****  
10724 TS406:  
10725 044652      012767 001400 125472 1$:  MOV      #1400,KIPAR5      ;MAP KERNEL PAGE 5 TO 24-28K  
10726 044660      012767 000406 125422      MOV      #406,KIPDR4      ;SETUP PDR4 FOR PAGE LENGTH ABORT  
10727 044666      012767 077402 125416      MOV      #77402,KIPDR5    ;SETUP PDR5 FOR R/O ABORT  
10728 044674      016767 132676 172452 2$:  MOV      SR2,WASSR2      ;READ SR2 TO SEE IF ITS TRACKING  
10729 044702      012701 044674      MOV      #2$,R1          ;PUT EXPECTED VIRTUAL PC IN R1  
10730 044706      020167 172442      CMP      R1,WASSR2      ;DID SR2 CONTAIN VIRTUAL PC AT 2$?  
10731 044712      001401      BEQ      4$  
10732 044714      104000      EMT  
10733                                     ;SR2 NOT TRACKING CORRECTLY  
10734                                     ;FOR TIGHTER SCOPE LOOP  
10735                                     ;REPLACE ERROR CALL WITH  
10736 044716      016767 132654 172430 4$:  MOV      SR2,WASSR2      ;READ SR2 TO SEE IF ITS TRACKING  
10737 044724      012701 044716      MOV      #4$,R1          ;PUT EXPECTED VIRTUAL PC IN R1  
10738 044730      020167 172420      CMP      R1,WASSR2      ;DID SR2 CONTAIN VIRTUAL PC AT 4$  
10739 044734      001401      BEQ      6$  
10740 044736      104000      EMT  
10741                                     ;SR2 NOT TRACKING CORRECTLY  
10742                                     ;FOR TIGHTER SCOPE LOOP  
10743                                     ;REPLACE ERROR CALL WITH  
10744 044740      012767 044756 133302 6$:  MOV      #7$,MMVEC      ;PUT ADDRESS OF 7$ IN M.M. TRAP VECTOR  
10745 044746      005067 172410      CLR      $TMP1          ;CLEAR ERROR INDICATOR  
10746 044752      005237 100500      INC      @#100500        ;CAUSE PAGE LENGTH ABORT - TRAP TO 7$  
10747 044756      012706 001000      7$:  MOV      #KERSTK,KSP      ;RESTORE STACK POINTER AFTER ABORT  
10748 044762      016767 132604 172370      MOV      SRO,$TMP0      ;SAVE SRO'S INFORMATION ON PG. LGTH. ABORT  
10749 044770      016767 132602 172366      MOV      SR2,$TMP2      ;SAVE SR2'S INFORMATION ON PG. LGTH. ABORT  
10750 044776      012767 045010 133244      MOV      #8$,MMVEC      ;PUT ADDRESS OF 8$ IN M.M. TRAP VECTOR  
10751 045004      005237 120000      INC      @#120000        ;CAUSE R/O ABORT - TRAP TO 8$  
10752 045010      012706 001000      8$:  MOV      #KERSTK,KSP      ;RESTORE STACK POINTER AFTER ABORT  
10753 045014      016767 132552 172330      MOV      SRO,WASSRO      ;READ SRO FOLLOWING SECOND KT ABORT  
10754 045022      016767 132550 172324      MOV      SR2,WASSR2      ;READ SR2 FOLLOWING SECOND KT ABORT  
10755 045030      026767 172324 172314      CMP      $TMP0,WASSRO    ;IS SRO STILL HOLDING INFO ON FIRST ABORT?  
10756 045036      001402      BEQ      9$  
10757 045040      005267 172316      INC      $TMP1          ;BRANCH IF YES  
10758 045044      026767 172314 172302 9$:  CMP      $TMP2,WASSR2    ;SET ERROR INDICATOR  
10759 045052      001402      BEQ      10$           ;DOES SR2 STILL HOLD PC OF FIRST ABORT?  
10760 045054      005267 172302      INC      $TMP1          ;BRANCH IF YES  
10761 045060      005767 172276      10$: TST      $TMP1          ;SET ERROR INDICATOR  
10762 045064      001401      BEQ      11$           ;WERE SRO OR SR2 CHANGED BY A SECOND ABORT?  
10763 045066      104000      EMT  
10764                                     ;ONE OF STATUS REGS. CHANGED BY SECOND ABORT  
10765                                     ;FOR TIGHTER SCOPE LOOP  
10766                                     ;REPLACE ERROR CALL WITH  
10767 045070      005067 172266      11$: CLR      $TMP1          ;"BR 6$" = 000726  
10768 045074      000005      RESET      ;CLEAR ERROR INDICATOR  
10769 045076      005067 132470      CLR      SRO          ;EXECUTE A RESET, APPLYING AN "INIT"  
10770 045102      016767 132464 172242      MOV      SRO,WASSRO      ;READ SRO  
10771 045110      005767 172236      TST      WASSRO        ;WAS SRO CLEARED BY THE RESET?  
10772 045114      001402      BEQ      12$           ;BRANCH IF YES
```

```

10773 045116 005267 172240          INC      $TMP1      ;SRO NOT CLEARED BY A RESET
10774 045122 016767 132450 172224 12$: MOV      SR2,WASSR2 ;READ SR2
10775 045130 022767 045122 172216    CMP      #12$,WASSR2 ;WAS SR2 UNLOCKED BY A RESET?
10776 045136 001402          BEQ      13$        ;BRANCH IF YES
10777 045140 005267 172216          INC      $TMP1      ;SR2 NOT UNLOCKED BY A RESET
10778 045144 005767 172212    13$: TST      $TMP1 ;WERE SRO & SR2 BOTH 'RESET' BY A RESET?
10779 045150 001401          BEQ      14$
10780 045152 104000          EMT
10781          ;SRO OR SR2 NOT 'RESET' BY A RESET
10782          ;FOR TIGHTER SCOPE LOOP
10783          ;REPLACE ERROR CALL WITH
10784 045154 012767 000001 132410 14$: MOV      #1,SRO    ;TURN MEMORY MANAGEMENT BACK ON
10785 045162 016767 132410 172164 15$: MOV      SR2,WASSR2 ;READ SR2 TO SEE IF ITS TRACKING AGAIN
10786 045170 012701 045162          MOV      #15$,R1   ;PUT EXPECTED VIRTUAL PC IN R1
10787 045174 020167 172154          CMP      R1,WASSR2 ;DID SR2 CONTAIN VIRTUAL PC AT 15$
10788 045200 001401          BEQ      16$
10789 045202 104000          EMT
10790          ;SR2 NOT TRACKING CORRECTLY
10791          ;FOR TIGHTER SCOPE LOOP
10792          ;REPLACE ERROR CALL WITH
10793 045204 012767 077406 125076 16$: MOV      #77406,KIPDR4 ;RESET PDR4 TO 128 BLKS, R/W
10794 045212 012767 077406 125072    MOV      #77406,KIPDR5 ;RESET PDR5 TO 128 BLKS, R/W
10795 045220 012767 021356 133022    MOV      #T0250,MMVEC ;RESTORE ADDRESS OF NORMAL MEMORY
10796          ;MANAGEMENT TRAP ROUTINE TO M.M. VECTOR
10797
10798
10799
10800          ;*****
10801          ;TEST 407      USER ABORT PICKS UP KERNEL SPACE VECTOR
10802          ;*****
10803 045226          TS407:
10804 045232 004767 002614    1$: JSR      PC,TOFF ;TURN OFF T-BIT TRAPPING FOR THIS TEST
10805 045236 012706 001000    2$: CLR      PSW      ;GO TO KERNEL MODE
10806 045242 012767 001400 132370    MOV      #KERSTK,KSP ;SETUP KERNEL STACK PTR.
10807 045250 012737 045320 000004    MOV      #1400,UIPARO ;MAP USER PAGE 0 TO 24K
10808 045256 012737 000340 000006    MOV      #4$,a#4     ;LOAD KERNEL VECTOR 4 (LOC.4) WITH 4$
10809 045264 012767 140000 132504    MOV      #340,a#6   ;LOAD VECTOR+2 WITH NEW PSW
10810 045272 012706 000600    MOV      #140000,PSW ;GO TO USER MODE
10811 045276 012737 045316 000004    MOV      #USESTK,USP ;SETUP USER STACK PTR.
10812 045304 012737 000340 000006    MOV      #3$,a#4     ;LOAD USER VECTOR 4 (LOC. 60004) WITH 3$
10813 045312 005767 112462    MOV      #340,a#6   ;LOAD VECTOR+2 WITH NEW PSW
10814          TST      160000 ;CAUSE TIMEOUT ERROR TRAP TO '4'
10815          ;SHOULD PICK UP NEW PC=4$ FROM KERNEL
10816 045316          3$: EMT      ;LOC. 4, NOT PC=3$ FROM USER LOC. 4 (=60004)
10817 045316 104000          ;DID NOT TRAP THRU KERNEL SPACE
10818          ;FOR TIGHTER SCOPE LOOP
10819          ;REPLACE ERROR CALL WITH
10820          ;'BR 2$' = 000740
10821 045320 005067 132452    4$: CLR      PSW      ;BE SURE BACK IN KERNEL MODE
10822 045324 012706 001000    MOV      #KERSTK,KSP ;RESTORE KERNEL S.P. IN CASE IT CHANGED
10823 045330 005067 132304    CLR      UIPARO     ;REMAP USER PAGE 0 TO 0-4K
10824 045334 012767 140000 132434    MOV      #140000,PSW ;GO TO USER MODE
10825 045342 012706 000600    MOV      #USESTK,USP ;RESTORE USER STACK POINTER
10826 045346 005067 132424    CLR      PSW      ;GO BACK TO KERNEL MODE
10827 045352 012737 021336 000004    MOV      #T04,a#4   ;RESTORE ADDR. OF NORMAL CPU TRAP HANDLER TO 4
10828 045360 004767 002516    JSR      PC,T0N    ;TURN T-BIT TRAPPING BACK ON

```



```
10829
10830
10831
10832
10833 045364
10834
10835 045364 012702 170000
10836 045370 010267 132402
10837 045374 012746 000340
10838 045400 012746 045406
10839 045404 000002
10840 045406 016701 132364
10841 045412 042701 007437
10842 045416 005067 132354
10843 045422 020201
10844 045424 001401
10845 045426 104000
10846
10847
10848
10849
10850
10851
10852
10853
10854 045430
10855 045430 012705 077006
10856 045434 010567 124656
10857 045440 012737 045460 000004
10858 045446 012737 045462 000250
10859 045454 005237 177700
10860 045460
10861 045460 104000
10862
10863
10864
10865 045462 012706 001000
10866 045466 016767 132100 171656
10867 045474 016767 132076 171652
10868 045502 012700 040017
10869 045506 020067 171640
10870 045512 001401
10871 045514 104000
10872
10873
10874
10875 045516 012701 045454
10876 045522 020167 171626
10877 045526 001401
10878 045530 104000
10879
10880
10881
10882 045532 042767 160000 132032
10883 045540 012737 021336 000004
10884 045546 012737 021356 000250
```

```
*****
:TEST 410 RTI IN USER MODE DOES NOT CHANGE PSW
*****
TS410:
2$: MOV #170000,R2 ;LOAD 'PRESENT & EXPECTED' PSW VALUE INTO R2
MOV R2,PSW ;GO TO USER MODE-PRIORITY 0
MOV #340,-(SP) ;PUT A NEW PSW (PRIORITY=7) ON STACK
MOV #3$,-(SP) ;PUT NEW PC ON THE STACK
RTI ;DO AN RTI FROM USER MODE
3$: MOV PSW,R1 ;READ NEW PSW INTO R1
BIC #7437,R1 ;MASK OFF COND. CODE, T-BIT, AND UNUSED BITS
CLR PSW ;GO BACK TO KERNEL MODE
CMP R2,R1 ;DID PSW STAY IN USER, PRIORITY=0?
BEQ TS411
EMT ;PSW CHANGED BY AN RTI FROM USER
;FOR A TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;'BR=2$' = 000760

*****
:TEST 411 KT ERROR SERVICED BEFORE TIMEOUT ERROR
*****
TS411:
1$: MOV #77006,R5 ;LOAD PDR7 DATA INTO R5
MOV R5,KIPDR7 ;MAP PAGE 7 R/W PLF=176
MOV #3$,@#4 ;SET CPU TRAP VECTOR TO ADDRESS OF 3$
MOV #4$,@#250 ;SET M.M. TRAP VECTOR TO ADDRESS OF 4$
2$: INC @#177700 ;CAUSE PLF ABORT AND POTENTIAL TIMEOUT
3$: EMT ;TRAPPED THRU CPU TRAP VECTOR BUT SHOULDN'T HAVE
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;'BR 2$' = 000776
4$: MOV #KERSTK,KSP ;RESTORE STACK POINTER AFTER TRAPPING
MOV SRO,WASSR0 ;READ STATUS REG.0
5$: MOV SR2,WASSR2 ;READ STATUS REG. 2
MOV #40017,R0 ;LOAD EXPECTED SRO CONTENTS INTO R0
CMP R0,WASSR0 ;SRO PLF ERROR BIT SET?
BEQ 6$
EMT ;SRO DIDN'T REPORT PLF ERROR
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;'BR 2$' = 000741
6$: MOV #2$,R1 ;LOAD EXPECTED SR2 CONTENTS INTO R1
CMP R1,WASSR2 ;WAS SR2 LOCKED BY PLF ABORT?
BEQ 7$
EMT ;SR2 DIDN'T LOCK UP VIRTUAL ADDRESS
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;'BR 2$' = 000741
7$: BIC #160000,SRO ;CLEAR ERROR BITS THAT WERE SET IN SRO
MOV #T04,@#4 ;RESTORE ADDRESS OF NORMAL CPU TRAP HANDLER
MOV #T0250,@#250 ;RESTORE ADDRESS OF NORMAL M.M. TRAP HANDLER
```

10885 045554 012767 077406 124534
10886
10887
10888
10889
10890
10891 045562
10892 045562 004767 002260
10893 045566 012767 001400 132052
10894 045574 012767 001400 132046
10895 045602 012767 077402 131776
10896 045610 012767 077406 131772
10897 045616 012737 045664 000004
10898 045624 012737 140017 000006
10899 045632 012737 045664 000250
10900 045640 012737 000340 000252
10901 045646 012767 140000 132122
10902 045654 012706 100002
10903 045660 005737 177700
10904
10905 045664 016601 000002
10906 045670 011603
10907 045672 016767 131674 171452
10908 045700 016767 131672 171446
10909 045706 042767 160000 131656
10910 045714 005067 132056
10911 045720 012706 001000
10912 045724 012767 140000 132044
10913 045732 012706 000600
10914 045736 005067 132034
10915 045742 005067 171412
10916 045746 020127 170017
10917
10918
10919
10920 045752 001402
10921 045754 005267 171400
10922 045760 020327 045664
10923
10924 045764 001402
10925 045766 005267 171366
10926 045772 026727 171354 020147
10927 046000 001402
10928 046002 005267 171352
10929 046006 026727 171342 045660
10930
10931 046014 001402
10932 046016 005267 171336
10933 046022 005767 171332
10934 046026 001401
10935 046030 104000
10936
10937
10938
10939
10940

```
MOV #77406,KIPDR7 ;REMAP PAGE 7 TO READ/WRITE PLF=177
:*****
:TEST 412 PC & PSW SAVED FOR KT ERROR DURING SERVICE OF TIMEOUT ERROR
:*****
TS412:
1$: JSR PC,TOFF ;TURN T-BIT TRAPPING OFF FOR THIS TEST
MOV #1400,UIPAR3 ;MAP USER PAGE 3 TO 24-28K
MOV #1400,UIPAR4 ;MAP USER PAGE 4 TO 24-28K
MOV #77402,UIPDR3 ;MAP USER PAGE 3 READ-ONLY
MOV #77406,UIPDR4 ;MAP USER PAGE 4 READ/WRITE
MOV #4$,@#4 ;LOAD ADDRESS OF 4$ IN CPU (TIMEOUT) VECTOR
MOV #140017,@#6 ;LOAD PSW THAT SHOULD BE PUT ON STACK IN VECTOR+2
MOV #4$,@#250 ;LOAD ADDRESS OF 4$ IN M.M. TRAP VECTOR
MOV #340,@#252 ;LOAD A KERNEL PSW IN MMVEC+2
2$: MOV #140000,PSW ;GO TO USER MODE
MOV #100002,USP ;SET USER STACK PTR. SO SECOND PUSH IS IN PG. 3
3$: TST @#177700 ;CAUSE TIMEOUT ERROR THAT WILL CAUSE
;R/O ERROR WHEN TRY TO SAVE OLD PC
4$: MOV 2(KSP),R1 ;PUT PSW SAVED ON KERNEL STACK INTO R1
MOV (KSP),R3 ;PUT PC SAVED ON KERNEL STACK INTO R3
MOV SRO,WASSRO ;READ THE CONTENTS OF M.M. STATUS REG. 0
MOV SR2,WASSR2 ;READ THE CONTENTS OF M.M. STATUS REG. 2
BIC #160000,SRO ;CLEAR THE ERROR BITS IN SRO
CLR PSW ;BE SURE IN KERNEL MODE
MOV #KERSTK,KSP ;RESTORE KERNEL STACK POINTER
MOV #140000,PSW ;GO TO USER MODE
MOV #USESTK,USP ;RESTORE USER STACK POINTER
CLR PSW ;GO BACK TO KERNEL MODE
CLR $TMP0 ;CLEAR ERROR INDICATOR
CMP R1,#170017 ;WAS THE PSW SAVED THE ONE PICKED UP BY THE
;TIMEOUT TRAP FROM ERRVEC+2?
;VALUE 170017 = PSW FROM LOC. 6 WITH
;PREVIOUS MODE BITS = USER
5$: BEQ 5$ ;BRANCH IF YES
INC $TMP0 ;WRONG PSW SAVED DURING 'DOUBLE ERROR' SEQUENCE
CMP R3,#3$+4 ;WAS THE PC AT THE TIME OF THE TIMEOUT ERROR
;SAVED ON THE STACK?
6$: BEQ 6$ ;BRANCH IF YES
INC $TMP0 ;WRONG PC SAVED DURING TRAP SEQUENCE
CMP WASSRO,#20147 ;DID SRO REPORT - USER, PAGE 3, R/O ABORT?
7$: BEQ 7$ ;BRANCH IF YES
INC $TMP0 ;SRO DID NOT REPORT R/O ABORT
CMP WASSR2,#3$ ;DID SR2 LOCK UP VIRTUAL ADDR. OF LAST
;INSTRUCTION SUCCESSFULLY FETCHED?
8$: BEQ 8$ ;BRANCH IF YES
INC $TMP0 ;SR2 DID NOT LOCK UP ADDR. OF TIMEOUT INST.
TST $TMP0 ;ANY 'ERRORS' DURING TRAP SEQUENCE?
9$: BEQ 9$
EMT ;THE WRONG PC OR PSW WERE SAVED
;OR SRO OR SR2 DID NOT REPORT R/O
;ERROR DURING TIMEOUT - KT TRAP
;SEQUENCE
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
```

10941
10942 046032 012737 021336 000004
10943 046040 012737 000340 000006
10944 046046 012737 021356 000250
10945 046054 012767 077406 131524
10946 046062 004767 002014
10947
10948
10949
10950
10951
10952
10953
10954
10955
10956
10957
10958 046066
10959 046066 005067 124246
10960 046072 012767 000200 124242
10961 046100 012767 000400 124236
10962 046106 012767 000600 124232
10963 046114 012767 001400 124226
10964 046122 012767 007600 124226
10965 046130 012700 077406
10966
10967 046134 012702 000010
10968 046140 012701 172300
10969 046144 010021
10970 046146 077202
10971 046150 012702 000010
10972 046154 012701 177600
10973 046160 010021
10974 046162 077202
10975 046164 012767 000000 131446
10976 046172 012767 000200 131442
10977 046200 012767 000400 131436
10978 046206 012767 000600 131432
10979 046214 012767 007600 131434
10980 046222
10981 046222 012767 077406 124060
10982 046230 012767 001400 124112
10983 046236 012767 001400 131404
10984 046244 012700 036514
10985 046250 010037 100000
10986 046254 012767 046552 131766
10987 046262 105067 124022
10988
10989
10990 046266 012767 030340 131502
10991 046274 006506
10992
10993 046276 022706 001000
10994 046302 001405
10995 046304 012600
10996 046306 012701 000600

9\$: MOV #T04,@#4 ;'BR 2\$' = 000710
MOV #340,@#6 ;RESTORE ADDRESS OF NORMAL CPU TRAP HANDLER
MOV #T0250,@#250 ;RELOAD ERRVEC+2 WITH KERNEL PSW
MOV #77406,UIPDR3 ;RESTORE ADDRESS OF NORMAL M.M. TRAP HANDLER
JSR PC,TON ;REMAP USER PAGE 3 READ/WRITE
;TURN T-BIT TRAPPING BACK ON

*
* THIS GROUP OF TESTS WILL TEST ALL THE LOGIC ASSOCIATED WITH
* THE 'MOVE FROM PREVIOUS' AND MOVE TO PREVIOUS' INSTRUCTIONS.
*

:TEST 413 MOVE FROM PREVIOUS (USER) I-SPACE

TS413:
1\$: CLR KIPARO ;MAP KERNEL PAGE 0 TO 0-4K
MOV #200,KIPAR1 ;MAP KERNEL PAGE 1 TO 4-8K
MOV #400,KIPAR2 ;MAP KERNEL PAGE 2 TO 8-12K
MOV #600,KIPAR3 ;MAP KERNEL PAGE 3 TO 12-16K
MOV #1400,KIPAR4 ;MAP KERNEL PAGE 4 TO 24-28K
MOV #7600,KIPAR7 ;MAP KERNEL PAGE 7 TO THE I/O PAGE
MOV #77406,R0 ;MAKE ALL KERNEL I-SPACE PAGES RESIDENT
;READ/WRITE, LENGTH 200 BLOCKS
MOV #10,R2 ;SET LOOP COUNTER TO 8
MOV #KIPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
2\$: MOV R0,(R1)+ ;LOAD PDR WITH 77406
SOB R2,2\$;LOOP TO 2\$ UNTIL ALL PDRS LOADED
MOV #10,R2 ;SET LOOP COUNTER TO 8
MOV #UIPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
3\$: MOV R0,(R1)+ ;LOAD PDR WITH 77406
SOB R2,3\$;LOOP TO 3\$ UNTIL ALL PDRS LOADED
MOV #000,UIPAR0 ;MAP USER I PAGE 0 TO 0-4K
MOV #200,UIPAR1 ;MAP USER I PAGE 1 TO 4-8K
MOV #400,UIPAR2 ;MAP USER I PAGE 2 TO 8-12K
MOV #600,UIPAR3 ;MAP USER I PAGE 3 TO 12-16K
MOV #7600,UIPAR7 ;MAP USER I PAGE 7 TO THE I/O PAGE
4\$: MOV #77406,KIPDR4 ;KERNEL I-SPACE PAGE 4 READ/WRITE
MOV #1400,KIPAR4 ;MAP KERNEL I PAGE 4 TO 24K
MOV #1400,UIPAR4 ;MAP USER I PAGE 4 TO 24K
MOV #36514,R0 ;LOAD DATA PATTERN INTO R0
MOV R0,@#100000 ;LOAD DATA PATTERN INTO PHY 140000
MOV #23\$,MMVEC ;SET M.M. VECTOR TO 23\$
CLRB KIPDR4 ;MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
;THE FOLLOWING WILL TEST DSTM=0 MFPI
5\$: MOV #030340,PSW ;MAKE PREVIOUS MODE USER
6\$: MFPI USP ;PUT USER STACK POINTER ON KERNEL
;STACK
CMP #KERSTK,KSP ;WAS SOMETHING PUSHED ON STACK AT 6\$
BEQ 7\$;BRANCH IF NOTHING WAS PUSHED
MOV (KSP)+,R0 ;POP KERNEL STACK INTO R0
MOV #USESTK,R1 ;EXPECTING TO GET 700 AS USP

```
10997 046312 020001          CMP    R0,R1          ;DID YOU GET THE RIGHT POINTER?
10998 046314 001401          BEQ    8$
10999 046316          7$:
11000 046316 104000          EMT                  ;WRONG THING WAS PUSHED ON STACK
11001                                     ;FOR TIGHTER SCOPE LOOP
11002                                     ;REPLACE ERROR CALL WITH
11003                                     ;'BR 5$' = 000763
11004 046320          8$:          ;THE FOLLOWING WILL TEST DSTM=1 MFPI.
11005 046320 012700 036514          MOV    #36514,R0      ;RELOAD DATA PATTERN IN R0
11006 046324 012767 030340 131444 9$:          MOV    #030340,PSW   ;MAKE PREVIOUS MODE USER
11007 046332 012702 100000          MOV    #100000,R2    ;LOAD VIRTUAL ADDRESS INTO R2
11008 046336 006512          MFPI   (R2)          ;READ FROM PHYSICAL 140000
11009 046340 012601          MOV    (KSP)+,R1     ;POP KERNEL STACK INTO R1
11010 046342 020001          CMP    R0,R1          ;WAS DATA FETCHED SAME AS STORED
11011 046344 001401          BEQ    10$
11012 046346 104000          EMT                  ;WRONG DATA WAS FETCHED
11013                                     ;FOR TIGHTER SCOPE LOOP
11014                                     ;REPLACE ERROR CALL WITH
11015                                     ;'BR 9$' = 000766
11016 046350          10$:         ;THE FOLLOWING WILL TEST DSTM=2 MFPI.
11017 046350 012767 030340 131420 11$:          MOV    #030340,PSW   ;MAKE PREVIOUS MODE USER
11018 046356 012702 100000          MOV    #100000,R2    ;LOAD VIRTUAL ADDRESS INTO R2
11019 046362 006522          MFPI   (R2)+         ;READ FROM PHYSICAL 140000
11020 046364 012601          MOV    (KSP)+,R1     ;POP KERNEL STACK INTO R1
11021 046366 020001          CMP    R0,R1          ;WAS DATA FETCHED SAME AS STORED
11022 046370 001401          BEQ    12$
11023 046372 104000          EMT                  ;WRONG DATA WAS FETCHED
11024                                     ;FOR TIGHTER SCOPE LOOP
11025                                     ;REPLACE ERROR CALL WITH
11026                                     ;'BR 11$' = 000766
11027 046374          12$:         ;THE FOLLOWING WILL TEST DSTM=3 MFPI.
11028 046374 012767 030340 131374 13$:          MOV    #030340,PSW   ;MAKE PREVIOUS MODE USER
11029 046402 006537 100000          MFPI   #100000       ;READ FROM PHYSICAL 140000
11030 046406 012601          MOV    (KSP)+,R1     ;POP KERNEL STACK INTO R1
11031 046410 020001          CMP    R0,R1          ;WAS DATA FETCHED SAME AS STORED
11032 046412 001401          BEQ    14$
11033 046414 104000          EMT                  ;WRONG DATA WAS FETCHED
11034                                     ;FOR TIGHTER SCOPE LOOP
11035                                     ;REPLACE ERROR CALL WITH
11036                                     ;'BR 13$' = 000767
11037 046416          14$:         ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
11038 046416 012767 030340 131352 15$:          MOV    #030340,PSW   ;MAKE PREVIOUS MODE USER
11039 046424 012702 100002          MOV    #100002,R2    ;LOAD VIRTUAL ADDRESS INTO R2
11040 046430 006542          MFPI   -(R2)         ;READ FROM PHYSICAL 140000
11041 046432 012601          MOV    (KSP)+,R1     ;POP KERNEL STACK INTO R1
11042 046434 020001          CMP    R0,R1          ;WAS DATA FETCHED SAME AS STORED
11043 046436 001401          BEQ    16$
11044 046440 104000          EMT                  ;WRONG DATA WAS FETCHED
11045                                     ;FOR TIGHTER SCOPE LOOP
11046                                     ;REPLACE ERROR CALL WITH
11047                                     ;'BR 15$' = 000766
11048 046442          16$:
11049                                     ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
11050                                     ;
11051 046442 012767 030340 131326 17$:          MOV    #030340,PSW   ;MAKE PREVIOUS MODE USER
11052 046450 012767 100000 170706          MOV    #100000,$TMP2 ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
```


CJKDJBO 11/23-B CPU CLUSTER DIAG.
CJKDJB.P11 26-MAY-82 11:14

DNMAC X24.07-563 26-MAY-82^{I 1} 11:18 PAGE 215
T413 MOVE FROM PREVIOUS (USER) I-SPACE

SEQ 0214

11087 ;REPLACE ERROR CALL WITH
11088 ;'BR 21\$' = 000762
11089 046554 012767 021356 131466 22\$: MOV #T0250,MMVEC ;SET M.M. VECTOR TO NORMAL ROUTINE
11090
11091

```
11092 ;*****
11093 ;TEST 414 MOVE TO PREVIOUS (USER) I-SPACE
11094 ;*****
11095 046562 TS414:
11096 046562 012767 077406 123520 1$: MOV #77406,KIPDR4 ;KERNEL I-SPACE PAGE 4 READ/WRITE
11097 046570 012767 077406 131012 MOV #77406,UIPDR4 ;USER I-SPACE PAGE 4 READ/WRITE
11098 046576 012767 001400 123544 MOV #1400,KIPAR4 ;MAP KERNEL I PAGE 4 TO 24K
11099 046604 012767 001400 131036 MOV #1400,UIPAR4 ;MAP USER I PAGE 4 TO 24K
11100 046612 012767 047304 131430 MOV #20$,MMVEC ;SET M.M. VECTOR TO 20$
11101 ;THE FOLLOWING WILL TEST DSTM=0 MTPI
11102 ;
11103 046620 012767 030340 131150 2$: MOV #030340,PSW ;MAKE PREVIOUS MODE USER
11104 046626 012746 007777 MOV #7777,-(KSP) ;PUSH DATA ON KERNEL STACK
11105 046632 006606 MTPI USP ;LOAD USER STACK POINTER
11106 046634 006506 MFPI USP ;READ USER STACK POINTER
11107 046636 012601 MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
11108 046640 022701 007777 CMP #7777,R1 ;WAS USER STACK POINTER CHANGED
11109 046644 001401 BEQ 3$
11110 046646 104000 EMT ;USER STACK POINTER NOT CHANGED
11111 ;FOR TIGHTER SCOPE LOOP
11112 ;REPLACE ERROR CALL WITH
11113 ;'BR 2$' = 000764
11114 046650 012767 030340 131120 3$: MOV #030340,PSW ;MAKE PREVIOUS MODE USER
11115 046656 012746 000600 MOV #USESTK,-(KSP) ;GET READY TO RESTORE USER S. POINT
11116 046662 006606 MTPI USP ;RESTORE USER STACK POINTER
11117 046664 4$: ;THIS WILL TEST DSTM = 1 MTPI.
11118 046664 012702 100000 MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
11119 046670 012700 125252 MOV #125252,R0 ;LOAD TEST DATA INTO R0
11120 046674 010046 5$: MOV RO,-(KSP) ;PUSH TEST DATA ON KERNEL STACK
11121 046676 105067 123406 CLRB KIPDR4 ;MAKE KERNEL I PAGE 4 NON-RESIDENT
11122 046702 006612 MTPI (R2) ;LOAD TEST DATA INTO PHYSICAL 140000
11123 046704 112767 000006 123376 MOVB #006,KIPDR4 ;MAKE KERNEL PAGE 4 RESIDENT
11124 046712 011201 MOV (R2),R1 ;READ FROM ADDRESS 140000
11125 046714 020001 CMP RO,R1 ;SEE IF DATA WAS STORED AT CORRECT PLACE
11126 046716 001401 BEQ 6$
11127 046720 104000 EMT ;INCORRECT STORE
11128 ;FOR TIGHTER SCOPE LOOP
11129 ;REPLACE ERROR CALL WITH
11130 ;'BR 5$' = 000765
11131 046722 6$: ;THE FOLLOWING WILL TEST DSTM=2 MTPI.
11132 ;
11133 046722 012767 030340 131046 MOV #030340,PSW ;MAKE PREVIOUS MODE USER
11134 046730 012700 125252 MOV #125252,R0 ;LOAD TEST DATA INTO R0
11135 046734 012702 100000 MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
11136 046740 010046 8$: MOV RO,-(KSP) ;PUSH TEST DATA ON KERNEL STACK
11137 046742 105067 123342 CLRB KIPDR4 ;MAKE KERNEL PAGE 4 NON-RESIDENT
11138 046746 006612 MTPI (R2) ;LOAD TEST DATA INTO PHYSICAL 140000
11139 046750 112767 000006 123332 MOVB #006,KIPDR4 ;MAKE KERNEL PAGE 4 RESIDENT
11140 046756 013701 100000 MOV #100000,R1 ;READ FROM ADDRESS 140000
11141 046762 020001 CMP RO,R1 ;SEE IF DATA WAS STORED CORRECTLY
11142 046764 001401 BEQ 9$
11143 046766 104000 EMT ;INCORRECT STORE
11144 ;FOR TIGHTER SCOPE LOOP
11145 ;REPLACE ERROR CALL WITH
11146 ;'BR 8$' = 000764
11147 046770 9$: ;THIS WILL TEST DSTM = 3 MTPI.
```

11148	046770	012767	030340	131000	MOV	#030340,PSW	;MAKE PREVIOUS MODE USER
11149	046776	012700	052525		MOV	#52525,R0	;LOAD TEST DATA INTO R0
11150	047002	010046			10\$: MOV	R0,-(KSP)	;PUSH TEST DATA ON KERNEL STACK
11151	047004	105067	123300		CLRB	KIPDR4	;MAKE KERNEL I PAGE 4 NON-RESIDENT
11152	047010	006637	100000		MTPI	@#100000	;LOAD TEST DATA INTO PHYSICAL 140000
11153	047014	112767	000006	123266	MOVB	#006,KIPDR4	;MAKE KERNEL PAGE 4 RESIDENT
11154	047022	013701	100000		MOV	@#100000,R1	;READ FROM ADDRESS 140000
11155	047026	020001			CMP	R0,R1	;SEE IF DATA WAS STORED CORRECTLY
11156	047030	001531			BEQ	TS415	
11157	047032	104000			EMT		
11158	047034	001401			BEQ	11\$	
11159	047036	104000			EMT		;INCORRECT STORE
11160							;FOR TIGHTER SCOPE LOOP
11161							;REPLACE ERROR CALL WITH
11162							;'BR 10\$' = 000763
11163	047040				11\$:	;THIS WILL TEST DSTM = 4 MTPI.	
11164	047040	012767	030340	130730	MOV	#030340,PSW	;MAKE PREVIOUS MODE USER
11165	047046	012700	125252		MOV	#125252,R0	;LOAD TEST DATA INTO R0
11166	047052	010046			12\$: MOV	R0,-(KSP)	;PUSH TEST DATA ON KERNEL STACK
11167	047054	012702	100002		MOV	#100002,R2	;LOAD VIRTUAL ADDRESS INTO R2
11168	047060	105067	123224		CLRB	KIPDR4	;MAKE KERNEL I PAGE 4 NON-RESIDENT
11169	047064	006642			MTPI	-(R2)	;LOAD TEST DATA INTO PHYSICAL 140000
11170	047066	112767	000006	123214	MOVB	#006,KIPDR4	;MAKE KERNEL PAGE 4 RESIDENT
11171	047074	013701	100000		MOV	@#100000,R1	;READ FROM ADDRESS 140000
11172	047100	020001			CMP	R0,R1	;SEE IF DATA WAS STORED CORRECTLY
11173	047102	001401			BEQ	13\$	
11174	047104	104000			EMT		;INCORRECT STORE
11175							;FOR TIGHTER SCOPE LOOP
11176							;REPLACE ERROR CALL WITH
11177							;'BR 12\$' = 000762
11178	047106				13\$:	;THE FOLLOWING WILL TEST DSTM=5 MTPI.	
11179							
11180	047106	012767	030340	130662	MOV	#030340,PSW	;MAKE PREVIOUS MODE USER
11181	047114	012700	052525		MOV	#52525,R0	;LOAD TEST DATA INTO R0
11182	047120	012702	037366		MOV	#<STMP2+2>,R2	;LOAD ADDR. OF LGC. STMP2+2 INTO R2
11183	047124	012767	100000	170232	MOV	#100000,STMP2	;LOAD VIRT. ADDR. OF TEST LOC. INTO STMP2
11184	047132	010046			14\$: MOV	R0,-(KSP)	;PUSH TEST DATA ON KERNEL STACK
11185	047134	105067	123150		CLRB	KIPDR4	;MAKE KERNEL PAGE 4 NON-RESIDENT
11186	047140	006652			MTPI	@-(R2)	;LOAD TEST DATA INTO PHYSICAL 140000
11187	047142	112767	000006	123140	MOVB	#006,KIPDR4	;MAKE KERNEL PAGE 4 RESIDENT
11188	047150	013701	100000		MOV	@#100000,R1	;READ FROM ADDRESS 140000
11189	047154	020001			CMP	R0,R1	;SEE IF DATA WAS STORED CORRECTLY
11190	047156	001401			BEQ	15\$	
11191	047160	104000			EMT		;INCORRECT STORE
11192							;FOR TIGHTER SCOPE LOOP
11193							;REPLACE ERROR CALL WITH
11194							;'BR 14\$' = 000764
11195	047162				15\$:	;THIS WILL TEST DSTM = 6 MTPI.	
11196							
11197	047162	012767	030340	130606	MOV	#030340,PSW	;MAKE PREVIOUS MODE USER
11198	047170	012700	052525		MOV	#52525,R0	;LOAD TEST DATA INTO R0
11199	047174	005002			CLR	R2	;MAKE REGISTER 2 ZERO
11200	047176	010046			16\$: MOV	R0,-(KSP)	;PUSH TEST DATA ON KERNEL STACK
11201	047200	105067	123104		CLRB	KIPDR4	;MAKE KERNEL I PAGE 4 NON-RESIDENT
11202	047204	006662	100000		MTPI	100000(R2)	;LOAD TEST DATA INTO PHYSICAL 140000
11203	047210	112767	000006	123072	MOVB	#006,KIPDR4	;MAKE KERNEL PAGE 4 RESIDENT


```
11204 047216 013701 100000      MOV    @#100000,R1    ;READ FROM ADDRESS 140000
11205 047222 020001              CMP    R0,R1         ;SEE IF DATA WAS STORED CORRECTLY
11206 047224 001401              BEQ    17$           ;
11207 047226 104000              EMT                    ;INCORRECT STORE
11208                                ;FOR TIGHTER SCOPE LOOP
11209                                ;REPLACE ERROR CALL WITH
11210                                ;'BR 16$' = 000763
11211 047230      17$:      ;THE FOLLOWING WILL TEST DSTM=7 MTP1.
11212                                ;
11213 047230 012767 030340 130540  MOV    #030340,PSW   ;MAKE PREVIOUS MODE USER
11214 047236 012700 125252              MOV    #125252,R0    ;LOAD TEST DATA INTO R0
11215 047242 012767 100000 170114  MOV    #100000,$TMP2 ;LOAD VIRT. ADDR. OF TEST LOCATION
11216                                ;INTO LOCATION $TMP2
11217 047250 012702 037364              MOV    # $TMP2,R2    ;LOAD ADDRESS OF $TMP2 INTO R2
11218 047254 010046      18$:      MOV    R0,-(KSP)      ;PUSH TEST DATA ON KERNEL STACK
11219 047256 105067              CLRB   KIPDR4        ;MAKE KERNEL PAGE 4 NON-RESIDENT
11220 047262 006672 000000              MTP1   @0(R2)        ;LOAD TEST DATA INTO PHYSICAL 140000
11221 047266 112767 000006 123014  MOVVB  #006,KIPDR4   ;MAKE KERNEL PAGE 4 RESIDENT
11222 047274 013701 100000              MOV    @#100000,R1  ;READ FROM ADDRESS 140000
11223 047300 020001              CMP    R0,R1         ;SEE IF DATA WAS STORED CORRECTLY
11224 047302 001401              BEQ    19$           ;
11225 047304      20$:      ;
11226 047304 104000              EMT                    ;INCORRECT STORE
11227                                ;FOR TIGHTER SCOPE LOOP
11228                                ;REPLACE ERROR CALL WITH
11229                                ;'BR 18$' = 000763
11230 047306 012767 021356 130734 19$:      MOV    #T0250,MMVEC ;RESTORE M.M. VECTOR TO NORMAL ROUTINE
11231
11232
11233
11234                                ;*****
11235                                ;TEST 415      MOVE FROM PREVIOUS (KERNEL) I-SPACE TO USER MODE
11236                                ;*****
11237 047314      TS415:
11238 047314 012700 077406      1$:      MOV    #77406,R0     ;MAKE ALL USER I-SPACE PAGES RESIDENT
11239                                ;READ/WRITE, LENGTH 200 BLOCKS
11240 047320 012702 000010              MOV    #10,R2        ;SET LOOP COUNTER TO 8
11241 047324 012701 177600              MOV    #UIPDR0,R1    ;LOAD ADDRESS OF FIRST PDR IN R1
11242 047330 010021      2$:      MOV    R0,(R1)+      ;LOAD PDR WITH 77406
11243 047332 077202              SOB    R2,2$         ;LOOP UNTIL 8 USER PDRS LOADED
11244 047334 012767 140340 130434  3$:      MOV    #140340,PSW   ;GO TO USER MODE FOR THIS TEST
11245 047342 012767 077406 122740  MOV    #77406,KIPDR4 ;KERNEL I-SPACE PAGE 4 READ/WRITE
11246 047350 012767 001400 122772  MOV    #1400,KIPAR4  ;MAP KERNEL I PAGE 4 TO 24K
11247 047356 012767 001400 130264  MOV    #1400,UIPAR4  ;MAP USER I PAGE 4 TO 24K
11248 047364 012700 036514              MOV    #36514,R0     ;LOAD DATA PATTERN INTO R0
11249 047370 010037 100000              MOV    R0,@#100000  ;LOAD DATA PATTERN INTO PHY 140000
11250 047374 012702 100000              MOV    #100000,R2   ;LOAD VIRTUAL ADDRESS INTO R2
11251                                ;THE FOLLOWING WILL TEST DSTM=0 MFPI
11252                                ;
11253 047400 012767 047676 130642  MOV    #21$,MMVEC   ;SET M.M. VECTOR TO 21$
11254 047406 105067 130176              CLRB   UIPDR4        ;MAKE USER I-SPACE PAGE 4 NON-RESIDENT
11255 047412 012767 140340 130356  MOV    #140340,PSW   ;MAKE PREVIOUS MODE KERNEL PRESENT USER
11256 047420 006506      4$:      MFPI   KSP           ;PUT KERNEL STACK POINTER ON USER STACK
11257 047422 022706 000600              CMP    #USESTK,USP   ;WAS SOMETHING PUSHED ON STACK AT 1$
11258 047426 001405              BEQ    5$           ;BRANCH IF NOTHING WAS PUSHED
11259 047430 012600              MOV    (USP)+,R0    ;POP USER STACK INTO R0
```

```
11260 047432 012701 001000      MOV    #KERSTK,R1      ;EXPECTING 1100 AS KSP
11261 047436 020001                CMP    RO,R1           ;DID YOU GET THE RIGHT POINTER?
11262 047440 001401                BEQ    6$
11263 047442                        5$:
11264 047442 104000                EMT                    ;WRONG THING WAS PUSHED ON STACK
11265                                ;FOR TIGHTER SCOPE LOOP
11266                                ;REPLACE ERROR CALL WITH
11267                                ;'BR 4$' = 000766
11268 047444                        6$:
11269 047444 012767 140340 130324 7$: ;THE FOLLOWING WILL TEST DSTM=1 MFPI.
11270 047452 012700 036514        MOV    #140340,PSW    ;MAKE PREVIOUS MODE KERNEL PRESENT USER
11271 047456 012702 100000        MOV    #36514,RO     ;LOAD DATA EXPECTED INTO RO
11272 047462 006512                MOV    #100000,R2    ;LOAD VIRTUAL ADDRESS INTO R2
11273 047464 012601                MFPI   (R2)          ;READ FROM PHYSICAL 140000
11274 047466 020001                MOV    (USP)+,R1    ;POP USER STACK INTO R1
11275 047470 001401                CMP    RO,R1        ;WAS DATA FETCHED SAME AS STORED
11276 047472 104000                BEQ    9$
11277                                EMT                    ;WRONG DATA WAS FETCHED
11278                                ;FOR TIGHTER SCOPE LOOP
11279                                ;REPLACE ERROR CALL WITH
11280                                ;'BR 7$' = 000764
11281 047474 012767 140340 130274 9$: ;THE FOLLOWING WILL TEST DSM=2 MFPI.
11282 047502 012702 100000        MOV    #140340,PSW    ;MAKE PREVIOUS MODE KERNEL PRESENT USER
11283 047506 006522                MOV    #100000,R2    ;LOAD VIRTUAL ADDRESS INTO R2
11284 047510 012601                MFPI   (R2)+        ;READ FROM PHYSICAL 140000
11285 047512 020001                MOV    (USP)+,R1    ;POP USER STACK INTO R1
11286 047514 001401                CMP    RO,R1        ;WAS DATA FETCHED SAME AS STORED
11287 047516 104000                BEQ    11$
11288                                EMT                    ;WRONG DATA WAS FETCHED
11289                                ;FOR TIGHTER SCOPE LOOP
11290                                ;REPLACE ERROR CALL WITH
11291                                ;'BR 9$' = 000766
11292 047520 012767 140340 130250 11$: ;THE FOLLOWING WILL TEST DSTM=3 MFPI.
11293 047526 006537 100000        MOV    #140340,PSW    ;MAKE PREVIOUS MODE KERNEL PRESENT USER
11294 047532 012601                MFPI   @#100000     ;READ FROM PHYSICAL 140000
11295 047534 0200.1                MOV    (USP)+,R1    ;POP USER STACK INTO R1
11296 047536 001401                CMP    RO,R1        ;WAS DATA FETCHED SAME AS STORED
11297 047540 104000                BEQ    13$
11298                                EMT                    ;WRONG DATA WAS FETCHED
11299                                ;FOR TIGHTER SCOPE LOOP
11300                                ;REPLACE ERROR CALL WITH
11301                                ;'BR 11$' = 000767
11302 047542 012767 140340 130226 13$: ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
11303 047550 012702 100002        MOV    #140340,PSW    ;MAKE PREVIOUS MODE DERNEL PRESENT USER
11304 047554 006542                MOV    #100002,R2    ;LOAD VIRTUAL ADDRESS INTO R2
11305 047556 012601                MFPI   -(R2)        ;READ FROM PHYSICAL 140000
11306 047560 020001                MOV    (USP)+,R1    ;POP USER STACK INTO R1
11307 047562 001401                CMP    RO,R1        ;WAS DATA FETCHED SAME AS STORED
11308 047564 104000                BEQ    15$
11309                                EMT                    ;WRONG DATA WAS FETCHED
11310                                ;FOR TIGHTER SCOPE LOOP
11311                                ;REPLACE ERROR CALL WITH
11312                                ;'BR 13$' = 000766
11313                                ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
11314 047566 012767 140340 130202 15$:
11315 047574 012767 100000 167562  MOV    #140340,PSW    ;MAKE PREVIOUS MODE KERNEL PRESENT USER
                                MOV    #100000,$TMP2 ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
```

```
11316 047602 012702 037366      MOV    #<STMP2+2>,R2 ;LOAD ADDRESS OF $TMP2+2 INTO R2
11317 047606 006552      MFPI   @-(R2) ;READ FROM PHYSICAL 140000
11318 047610 012601      MOV    (USP)+,R1 ;POP USER STACK INTO R1
11319 047612 020001      CMP    R0,R1 ;WAS DATA FETCHED SAME AS STORED
11320 047614 001401      BEQ    17$
11321 047616 104000      EMT ;WRONG DATA WAS FETCHED
11322 ;FOR TIGHTER SCOPE LOOP
11323 ;REPLACE ERROR CALL WITH
11324 ;'BR 15$' = 000763
11325 ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
11326 ;
11327 047620 012767 140340 130150 17$: MOV    #140340,PSW ;MAKE PREVIOUS MODE KERNEL PRESENT USER
11328 047626 005002      CLR    R2 ;MAKE REGISTER 2 A ZERO
11329 047630 006562 100000      MFPI   100000(R2) ;READ FROM PHYSICAL 140000
11330 047634 012601      MOV    (USP)+,R1 ;POP USER STACK INTO R1
11331 047636 020001      CMP    R0,R1 ;WAS DATA FETCHED SAME AS STORED
11332 047640 001401      BEQ    19$
11333 047642 104000      EMT ;WRONG DATA WAS FETCHED
11334 ;FOR TIGHTER SCOPE LOOP
11335 ;REPLACE ERROR CALL WITH
11336 ;'BR 17$' = 000766
11337 ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
11338 ;
11339 047644 012767 140340 130124 19$: MOV    #140340,PSW ;MAKE PREVIOUS MODE KERNEL PRESENT USER
11340 047652 012767 100000 167504      MOV    #100000,$TMP2 ;LOAD TEST LOC. VIRT. ADDR. INTO $TMP2
11341 047660 012702 037364      MOV    #$TMP2,R2 ;LOAD ADDRESS OF $TMP2 INTO R2
11342 047664 006572 000000      MFPI   @0(R2) ;READ FROM PHYSICAL 140000
11343 047670 012601      MOV    (USP)+,R1 ;POP USER STACK INTO R1
11344 047672 020001      CMP    R0,R1 ;WAS DATA FETCHED SAME AS STORED
11345 047674 001401      BEQ    20$
11346 047676      21$: EMT ;WRONG DATA WAS FETCHED
11347 047676 104000      EMT ;FOR TIGHTER SCOPE LOOP
11348 ;REPLACE ERROR CALL WITH
11349 ;'BR 19$' = 000762
11350 ;
11351 047700 012767 021356 130342 20$: MOV    #T0250,MMVEC ;SET M.M. VECTOR TO NORMAL ROUTINE
11352 047706 012767 000340 130062      MOV    #00340,PSW ;GO BACK TO KERNEL MODE, PREVIOUS KERNEL
11353 ;
11354 ;*****
11355 ;TEST 416 MOVE FROM/TO D-SPACE = MOVE FROM/TO I-SPACE
11356 ;*****
11357 047714      TS416:
11358 047714 012767 030340 130054 1$: MOV    #030340,PSW ;MAKE PREVIOUS MODE=USER,CURRENT=KERNEL
11359 047722 106506      MFPD   USP ;MFPD SHOULD ACT LIKE MFPI PUTTING
11360 ;USER STACK POINTER ON THE KERNEL STACK
11361 047724 022706 001000      CMP    #KERSTK,KSP ;WAS SOMETHING PUSHED ON KERNEL STACK?
11362 047730 001405      BEQ    2$ ;BRANCH IF NO
11363 047732 012600      MOV    (KSP)+,R0 ;POP KERNEL STACK INTO R0
11364 047734 012701 000600      MOV    #USESTK,R1 ;EXPECTING TO GET 700 AS USP
11365 047740 020001      CMP    R0,R1 ;DID GET RIGHT POINTER VALUE?
11366 047742 001401      BEQ    4$
11367 047744      2$: EMT ;WRONG THING WAS PUSHED ON STACK
11368 047744 104000      EMT ;FOR TIGHTER SCOPE LOOP
11369 ;REPLACE ERROR CALL WITH
11370 ;'BR 1$' = 000763
11371
```

11372 047746 012746 007777
11373 047752 106606
11374 047754 106506
11375 047756 012601
11376 047760 022701 007777
11377 047764 001401
11378 047766 104000
11379
11380
11381
11382 047770 012746 000600
11383 047774 106606
11384
11385
11386
11387
11388 047776
11389 047776 005037 177776
11390 050002 012700 001000
11391 050006 010006
11392 050010 006506
11393
11394 050012 011601
11395 050014 020001
11396
11397 050016 001401
11398 050020 104000
11399
11400
11401
11402 050022 005740
11403 050024 020600
11404 050026 001401
11405 050030 104000
11406
11407
11408
11409 050032 012706 001000
11410 050036 005067 127530
11411 050042 000167 000472

4\$: MOV #7777,-(KSP) ;PUSH DATA ON KERNEL STACK
MTPD USP ;LOAD THE USER STACK POINTER
MFPD USP ;READ USER STACK POINTER
MOV (KSP),R1 ;POP KERNEL STACK INTO R1
CMP #7777,R1 ;WAS USER STACK POINTER CHANGED?
BEQ 5\$
EMT ;USER STACK POINTER NOT CHANGED
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;'BR 4\$' = 000767
5\$: MOV #USESTK,-(KSP) ;GET READY TO RESTORE USER STK. PTR.
MTPD USP ;RESTORE USER STACK POINTER
:*****
:TEST 417 MOVE FROM PREVIOUS I=SPACE (PREVIOUS=CURRENT=KERNEL)
:*****
TS417:
1\$: CLR @#PSW ;SET PREVIOUS = CURRENT = KERNEL
MOV #KERSTK,R0 ;SETUP VALUE FOR STACK POINTER
MOV R0,KSP ;LOAD STACK POINTER
MFPD KSP ;THE VALUE 'STACK' SHOULD BE PUSHED
;BEFORE BEING DECREMENTED
MOV (KSP),R1 ;READ DATA WHICH WAS PUSHED
CMP R0,R1 ;WAS THE ORIGINAL VALUE OF THE
;STACK POINTER PUSHED?
BEQ 2\$
EMT ;MFPD FETCHED WRONG DATA
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;'BR 1\$' = 000766
2\$: TST -(R0) ;SETUP EXPECTED STACK POINTER VALUE
CMP KSP,R0 ;WAS THE STACK POINTER DECREMENTED?
BEQ 3\$
EMT ;STACK NOT PUSHED BY THE MFPD
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;'BR 1\$' = 000762
3\$: MOV #KERSTK,KSP ;RESTORE STACK POINTER
CLR SRO ;TURN OFF MEMORY MANAGEMENT UNIT
JMP EXATST ;GET OVER SUBROUTINES TO EXTENDED ADRS TESTS

11412
11413
11414
11415
11416
11417
11418
11419
11420
11421
11422
11423 050046 036727 127724 000020
11424 050054 001411
11425 050056 016746 127714
11426 050062 011667 167270
11427
11428 050066 042710 000020
11429 050072 012746 050100
11430 050076 000006
11431 050100 000207
11432
11433
11434
11435
11436
11437
11438
11439
11440
11441 050102 036727 167250 000020
11442 050110 001410
11443 050112 016746 167240
11444 050116 012767 000340 167232
11445 050124 012746 050132
11446 050130 000006
11447 050132 000207
11448
11449
11450
11451
11452
11453
11454
11455
11456
11457
11458
11459
11460 050134 012702 000010
11461 050140 012701 172300
11462 050144 012721 177777
11463 050150 077203
11464 050152 012702 000010
11465 050156 012701 172340
11466 050162 012721 177777
11467 050166 077203

```
.SBTTL ***** SUBROUTINES USED BY THIS PROGRAM *****

.SBTTL TURN OFF T-BIT AND SAVE CURRENT PSW
:*****
:*
:* THIS SUBROUTINE IS USED TO TURN OFF THE TRACE TRAP BIT IN THE PSW
:* IF IT IS ON. THE PROCESSOR STATUS IS SAVED IN 'TBITPS' SO THAT
:* THE PSW CAN BE RESTORED TO ITS PREVIOUS CONDITION WHEN CONDITIONS
:* WARRANT T-BIT TRAPPING.
:*
:*****
TOFF: BIT PSW,#TBIT ;IS THE T-BIT SET IN THE PSW?
      BEQ 1$ ;EXIT IF NO
      MOV PSW,-(SP) ;PUSH PRESENT PSW ON THE STACK
      MOV (SP),TBITPS ;ALSO SAVE IT IN 'TBITPS' FOR
                        ;RESTORING LATER
      BIC #TBIT,(SP) ;CLEAR THE T-BIT (BIT 4) IN THE PSW
      MOV #1$,-(SP) ;PUSH PC OF 'RTS' ON STACK
      RTT ;'RETURN' TO 1$ WITH T-BIT OFF
1$: RTS PC ;RETURN TO PROGRAM

.SBTTL TURN ON T-BIT AND RESTORE PREVIOUS PSW
:*****
:*
:* THIS SUBROUTINE IS USED TO RESTORE THE PROCESSOR STATUS TO ITS
:* PREVIOUS CONDITION BY RESTORING THE 'T-BIT PSW' SAVED BY THE
:* 'TOFF' SUBROUTINE IN THE 'TBITPS' LOCATION.
:*
:*****
TON: BIT TBITPS,#TBIT ;WAS T-BIT ON IN THE PREVIOUS PSW?
      BEQ 1$ ;EXIT IF NO
      MOV TBITPS,-(SP) ;PUSH PREVIOUS PSW ON THE STACK
      MOV #340,TBITPS ;RESET THE 'TBITPS' LOCATION
      MOV #1$,-(SP) ;PUSH PC OF 'RTS' ON STACK
      RTT ;'RETURN' TO 1$ WITH T-BIT RESTORED
1$: RTS PC ;RETURN TO PROGRAM

.SBTTL SET ALL WRITEABLE BITS IN ALL PAR/PDR'S
:*****
:*
:* THIS SUBROUTINE IS USED BY THE PAR/PDR DUAL ADDRESSING TEST
:* TO SET ALL WRITEABLE BITS IN ALL KERNEL AND USE PAR'S AND
:* PDR'S TO A 1. THE 'INITIAL STATE' OF HAVING ALL BITS=1 IS
:* USED TO SEE THAT ONLY ONE REGISTER IS CLEARED IN RESPONSE TO
:* A SINGLE PAR OR PDR ADDRESS.
:*
:*****
SETREG: MOV #10,R2 ;LOAD LOOP COUNTER WITH AN 8
        MOV #KIPDRO,R1 ;LOAD ADDRESS OF FIRST PDR INTO R1
1$: MOV #-1,(R1)+ ;SET BITS IN KERNEL PDR TO 1
      SOB R2,1$ ;LOOP TO 1$ UNTIL ALL KERNEL PDR'S LOADED
        MOV #10,R2 ;LOAD LOOP COUNTER WITH AN 8
        MOV #KIPARO,R1 ;LOAD ADDRESS OF FIRST PAR INTO R1
2$: MOV #-1,(R1)+ ;SET BITS IN A KERNEL PAR TO 1
      SOB R2,2$ ;LOOP TO 2$ UNTIL ALL KERNEL PAR'S LOADED
```

11468 050170 012702 000010
11469 050174 012701 177600
11470 050200 012721 177777
11471 050204 077203
11472 050206 012702 000010
11473 050212 012701 177640
11474 050216 012721 177777
11475 050222 077203
11476 050224 000207
11477
11478
11479
11480
11481
11482
11483
11484
11485
11486
11487
11488 050226
11489 050226 012701 172300
11490 050232 012704 000010
11491 050236 012705 077416
11492 050242 021105
11493 050244 001403
11494 050246 020100
11495 050250 001401
11496 050252 104000
11497
11498
11499
11500 050254 062701 000002
11501 050260 077410
11502 050262 012701 172340
11503 050266 012704 000010
11504
11505 050272 012705 177777
11506 050276 021105
11507 050300 001403
11508 050302 020100
11509 050304 001401
11510 050306 104000
11511
11512
11513
11514 050310 062701 000002
11515 050314 077410
11516 050316 012701 177600
11517 050322 012704 000010
11518 050326 012705 077416
11519 050332 021105
11520 050334 001403
11521 050336 020100
11522 050340 001401
11523 050342 104000

```
MOV #10,R2 ;LOAD LOOP COUNTER WITH AN 8
MOV #UIPDRO,R1 ;LOAD ADDRESS OF FIRST PDR INTO R1
3$: MOV #-1,(R1)+ ;SET BITS IN A USER PDR TO 1
SOB R2,3$ ;LOOP TO 3$ UNTIL ALL USER PDR'S LOADED
MOV #10,R2 ;LOAD LOOP COUNTER WITH AN 8
MOV #UIPARO,R1 ;LOAD ADDRESS OF FIRST PAR INTO R1
4$: MOV #-1,(R1)+ ;SET BITS IN A USER PAR TO 1
SOB R2,4$ ;LOOP TO 4$ UNTIL ALL USER PAR'S LOADED
RTS PC ;RETURN TO TEST

.SBTTL READ & COMPARE KERNEL & USER PAR/PDR'S
:*****
:*
:* THIS SUBROUTINE IS USED BY PAR/PDR DUAL ADDRESSING TEST TO
:* READ ALL THE PAR'S AND PDR'S TO SEE THAT ONLY ONE REGISTER
:* WAS CLEARED IN RESPONSE TO A SINGLE PAR OR PDR ADDRESS.
:* ANY FAILURES FOUND BY THE PAR/PDR DUAL ADDRESSING TEST WILL
:* BE REPORTED BY THIS SUBROUTINE.
:*
:*****
CMPREG:
MOV #KIPDRO,R1 ;LOAD ADDRESS OF FIRST KERNEL PDR IN R1
MOV #10,R4 ;LOAD LOOP COUNTER WITH AN 8
MOV #77416,R5 ;PUT EXPECTED PDR CONTENTS IN R5
1$: CMP (R1),R5 ;ARE ALL WRITEABLE BITS SET AS EXPECTED?
BEQ 2$ ;BRANCH IF YES
CMP R1,R0 ;WAS IT THE REG. THAT WAS CLEARED?
BEQ 2$
EMT ;A PDR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PRO
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;AN 'RTS PC' = 000207
2$: ADD #2,R1 ;FORM NEXT ADDRESS
SOB R4,1$ ;LOOP TO 1$ UNTIL ALL KERNEL PDR'S CHECKED
MOV #KIPARO,R1 ;LOAD ADDRESS OF FIRST KERNEL PAR IN R1
MOV #10,R4 ;LOAD LOOP COUNTER WITH AN 8
;****F11 CHANGE**** FROM #7777 TO #177777
MOV #177777,R5 ;PUT EXPECTED PAR CONTENTS IN R5
3$: CMP (R1),R5 ;ARE ALL WRITEABLE BITS SET AS EXPECTED?
BEQ 4$ ;BRANCH IF YES
CMP R1,R0 ;WAS IT THE REG. THAT WAS CLEARED?
BEQ 4$
EMT ;A PAR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;AN 'RTS PC' = 000207
4$: ADD #2,R1 ;FORM NEXT ADDRESS
SOB R4,3$ ;LOOP TO 3$ UNTIL ALL KERNEL PAR'S CHECKED
MOV #UIPDRO,R1 ;LOAD ADDRESS OF FIRST USER PDR IN R1
MOV #10,R4 ;LOAD LOOP COUNTER WITH AN 8
MOV #77416,R5 ;PUT EXPECTED PDR CONTENTS IN R5
5$: CMP (R1),R5 ;ARE ALL WRITEABLE BITS SET AS EXPECTED?
BEQ 6$ ;BRANCH IF YES
CMP R1,R0 ;WAS IT THE REG. THAT WAS CLEARED?
BEQ 6$
EMT ;A PDR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
```

```
11524 ;FOR TIGHTER SCOPE LOOP
11525 ;REPLACE ERROR CALL WITH
11526 ;AN 'RTS PC' = 000207
11527 050344 062701 000002 6$: ADD #2,R1 ;FORM NEXT ADDRESS
11528 050350 077410 SOB R4,5$ ;LOOP TO 5$ UNTIL ALL USER PDR'S CHECKED
11529 050352 012701 177640 MOV #UIPAR0,R1 ;LOAD ADDRESS OF FIRST USER PAR IN R1
11530 050356 012704 000010 MOV #10,R4 ;LOAD LOOP COUNTER WITH AN 8
11531 ;****F11 CHANGE**** FROM #7777 TO #177777
11532 050362 012705 177777 7$: MOV #177777,R5 ;PUT EXPECTED PAR CONTENTS IN R5
11533 050366 021105 CMP (R1),R5 ;ARE ALL WRITEABLE BITS SET AS EXPECTED?
11534 050370 001403 BEQ 8$ ;BRANCH IF YES
11535 050372 020100 CMP R1,R0 ;WAS IT THE REG. THAT WAS CLEARED?
11536 050374 001401 BEQ 8$
11537 050376 104000 EMT ;A PAR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
11538 ;FOR TIGHTER SCOPE LOOP
11539 ;REPLACE ERROR CALL WITH
11540 ;AN 'RTS PC' = 000207
11541 050400 062701 000002 8$: ADD #2,R1 ;FORM NEXT ADDRESS
11542 050404 077410 SOB R4,7$ ;LOOP TO 7$ UNTIL ALL USER PAR'S CHECKED
11543 050406 000207 RTS PC ;RETURN TO TEST
11544
11545 .SBTTL INHIBIT 'RESETS' WHILE UNDER APT
11546 ;*****
11547 ;*
11548 ;* THIS SUBROUTINE CONTROLS THE USAGE OF RESET INST'S WHILE
11549 ;* RUNNING UNDER APT. RESETS ARE ALLOWED DURING THE FIRST
11550 ;* PASS OF THE DIAGNOSTIC.
11551 ;*
11552 050410 126727 130404 000001 CHKAPT: CMPB $ENV,#1 ;ARE WE RUNNING UNDER APT?
11553 050416 001003 BNE 1$ ;NO BRANCH
11554 050420 005767 130362 TST $PASS ;IS THIS THE FIRST PASS?
11555 050424 001002 BNE RETA ;NO BRANCH
11556 050426 062705 000002 1$: ADD #2,R5 ;BUMP RETURN ADDRESS FOR NORMAL TESTING
11557 050432 000205 RETA: RTS R5 ;RETURN
11558
11559
11560 .SBTTL ERROR ROUTINE FOR MEMORY MANAGEMENT TEST
11561 ;*****
11562 ;* THIS IS THE ONLY ERROR REPORT FOR ALL THE MMU TESTS
11563 ;*****
11564
11565 050434 012737 000002 001002 ERROR3: MOV #2,$FATAL ;SET UP FATAL ERROR NUMBER
11566 050442 012767 000001 130330 MOV #1,$MSGTY ;SET FATAL ERROR FLAG
11567 050450 032737 000001 001020 BIT #1,$ENV ;UNDER APT ?
11568 050456 001004 BNE MMUHLT
11569 050460 012700 050472 MOV #MMUMSG,R0
11570 050464 004767 062640 JSR PC,TYPE
11571 050470 000777 MMUHLT: BR . ;STAY IN LOOP
11572
11573 050472 040506 046111 042105 MMUMSG: .ASCIZ /FAILED DURING MMU TESTING/<12><15>
11574 050500 042040 051125 047111
11575 050506 020107 046515 020125
11576 050514 042524 052123 047111
11577 050522 005107 000015
11578
11579 .EVEN
```

11580
 11581
 11582
 11583
 11584
 11585
 11586
 11587
 11588 050526 000000
 11589 050530 000000
 11590 050532 000000
 11591 050534 000002
 11592
 11593
 11594
 11595 050540
 11596
 11597
 11598
 11599
 11600
 11601
 11602
 11603 050540 012767 177600 121610
 11604 050546 012737 000007 001004
 11605 050554 012737 051332 000030
 11606 050562 000244
 11607 050564 032777 000200 150612
 11608 050572 001001
 11609 050574 000470
 11610
 11611 050576 016767 127202 177722
 11612 050604 016767 127176 177716
 11613 050612 012767 050714 127164
 11614 050620 012767 000340 127160
 11615 050626 012767 004000 121514
 11616 050634 012767 000001 126730
 11617 050642 012767 000020 121646
 11618
 11619 050650 012767 000000 177656
 11620 050656 026727 177652 000005
 11621 050664 001420
 11622 050666 013737 100000 100000
 11623 050674 000240
 11624 050676 012767 000000 126666
 11625 050704 012767 000000 121604
 11626 050712 104000
 11627 050714 006367 121430
 11628 050720 005267 177610
 11629 050724 000754
 11630 050726 016767 177574 127050
 11631 050734 016767 177570 127044
 11632 050742 012767 000000 126622
 11633 050750 012767 000000 121540
 11634
 11635

```

*****
STORAGE AREAS FOR THE FOLLOWING FEW TESTS
*****
TMP:      .WORD  0
TMP1:     .WORD  0
MEM:      .WORD  0
CNTR:     .BLKW  2
*****
:TEST 420      TEST ADDRESS BITS 17-21
*****
TS420:
*****
THIS TEST WILL DETERMINE WHETHER THE KDF11-B CAN
DRIVE ADDRESS BITS 17-21.
*****
EXATST:  MOV      #177600,KIPAR7      ;I 0 PAGE
          MOV      #7,@#STESTN
          MOV      #ERRORA,@#30
          CLZ
          BIT      #200,@SWR          ;CLR THE Z BIT
          BNE     ESR3                ;WANT TO TEST THIS ?
          BR      NEXT
ESR3:    MOV      ERRVEC,TMP
          MOV      ERRVEC+2,TMP1
          MOV      #ESRO,ERRVEC      ;PREPARE FOR NEW
          MOV      #340,ERRVEC+2    ;INTERRUPT
          MOV      #4000,KIPAR4     ;SET FOR ADRS BIT 17
          MOV      #BIT0,SRO
          MOV      #BIT4,SR3        ;TURN 22 BIT ADRSG AND
                                     ;MEMORY MANG ON
ESR1:    MOV      #0,CNTR
          CMP      CNTR,#5
          BEQ     ESR2
          MOV      @#100000,@#100000 ;TRY ADRSG IT
          NOP
          MOV      #0,SRO
          MOV      #0,SR3          ;TURN OFF MM AND 22 BIT
          EMT                       ;ERROR
ESR0:    ASL      KIPAR4            ;SHOULD TIME OUT
          INC      CNTR             ;SHIFT 1 TO THE LEFT
          BR      ESR1
ESR2:    MOV      TMP,ERRVEC
          MOV      TMP1,ERRVEC+2    ;RESTORE
          MOV      #0,SRO
          MOV      #0,SR3          ;TURN OFF MM AND 22 BIT
  
```


11636
11637
11638
11639 050756
11640
11641
11642
11643
11644
11645
11646
11647 050756 012767 000001 126606
11648 050764 016767 127014 177534
11649 050772 016767 127010 177 30
11650 051000 012767 051062 126776
11651 051006 012767 000340 126772
11652 051014 012737 123456 000000
11653 051022 012767 002000 121320
11654 051030 013737 100000 100000
11655 051036 023727 000000 123456
11656 051044 001407
11657 051046 012767 000000 126516
11658 051054 012767 000000 121434
11659 051062 104000
11660 051064 005067 126502
11661 051070 016767 177432 126706
11662 051076 016767 177426 126702
11663
11664
11665
11666
11667
11668
11669 051104
11670
11671
11672
11673
11674
11675
11676
11677
11678
11679 051104 000244
11680 051106 032777 001000 150270
11681 051114 001002
11682 051116 000167 000430
11683
11684 051122 012767 000001 000370
11685 051130 016767 126650 177376
11686 051136 016767 126644 177372
11687 051144 012767 051162 126632
11688 051152 012704 172100
11689 051156 005714
11690 051160 000407
11691 051162 022704 172136

```
*****  
:TEST 421 TEST ADDRESS BIT 16  
*****  
TS421:  
*****  
: THIS TEST WILL CHECK TO SEE IF THE 17TH BIT (16) OF AN 18 BIT  
: ADDRESS CAN BE DRIVEN.  
*****  
NEXT: MOV #BIT0,SRO ;TURN ON MEM MANG  
MOV ERRVEC,TMP  
MOV ERRVEC+2,TMP1  
MOV #ESR4,ERRVEC  
MOV #340,ERRVEC+2  
MOV #123456,@#0 ;MODIFY LOCATION ZERO  
MOV #2000,KIPAR4  
MOV @#100000,@#100000  
CMP @#0,#123456 ;CHANGED ?  
BEQ ESR5  
MOV #0,SRO  
MOV #0,SR3 ;TURN OFF MM AND 22 BIT  
ESR4: EMT ;LOC 0 WAS ALTERED  
ESR5: CLR SRO ;TURN OFF MM  
MOV TMP,ERRVEC  
MOV TMP1,ERRVEC+2 ;RESTORE VECTORS
```

```
*****  
:TEST 422 TEST PARITY ERROR DETECTION LOGIC  
*****  
TS422:  
*****  
: THIS TEST WILL USE THE MEMORY PARITY CSR TO GENERATE  
: A PARITY ERROR THAT THE CPU SHOULD DETECT.  
*****  
CLZ ;CLR THE Z BIT  
BIT #BIT9,@SWR ;TEST THIS ?  
BNE ESR57  
JMP Q22TST ;NO  
ESR57: MOV #1,ERRORE ;DUMMY BIT  
MOV ERRVEC,CNTR  
MOV ERRVEC+2,CNTR+2 ;SAVE VECTORS  
MOV #2$,ERRVEC ;NEW TIME OUT VECOTR  
MOV #172100,R4 ;PLACE TO START  
1$: TST (R4) ;IS IT THERE  
BR ESR7  
2$: CMP #172136,R4 ;TOP YET ?
```

```

11692 051166 100403          BMI      ESR6
11693 051170 062704 000002    ADD      #2,R4
11694 051174 000770          BR       1$
11695 051176 104000          ESR6:   EMT
11696 051200 016767 126710 177320 ESR7:   MOV      114,TMP          ;COULD NOT FIND MEM PAR CSR
11697 051206 012767 051254 126700 ESR7:   MOV      #ESR8,114      ;SAVE CONTENTS OF 114
11698 051214 012714 000005    MOV      #5,(R4)          ;PARITY TRAP VECTOR
11699 051220 010167 177306    MOV      R1,MEM          ;BITS 0,2 IN MEM CSR
11700 051224 042714 000004    BIC      #4,(R4)          ;TURN OFF WRT WRG PAR BIT
11701 051230 016701 177276    MOV      MEM,R1          ;SOME DATA TO MEMORY FROM
11702 051234 012714 000000    ESR9:   MOV      #0,(R4)    ;TURN OFF P ERR
11703 051240 010167 177266    MOV      R1,MEM
11704 051244 016767 177256 126642 MOV      TMP,114          ;RESTORE VECTOR CONTENTS
11705                                ;CPU, SHOULD SET A PAR ERR
11706 051252 104000          EMT
11707 051254 032714 100000    ESR8:   BIT      #BIT15,(R4) ;SEE IF BIT 15 SET IN MEM CSR
11708 051260 001765          BEQ      ESR9            ;Z IS SET IF AN ERROR
11709 051262 005000          CLR      R0
11710 051264 005200    ESR19: INC      R0          ;TIME FOR LED TO BE SEEN
11711 051266 005700          TST      R0
11712 051270 100375          BPL      ESR19
11713 051272 012714 000000    MOV      #0,(R4)
11714 051276 010167 177230    MOV      R1,MEM
11715 051302 016767 177220 126604 MOV      TMP,114
11716 051310 016767 177220 126466 MOV      CNTR,ERRVEC
11717 051316 016767 177214 126462 MOV      CNTR+2,ERRVEC+2
11718 051324 005067 000170    CLR      ERRORE          ;RESTORE DUMMY BIT
11719 051330 000510          BR       Q22TST         ;GO FIND A Q22BE
11720                                ;*****
11721                                ;*****
11722 051332 012737 000007 001002 ERRORA: MOV      #7,#$FATAL
11723 051340 012767 000001 127432 ERRORA: MOV      #1,$MSGTY
11724 051346 032737 000001 001020 ERRORA: BIT      #1,#$SENV          ;UNDER APT ?
11725 051354 001013          BNE      EXADHT
11726 051356 026727 000136 000001 ERRORA: CMP      ERRORE,#1          ;DUMMY BIT THERE ?
11727 051364 001003          BNE      ESR34
11728 051366 012700 051451          MOV      #CSRMSG,R0
11729 051372 000402          BR       ESR34+4
11730 051374 012700 051406    ESR34: MOV      #EXTMSG,R0
11731 051400 004767 061724    ESR34: JSR      PC,TYPE
11732 051404 000777          EXADHT: BR       .
11733
11734
11735 051406 040506 046111 042105 EXTMSG: .ASCIZ  /FAILED DURING EXTENDED ADRS TEST/<12><15>
11736 051414 042040 051125 047111
11737 051422 020107 054105 042524
11738 051430 042116 042105 040440
11739 051436 051104 020123 042524
11740 0: 444 052123 006412 000
11741
11742 051451 106 044501 042514 CSRMSG: .ASCIZ  /FAILED MEM PARITY ERROR DETECT TEST/<12><15>
11743 051456 020104 042515 020115
11744 051464 040520 044522 054524
11745 051472 042440 051122 051117
11746 051500 042040 052105 041505
11747 051506 020124 042524 052123

```

11748 051514 006412 000
11749
11750 051520 051520
11751 051520 000000
11752
11753
11754 051522 000510
11755 051524 000000
11756 051526 170000
11757 051530 000000
11758 051532 000000
11759 051534 000000
11760 051536 000000
11761 051540 000000
11762 051542 000000
11763 051544 000000
11764 051546 000000
11765 051550 000000
11766
11767
11768
11769
11770
11771 051552
11772
11773
11774
11775
11776
11777
11778
11779
11780
11781
11782 051552 000244
11783 051554 032777 000100 147622
11784 051562 001002
11785 051564 000167 001476
11786 051570 012737 000011 001004
11787 051576 012737 053160 000030
11788
11789
11790 051604 005067 125762
11791 051610 016767 126170 176714
11792 051616 012767 052006 126160
11793 051624 016767 177676 177676
11794 051632 016767 177664 177664
11795 051640 005777 177664
11796
11797 051644 016767 177660 177660
11798 051652 016767 177654 177654
11799 051660 062767 000002 177646
11800 051666 016767 177640 177642
11801 051674 062767 000004 177634
11802 051702 016767 177624 177630
11803 051710 062767 000006 177622

```
.EVEN  
ERRORE: .WORD 0 ;DUMMY BIT  
:*****  
VECT1: .WORD 510 ;FIRST DEV VECTOR Q22BE  
DEVECT: .WORD 0  
DEV1: .WORD 170000 ;FIRST DEV ADRS Q22BE  
DEVADR: .WORD 0  
CSR1: .WORD 0  
CSR2: .WORD 0  
BA: .WORD 0  
WC: .WORD 0  
DATA: .WORD 0  
LATCNT: .WORD 0  
MVL CNT: .WORD 0  
SIMGOA: .WORD 0
```

```
:*****  
:TEST 423 SEE IF A Q22BE(QBE) IS THERE  
:*****  
TS423:
```

```
:*****  
:ROUTINE TO SIZE FOR THE Q22BE(QBE) DEVICE ADDRESS  
:WE WILL LOOK FOR A Q22BE(QBE). IF IT ISN'T THERE  
:WE WILL CAUSE AN ERROR VIA THE EMT INSTRUCTION.  
:*****
```

```
Q22TST: CLZ  
BIT #BIT6,@SWR  
BNE 1$  
JMP BDVTST  
1$: MOV #11,@#$TESTN ;TEST NUM IN MAILBOX  
MOV #ERRORB,@#30 ;SET UP FOR CORRECT EMT  
  
CLR SRO  
MOV ERRVEC, MEM ;STORE ERRVEC CONTENTS  
MOV #ESR99,ERRVEC  
MOV DEV1,DEVADR  
MOV VECT1,DEVECT  
ESR11: TST @DEVADR ;SEE IF IT RESPONDS  
  
MOV DEVADR,CSR1  
MOV CSR1,CSR2  
ADD #2,CSR2 ;YES IT DID  
MOV CSR1,BA  
ADD #4,BA  
MOV CSR1,WC  
ADD #6,WC
```

11804 051716 016767 177610 177616
11805 051724 062767 000010 177610
11806 051732 016767 177574 177604
11807 051740 062767 000012 177576
11808 051746 016767 177560 177572
11809 051754 062767 000014 177564
11810 051762 016767 177544 177560
11811 051770 062767 000016 177552
11812 051776 016767 176530 126000
11813 052004 000413
11814 052006 062767 000020 177514
11815 052014 062767 000004 177502
11816 052022 026727 177476 000550
11817 052030 001303
11818 052032 104000
11819 052034 016700 177464
11820 052040 062700 000002
11821 052044 012710 000340
11822
11823
11824
11825
11826
11827 052050
11828
11829
11830
11831
11832
11833
11834
11835
11836
11837
11838
11839 052050 012777 052114 177446
11840 052056 012777 000001 177446
11841 052064 106427 000140
11842 052070 012777 000003 177436
11843 052076 000240
11844 052100 012777 000002 177426
11845 052106 000240
11846 052110 000240
11847 052112 104000
11848 052114 012777 052150 177402
11849 052122 106427 000200
11850 052126 012777 000003 177400
11851 052134 000240
11852 052136 012777 000000 177370
11853 052144 000240
11854 052146 000401
11855 052150 104000
11856
11857
11858
11859

```
MOV CSR1,DATA  
ADD #10,DATA  
MOV CSR1,LATCNT  
ADD #12,LATCNT  
MOV CSR1,MVLCNT  
ADD #14,MVLCNT  
MOV CSR1,SIMGOA  
ADD #16,SIMGOA  
MOV MEM,ERRVEC ;RESTORE ERROR VECTOR  
BR ESR98  
ESR99: ADD #20,DEVADR  
ADD #4,DEVECT  
CMP DEVECT,#550  
BNE ESR11 ;GO TRY ANOTHER ADRS  
ESR98: MOV DEVECT,RO  
ADD #2,RO  
MOV #340,(RO)
```

```
*****  
:TEST 424 USE Q22BE(QBE) TO ALTER THE INTERRUPT LEVEL BITS  
:*****  
TS424:
```

```
*****  
:  
: GENERATE A Q22BE(QBE) SOFTWARE INTR AT LEVEL 4.  
: THE FOLLOWING CODE WILL USE THE Q22(QBE) BUS EXERCISER  
: TO GENERATE INTERRUPTS THAT THE CPU SHOULD EITHER  
: HONOR OR IGNORE DEPENDING ON THE INTR LEVEL BITS.  
:*****
```

```
MOV #ESR21,@DEVECT ;INTERRUPT TO ESR21  
MOV #1,@CSR1  
MTPS #140 ;CPU AT LEVEL THREE  
MOV #3,@CSR2  
NOP  
MOV #2,@CSR2  
NOP  
NOP  
EMT  
ESR21: MOV #ESR22,@DEVECT ;CHANGE INTR VECTOR  
MTPS #200  
MOV #3,@CSR2  
NOP  
MOV #0,@CSR2 ;INTR LEVEL IS 4  
NOP ;INTR SHOULD NOT HONORED  
BR ESR23  
ESR22: EMT ;INTR HONORED A ERROR  
:*****  
: GENERATE AN INTERRUPT AT LEVEL FIVE  
:*****
```

```
11860 052152 012777 052214 177344 ESR23: MOV #ESR24,@DEVECT
11861 052160 106427 000200 MTPS #200 ;CPU AT LEVEL FOUR
11862 052164 012777 000001 177340 MOV #1,@CSR1 ;TRY TO CAUSE AN INTERRUPT
11863 052172 012777 000007 177334 MOV #7,@CSR2 ;AT LEVEL FIVE
11864 052200 000240 NOP
11865 052202 012777 000006 177324 MOV #6,@CSR2 ;CLRS GO SETS DONE
11866 052210 000240 NOP
11867 052212 104000 EMT ;INTR DID NOT HAPPEN
11868 052214 012777 052250 177302 ESR24: MOV #ESR27,@DEVECT ;ALTER INTR VECTOR
11869 052222 106427 000240 MTPS #240 ;CHANGE LEVEL TO FIVE
11870 052226 012777 000007 177300 MOV #7,@CSR2
11871 052234 000240 NOP
11872 052236 012777 000000 177270 MOV #0,@CSR2
11873 052244 000240 NOP
11874 052246 000401 BR ESR28
11875 052250 104000 ESR27: EMT ;IF HERE, AN ERROR
11876 :*****
11877 : TRY INTERRUPT AT LEVEL SIX
11878 :*****
11879 052252 012777 052310 177244 ESR28: MOV #ESR29,@DEVECT
11880 052260 012777 000001 177244 MOV #1,@CSR1 ;INTR RQST BITS TO 6
11881 052266 012777 000013 177240 MOV #13,@CSR2
11882 052274 000240 NOP
11883 052276 012777 000012 177230 MOV #12,@CSR2 ;INTR SHOULD BE HONORED
11884 052304 000240 NOP
11885 052306 104000 EMT ;OTHERWISE AN ERROR
11886 052310 012777 052344 177206 ESR29: MOV #ESR31,@DEVECT ;ALTER CPU INTR VECTOR
11887 052316 106427 000300 MTPS #300
11888 052322 012777 000013 177204 MOV #13,@CSR2 ;SETS GO, CLRS DONE
11889 052330 000240 NOP
11890 052332 012777 000000 177174 MOV #0,@CSR2
11891 052340 000240 NOP
11892 052342 000401 BR ESR30
11893 052344 104000 ESR31: EMT ;INTR SHOULD NOT BE HONORED
11894 : ;BUT IF HERE, IT WAS
11895 :*****
11896 : GENERATE AN INTERRUPT AT LEVEL SEVEN
11897 :*****
11898 052346 012777 052404 177150 ESR30: MOV #ESR33,@DEVECT
11899 052354 012777 000001 177150 MOV #1,@CSR1 ;CPU AT LEVEL 6, Q22 AT 7
11900 052362 012777 000033 177144 MOV #33,@CSR2
11901 052370 000240 NOP
11902 052372 012777 000032 177134 MOV #32,@CSR2
11903 052400 000240 NOP
11904 052402 104000 EMT
11905 052404 012777 052460 177112 ESR33: MOV #ESR35,@DEVECT ;MODIFY INTR VECTOR IN CPU
11906 052412 106427 000340 MTPS #340
11907 052416 012777 000033 177110 MOV #33,@CSR2
11908 052424 000240 NOP
11909 052426 012777 000032 177100 MOV #32,@CSR2
11910 052434 000240 NOP
11911 052436 012777 000001 177070 MOV #1,@CSR2 ;THIS WILL REMOVE THE
11912 052444 012777 000000 177062 MOV #0,@CSR2 ;PENDING INTERRUPT
11913 052452 106427 000004 MTPS #4 ;RESTORE THE PSW
11914 052456 000401 BR Q22TS1 ;NADA SHOULD INTERRUPT
11915 052460 104000 ESR35: EMT ;ERROR IF IT DOES
```

11916
11917
11918
11919
11920 052462
11921
11922
11923
11924
11925
11926
11927
11928 052462 012777 065432 177052
11929 052470 012777 001601 177034
11930 052476 005077 177032
11931 052502 012777 177776 177030
11932 052510 012777 050534 177020
11933 052516 005067 176012
11934 052522 005067 176010
11935 052526 012777 000001 177014
11936 052534 032777 000200 176772
11937 052542 001001
11938 052544 000773
11939 052546 022767 065432 175760
11940 052554 001004
11941 052556 022767 065432 175752
11942 052564 001401
11943 052566 104000
11944
11945
11946
11947
11948
11949
11950 052570
11951
11952
11953
11954
11955
11956
11957
11958
11959
11960 052570 000244
11961 052572 032777 000400 146604
11962 052600 001002
11963 052602 000167 000460
11964
11965 052606 016767 125212 175712
11966 052614 016767 125206 175706
11967 052622 012767 052674 125174
11968 052630 012767 000340 125170
11969 052636 010667 175672
11970 052642 012706 000420
11971 052646 012767 052706 125130

:TEST 425 DMA DATO TRANSFER
:*****
TS425:

: THIS TEST WILL SET UP AN ACTUAL DMA TRANSFER
:*****

Q22TS1: MOV #65432,@DATA
MOV #1601,@CSR1
CLR @CSR2 ;NO INTERRUPT BITS
MOV #177776,@WC ;TWO WORDS
MOV #CNTR,@BA
CLR CNTR
CLR CNTR+2 ;CLEAR THESE FIRST
MOV #1,@SIMGOA ;SIMULT. GO BIT
ESR26: BIT #BIT7,@CSR2 ;WAIT FOR DONE
BNE ESR25
BR ESR26
ESR25: CMP #65432,CNTR
BNE ESR32
CMP #65432,CNTR+2 ;GOOD DATA
BEQ Q22TS2
ESR32: EMT ;NO, BAD DATA

:TEST 426 TEST PWR OK LOGIC
:*****
TS426:

: THIS TEST WILL DETERMINE IF THE CPU WILL RESPOND
: TO THE BPOK LINE BEING ALTERED. THE Q22BE
: WILL BE USED TO CONTROL THAT LINE ON THE Q BUS.
:*****

Q22TS2: CLZ
BIT #BIT8,@SWR ;IS A Q BUS EXCER THERE
BNE ESR20 ;NO, IT'S A Q22
JMP BDVTST ;YES, THEN BYPASS THESE TESTS
ESR20: MOV 24,TMP
MOV 26,TMP1 ;SAVE FOR LATER
MOV #ESR12,24
MOV #340,26 ;SET UP NEW VECTOR
MOV SP,CNTR ;SAVE STACK POINTER
MOV #420,SP ;MOVE SP WAY DOWN
MOV #ESR13,4 ;WHERE A STK OVFL WLD GO

```
11972 052654 012767 000340 125124      MOV    #340,6
11973 052662 012777 000040 176644      MOV    #40,@CSR2      ;SET BIT 5 IN Q22BE CSR2
11974 052670 000240                NOP                    ;WILL PULL BPOK H LOW
11975 052672 104000                EMT                    ;DIDNT CAUSE CPU TO GO
11976 052674 012777 000000 176632 ESR12: MOV    #0,@CSR2    ;THRU LOC 24
11977 052702 000240                NOP                    ;BPOK H WILL GO HIGH
11978 052704 000401                BR     ESR14
11979 052706 104000                ESR13: EMT            ;IF HERE, AN ERROR BY GOING
11980                                ;THRU LOC 4
11981 052710 016706 175620 ESR14: MOV    CNTR,SP    ;RESTORE SOME STUFF
11982 052714 016767 175606 125102      MOV    TMP,24
11983 052722 016767 175602 125076      MOV    TMP1,26
11984 052730 000167 000000                JMP    Q22TS3        ;GO TO NEXT TEST
```

```
*****
:TEST 427 TEST INDIVIDUAL EXTENDED ADRS BITS
*****
TS427:
```

```
*****
:
: THIS TEST WILL UTILIZE THE Q22BE LATENCY CNTR
: TO CAPTURE THE EXTENDED ADDRESS BITS ON THE Q BUS.
: THE Q22BE LATENCY COUNTER BITS 15-12 CORRESPOND
: TO ADDRESS BITS 21-18.
*****
```

```
12001
12002 052734 016767 125044 175564 Q22TS3: MOV    4,TMP
12003 052742 016767 125040 175560      MOV    6,TMP1        ;STORE FOR LATER
12004 052750 012767 053046 125026      MOV    #ESR15,4
12005 052756 012767 000340 125022      MOV    #340,6        ;NEW VECTOR AND PSW
12006
12007 052764 017700 176554                MOV    @LATCNT,RO    ;READ IT TO CLR IT
12008 052770 012767 000020 117520      MOV    #BIT4,SR3     ; 22 BIT ADRSNG
12009 052776 012767 010000 117344      MOV    #10000,KIPAR4 ;SET FOR ADRS BIT 18
12010 053004 012767 010000 175524      MOV    #10000,CNTR+2
12011 053012 012767 000000 175514      MOV    #0,CNTR       ;ZERO THESE
12012 053020 012767 000001 124544      MOV    #BIT0,SRO    ;TURN ON MEM MANG
12013 053026 026727 175502 000017 ESR17: CMP    CNTR,#17   ;FINISHED ?
12014 053034 001433                BEQ    ESR16
12015 053036 105037 100000                CLRB   @#100000     ;IF THERE IS NO MEMORY HERE
12016 053042 000240                NOP                    ;SHOULD HAVE TIMED OUT
12017 053044 000240                NOP                    ;WE ARE IN BIG TROUBLE
12018
12019 053046 017700 176472 ESR15: MOV    @LATCNT,RO    ;READ LATENCY COUNTER
12020 053052 042700 007777      BIC    #7777,RO     ;CLR DONT CARES
12021 053056 026700 175454      CMP    CNTR+2,RO    ;EXPECTED ?
12022 053062 001407      BEQ    ESR18        ;EQUALS RECVD
12023 053064 012767 000000 124500      MOV    #0,SRO       ;TURN OFF MM AND 22 BITS
12024 053072 012767 000000 117416      MOV    #0,SR3
12025 053100 104000                EMT
12026 053102 005267 175426 ESR18: INC    CNTR
12027 053106 062767 010000 117234      ADD    #10000,KIPAR4 ;DATA NOT GOOD
```

12028 053114 062767 010000 175414
12029 053122 000741
12030
12031 053124 016767 175376 124652
12032 053132 016767 175372 124646
12033 053140 012767 000000 124424
12034 053146 012767 000000 117342
12035 053154 000167 000106
12036
12037
12038
12039
12040 053160 012737 000011 001002
12041 053166 012767 000001 125604
12042 053174 032737 000001 001020
12043 053202 001004
12044 053204 012700 053216
12045 053210 004767 060114
12046 053214 000777
12047 053216 040506 046111 051125
12048 053224 020195 052504 044522
12049 053232 043516 050440 041040
12050 053240 051525 042440 042530
12051 053246 041522 051511 051105
12052 053254 052040 051505 051524
12053 053262 006412 000
12054 053266
12055
12056
12057
12058
12059
12060
12061
12062
12063
12064
12065
12066
12067
12068
12069
12070 053266
12071
12072
12073
12074 053266 012767 000017 124230
12075 053274 012737 000012 001004
12076 053302 012737 054532 000030
12077 053310 000244
12078 053312 032777 002000 146064
12079 053320 001410
12080 053322 016767 124176 175176
12081 053330 026727 175172 000113
12082 053336 001401
12083 053340 104000

ADD #10000,CNTR+2 ;INCREASE ADRS BITS BY ONE
BR ESR17 ;CONTINUE
ESR16: MOV TMP,4
MOV TMP,6 ;RESTORE THIS VECTOR
MOV #0,SR0 ;TURN OFF MM AND 22 BITS
MOV #0,SR3
JMP BDVTST

ERRORB: MOV #11,@#SFATAL
MOV #1,\$MSGTY
BIT #1,@#SENV
BNE Q22HLT
MOV #Q22MSG,R0
JSR PC,TYPE
Q22HLT: BR
Q22MSG: .ASCIZ /FAILURE DURING Q BUS EXERCISER TESTS/<12><15>

.EVEN

BDV TESTS

:TEST 430 TEST FOR FUNCTIONALITY OF R/W REGISTER

TS430:

BDVTST: MOV #17,LSREG ;TURN OFF THE 4 LEDS
MOV #12,@#STESTN ;TST NUMBER FOR APT
MOV #ERRORD,@#30 ;ERROR TRAP
CLZ
BIT #BIT10,@SWR ;WANT TO TEST E102 ?
BEQ ESR10 ;NO
MOV LSREG,TMP ;RD THIS TO GET SWITCHES
CMP TMP,#113 ;SHOULD BE THIS VALUE
BEQ ESR10 ;EQUAL
EMT

12084	053342	012737	054436	000030	ESR10:	MOV	#ERRORC,@#30		:ERROR TRAP
12085	053350	005067	124146			CLR	RWREG		:THE NEXT FEW INSTRUCTIONS
12086	053354	016701	124142			MOV	RWREG,R1		:WILL TEST THE R/W ABILITY
12087	053360	020127	000000			CMP	R1,#0		:OF THE R/W MAINT. REGISTER
12088	053364	001401				BEQ	ESR40		
12089	053366	104000				EMT			
12090	053370	012767	177777	124124	ESR40:	MOV	#-1,RWREG		
12091	053376	016701	124120			MOV	RWREG,R1		
12092	053402	022701	177777			CMP	#177777,R1		
12093	053406	001401				BEQ	ESR41		
12094	053410	104000				EMT			
12095	053412	012767	125252	124102	ESR41:	MOV	#125252,RWREG		
12096	053420	016701	124076			MOV	RWREG,R1		
12097	053424	020127	125252			CMP	R1,#125252		
12098	053430	001401				BEQ	ESR42		
12099	053432	104000				EMT			
12100	053434	105067	124062		ESR42:	CLRB	RWREG		:BYTES
12101	053440	016701	124056			MOV	RWREG,R1		
12102	053444	020127	125000			CMP	R1,#125000		
12103	053450	001401				BEQ	ESR43		
12104	053452	104000				EMT			
12105	053454	000367	124042		ESR43:	SWAB	RWREG		
12106	053460	016701	124036			MOV	RWREG,R1		
12107	053464	020127	000252			CMP	R1,#252		
12108	053470	001401				BEQ	ESR44		
12109	053472	104000				EMT			
12110	053474	012767	052525	124020	ESR44:	MOV	#52525,RWREG		
12111	053502	016701	124014			MOV	RWREG,R1		
12112	053506	020127	052525			CMP	R1,#52525		
12113	053512	001401				BEQ	ESR45		
12114	053514	104000				EMT			
12115	053516				ESR45:				
12116	053516	105067	124001			CLRB	RWREG+1		
12117	053522	016701	123774			MOV	RWREG,R1		
12118	053526	020127	000125			CMP	R1,#125		
12119	053532	001401				BEQ	ESR46		
12120	053534	104000				EMT			
12121	053536	000367	123760		ESR46:	SWAB	RWREG		
12122	053542	016701	123754			MOV	RWREG,R1		
12123	053546	020127	052400			CMP	R1,#52400		
12124	053552	001401				BEQ	ESR47		
12125	053554	104000				EMT			
12126	053556	005067	123740		ESR47:	CLR	RWREG		
12127	053562	052767	100000	123732		BIS	#BIT15,RWREG		:SET 15
12128	053570	016701	123726			MOV	RWREG,R1		
12129	053574	020167	123722		ROTLP1:	CMP	R1,RWREG		:ARE THEY THE SAME
12130	053600	001401				BEQ	ESR48		
12131	053602	104000				EMT			
12132	053604	006001			ESR48:	ROR	R1		:NEXT BIT TO THE RIGHT
12133	053606	001403				BEQ	ESR49		
12134	053610	006067	123706			ROR	RWREG		
12135	053614	000767				BR	ROTLP1		:CONTINUE TESTING
12136	053616	012767	177777	123676	ESR49:	MOV	#-1,RWREG		
12137	053624	042767	100000	123670		BIC	#BIT15,RWREG		
12138	053632	016701	123664			MOV	RWREG,R1		
12139	053636	026701	123660		ROTLP2:	CMP	RWREG,R1		:SAME ?

12140 053642 001401
12141 053644 104000
12142 053646 000261
12143 053650 006067 123646
12144 053654 006001
12145 053656 020127 077777
12146 053662 001365
12147 053664 000461

BEQ ESR50 ;YES
EMT
ESR50: SEC ;SET THE C BIT
ROR RWREG
ROR R1
CMP R1,#77777
BNE ROTLP2 ;NOT FINISHED YET
BR BDVTS2

12148
12149
12150
12151
12152
12153 053666

:TEST 431 ROM CHECKSUM TEST

TS431:

12154
12155
12156
12157
12158
12159
12160
12161
12162
12163
12164

:
: THE PREVIOUS TEST CHECKED OUT THE ROM R/W REGISTER
: NOW WE WILL TEST THE 2K DIAGNOSTIC ROM FOR THE
: CORRECT CHECKSUM AND CHECKWORD
:*****

12165 053666
12166 053666 017042
12167 053670 020656
12168 053672 065162
12169 053674 161744
12170 053676 124453
12171 053700 113667
12172 053702 056040
12173 053704 044734

:THE CHECKWORDS CORRESPONDING TO ROM CHIPS #23-045E2 AND #23-046E2 FOLLOW:
SFPTBL:

.WORD 17042 ;ROMA: PAGE 0,1
.WORD 20656 ;ROMB: PAGE 2,3
.WORD 65162 ;ROMC: PAGE 4,5
.WORD 161744 ;ROMD: PAGE 6,7
.WORD 124453 ;ROME: PAGE 10,11
.WORD 113667 ;ROMF: PAGE 12,13
.WORD 56040 ;ROMG: PAGE 14,15
.WORD 44734 ;ROMH: PAGE 16,17

12174
12175
12176 053706 166020
12177 053710 020232
12178 053712 045651
12179 053714 036474
12180 053716 066675
12181 053720 163100
12182 053722 005407
12183 053724 022243

:THE CHECKWORDS CORRESPONDING TO ROM CHIPS #23-339E2 AND #23-340E2 FOLLOW:

.WORD 166020 ;ROMA: PAGE 0,1
.WORD 20232 ;ROMB: PAGE 2,3
.WORD 45651 ;ROMC: PAGE 4,5
.WORD 36474 ;ROMD: PAGE 6,7
.WORD 66675 ;ROME: PAGE 10,11
.WORD 163100 ;ROMF: PAGE 12,13
.WORD 5407 ;ROMG: PAGE 14,15
.WORD 22243 ;ROMH: PAGE 16,17

12184
12185
12186 053726 031547
12187 053730 014036
12188 053732 065162
12189 053734 124632
12190 053736 032040
12191 053740 167124
12192 053742 155461
12193 053744 032257

:THE CHECKWORDS CORRESPONDING TO ROM CHIPS #23-010E2 AND #23-011E2 FOLLOW:

.WORD 31547 ;ROMA: PAGE 0,1
.WORD 14036 ;ROMB: PAGE 2,3
.WORD 65162 ;ROMC: PAGE 4,5
.WORD 124632 ;ROMD: PAGE 6,7
.WORD 32040 ;ROME: PAGE 10,11
.WORD 167124 ;ROMF: PAGE 12,13
.WORD 155461 ;ROMG: PAGE 14,15
.WORD 32257 ;ROMH: PAGE 16,17

12194
12195

12196
12197
12198
12199
12200 053746 000000
12201 053750 000000
12202 053752 000000
12203 053754 000001
12204 053756 000000
12205 053760 000000
12206 053762 000000
12207
12208
12209
12210
12211
12212
12213
12214
12215
12216
12217 053764 012701 173776
12218 053770 066701 177754
12219 053774 005067 177762
12220 054000 012702 173000
12221 054004 066702 177740
12222 054010 111204
12223 054012 060467 177744
12224 054016 062702 000002
12225 054022 020201
12226 054024 002771
12227 054026 000207
12228
12229
12230
12231
12232
12233
12234
12235
12236
12237 054030 012767 000400 000352
12238 054036 016767 000346 177702
12239 054044 016767 177676 123446
12240 054052 012767 000010 177672
12241 054060 012705 053666
12242 054064 012767 000001 177664
12243 054072 005067 177652
12244 054076 122737 177777 173774
12245 054104 001001
12246 054106 104000
12247 054110 004767 177650
12248 054114 113767 173776 177636
12249 054122 066767 177634 177630
12250 054130 105767 177624
12251 054134 001401

```
:  
: DATA SECTION FOR THE NEXT TEST  
:*****  
:VRTPCR: .WORD 0 ;VIRTUAL PAGE CONTROL REGISTER  
:BCF: .WORD 0  
:COUNTR: .WORD 0  
:ANSR: .WORD 1  
:RFLAG: .WORD 0  
:EXPSUM: .WORD 0  
:ACTSUM: .WORD 0  
:*****  
:FUNCTIONAL DESCRIPTION:  
:SUBROUTINE TO COMPUTE A CHECKSUM IN A ROM/EPROM  
:INPUT: CONTENTS OF BCF  
:IMPLICIT INPUTS: CONTENTS OF PCR  
:OUTPUT: A CHECKSUM VALUE STORED IN LOCATION ACTSUM  
:CALLING SEQUENCE: JSR PC,CHKSUM  
:*****  
CHKSUM: MOV #173776,R1 ;STORE THE HIGHEST ADDRESS IN THE ROM  
:ADD BCF,R1 ;FOR EITHER LOW OR HIGH BYTES  
:CLR ACTSUM ;CLEAR LOCATION WHICH WILL HOLD THE CHECKSUM  
:MOV #173000,R2 ;COMPUTE THE LOWEST ADDRESS IN THE ROM  
:ADD BCF,R2 ;WHERE THE DATA WILL START  
1$: MOV (R2),R4 ;GET DATA IN BYTES  
:ADD R4,ACTSUM ;ADD CONTENTS OF EACH LOCATION TO THE CHECKSUM  
:ADD #2,R2 ;ADJUST ADDRESS  
:CMP R2,R1 ;COMPARE CURRENT ADDRESS WITH HIGHEST ADDRESS  
:BLT 1$ ;BR IF LESS THAN  
:RTS PC ;RETURN  
:***** *****  
:TEST TO PERFORM CHECKSUM AND CHECKWORD VERIFICATION ON THE 2K  
:OF DIAGNOSTIC ROM. IN UNATTENDED MODE, THE ROM WILL BE ADDRESSED  
:FROM 0-2K. IN STAND-ALONE MODE, THE OPERATOR MAY CHANGE THE  
:ADDRESS BY RESPONDING TO QUESTIONS GENERATED ON THE FIRST PASS.  
:***** *****  
BDVTS2: MOV #400,DRLP ;STORE STARTING ADDRESS  
:MOV DRLP,VRTPCR ;SET UP PCR  
:MOV VRTPCR,PCR  
:MOV #10,COUNTR ;SET NUMBER OF CHECKWORDS TO CHECK  
:MOV #SFPTBL,R5 ;LOCATION OF CHECKWORDS  
:MOV #1,RFLAG ;INDICATE ROM  
DLOOP: CLR BCF ;SIGNAL LOW BYTES ARE BEING CHECKED  
:CMPB #-1,@#173774 ;DOES THE ROM EXIST?  
:BNE 1$ ;BR IF YES  
:EMT  
1$: JSR PC,CHKSUM ;COMPUTE THE ACTUAL CHECKSUM  
:MOV @#173776,EXPSUM ;GET THE STORED CHECKSUM  
:ADD ACTSUM,EXPSUM ;ADD THE EXPECTED AND ACTUAL CHECKSUMS  
:TSTB EXPSUM ;BYTE RESULT = 0?  
:BEQ 2$ ;BR IF YES
```

```
12252 054136 104000
12253 054140 012767 000001 177602 2$: EMT
12254 054146 122737 177777 173775 MOV #1,BCF ;SET BCF TO DENOTE HIGH BYTES
12255 054154 001001 CMPB #-1,@#173775 ;DOES THE ROM EXIST?
12256 054156 104000 BNE 3$ ;BR IF YES
12257 054160 004767 177600 3$: EMT
12258 054164 113767 173777 177566 JSR PC,CHKSUM ;COMPUTE THE ACTUAL CHECKSUM
12259 054172 066767 177564 177560 MOVB @#173777,EXPSUM ;GET EXPECTED CHECKSUM
12260 054200 105767 177554 ADD ACTSUM,EXPSUM ;ADD THE EXPECTED AND ACTUAL CHECKSUMS
12261 054204 001401 TSTB EXPSUM ;BYTE RESULT = 0?
12262 054206 104000 BEQ 4$ ;BR IF YES
12263 054210 062767 001002 177530 4$: EMT
12264 054216 016767 177524 123274 ADD #1002,VRTPCR ;NEXT PAGE IN PCR
12265 054224 005367 177522 MOV VRTPCR,PCR
12266 054230 001320 DEC COUNTR ;DECREMENT CHECKWORD COUNT
12267
12268
12269 ;GET THE CHECKWORDS FROM THE ROMS AND PUT INTO TABLE 'CHKWRD'
12270 054232 012702 054412 MOV #CHKWRD,R2
12271 054236 012767 000001 177502 MOV #1,VRTPCR
12272 054244 012767 000010 177500 MOV #10,COUNTR
12273 054252 016767 177470 123240 5$: MOV VRTPCR,PCR
12274 054260 013722 173376 MOV @#173376,(R2)+
12275 054264 062767 000002 177454 ADD #2,VRTPCR
12276 054272 005367 177454 DEC COUNTR
12277 054276 001365 BNE 5$
12278
12279
12280 ;TRY TO IDENTIFY THE ROM CHIPS
12281 054300 016701 000126 MOV TABLES,R1
12282 054304 012767 053666 000122 MOV #SFPTBL,PNTR
12283 054312 016700 000116 SIXDLR: MOV PNTR,RO
12284 054316 012702 054412 MOV #CHKWRD,R2
12285 054322 012767 000010 177422 MOV #10,COUNTR
12286 054330 022022 7$: CMP (RO)+,(R2)+ ;ARE THE CHECKWORDS EQUAL?
12287 054332 001017 BNE NOTEQ ;BRANCH IF NOT
12288 054334 005367 177412 DEC COUNTR ;DONE CHECKING THIS TABLE?
12289 054340 001373 BNE 7$ ;BRANCH IF NOT
12290 054342 020127 000003 CMP R1,#3 ;DID THE FIRST TABLE OF CHECKWORDS COMPARE?
12291 054346 001001 BNE 8$ ;BRANCH IF NOT
12292 054350 000416 BR COMPLE
12293 054352 020127 000002 8$: CMP R1,#2 ;DID THE SECOND TABLE OF CHECKWORDS COMPARE?
12294 054356 001001 BNE NOCMP ;BRANCH IF NOT
12295 054360 000412 BR COMPLE
12296 054362 020127 000001 NOCMP: CMP R1,#1
12297 054366 001001 BNE NOTEQ
12298 054370 000406 BR COMPLE
12299 054372 062767 000020 000034 NOTEQ: ADD #20,PNTR
12300 054400 005301 DEC R1 ;ANY MORE TABLES TO CHECK?
12301 054402 001343 BNE SIXDLR ;BRANCH IF YES
12302 054404 104000 EMT
12303 054406 000515 COMPLE: BR FPSTRT
12304
12305
12306
12307
```

12308 054410 000000
12309
12310
12311 054412 000010
12312
12313 054432 000003
12314 054434 000000
12315
12316
12317
12318
12319
12320
12321
12322 054436 012737 000012 001002
12323 054444 012767 000001 124326
12324 054452 032737 000001 001020
12325 054460 001004
12326 054462 012700 054474
12327 054466 004767 056636
12328 054472 000777
12329
12330 054474 040506 046111 042105
12331 054502 042040 051125 047111
12332 054510 020107 044124 020105
12333 054516 042102 020126 042524
12334 054524 052123 005123 000015
12335
12336
12337 054532 012737 000012 001002
12338 054540 012767 000001 124232
12339 054546 032737 000001 001020
12340 054554 001346
12341 054556 012700 054570
12342 054562 004767 056542
12343 054566 000000
12344 054570 044103 041505 020113
12345 054576 053523 052111 044103
12346 054604 051505 047440 020116
12347 054612 030505 031060 020054
12348 054620 042522 052123 051101
12349 054626 020124 052101 031040
12350 054634 030060 006412 000
12351 054642
12352
12353
12354
12355
12356
12357
12358
12359
12360
12361
12362 000244
12363

DRLP: .WORD 0
CHKWRD: .BLKW 10 ;TABLE TO STORE THE CHECKWORDS
TABLES: .WORD 3 ;NUMBER OF CHECKWORD TABLES
PNTR: .WORD 0 ;WILL BE USED AS A POINTER
:*****
: ERROR ROUTINE FOR THE BDV TESTING
:*****
ERRORC: MOV #12,@#FATAL
MOV #1,\$MSGTY
BIT @#SENV ;UNDER APT ?
BNE BDVHLT
MOV #BDVMSG,RO
JSR PC,TYPE
BDVHLT: BR .
BDVMSG: .ASCIZ /FAILED DURING THE BDV TESTS/<12><15>
ERRORD: MOV #12,@#FATAL
MOV #1,\$MSGTY
BIT @#SENV
BNE BDVHLT
MOV #SWMSG,RO
JSR PC,TYPE
HALT
SWMSG: .ASCIZ /CHECK SWITCHES ON E102. RESTART AT 200/<12><15>
.EVEN
:* *****
FPVECT=244
.SBTTL FPP REGISTER DEFINITIONS

12364	000000				AC0	=%0	
12365	000001				AC1	=%1	
12366	000002				AC2	=%2	
12367	000003				AC3	=%3	
12368	000004				AC4	=%4	
12369	000005				AC5	=%5	
12370	000006				AC6	=%6	
12371	000007				AC7	=%7	
12372							
12373							
12374	054642	012706	001000		FPSTRT:	MOV #STBOT,SP	;SET UP STACK POINTER
12375	054646	000244				CLZ	
12376	054650	032777	000002	144526		BIT #2,@SWR	
12377	054656	001002				BNE 1\$	
12378	054660	000167	047772			JMP SLU1ST	
12379	054664	012737	124476	000030	1\$:	MOV #ERROR4,@#30	;SETUP FOR CORRECT ERROR CALL
12380	054672	012737	000003	001004		MOV #3,@#STESTN	;PUT TEST NUMBER IN MAILBOX
12381							
12382							
12383							
12384							
12385							
12386	054700						
12387	054700	012700	177777				
12388	054704	012737	054756	000244		MOV #AERR1,@#FPVECT	;SET UP FOR UNABLE TO DECODE
12389	054712	012737	054756	000010		MOV #AERR1,@#10	;FPP INSTRUCTION TRAP TO 244 OR 10.
12390	054720	012737	054756	000004		MOV #AERR1,@#ERRVECT	;IF EITHER INSTRUCTION
12391							;FAILS TO GO THROUGH THE
12392							;CORRECT SRC OR DST MODE AN
12393							;ODD ADDRESS TRAP WILL OCCUR.
12394	054726				A1:		
12395	054726	010004			A11:	MOV R0,R4	
12396	054730	042704	030020			BIC #30020,R4	
12397	054734	170104				LDFPS R4	;TEST INSTRUCTION.
12398							
12399	054736	012701	177777			MOV #-1,R1	
12400	054742	170201			A12:	STFPS R1	;TEST INSTRUCTION.
12401	054744	010004				MOV R0,R4	;MASK OFF UNSETTABLE BITS.
12402	054746	042704	030020			BIC #30020,R4	
12403	054752	020401				CMP R4,R1	;COMPARE DATA EXPECTED WITH
12404							;THE DATA READ.
12405	054754	001401				BEQ A2	
12406	054756				AERR1:		
12407	054756	104000				EMT	:
12408							
12409	054760	012700	000001		A2:	MOV #1,R0	;NEXT PATTERN WILL BE ALL ZERO
12410	054764	077020				SUB R0,A1	;DECREMENT COUNT PATTERN
12411	054766				ADONE:		
12412	054766	004767	047600			JSR PC,.RSET	;GO INITIALIZE THE FPS AND STACK; AND
12413							;SEE IF THE USER HAS EXPRESSED
12414							;THE DESIRE TO CHANGE THE SOFTWARE
12415							;VIRTUAL CONSOLE SWITCH REGISTER (HAS
12416							;THE USER TYPED CONTROL G?).
12417							
12418							
12419							

```

12420
12421
12422 054772
12423 054772 012700 000017
12424
12425 054776
12426 054776 170100
12427
12428 055000
12429 055000 170000
12430
12431 055002 013703 177776
12432 055006 042703 177760
12433 055012 020003
12434 055014 001401
12435 055016 104000
12436 055020 077012
12437 055022
12438 055022 004767 047544
12439
12440
12441
12442
12443
12444
12445
12446
12447 055026
12448 055026 005000
12449
12450 055030 170100
12451 055032 170001
12452
12453 055034 170201
12454 055036 005002
12455 055040 020201
12456 055042 001401
12457 055044 104000
12458 055046 012700 147757
12459
12460 055052 170100
12461 055054 170001
12462
12463 055056 170201
12464 055060 012702 147557
12465 055064 020102
12466 055066 001401
12467 055070 104000
12468 055072 012700 147757
12469
12470 055076 170100
12471 055100 170011
12472
12473 055102 170201
12474 055104 012702 147757
12475 055110 020102
  
```

```

;TEST 433      CFCC TEST
:*****
TS433:
      MOV      #17,R0          ;R0 CONTAINS TO TEST PATTERN.
B1:   LDFPS   R0              ;LOAD THE TEST PATTERN
B2:   CFCC
      ;COPY CONDITION CODES.
      MOV      @#PSW,R3       ;SEE IF PATTERN TRANSFERED.
      BIC      #177760,R3
      CMP      R0,R3
      BEQ      B3
B3:   EMT
      SOB      R0,B1          ;
BDONE: JSR      PC,.RSET      ;GO INITIALIZE THE FPS AND STACK; AND
      ;SEE IF THE USER HAS EXPRESSED
      ;THE DESIRE TO CHANGE THE SOFTWARE
      ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
      ;THE USER TYPED CONTROL G?).
:*****
;TEST 434      SETF, SETD, SETI AND SETL TEST
:*****
TS434:
      CLR      R0
C15:  LDFPS   R0              ;CLEAR THE FPS.
      SETF
      ;TEST INSTRUCTION.
      STFPS   R1              ;GET RESULT.
      CLR      R2
      CMP      R2,R1          ;DID AN ERROR OCCUR?
      BEQ      C2
C2:   EMT
      MOV      #147757,R0
      ;
C25:  LDFPS   R0              ;PUT 147757 IS FPS
      SETF
      ;CLEAR FD BIT.
      STFPS   R1              ;GET RESULT
      MOV      #147557,R2
      CMP      R1,R2          ;RESULT CORRECT.
      BEQ      C3
C3:   EMT
      MOV      #147757,R0
      ;
C35:  LDFPS   R0              ;LOAD 147757 INTO FPS.
      SETD
      ;SETD FD BIT.
      STFPS   R1
      MOV      #147757,R2
      CMP      R1,R2          ;RESULT CORRECT?
  
```

12476	055112	001401		BEQ	C4		
12477	055114	104000		EMT			
12478	055116	005000		C4:	CLR	R0	
12479	055120	170100			LDFPS	R0	;CLEAR FPS.
12480	055122	170011		C45:	SETD		;SET FD BIT.
12481							
12482	055124	170201			STFPS	R1	;GET RESULT.
12483	055126	012702	000200		MOV	#200,R2	
12484	055132	020102			CMP	R1,R2	;RESULT CORRECT?
12485	055134	001401			BEQ	C5	
12486	055136	104000			EMT		
12487	055140	005000		C5:	CLR	R0	
12488							
12489	055142	170100			LDFPS	R0	;CLEAR FPS
12490	055144	170002		C55:	SETI		;CLEAR FL BIT.
12491							
12492	055146	170201			STFPS	R1	;GET RESULT.
12493	055150	005002			CLR	R2	
12494	055152	020201			CMP	R2,R1	;RESULT CORRECT?
12495	055154	001401			BEQ	C6	
12496	055156	104000			EMT		
12497	055160	012700	147757	C6:	MOV	#147757,R0	
12498	055164	170100			LDFPS	R0	;PUT 147757 INTO FPS
12499	055166	170002		C65:	SETI		;CLEAR FL BIT.
12500							
12501	055170	170201			STFPS	R1	;GET THE RESULT.
12502	055172	012702	147657		MOV	#147657,R2	
12503	055176	020102			CMP	R1,R2	;RESULT CORRECT?
12504	055200	001401			BEQ	C7	
12505	055202	104000			EMT		
12506	055204	012700	147757	C7:	MOV	#147757,R0	
12507	055210	170100			LDFPS	R0	;SET FPS TO 147757.
12508	055212	170012		C75:	SETL		;SET FL BIT.
12509							
12510	055214	170201			STFPS	R1	;GET THE RESULT.
12511	055216	012702	147757		MOV	#147757,R2	
12512	055222	020102			CMP	R1,R2	;RESULT CORRECT?
12513	055224	001401			BEQ	C8	
12514	055226	104000			EMT		
12515	055230	005000		C8:	CLR	R0	
12516	055232	170100			LDFPS	R0	;CLEAR FPS.
12517	055234	170012		C85:	SETL		;SET FL BIT.
12518							
12519	055236	170201			STFPS	R1	
12520	055240	012702	000100		MOV	#100,R2	
12521	055244	020102			CMP	R1,R2	;RESULT CORRECT.
12522	055246	001401			BEQ	CDONE	
12523	055250	104000			EMT		
12524	055252			CDONE:			
12525	055252	004767	047314		JSR	PC..RSET	;GO INITIALIZE THE FPS AND STACK; AND
12526							;SEE IF THE USER HAS EXPRESSED
12527							;THE DESIRE TO CHANGE THE SOFTWARE
12528							;VIRTUAL CONSOLE SWITCH REGISTER (HAS
12529							;THE USER TYPED CONTROL G?).
12530							
12531							


```

:*****
:TEST 435      ILLEGAL FPP OP CODES AND STST TEST
:*****
TS435:
      MOV      #170003,R5      ;INITIAL OP CODE.
      MOV      #DERR2,@#ERRVECT
      MOV      #DERR1,@#FPVECT

D1:   CLR      R0              ;CLEAR FPS.
      LDFPS   R0
      CLR      R2
      MOV      R5,@#D2        ;SET UP THE ILLEGAL INSTRUCTION.

D2:   .WORD   0
D3:   CFCC
      INC      R2
D4:   INC      R2

      STFPS   R1              ;REPORT FAILURE. DID NOT TRAP.
DERR2:

D5:   EMT
      CMP      #170010,R5      ;
      BNE     D6                ;COMPUTE NEXT OP CODE
      MOV     #170013,R5
      BR      D1

D6:   CMP      #170077,R5
      BNE     D7
      BR      DDONE

D7:   INC      R5
      BR      D1

DERR1: CMP      #D3,(SP)        ;DID TRAP OCCUR ON TEST INSTRUCTION?
      BEQ     1$

1$:   EMT
      CMP      (SP)+,(SP)+      ;
      STFPS   R1                ;GET THE FPS AND SEE IF IT IS
      CMP      #100000,R1        ;SET CORRECTLY.
      BEQ     3$

3$:   MOV      #1,R4
      BR      D8

D8:   STST    R4                ;GET THE FEC CODE. NOTE THAT
                                      ;IF THE DESTINATION MODE IS
                                      ;IMPROPERLY DECODED AN ODD
                                      ;ADDRESS TRAP TO 4 SHOULD OCCUR.
                                      ;WAS FEC CORRECT?

D9:   EMT
                                      ;REPORT STST FAILURE

DDONE: JSR     PC,,RSET        ;GO INITIALIZE THE FPS AND STACK; AND
                                      ;SEE IF THE USER HAS EXPRESSED
                                      ;THE DESIRE TO CHANGE THE SOFTWARE
                                      ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
                                      ;THE USER TYPED CONTROL G?).
  
```

12532
 12533
 12534
 12535 055256
 12536 055256 012705 170003
 12537 055262 012737 055322 000004
 12538 055270 012737 055354 000244
 12539
 12540 055276 005000
 12541 055300 170100
 12542 055302 005002
 12543 055304 010537 055310
 12544 055310 000000
 12545 055312 170000
 12546 055314 005202
 12547 055316 005202
 12548
 12549 055320 170201
 12550 055322
 12551 055322 104000
 12552 055324 022705 170010
 12553 055330 001003
 12554 055332 012705 170013
 12555 055336 000757
 12556
 12557 055340 022705 170077
 12558 055344 001001
 12559 055346 000424
 12560 055350 005205
 12561 055352 000751
 12562
 12563 055354 022716 055312
 12564 055360 001401
 12565 055362 104000
 12566 055364 022626
 12567 055366 170201
 12568 055370 022701 100000
 12569 055374 001401
 12570 055376 104000
 12571 055400 012704 000001
 12572 055404 170304
 12573
 12574
 12575
 12576 055406 022704 000002
 12577 055412 001001
 12578 055414 000743
 12579
 12580 055416
 12581 055416 104000
 12582 055420
 12583 055420 004767 047146
 12584
 12585
 12586
 12587

12588
12589
12590
12591
12592 055424
12593 055424 012737 055464 000244
12594
12595 055432 012700 040000
12596 055436 170100
12597 055440 170020
12598 055442 170000
12599
12600 055444 170201
12601 055446 022701 140000
12602 055452 001004
12603
12604 055454 170304
12605 055456 022704 000002
12606 055462 001401
12607 055464
12608 055464 104000
12609 055466
12610 055466 004767 047100
12611
12612
12613
12614
12615
12616
12617
12618
12619
12620 055472
12621
12622 055472 005000
12623 055474 170100
12624 055476 170011
12625 055500 012701 055742
12626 055504 012702 056006
12627 055510 012703 000010
12628
12629 055514 012221
12630 055516 077302
12631
12632 055520 012700 055752
12633 055524 012737 055740 000004
12634
12635 055532 005003
12636
12637 055534 172410
12638 055536 005203
12639 055540 005203
12640
12641 055542 020027 055752
12642 055546 001401
12643 055550 104000

```
*****  
:TEST 436      FID, INTERRUPT DISABLE, BIT TEST  
*****  
TS436:  
      MOV      #EERRO,@#FPVTECT      ;SETUP FOR THE INTERRUPT.  
E1:    MOV      #40000,R0  
      LDFPS    R0                      ;SET FID.  
E3:    .WORD   170020                  ;ILLEGAL FPP INSTRUCTION.  
E4:    CFCC  
  
      STFPS    R1                      ;SEE IF ERROR WAS DETECTED.  
      CMP      #140000,R1  
      BNE      EERRO  
  
      STST     R4                      ;SEE IF FEC=2  
      CMP      #2,R4  
      BEQ      EDONE  
EERRO: EMT  
EDONE: JSR      PC,,RSET                ;GO INITIALIZE THE FPS AND STACK; AND  
                                           ;SEE IF THE USER HAS EXPRESSED  
                                           ;THE DESIRE TO CHANGE THE SOFTWARE  
                                           ;VIRTUAL CONSOLE SWITCH REGISTER (HAS  
                                           ;THE USER TYPED CONTROL G?).  
  
*****  
:TEST 437      LDD AND STD, WITH SRC AND DST MODE 1, TEST  
*****  
TS437:  
      CLR      R0  
      LDFPS    R0  
      SETD  
      MOV      #FDAT10,R1              ;SET UP THE LOAD DATA.  
      MOV      #FXDAT0,R2  
      MOV      #10,R3  
F2:    MOV      (R2)+,(R1)+  
      SOB     R3,F2  
  
      MOV      #FDAT14,R0              ;SETUP R0 FOR THE LDD (R0),ACO.  
      MOV      #FERR20,@#ERRVTECT    ;IF THE SRC FLOWS FAIL THEN  
                                           ;AN ODD ADDRESS MAY OCCUR.  
  
      CLR      R3  
F3:    LDD      (R0),ACO  
F4:    INC      R3  
      INC      R3  
  
      CMP      R0,#FDAT14              ;WAS R0 AFFECTED?  
      BEQ     F5  
      EMT  
      ;
```

12644	055552	020327	000002	F5:	CMP R3,#2	;SEE IF THE PC WAS ADVERSELY
12645	055556	001401			BEQ 1\$	
12646	055560	104000			EMT	
12647	055562	012701	055742	1\$:	MOV #FDAT10,R1	;MAKE SURE THE SOURCE DATA WAS
12648	055566	012702	056006		MOV #FXDAT0,R2	;NOT AFFECTED.
12649	055572	012703	000010		MOV #10,R3	
12650	055576	022122		2\$:	CMP (R1)+,(R2)+	
12651	055600	001401			BEQ 3\$	
12652	055602	104000			EMT	
12653	055604	077304		3\$:	SOB R3,2\$	
12654						
12655	055606	170201			STFPS R1	;MAKE SURE THE FPS IS CORRECT.
12656	055610	022701	000200		CMP #200,R1	
12657	055614	001401			BEQ F6	
12658	055616	104000			EMT	
12659	055620	012703	177777	F6:	MOV #-1,R3	
12660	055624	012704	000010		MOV #10,R4	
12661	055630	012705	055764		MOV #FDAT00,R5	;SET UP THE OUTPUT DATA BUFFER.
12662	055634	010325		F7:	MOV R3,(R5)+	
12663	055636	077402			SOB R4,F7	
12664						
12665	055640	012700	055774		MOV #FDAT04,R0	;SET UP R0 FOR DST MODE 1 REG 0.
12666	055644	012737	055740 000004		MOV #FERR20,@#ERRVECT	;IF THE DST FLOWS FAIL AN ODD
12667						;ADDRESS COULD OCCUR.
12668	055652	005003			CLR R3	
12669						
12670	055654	174010		F10:	STD AC0,(R0)	;TEST INSTRUCTION.
12671	055656	005203		F11:	INC R3	
12672	055660	005203			INC R3	
12673						
12674	055662	020027	055774		CMP R0,#FDAT04	;WAS R0 MODIFIED?
12675	055666	001401			BEQ F12	
12676	055670	104000			EMT	
12677	055672	020327	000002	F12:	CMP R3,#2	;WAS THE PC AFFECTED CORRECTLY?
12678	055676	001401			BEQ F135	
12679	055700	104000			EMT	
12680	055702	012701	055764	F135:	MOV #FDAT00,R1	
12681	055706	012702	056006		MOV #FXDAT0,R2	
12682	055712	012703	000010		MOV #10,R3	;SETUP LOOP COUNT
12683	055716	022122		F13:	CMP (R1)+,(R2)+	;WAS DATA OUTPUT CORRECTLY
12684	055720	001401			BEQ F14	
12685	055722	104000			EMT	
12686	055724	077304		F14:	SOB R3,F13	;SUBTRACT 1 FROM LOOP COUNT AND LOOP IF NOT ZERO
12687	055726	005001		F22:	CLR R1	
12688	055730	170201			STFPS R1	;MAKE SURE FPS IS CORRECT.
12689	055732	022701	000200		CMP #200,R1	
12690	055736	001433			BEQ FDONE	
12691	055740			FERR20:		
12692	055740	104000			EMT	
12693						
12694	055742	177777		FDAT10:	-1	
12695	055744	177777		FDAT11:	-1	
12696	055746	177777		FDAT12:	-1	
12697	055750	177777		FDAT13:	-1	
12698	055752	177777		FDAT14:	-1	
12699	055754	177777		FDAT15:	-1	

12700 055756 177777
 12701 055760 177777
 12702 055762 177777
 12703 055764 177777
 12704 055766 177777
 12705 055770 177777
 12706 055772 177777
 12707 055774 177777
 12708 055776 177777
 12709 056000 177777
 12710 056002 177777
 12711 056004 177777
 12712 056006 177777
 12713 056010 177777
 12714 056012 177777
 12715 056014 177777
 12716 056016 052525
 12717 056020 031463
 12718 056022 007417
 12719 056024 000477
 12720
 12721
 12722 056026
 12723 056026 004767 046540
 12724
 12725
 12726
 12727
 12728
 12729
 12730
 12731
 12732
 12733 056032
 12734 056032
 12735 056032 170011
 12736 056034 012700 056320
 12737 056040 012701 056270
 12738 056044 012702 000004
 12739 056050 012120
 12740 056052 077202
 12741
 12742 056054 012700 056320
 12743 056060 172510
 12744
 12745 056062 012700 056300
 12746 056066 172410
 12747
 12748 056070 012701 000001
 12749 056074 172401
 12750 056076 000240
 12751 056100 000240
 12752
 12753 056102 012700 056310
 12754 056106 174010
 12755

FDATE6: -1
 FDATE7: -1
 -1
 FDATE0: -1
 FDATE01: -1
 FDATE02: -1
 FDATE03: -1
 FDATE04: -1
 FDATE05: -1
 FDATE06: -1
 FDATE07: -1
 -1
 FXDATE0: -1
 FXDATE1: -1
 FXDATE2: -1
 FXDATE3: -1
 FXDATE4: 052525
 FXDATE5: 031463
 FXDATE6: 007417
 FXDATE7: 000477

FDONE: JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND
 ;SEE IF THE USER HAS EXPRESSED
 ;THE DESIRE TO CHANGE THE SOFTWARE
 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
 ;THE USER TYPED CONTROL G?).

 ;TEST 440 FSRC MODE 0 TEST

TS440:
 I1: SETD ;SET FD.
 MOV #IDATIO,R0
 MOV #IPATIO,R1
 MOV #4,R2
 I2: MOV (R1)+,(R0)+ ;SET UP THE INPUT DATA BUFFER.
 SOB R2,I2
 MOV #IDATIO,R0 ;LOAD AC1
 LDD (R0),AC1
 MOV #IPATIO,R0 ;LOAD AC0
 LDD (R0),AC0
 MOV #1,R1 ;IN CASE THE FSRC FLOWS FAIL
 I3: LDD AC1,AC0 ;TEST INSTRUCTION.
 I4: NOP
 I5: NOP
 MOV #IDATIO,R0 ;GET AC0, THE RESULTS.
 STD AC0,(R0)

12756	056110	012700	056310		MOV	#IDAT00,R0		;SEE IF DATA IS CORRECT.
12757	056114	012701	056320		MOV	#IDAT10,R1		
12758	056120	012702	000004		MOV	#4,R2		
12759	056124	022021		I6:	CMP	(R0)+,(R1)+		
12760	056126	001401			BEQ	I105		
12761	056130	104000			EMT			
12762	056132	077204		I105:	SOB	R2,I6		
12763								
12764								
12765								
12766	056134	012700	056270	I12:	MOV	#IPAT10,R0		
12767	056140	012701	056320		MOV	#IDAT10,R1		
12768	056144	012702	000004		MOV	#4,R2		
12769	056150	012021		I13:	MOV	(R0)+,(R1)+		
12770	056152	077202			SOB	R2,I13		
12771								
12772	056154	012700	056320		MOV	#IDAT10,R0		;SET UP AC1
12773	056160	172510			LDD	(R0),AC1		
12774								
12775	056162	012700	056300		MOV	#IPAT20,R0		;SET UP AC0
12776	056166	172410			LDD	(R0),AC0		
12777								
12778	056170	012701	000001		MOV	#1,R1		
12779	056174	170001			SETF			;CLEAR FD.
12780								
12781	056176	172401		I14:	LDF	AC1,AC0		;TEST INSTRUCTION.
12782	056200	000240		I15:	NOP			
12783	056202	000240		I16:	NOP			
12784								
12785	056204	170200			STFPS	R0		;SEE IF FPS IS STILL CLEAR.
12786	056206	022700	000004		CMP	#4,R0		
12787	056212	001401			BEQ	I17		
12788	056214	104000			EMT			
12789	056216			I17:				;RESET TO DOUBLE MODE.
12790	056216	170011			SETD			
12791								
12792	056220	012700	056310		MOV	#IDAT00,R0		
12793	056224	174010			STD	AC0,(R0)		;GET AC0
12794								
12795	056226	012737	177777	056324	MOV	#-1,@#IDAT12		
12796	056234	012737	177777	056326	MOV	#-1,@#IDAT13		
12797	056242	012700	056310		MOV	#IDAT00,R0		
12798	056246	012701	056320		MOV	#IDAT10,R1		
12799	056252	012702	000004		MOV	#4,R2		
12800	056256	022021		I20:	CMP	(R0)+,(R1)+		;SEE IF AC0 WAS CORRECT.
12801	056260	001401			BEQ	I23		
12802	056262	104000			EMT			
12803	056264	077204		I23:	SOB	R2,I20		
12804	056266	000420			BR	IDONE		;NO ERRORS.
12805								
12806	056270	000000			IPAT10:	0		
12807	056272	170360			IPAT11:	170360		
12808	056274	016161			IPAT12:	016161		
12809	056276	052525			IPAT13:	052525		
12810								
12811	056300	177777			IPAT20:	-1		

12812 056302 177777
12813 056304 177777
12814 056306 177777
12815
12816 056310 000000
12817 056312 000000
12818 056314 000000
12819 056316 000000
12820
12821 056320 000000
12822 056322 000000
12823 056324 000000
12824 056326 000000
12825
12826 056330
12827 056330 004767 046236
12828
12829
12830
12831
12832
12833
12834
12835
12836
12837 056334
12838 056334 170011
12839 056336 012700 056574
12840 056342 012701 056624

IPAT21: -1
IPAT22: -1
IPAT23: -1

IDAT00: 0
IDAT01: 0
IDAT02: 0
IDAT03: 0

IDAT10: 0
IDAT11: 0
IDAT12: 0
IDAT13: 0

IDONE:

JSR PC,.RSET

;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;TEST 441 FDST MODE 0 TEST

TS441:

SETD ;SET FD
MOV #TPAT10,R0
MOV #TDAT10,R1

CJKDJBO 11/23-B CPU CLUSTER DIAG.
CJKDJB.P11 26-MAY-82 11:14

DNMAC X24.07-563 26-MAY-82^{C 4} 11:18 PAGE 248
T441 FDST MODE 0 TEST

SEQ 0247

12841 056346 012702 000004

MOV #4,R2

CJKDJBO 11/23-B CPU CLUSTER DIAG.
CJKDJB.P11 26-MAY-82 11:14

DNMAC X24.07-563 26-MAY-82^{D 4} 11:18 PAGE 249
T441 FDST MODE 0 TEST

SEQ 0248

12842 056352 012021
12843 056354 077202
12844

T2: MOV (R0)+,(R1)+ ;SET UP THE INPUT DATA BUFFER.
SOB R2,T2

12845	056356	012700	056624		MOV	#TDAT10,R0		:LOAD AC0
12846	056362	172410			LDD	(R0),AC0		
12847								
12848	056364	012700	056604		MOV	#TPAT20,R0		:LOAD AC1
12849	056370	172510			LDD	(R0),AC1		
12850								
12851	056372	012701	000001		MOV	#1,R1		:IF THE (BUT FDST) FORK FAILS
12852	056376	174001		T3:	STD	AC0,AC1		
12853	056400	000240		T4:	NOP			
12854	056402	000240		T5:	NOP			
12855								
12856	056404	012700	056614		MOV	#TDAT00,R0		:GET THE DATA.
12857	056410	174110			STD	AC1,(R0)		
12858								
12859	056412	012703	056614		MOV	#TDAT00,R3		:SEE IF THE DATA IS CORRECT.
12860	056416	012704	056624		MOV	#TDAT10,R4		
12861	056422	012705	000004		MOV	#4,R5		
12862	056426	022324		T6:	CMP	(R3)+,(R4)+		
12863	056430	001401			BEQ	T105		
12864	056432	104000			EMT			
12865	056434	077504		T105:	SOB	R5,T6		
12866								
12867								:NOW TEST THE STF AC0,AC1 INSTRUCTION.
12868								
12869	056436	012700	056574	T12:	MOV	#TPAT10,R0		:SET UP THE INPUT DATA BUFFER.
12870	056442	012701	056624		MOV	#TDAT10,R1		
12871	056446	012702	000004		MOV	#4,R2		
12872	056452	012021		T13:	MOV	(R0)+,(R1)+		
12873	056454	077202			SOB	R2,T13		
12874								
12875	056456	012700	056624		MOV	#TDAT10,R0		:SET UP AC0
12876	056462	172410			LDD	(R0),AC0		
12877								
12878	056464	012700	056604		MOV	#TPAT20,R0		:SET UP AC1
12879	056470	172510			LDD	(R0),AC1		
12880								
12881	056472	012701	000001		MOV	#1,R1		
12882	056476	170001			SETF			:CLEAR FD
12883	056500	174001		T14:	STF	AC0,AC1		
12884	056502	000240		T15:	NOP			
12885	056504	000240		T16:	NOP			
12886								
12887	056506	005000			CLR	R0		
12888	056510	170200			STFPS	R0		:SEE IF FPS IS CLEAR.
12889	056512	022700	000010		CMP	#10,R0		
12890	056516	001401			BEQ	T17		
12891	056520	104000			EMT			
12892	056522			T17:				
12893	056522	170011			SETD			:SET FD.
12894								
12895	056524	012700	056614		MOV	#TDAT00,R0		:PICK UP AC1.
12896	056530	174110			STD	AC1,(R0)		
12897								
12898	056532	012737	177777	056630	MOV	#-1,@#TDAT12		
12899	056540	012737	177777	056632	MOV	#-1,@#TDAT13		
12900	056546	012703	056614		MOV	#TDAT00,R3		

12901 056552 012704 056624
12902 056556 012705 000004
12903 056562 022324
12904 056564 001401
12905 056566 104000
12906 056570 077504
12907 056572 000420
12908
12909
12910 056574 000000
12911 056576 170360
12912 056600 016161
12913 056602 052525
12914
12915 056604 177777
12916 056606 177777
12917 056610 177777
12918 056612 177777
12919
12920 056614 000000
12921 056616 000000
12922 056620 000000
12923 056622 000000
12924
12925 056624 000000
12926 056626 000000
12927 056630 000000
12928 056632 000000
12929
12930 056634
12931 056634 004767 045732
12932
12933
12934
12935
12936
12937
12938
12939
12940
12941
12942 056640
12943 056640 170011
12944
12945 056642 012700 060374
12946 056646 012701 060434
12947 056652 004737 060246
12948 056656 012703 000102
12949 056662
12950 056662 172410
12951 056664 174000
12952 056666 172400
12953 056670 174011
12954 056672 004737 060344
12955
12956 056676 005737 060370

MOV #TDATIO,R4
MOV #4,R5
T20: CMP (R3)+,(R4)+ ;WAS THE DATA TRANSFERRED CORRECTLY?
BEQ T23
EMT
T23: SOB R5,T20
BR TDONE

TPAT10: 0
TPAT11: 170360
TPAT12: 016161
TPAT13: 052525

TPAT20: -1
TPAT21: -1
TPAT22: -1
TPAT23: -1

TDAT00: 0
TDAT01: 0
TDAT02: 0
TDAT03: 0

TDATIO: 0
TDATI1: 0
TDATI2: 0
TDATI3: 0

TDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

:TEST 442 ACCUMULATORS DATA PATTERNS TEST

TS442: SETD ;SET FD.
:TEST ACCUMULATOR 0 WITH FLOATING ONE
MOV #GPAT00,R0
MOV #GDAT00,R1
JSR PC,@#GSETUP ;LOAD TEST PATTERN.
MOV #102,R3
G1: LDD (R0),ACO
STD ACO,ACO
LDD ACO,ACO ;STORE THE TEST PATTERN.
STD ACO,(R1)
JSR PC,@#GCMP ;COMPARE THE DATA READ WITH
;THAT WHICH WAS WRITTEN.
TST @#GFLAG1

12957	056702	001004		BNE	G2	
12958	056704	005137	060370	COM	@#GFLAG1	
12959	056710	000261		SEC		
12960	056712	000401		BR	G3	
12961	056714	000241		G2:	CLC	
12962	056716	006160	000006	G3:	ROL	6(R0)
12963	056722	006160	000004			:GENERATE THE NEXT TEST PATTERN.
12964	056726	006160	000002		ROL	4(R0)
12965	056732	006110			ROL	2(R0)
12966	056734	004737	060324		ROL	(R0)
12967					JSR	PC,@#GRESET
12968	056740	077330				:RESET DEFAULT PATTERN IN OUTPUT
12969					SOB	R3,G1
12970						:BUFFER.
12971	056742	012700	060404			:TEST ACCUMULATOR 0 WITH FLOATING ZERO
12972	056746	012701	060434		MOV	#GPAT0,R0
12973	056752	004737	060246		MOV	#GDAT00,R1
12974	056756	012703	000102		JSR	PC,@#GSETUP
12975	056762					:LOAD TEST PATTERN.
12976	056762	172410		G4:	MOV	#102,R3
12977	056764	174000			LDD	(R0),ACO
12978	056766	172400			STD	ACO,ACO
12979	056770	174011			LDD	ACO,ACO
12980	056772	004737	060344		STD	ACO,(R1)
12981					JSR	PC,@#GCMP
12982	056776	005737	060370			:STORE THE TEST PATTERN.
12983	057002	001004			JSR	PC,@#GCMP
12984	057004	005137	060370			:COMPARE THE DATA READ WITH
12985	057010	000241			TST	@#GFLAG1
12986	057012	000401			BNE	G5
12987	057014	000261			COM	@#GFLAG1
12988	057016	006160	000006		CLC	
12989	057022	006160	000004		BR	G6
12990	057026	006160	000002	G5:	SEC	
12991	057032	006110		G6:	ROL	6(R0)
12992	057034	004737	060324		ROL	4(R0)
12993					ROL	2(R0)
12994	057040	077330			ROL	(R0)
12995					JSR	PC,@#GRESET
12996						:RESET DEFAULT PATTERN IN OUTPUT
12997	057042	012700	060374			:BUFFER.
12998	057046	012701	060434		SOB	R3,G4
12999	057052	004737	060246			:TEST ACCUMULATOR 1 WITH FLOATING ONE
13000	057056	012703	000102		MOV	#GPAT0,R0
13001	057062				MOV	#GDAT00,R1
13002	057062	172410			JSR	PC,@#GSETUP
13003	057064	174001			MOV	#102,R3
13004	057066	172401		G7:	LDD	(R0),ACO
13005	057070	174011			STD	ACO,AC1
13006	057072	004737	060344		LDD	AC1,ACO
13007					STD	ACO,(R1)
13008	057076	005737	060370		JSR	PC,@#GCMP
13009	057102	001004				:STORE THE TEST PATTERN.
13010	057104	005137	060370			:COMPARE THE DATA READ WITH
13011	057110	000261			TST	@#GFLAG1
13012	057112	000401			BNE	G10
					COM	@#GFLAG1
					SEC	
					BR	G11

13013	057114	000241	
13014	057116	006160	000006
13015	057122	006160	000004
13016	057126	006160	000002
13017	057132	006110	
13018	057134	004737	060324
13019			
13020	057140	077330	
13021			
13022			
13023	057142	012700	060404
13024	057146	012701	060434
13025	057152	004737	060246
13026	057156	012703	000102
13027	057162		
13028	057162	172410	
13029	057164	174001	
13030	057166	172401	
13031	057170	174011	
13032	057172	004737	060344
13033			
13034	057176	005737	060370
13035	057202	001004	
13036	057204	005137	060370
13037	057210	000241	
13038	057212	000401	
13039	057214	000261	
13040	057216	006160	000006
13041	057222	006160	000004
13042	057226	006160	000002
13043	057232	006110	
13044	057234	004737	060324
13045			
13046	057240	077330	
13047			
13048			
13049	057242	012700	060374
13050	057246	012701	060434
13051	057252	004737	060246
13052	057256	012703	000102
13053	057262		
13054	057262	172410	
13055	057264	174002	
13056	057266	172402	
13057	057270	174011	
13058	057272	004737	060344
13059			
13060	057276	005737	060370
13061	057302	001004	
13062	057304	005137	060370
13063	057310	000261	
13064	057312	000401	
13065	057314	000241	
13066	057316	006160	000006
13067	057322	006160	000004
13068	057326	006160	000002

```

G10:   CLC
G11:   ROL      6(R0)           ;GENERATE THE NEXT TEST PATTERN.
        ROL      4(R0)
        ROL      2(R0)
        ROL      (R0)
        JSR      PC,@#GRESET   ;RESET DEFAULT PATTERN IN OUTPUT
        SOB      R3,G7        ;BUFFER.

;TEST ACCUMULATOR 1 WITH FLOATING ZERO
        MOV      #GPAT10,R0
        MOV      #GDAT00,R1
        JSR      PC,@#GSETUP   ;LOAD TEST PATTERN.
        MOV      #102,R3

G12:   LDD      (R0),AC0
        STD      AC0,AC1
        LDD      AC1,AC0       ;STORE THE TEST PATTERN.
        STD      AC0,(R1)
        JSR      PC,@#GCMP     ;COMPARE THE DATA READ WITH
        ;THAT WHICH WAS WRITTEN.

        TST      @#GFLAG1
        BNE      G13
        COM      @#GFLAG1
        CLC
        BR      G14

G13:   SEC
G14:   ROL      6(R0)           ;GENERATE THE NEXT TEST PATTERN.
        ROL      4(R0)
        ROL      2(R0)
        ROL      (R0)
        JSR      PC,@#GRESET   ;RESET DEFAULT PATTERN IN OUTPUT
        SOB      R3,G12       ;BUFFER.

;TEST ACCUMULATOR 2 WITH FLOATING ONE
        MOV      #GPAT00,R0
        MOV      #GDAT00,R1
        JSR      PC,@#GSETUP   ;LOAD TEST PATTERN.
        MOV      #102,R3

G15:   LDD      (R0),AC0
        STD      AC0,AC2
        LDD      AC2,AC0       ;STORE THE TEST PATTERN.
        STD      AC0,(R1)
        JSR      PC,@#GCMP     ;COMPARE THE DATA READ WITH
        ;THAT WHICH WAS WRITTEN.

        TST      @#GFLAG1
        BNE      G16
        COM      @#GFLAG1
        SEC
        BR      G17

G16:   CLC
G17:   ROL      6(R0)           ;GENERATE THE NEXT TEST PATTERN.
        ROL      4(R0)
        ROL      2(R0)
  
```

13069	057332	006110		ROL	(R0)	
13070	057334	004737	060324	JSR	PC,@#GRESET	;RESET DEFAULT PATTERN IN OUTPUT
13071						;BUFFER.
13072	057340	077330		SOB	R3,G15	
13073						
13074						;TEST ACCUMULATOR 2 WITH FLOATING ZERO
13075	057342	012700	060404	MOV	#GPAT10,R0	
13076	057346	012701	060434	MOV	#GDAT00,R1	
13077	057352	004737	060246	JSR	PC,@#GSETUP	;LOAD TEST PATTERN.
13078	057356	012703	000102	MOV	#102,R3	
13079	057362			G20:		
13080	057362	172410		LDD	(R0),AC0	
13081	057364	174002		STD	AC0,AC2	
13082	057366	172402		LDD	AC2,AC0	;STORE THE TEST PATTERN.
13083	057370	174011		STD	AC0,(R1)	
13084	057372	004737	060344	JSR	PC,@#GCMP	;COMPARE THE DATA READ WITH
13085						;THAT WHICH WAS WRITTEN.
13086	057376	005737	060370	TST	@#GFLAG1	
13087	057402	001004		BNE	G21	
13088	057404	005137	060370	COM	@#GFLAG1	
13089	057410	000241		CLC		
13090	057412	000401		BR	G22	
13091	057414	000261		G21:	SEC	
13092	057416	006160	000006	G22:	ROL	6(R0)
13093	057422	006160	000004		ROL	4(R0)
13094	057426	006160	000002		ROL	2(R0)
13095	057432	006110			ROL	(R0)
13096	057434	004737	060324	JSR	PC,@#GRESET	;RESET DEFAULT PATTERN IN OUTPUT
13097						;BUFFER.
13098	057440	077330		SOB	R3,G20	
13099						
13100						;TEST ACCUMULATOR 3 WITH FLOATING ONE
13101	057442	012700	060374	MOV	#GPAT00,R0	
13102	057446	012701	060434	MOV	#GDAT00,R1	
13103	057452	004737	060246	JSR	PC,@#GSETUP	;LOAD TEST PATTERN.
13104	057456	012703	000102	MOV	#102,R3	
13105	057462			G23:		
13106	057462	172410		LDD	(R0),AC0	
13107	057464	174003		STD	AC0,AC3	
13108	057466	172403		LDD	AC3,AC0	;STORE THE TEST PATTERN.
13109	057470	174011		STD	AC0,(R1)	
13110	057472	004737	060344	JSR	PC,@#GCMP	;COMPARE THE DATA READ WITH
13111						;THAT WHICH WAS WRITTEN.
13112	057476	005737	060370	TST	@#GFLAG1	
13113	057502	001004		BNE	G24	
13114	057504	005137	060370	COM	@#GFLAG1	
13115	057510	000261		SEC		
13116	057512	000401		BR	G25	
13117	057514	000241		G24:	CLC	
13118	057516	006160	000006	G25:	ROL	6(R0)
13119	057522	006160	000004		ROL	4(R0)
13120	057526	006160	000002		ROL	2(R0)
13121	057532	006110			ROL	(R0)
13122	057534	004737	060324	JSR	PC,@#GRESET	;RESET DEFAULT PATTERN IN OUTPUT
13123						;BUFFER.
13124	057540	077330		SOB	R3,G23	

13125			
13126			
13127	057542	012700	060404
13128	057546	012701	060434
13129	057552	004737	060246
13130	057556	012703	000102
13131	057562		
13132	057562	172410	
13133	057564	174003	
13134	057566	172403	
13135	057570	174011	
13136	057572	004737	060344
13137			
13138	057576	005737	060370
13139	057602	001004	
13140	057604	005137	060370
13141	057610	000241	
13142	057612	000401	
13143	057614	000261	
13144	057616	006160	000006
13145	057622	006160	000004
13146	057626	006160	000002
13147	057632	006110	
13148	057634	004737	060324
13149			
13150	057640	077330	
13151			
13152			
13153	057642	012700	060374
13154	057646	012701	060434
13155	057652	004737	060246
13156	057656	012703	000102
13157	057662		
13158	057662	172410	
13159	057664	174004	
13160	057666	172404	
13161	057670	174011	
13162	057672	004737	060344
13163			
13164	057676	005737	060370
13165	057702	001004	
13166	057704	005137	060370
13167	057710	000261	
13168	057712	000401	
13169	057714	000241	
13170	057716	006160	000006
13171	057722	006160	000004
13172	057726	006160	000002
13173	057732	006110	
13174	057734	004737	060324
13175			
13176	057740	077330	
13177			
13178			
13179	057742	012700	060404
13180	057746	012701	060434

```

;TEST ACCUMULATOR 3 WITH FLOATING ZERO
MOV #GPAT10,R0
MOV #GDAT00,R1
JSR PC,@#GSETUP ;LOAD TEST PATTERN.
MOV #102,R3
G26: LDD (R0),AC0
STD AC0,AC3 ;STORE THE TEST PATTERN.
LDD AC3,AC0
STD AC0,(R1)
JSR PC,@#GCMP ;COMPARE THE DATA READ WITH
;THAT WHICH WAS WRITTEN.
TST @#GFLAG1
BNE G27
COM @#GFLAG1
CLC
BR G30
G27: SEC
G30: ROL 6(R0) ;GENERATE THE NEXT TEST PATTERN.
ROL 4(R0)
ROL 2(R0)
ROL (R0)
JSR PC,@#GRESET ;RESET DEFAULT PATTERN IN OUTPUT
;BUFFER.
SOB R3,G26

;TEST ACCUMULATOR 4 WITH FLOATING ONE
MOV #GPAT00,R0
MOV #GDAT00,R1
JSR PC,@#GSETUP ;LOAD TEST PATTERN.
MOV #102,R3
G31: LDD (R0),AC0
STD AC0,AC4 ;STORE THE TEST PATTERN.
LDD AC4,AC0
STD AC0,(R1)
JSR PC,@#GCMP ;COMPARE THE DATA READ WITH
;THAT WHICH WAS WRITTEN.
TST @#GFLAG1
BNE G32
COM @#GFLAG1
SEC
BR G33
G32: CLC
G33: ROL 6(R0) ;GENERATE THE NEXT TEST PATTERN.
ROL 4(R0)
ROL 2(R0)
ROL (R0)
JSR PC,@#GRESET ;RESET DEFAULT PATTERN IN OUTPUT
;BUFFER.
SOB R3,G31

;TEST ACCUMULATOR 4 WITH FLOATING ZERO
MOV #GPAT10,R0
MOV #GDAT00,R1

```

13181	057752	004737	060246		JSR	PC,@#GSETUP		;LOAD TEST PATTERN.
13182	057756	012703	000102		MOV	#102,R3		
13183	057762			G34:				
13184	057762	172410			LDD	(R0),AC0		
13185	057764	174004			STD	AC0,AC4		
13186	057766	172404			LDD	AC4,AC0		;STORE THE TEST PATTERN.
13187	057770	174011			STD	AC0,(R1)		
13188	057772	004737	060344		JSR	PC,@#GCMP		;COMPARE THE DATA READ WITH ;THAT WHICH WAS WRITTEN.
13189								
13190	057776	005737	060370		TST	@#GFLAG1		
13191	060002	001004			BNE	G35		
13192	060004	005137	060370		COM	@#GFLAG1		
13193	060010	000241			CLC			
13194	060012	000401			BR	G36		
13195	060014	000261		G35:	SEC			
13196	060016	006160	000006	G36:	ROL	6(R0)		;GENERATE THE NEXT TEST PATTERN.
13197	060022	006160	000004		ROL	4(R0)		
13198	060026	006160	000002		ROL	2(R0)		
13199	060032	006110			ROL	(R0)		
13200	060034	004737	060324		JSR	PC,@#GRESET		;RESET DEFAULT PATTERN IN OUTPUT ;BUFFER.
13201								
13202	060040	077330			SOB	R3,G34		
13203								
13204								;TEST ACCUMULATOR 5 WITH FLOATING ONE
13205	060042	012700	060374		MOV	#GPAT00,R0		
13206	060046	012701	060434		MOV	#GDAT00,R1		
13207	060052	004737	060246		JSR	PC,@#GSETUP		;LOAD TEST PATTERN.
13208	060056	012703	000102		MOV	#102,R3		
13209	060062			G37:				
13210	060062	172410			LDD	(R0),AC0		
13211	060064	174005			STD	AC0,AC5		
13212	060066	172405			LDD	AC5,AC0		;STORE THE TEST PATTERN.
13213	060070	174011			STD	AC0,(R1)		
13214	060072	004737	060344		JSR	PC,@#GCMP		;COMPARE THE DATA READ WITH ;THAT WHICH WAS WRITTEN.
13215								
13216	060076	005737	060370		TST	@#GFLAG1		
13217	060102	001004			BNE	G40		
13218	060104	005137	060370		COM	@#GFLAG1		
13219	060110	000261			SEC			
13220	060112	000401			BR	G41		
13221	060114	000241		G40:	CLC			
13222	060116	006160	000006	G41:	ROL	6(R0)		;GENERATE THE NEXT TEST PATTERN.
13223	060122	006160	000004		ROL	4(R0)		
13224	060126	006160	000002		ROL	2(R0)		
13225	060132	006110			ROL	(R0)		
13226	060134	004737	060324		JSR	PC,@#GRESET		;RESET DEFAULT PATTERN IN OUTPUT ;BUFFER.
13227								
13228	060140	077330			SOB	R3,G37		
13229								
13230								;TEST ACCUMULATOR 5 WITH FLOATING ZERO
13231	060142	012700	060404		MOV	#GPAT10,R0		
13232	060146	012701	060434		MOV	#GDAT00,R1		
13233	060152	004737	060246		JSR	PC,@#GSETUP		;LOAD TEST PATTERN.
13234	060156	012703	000102		MOV	#102,R3		
13235	060162			G42:				
13236	060162	172410			LDD	(R0),AC0		

13237	060164	174005		STD	AC0,AC5	
13238	060166	172405		LDD	AC5,AC0	:STORE THE TEST PATTERN.
13239	060170	174011		STD	AC0,(R1)	
13240	060172	004737	060344	JSR	PC,@#GCMP	:COMPARE THE DATA READ WITH :THAT WHICH WAS WRITTEN.
13241						
13242	060176	005737	060370	TST	@#GFLAG1	
13243	060202	001004		BNE	G43	
13244	060204	005137	060370	COM	@#GFLAG1	
13245	060210	000241		CLC		
13246	060212	000401		BR	G44	
13247	060214	000261		G43:	SEC	
13248	060216	006160	000006	G44:	ROL 6(R0)	:GENERATE THE NEXT TEST PATTERN.
13249	060212	006160	000004		ROL 4(R0)	
13250	060226	006160	000002		ROL 2(R0)	
13251	060232	006110			ROL (R0)	
13252	060234	004737	060324	JSR	PC,@#GRESET	:RESET DEFAULT PATTERN IN OUTPUT :BUFFER.
13253						
13254	060240	077330		SOB	R3,G42	
13255						
13256						
13257	060242	000137	060444	JMP	@#GDONE	
13258						
13259						:USE THIS ROUTINE TO INITIALIZE ALL THE DATA BUFFERS.
13260	060246	012705	060370	GSETUP:	MOV #GFLAG1,R5	
13261	060252	012704	000026		MOV #26,R4	
13262	060256	005025		1\$:	CLR (R5)+	
13263	060260	077402			SOB R4,1\$	
13264						
13265	060262	012705	060404		MOV #GPAT10,R5	
13266	060266	012704	000010		MOV #10,R4	
13267	060272	005125		2\$:	COM (R5)+	
13268	060274	077402			SOB R4,2\$	
13269						
13270	060276	020067	000072	GS1:	CMP R0,GPAT00	
13271	060302	001401			BEQ 3\$	
13272	060304	000207			RTS PC	
13273						
13274	060306	012705	060434	3\$:	MOV #GDAT00,R5	
13275	060312	012704	000004		MOV #4,R4	
13276	060316	005125		4\$:	COM (R5)+	
13277	060320	077402			SOB R4,4\$	
13278	060322	000207			RTS PC	
13279						
13280	060324	012705	060434	GRESET:	MOV #GDAT00,R5	
13281	060330	012704	000004		MOV #4,R4	
13282	060334	005025		1\$:	CLR (R5)+	
13283	060336	077402			SOB R4,1\$	
13284	060340	000137	060276		JMP @#GS1	
13285						
13286						:SEE IF THE DATA WRITTEN MATCHES THE DATA READ.
13287	060344	012705	060434	GCMP:	MOV #GDAT00,R5	
13288	060350	012704	000004		MOV #4,R4	
13289	060354	010002			MOV R0,R2	
13290	060356	022225		1\$:	CMP (R2)+,(R5)+	
13291	060360	001401			BEQ 2\$	
13292	060362	104000			EMT	:

13293	060364	077404	
13294	060366	000207	
13295			
13296			
13297			
13298			
13299	060370	000000	
13300	060372	000000	
13301			
13302	060374	000000	
13303	060376	000000	
13304	060400	000000	
13305	060402	000000	
13306			
13307	060404	177777	
13308	060406	177777	
13309	060410	177777	
13310	060412	177777	
13311			
13312	060414	177777	
13313	060416	177777	
13314	060420	177777	
13315	060422	177777	
13316			
13317	060424	000000	
13318	060426	000000	
13319	060430	000000	
13320	060432	000000	
13321			
13322	060434	000000	
13323	060436	000000	
13324	060440	000000	
13325	060442	000000	
13326			
13327			
13328	060444		
13329	060444	004767	044122
13330			
13331			
13332			
13333			
13334			
13335			
13336			
13337			
13338			
13339	060450		
13340	060450	005037	061140
13341	060454	012700	061142
13342	060460	012701	061262
13343	060464	012703	000024
13344	060470	012120	
13345	060472	077302	
13346			
13347	060474	004767	000420
13348			

2\$: SOB R4,1\$
RTS PC

GFLAG1: 0
GFLAG2: 0

GPAT00: 0
GPAT01: 0
GPAT02: 0
GPAT03: 0

GPAT10: -1
GPAT11: -1
GPAT12: -1
GPAT13: -1

GAND0: -1
GAND1: -1
GAND2: -1
GAND3: -1

GOR0: 0
GOR1: 0
GOR2: 0
GOR3: 0

GDAT00: 0
GDAT01: 0
GDAT02: 0
GDAT03: 0

GDONE: JSR PC,,RSET

:GO INITIALIZE THE FPS AND STACK; AND
:SEE IF THE USER HAS EXPRESSED
:THE DESIRE TO CHANGE THE SOFTWARE
:VIRTUAL CONSOLE SWITCH REGISTER (HAS
:THE USER TYPED CONTROL G?).

:*****
:TEST 443 FPP ACCUMULATORS DUAL ADDRESS TEST
:*****

TS443:
H1: CLR @HFLAG
MOV #HA1W,RO ;INITIALIZE THE LOAD BUFFER DATA.
MOV #HDAT1,R1
MOV #24,R3

H2: MOV (R1)+,(R0)+
SOB R3,H2

JSR PC,HCLR ;CLEAR THE OUTPUT DATA BUFFER.

13349	060500	170011	
13350			
13351	060502	012700	061142
13352	060506	172410	
13353	060510	174001	
13354			
13355	060512	012700	061152
13356	060516	172410	
13357	060520	174002	
13358			
13359	060522	012700	061162
13360	060526	172410	
13361	060530	174003	
13362			
13363	060532	012700	061172
13364	060536	172410	
13365	060540	174004	
13366			
13367	060542	012700	061202
13368	060546	172410	
13369	060550	174005	
13370			
13371	060552	004737	061006
13372			
13373	060556	004737	061064
13374			
13375			
13376			
13377			
13378	060562	012700	061142
13379	060566	012702	000004
13380	060572	010001	
13381	060574	005121	
13382	060576	172410	
13383	060600	174001	
13384	060602	004737	061006
13385	060606	004737	061064
13386	060612	077210	
13387			
13388			
13389			
13390			
13391	060614	012700	061152
13392	060620	012702	000004
13393	060624	010001	
13394	060626	005121	
13395	060630	172410	
13396	060632	174002	
13397	060634	004737	061006
13398	060640	004737	061064
13399	060644	077210	
13400			
13401			
13402			
13403			
13404	060646	012700	061162

```

H3:  SETD
      ;LOAD ACCUMULATOR 1
      MOV  #HA1W,R0
      LDD  (R0),ACO
      STD  ACO,AC1
      ;LOAD ACCUMULATOR 2
      MOV  #HA2W,R0
      LDD  (R0),ACO
      STD  ACO,AC2
      ;LOAD ACCUMULATOR 3
      MOV  #HA3W,R0
      LDD  (R0),ACO
      STD  ACO,AC3
      ;LOAD ACCUMULATOR 4
      MOV  #HA4W,R0
      LDD  (R0),ACO
      STD  ACO,AC4
      ;LOAD ACCUMULATOR 5
      MOV  #HA5W,R0
      LDD  (R0),ACO
      STD  ACO,AC5

H4:   JSR  PC,@#HSTD           ;GO READ ALL ACCUMULATORS BACK.
      JSR  PC,@#HCMP           ;SEE IF DATA IS CORRECT.

      ;COMPLIMENT EACH WORD OF THE DATA STORED IN ACCUMULATOR 1,
      ;RELOAD THAT ACCUMULATOR, READ ALL THE ACCUMULATORS BACK AND CHECK
      ;THE DATA.
      MOV  #HA1W,R0
      MOV  #4,R2
      MOV  R0,R1
H5:   COM  (R1)+
      LDD  (R0),ACO
      STD  ACO,AC1
      JSR  PC,@#HSTD           ;READ ALL THE ACCUMULATORS BACK.
      JSR  PC,@#HCMP           ;CHECK THE DATA.
      SOB  R2,H5

      ;COMPLIMENT EACH WORD OF THE DATA STORED IN ACCUMULATOR 2,
      ;RELOAD THAT ACCUMULATOR, READ ALL THE ACCUMULATORS BACK AND CHECK
      ;THE DATA.
      MOV  #HA2W,R0
      MOV  #4,R2
      MOV  R0,R1
H6:   COM  (R1)+
      LDD  (R0),ACO
      STD  ACO,AC2
      JSR  PC,@#HSTD           ;READ ALL THE ACCUMULATORS BACK.
      JSR  PC,@#HCMP           ;CHECK THE DATA.
      SOB  R2,H6

      ;COMPLIMENT EACH WORD OF THE DATA STORED IN ACCUMULATOR 3,
      ;RELOAD THAT ACCUMULATOR, READ ALL THE ACCUMULATORS BACK AND CHECK
      ;THE DATA.
      MOV  #HA3W,R0
  
```

13405	060652	012702	000004
13406	060656	010001	
13407	060660	005121	
13408	060662	172410	
13409	060664	174003	
13410	060666	004737	061006
13411	060672	004737	061064
13412	060676	077210	
13413			
13414			
13415			
13416			
13417	060700	012700	061172
13418	060704	012702	000004
13419	060710	010001	
13420	060712	005121	
13421	060714	172410	
13422	060716	174004	
13423	060720	004737	061006
13424	060724	004737	061064
13425	060730	077210	
13426			
13427			
13428			
13429			
13430	060732	012700	061202
13431	060736	012702	000004
13432	060742	010001	
13433	060744	005121	
13434	060746	172410	
13435	060750	174005	
13436	060752	004737	061006
13437	060756	004737	061064
13438	060762	077210	
13439			
13440			
13441	060764	005737	061140
13442	060770	001402	
13443	060772	000137	061332
13444			
13445	060776	005137	061140
13446	061002	000137	060500
13447			
13448			
13449	061006	004737	061120
13450			
13451	061012	012704	061212
13452	061016	172401	
13453	061020	174014	
13454			
13455	061022	012704	061222
13456	061026	172402	
13457	061030	174014	
13458			
13459	061032	012704	061232
13460	061036	172403	

```

MOV #4,R2
MOV R0,R1
H7: COM (R1)+
LDD (R0),ACO
STD ACO,AC3
JSR PC,@#HSTD ;READ ALL THE ACCUMULATORS BACK.
JSR PC,@#HCMP ;CHECK THE DATA.
SOB R2,H7

;COMPLIMENT EACH WORD OF THE DATA STORED IN ACCUMULATOR 4,
;RELOAD THAT ACCUMULATOR, READ ALL THE ACCUMULATORS BACK AND CHECK
;THE DATA.
MOV #HA4W,R0
MOV #4,R2
MOV R0,R1
H10: COM (R1)+
LDD (R0),ACO
STD ACO,AC4
JSR PC,@#HSTD ;READ ALL THE ACCUMULATORS BACK.
JSR PC,@#HCMP ;CHECK THE DATA.
SOB R2,H10

;COMPLIMENT EACH WORD OF THE DATA STORED IN ACCUMULATOR 5,
;RELOAD THAT ACCUMULATOR, READ ALL THE ACCUMULATORS BACK AND CHECK
;THE DATA.
MOV #HASW,R0
MOV #4,R2
MOV R0,R1
H11: COM (R1)+
LDD (R0),ACO
STD ACO,AC5
JSR PC,@#HSTD ;READ ALL THE ACCUMULATORS BACK.
JSR PC,@#HCMP ;CHECK THE DATA.
SOB R2,H11

TST @#HFLAG
BEQ H12
JMP @#HDONE

H12: COM @#HFLAG
JMP @#H3

;STORE ALL ACCUMULATORS IN THE OUTPUT BUFFERS.
HSTD: JSR PC,@#HCLR ;CLEAR ALL OUTPUT BUFFERS.
;STORE ACCUMULATOR 1
MOV #HA1R,R4
LDD AC1,ACO
STD ACO,(R4)
;STORE ACCUMULATOR 2
MOV #HA2R,R4
LDD AC2,ACO
STD ACO,(R4)
;STORE ACCUMULATOR 3
MOV #HA3R,R4
LDD AC3,ACO

```

13461	061040	174014		
13462				
13463	061042	012704	061242	
13464	061046	172404		
13465	061050	174014		
13466				
13467	061052	012704	061252	
13468	061056	172405		
13469	061060	174014		
13470	061062	000207		
13471				
13472				
13473	061064	012637	061136	
13474	061070	012703	061142	
13475	061074	012704	061212	
13476	061100	012705	000024	
13477	061104	022324		
13478	061106	001401		
13479	061110	104000		
13480	061112	077504		
13481	061114	000177	000016	
13482				
13483				
13484	061120	012704	061212	
13485	061124	012705	000024	
13486	061130	005024		
13487	061132	077502		
13488	061134	000207		
13489				
13490	061136	000000		
13491	061140	000000		
13492				
13493	061142	000000	000000	000000
13494	061150	000000		
13495	061152	000000	000000	000000
13496	061160	000000		
13497	061162	000000	000000	000000
13498	061170	000000		
13499	061172	000000	000000	000000
13500	061200	000000		
13501	061202	000000	000000	000000
13502	061210	000000		
13503				
13504	061212	000000	000000	000000
13505	061220	000000		
13506	061222	000000	000000	000000
13507	061230	000000		
13508	061232	000000	000000	000000
13509	061240	000000		
13510	061242	000000	000000	000000
13511	061250	000000		
13512	061252	000000	000000	000000
13513	061260	000000		
13514				
13515	061262	073567	073567	073567
13516	061270	073567		

```

STD     ACO,(R4)
:STORE ACCUMULATOR 4
MOV     #HA4R,R4
LDD     AC4,ACO
STD     ACO,(R4)
:STORE ACCUMULATOR 5
MOV     #HA5R,R4
LDD     AC5,ACO
STD     ACO,(R4)
RTS     PC

:COMPARE DATA LOADED WITH DATA READ.
HCMP:   MOV     (SP)+,@#HADR           ;SAVE RETURN ADDRESS.
MOV     #HA1W,R3
MOV     #HA1R,R4
MOV     #24,R5
HCMP1:  CMP     (R3)+,(R4)+
BEQ     HCMP2
EMT
HCMP2:  SOB     R5,HCMP1
JMP     @HADR

:CLEAR THE DATA OUTPUT BUFFER.
HCLR:   MOV     #HA1R,R4
MOV     #24,R5
HCLR1:  CLR     (R4)+
SOB     R5,HCLR1
RTS     PC

HADR:   0
HFLAG:  0

HA1W:   .WORD  0,0,0,0
HA2W:   .WORD  0,0,0,0
HA3W:   .WORD  0,0,0,0
HA4W:   .WORD  0,0,0,0
HA5W:   .WORD  0,0,0,0

HA1R:   .WORD  0,0,0,0
HA2R:   .WORD  0,0,0,0
HA3R:   .WORD  0,0,0,0
HA4R:   .WORD  0,0,0,0
HA5R:   .WORD  0,0,0,0

HDAT1:  .WORD  73567,73567,73567,73567
  
```

13517 061272 063146 063146 063146 HDAT2: .WORD 63146,63146,63146,63146
13518 061300 063146
13519 061302 010421 010421 010421 HDAT3: .WORD 10421,10421,10421,10421
13520 061310 010421
13521 061312 031463 031463 031463 HDAT4: .WORD 31463,31463,31463,31463
13522 061320 031463
13523 061322 042104 042104 042104 HDAT5: .WORD 42104,42104,42104,42104
13524 061330 042104

13525
13526 061332 HDONE:
13527 061332 004767 043234 JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
13528 ;SEE IF THE USER HAS EXPRESSED
13529 ;THE DESIRE TO CHANGE THE SOFTWARE
13530 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
13531 ;THE USER TYPED CONTROL G?).
13532
13533

13534 ;*****
13535 ;TEST 444 FSRC MODE 0 WITH ILLEGAL ACCUMULATOR TEST
13536 ;*****
13537 TS444:

13537 061336
13538 061336 170011 ;SET FD
13539 061340 012700 061620 MOV #SPAT10,R0 ;LOAD ACO
13540 061344 172410 LDD (R0),ACO
13541
13542 061346 012737 061524 000244 MOV #SERR0,@#FPVECT ;USE OF THE NON-EXISTENT AC-
13543 ;CUMULATOR SHOULD RESULT IN
13544 ;A TRAP TO 244.
13545 061354 012700 000001 MOV #1,R0 ;A FAILURE IN THE FSRC FLOWS
13546 ;WILL RESULT IN AN ODD ADDRESS
13547 061360 012737 061432 000004 MOV #SERR1,@#ERRVECT ;TRAP TO 4.
13548 061366 005003 CLR R3

13549
13550 061370 172407 SX2: LDD AC7,ACO
13551 061372 170000 SX3: CFCC
13552 061374 005203 INC R3
13553 061376 005203 SX4: INC R3
13554
13555 061400 012701 061630 MOV #SDAT00,R1 ;NO TRAP OCCURRED!!
13556 061404 174011 STD ACO,(R1) ;SEE IF ACO WAS MODIFIED.
13557

13558 061406 012701 061630 MOV #SDAT00,R1
13559 061412 012702 061620 MOV #SPAT10,R2
13560 061416 012703 000004 MOV #4,R3
13561 061422 022122 SX5: CMP (R1)+,(R2)+
13562 061424 001401 BEQ SX6
13563 061426 104000 EMT
13564 061430 077304 SX6: SOB R3,SX5 ;
13565 061432 SERR1: SOB R3,SX5 ;
13566 061432 104000 EMT ;
13567

13568 ;NOW TEST AC6.
13569 061434 170011 SX7: SETD
13570 061436 012700 061620 MOV #SPAT10,R0 ;LOAD ACO
13571 061442 172410 LDD (R0),ACO
13572 061444 012737 061574 000244 MOV #SERR4,@#FPVECT

```

13573 061452 012700 000001      MOV    #1,R0
13574 061456 005003              CLR    R3
13575                                SX9:  LDD    AC6,ACO
13576 061460 172406              SX9:  CFCC
13577 061462 170000              INC    R3
13578 061464 005203              SX10: INC    R3
13579 061466 005203
13580                                MOV    #SDAT00,R1
13581 061470 012701 061630      STD    ACO,(R1)          ;NO TRAP! GET ACO.
13582 061474 174011
13583                                MOV    #SDAT00,R1          ;WAS ACO MODIFIED.
13584 061476 012701 061630      MOV    #SPAT10,R2
13585 061502 012702 061620      MOV    #4,R3
13586 061506 012703 000004      SX11: CMP    (R1)+,(R2)+
13587 061512 022122              BEQ    SX12
13588 061514 001401              EMT
13589 061516 104000              SX12: SOB   R3,SX11
13590 061520 077304              EMT
13591 061522 104000
13592
13593      ;TRAPPED TO 244.
13594 061524 021627 061372      SERR0: CMP    (SP),#SX3          ;PC OF TRAP CORRECT?
13595 061530 001401              BEQ    1$
13596 061532 104000              EMT
13597 061534 012737 061434 061614 1$:  MOV    #SX7,@#SADR
13598 061542 022626      SERR10: CMP    (SP)+,(SP)+
13599 061544 005004              CLR    R4
13600 061546 170204              STFPS  R4          ;IS FPS CORRECT?
13601 061550 022704 100200      CMP    #100200,R4
13602 061554 001326              BNE    SERR1
13603
13604 061556 005004              CLR    R4
13605 061560 170304              STST  R4          ;IS FEC CORRECT?
13606 061562 022704 000002      CMP    #2,R4
13607 061566 001321              BNE    SERR1
13608 061570 000177 000020      JMP    @SADR
13609
13610 061574 021627 061462      SERR4: CMP    (SP),#SX9
13611 061600 001401              BEQ    1$
13612 061602 104000              EMT
13613 061604 012737 061640 061614 1$:  MOV    #SDONE,@#SADR
13614 061612 000753              BR    SERR10
13615
13616 061614 000000      SADR:  0
13617 061616 177777              -1
13618 061620 010421      SPAT10: 10421
13619 061622 021042      SPAT11: 21042
13620 061624 031463      SPAT12: 31463
13621 061626 042104      SPAT13: 42104
13622
13623 061630 000000      SDAT00: 0
13624 061632 000000      SDAT01: 0
13625 061634 000000      SDAT02: 0
13626 061636 000000      SDAT03: 0
13627
13628 061640      SDONE:

```

13629 061640 004767 042726
 13630
 13631
 13632
 13633
 13634
 13635
 13636
 13637
 13638 061644
 13639 061644
 13640 061644 170011
 13641
 13642 061646 012700 061774
 13643 061652 172410
 13644
 13645 061654 012700 061754
 13646 061660 005003
 13647
 13648 061662 172420
 13649 061664 005203
 13650 061666 005203
 13651
 13652 061670 012701 061764
 13653 061674 174011
 13654
 13655 061676 020027 061744
 13656 061702 001001
 13657 061704 104000
 13658 061706 012702 061754
 13659 061712 012703 061764
 13660 061716 012704 000004
 13661 061722 022223
 13662 061724 001401
 13663 061726 104000
 13664 061730 077404
 13665
 13666 061732 022700 061764
 13667 061736 001401
 13668 061740 104000
 13669 061742 000420
 13670
 13671 061744 010421
 13672 061746 021042
 13673 061750 042104
 13674 061752 031463
 13675
 13676 061754 052525
 13677 061756 114631
 13678 061760 063146
 13679 061762 073567
 13680
 13681 061764 000000
 13682 061766 000000
 13683 061770 000000
 13684 061772 000000

JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
 ;SEE IF THE USER HAS EXPRESSED
 ;THE DESIRE TO CHANGE THE SOFTWARE
 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
 ;THE USER TYPED CONTROL G?).

 :TEST 445 FSRC MODE 2 TEST
 :*****
 TS445:
 J1: SETD ;SET DOUBLE MODE
 MOV #JDAT0,R0
 LDD (R0),ACO ;LOAD ACO=ALL 1
 MOV #JDATIO,R0
 CLR R3
 J2: LDD (R0)+,ACO ;TEST INSTRUCTION
 J3: INC R3
 J4: INC R3
 MOV #JDAT00,R1
 STD ACO,(R1) ;PICK UP RESULTS
 CMP R0,#JBUF0 ;WAS AN AUTO
 BNE 1\$
 EMT ;
 1\$: MOV #JDATIO,R2 ;IS DATA CORRECT?
 MOV #JDAT00,R3
 MOV #4,R4
 J5: CMP (R2)+,(R3)+
 BEQ J6
 EMT ;
 J6: SOB R4,J5
 CMP #JDATIO+10,R0 ;WAS RO INCREM.
 BEQ J7
 EMT ;
 J7: BR JDONE
 JBUF0: .WORD 010421
 JBUF1: 021042
 JBUF2: 042104
 JBUF3: 031463
 JDATIO: 052525
 JDATIO1: 114631
 JDATIO2: 063146
 JDATIO3: 073567
 JDAT00: 0
 JDAT01: 0
 JDAT02: 0
 JDAT03: 0

13685
13686 061774 177777
13687 061776 177777
13688 062000 177777
13689 062002 177777
13690
13691
13692 062004
13693 062004 004767 042562
13694
13695
13696
13697
13698
13699
13700
13701
13702
13703 062010
13704 062010 170011
13705
13706 062012 012700 062140
13707 062016 172410
13708
13709 062020 012700 062120
13710 062024 005003
13711 062026 172440
13712 062030 005203
13713 062032 005203
13714
13715 062034 012701 062130
13716 062040 174011
13717
13718 062042 020027 062130
13719 062046 001001
13720 062050 104000
13721 062052 012702 062110
13722 062056 012703 062130
13723 062062 012704 000004
13724 062066 022223
13725 062070 001401
13726 062072 104000
13727 062074 077404
13728
13729 062076 022700 062110
13730 062102 001401
13731 062104 104000
13732 062106 000420
13733
13734 062110 052525
13735 062112 114631
13736 062114 063140
13737 062116 073567
13738
13739 062120 010421
13740 062122 031463

JDAT0: -1
JDAT1: -1
JDAT2: -1
JDAT3: -1

JDONE: JSR PC, .RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

:TEST 446 FSRC MODE 4 TEST
:*****
TS446:

SETD ;SET DOUBLE MODE

MOV #KPATO,R0
LDD (R0),ACO ;LOAD A DEFAULT
;PATTERN INTO ACO

MOV #KBUF0,R0
CLR R3
LDD -(R0),ACO ;TEST INSTRUCTION
KX2: LDD R3
KX3: INC R3
KX4: INC R3

MOV #KDAT00,R1
STD ACO,(R1) ;PICK UP THE RESULT

CMP R0,#KBUF0+10 ;WAS AN AUTO
BNE 1\$
; IS DATA CORRECT?
1\$: MOV #KDAT10,R2
MOV #KDAT00,R3
MOV #4,R4
KX5: CMP (R2)+,(R3)+
BEQ KX6
; EMT
KX6: SOB R4,KX5
; WAS R0 DECREMENTED
CMP #KBUF0-10,R0
BEQ KX7
; EMT
KX7: BR KDONE

KDAT10: .WORD 052525
KDAT11: 114631
KDAT12: 063140
KDAT13: 073567

KBUF0: 010421
KBUF1: 031463

13741 062124 042104
 13742 062126 021042
 13743
 13744 062130 000000
 13745 062132 000000
 13746 062134 000000
 13747 062136 000000
 13748
 13749 062140 177777
 13750 062142 177777
 13751 062144 177777
 13752 062146 177777
 13753
 13754 062150
 13755 062150 004767 042416
 13756
 13757
 13758
 13759
 13760
 13761
 13762
 13763
 13764
 13765 062154
 13766 062154
 13767 062154 170011
 13768
 13769 062156 012700 062302
 13770 062162 172410
 13771
 13772 062164 012700 062324
 13773 062170 012701 062312
 13774 062174 012702 000004
 13775
 13776 062200 012120
 13777 062202 077202
 13778
 13779 062204 012700 062324
 13780 062210 005003
 13781 062212 170001
 13782
 13783 062214 172420
 13784 062216 005203
 13785
 13786 062220
 13787 062220 170011
 13788
 13789 062222 012701 062336
 13790 062226 174011
 13791
 13792 062230 020027 062330
 13793 062234 001401
 13794 062236 104000
 13795 062240 012737 177777 062332
 13796 062246 012737 177777 062332

KBUF2: 042104
 KBUF3: 021042
 KDAT00: 0
 KDAT01: 0
 KDAT02: 0
 KDAT03: 0
 KPAT0: -1
 KPAT1: -1
 KPAT2: -1
 DPAT3: -1

KDONE: JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND
 ;SEE IF THE USER HAS EXPRESSED
 ;THE DESIRE TO CHANGE THE SOFTWARE
 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
 ;THE USER TYPED CONTROL G?).

 ;TEST 447 FSRC MODE 2, WITH FD=0, TEST

TS447:
 L1: SETD ;SET DOUBLE MODE
 MOV #LPAT10,R0 ;LOAD ACO
 LDD (R0),AC0 ;LOAD ACO
 MOV #LDAT10,R0 ;SET UP THE INPUT
 MOV #LPAT20,R1 ;DATA
 MOV #4,R2
 1\$: MOV (R1)+,(R0)+
 SOB R2,1\$
 MOV #LDAT10,R0
 CLR R3 ;CLEAR FD.
 SETF
 L2: LDF (R0)+,AC0
 L3: INC R3
 L4: SETD ;SET FD
 MOV #LDAT00,R1 ;PICK UP RESULTS
 STD ACO,(R1)
 CMP R0,#LDAT12 ;WAS R0 INCREMENTED
 BEQ 1\$
 EMT
 1\$: MOV #-1,@#LDAT12
 MOV #-1,@#LDAT13

13797 062254 012702 062324
13798 062260 012703 062336
13799 062264 012704 000004
13800
13801 062270 022223
13802 062272 001401
13803 062274 104000
13804 062276 077404
13805 062300 000422
13806
13807 062302 177777
13808 062304 177777
13809 062306 177777
13810 062310 177777
13811
13812 062312 052525
13813 062314 114631
13814 062316 063142
13815 062320 073567
13816 062322 000001
13817 062324 000000
13818 062326 000000
13819 062330 000000
13820 062332 000000
13821 062334 000001
13822 062336 000000
13823 062340 000000
13824 062342 000000
13825 062344 000000
13826
13827 062346
13828 062346 004767 042220
13829
13830
13831
13832
13833
13834
13835
13836
13837
13838 062352
13839
13840 062352
13841 062352 170011
13842
13843 062354 012700 062454
13844 062360 172410
13845
13846 062362 005004
13847 062364 012737 062412 000004
13848
13849 062372 172427 000000
13850 062374 062374
13851 062374 005204
13852 062376 005204

MOV #LDAT10,R2 ;IS DATA CORRECT
MOV #LDAT00,R3
MOV #4,R4
L5: CMP (R2)+,(R3)+
BEQ L6
EMT ;
L6: SOB R4,L5
BR LDONE
LPAT10: .WORD -1
LPAT11: -1
LPAT12: -1
LPAT13: -1
LPAT20: 052525
LPAT21: 114631
LPAT22: 063142
LPAT23: 073567
.WORD 000001
LDAT10: 0
LDAT11: 0
LDAT12: 0
LDAT13: 0
.WORD 00001
LDAT00: 0
LDAT01: 0
LDAT02: 0
LDAT03: 0
LDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).
;*****
;TEST 450 FSRC MODE 2 WITH GR7, IMMEDIATE MODE, TEST
;*****
TS450:
M1: SETD
MOV #MPAT10,R0
LDD (R0),AC0 ;LOAD BACKGROUND
;PATTERN INTO AC0.
CLR R4
MOV #MERR3,@#ERRVECT
M15: LDD #0,AC0 ;TEST INSTRUCTION
;EFFECTIVELY: 05204 IS PUT IN THE FIRST
;.=-2 ;16 BIT WORD, OR THE 'EXP-FRACTION' WORD.
.WORD 5204
M2: INC R4 ;NOTE THAT

13853 062400 005204
13854 062402 005204
13855 062404 020427 000003
13856 062410 001401
13857 062412
13858 062412 104000
13859 062414 012700 062474
13860 062420 174010
13861
13862 062422 012700 062474
13863 062426 022720 005204
13864 062432 001401
13865 062434 104000
13866 062436 012701 000003
13867 062442 005720
13868 062444 001401
13869 062446 104000
13870 062450 077104
13871 062452 000414
13872
13873 062454 177777
13874 062456 177777
13875 062460 177777
13876 062462 177777
13877
13878 062464 005204
13879 062466 005204
13880 062470 005204
13881 062472 005204
13882
13883 062474 000000
13884 062476 000000
13885 062500 000000
13886 062502 000000
13887
13888 062504
13889 062504 004767 042062
13890
13891
13892
13893
13894
13895
13896
13897
13898
13899 062510
13900
13901 062510
13902 062510 170011
13903
13904 062512 012700 062640
13905 062516 172410
13906
13907 062520 012700 062626
13908 062524 005003

M3: INC R4 :005204=INC R4
M4: INC R4
CMP R4,#3 :SEE IF THE PC
BEQ M8
MERR3:
M8: EMT ;
MOV #MDAT00,R0 ;
STD ACO,(R0) :GET THE DATA
MOV #MDAT00,R0 ;
CMP #5204,(R0)+ :IS THE DATA CORRECT?
BEQ M5 ;
EMT ;
M5: MOV #3,R1 ;
M6: TST (R0)+ ;
BEQ M7 ;
EMT ;
M7: SOB R1,M6 ;
BR MDONE
MPAT10: -1
MPAT11: -1
MPAT12: -1
MPAT13: -1
MPAT20: 5204
MPAT21: 5204
MPAT22: 5204
MPAT23: 5204
MDAT00: 0
MDAT01: 0
MDAT02: 0
MDAT03: 0
MDONE: JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;TEST 451 FSRC MODE 3 TEST

TS451:

N1: SETD ;SET FD MODE
MOV #NPAT10,R0 ;LOAD ACO WITH A DEFAULT
LDD (R0),ACO ;PATTERN
MOV #NPAT20,R0
CLR R3

```
13909 062526 012737 062600 000004      MOV      #NERRO,@#ERRVECT      ;IF A FAILURE OCCURS
13910                                     ;IN THE FSRC FLOWS AN
13911                                     ;ODD TRAP TO 4 COULD OCCUR
13912 062534 172430      N2:      LDD      @(R0)+,ACO      ;TEST INSTRUCTION.
13913 062536 005203      N3:      INC      R3
13914 062540 005203      N4:      INC      R3
13915
13916 062542 012701 062606      MOV      #NDAT00,R1
13917 062546 174011      STD      ACO,(R1)      ;GET THE DATA
13918
13919 062550 020027 062630      CMP      R0,#NPAT20+2      ;WAS R0 INCREMENTED
13920 062554 001401      BEQ      N12
13921 062556 104000      EMT
13922 062560 012702 062606      N12:     MOV      #NDAT00,R2      ;DATA CORRECT
13923 062564 012703 062650      MOV      #NDAT10,R3
13924 062570 012704 000004      MOV      #4,R4
13925 062574 022223      N13:     CMP      (R2)+,(R3)+
13926 062576 001401      BEQ      N14
13927 062600      NERRO:
13928 062600 104000      EMT
13929 062602 077404      N14:     SOB      R4,N13
13930 062604 000425      BR       NDONE
13931
13932 062606 000000      NDAT00:  .WORD   0
13933 062610 000000      NDAT01:  .WORD   0
13934 062612 000000      NDAT02:  .WORD   0
13935 062614 000000      NDAT03:  .WORD   0
13936
13937 062616 052525 052525 052525      .WORD   52525,52525,52525,52525
13938 062624 052525
13939 062626 062650      NPAT20:  .WORD   NDAT10
13940 062630 070707      NPAT21:  .WORD   070707
13941 062632 070707      NPAT22:  .WORD   070707
13942 062634 070707      NPAT23:  .WORD   070707
13943 062636 000001      .WORD   1
13944 062640 177777      NPAT10:  .WORD  -1
13945 062642 177777      NPAT11:  .WORD  -1
13946 062644 177777      NPAT12:  .WORD  -1
13947 062646 177777      NPAT13:  .WORD  -1
13948
13949 062650 010421      NDAT10:  .WORD   010421
13950 062652 021042      NDAT11:  .WORD   021042
13951 062654 031463      NDAT12:  .WORD   031463
13952 062656 042104      NDAT13:  .WORD   042104
13953
13954 062660      NDONE:
13955 062660 004767 041706      JSR      PC,,RSET      ;GO INITIALIZE THE FPS AND STACK; AND
13956                                     ;SEE IF THE USER HAS EXPRESSED
13957                                     ;THE DESIRE TO CHANGE THE SOFTWARE
13958                                     ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
13959                                     ;THE USER TYPED CONTROL G?).
13960
13961
13962
13963
13964
```

```

13965 062664
13966
13967 062664
13968 062664 170011
13969
13970 062666 012700 063014
13971 062672 172410
13972
13973 062674 012700 063002
13974 062700 005003
13975 062702 012737 062732 000004
13976
13977
13978
13979 062710 172450
13980 062712 005203
13981 062714 005203
13982
13983 062716 012701 062762
13984 062722 174011
13985
13986 062724 020027 063000
13987 062730 001401
13988 062732
13989 062732 104000
13990 062734 012702 062762
13991 062740 012703 063024
13992 062744 012704 000004
13993 062750 022223
13994 062752 001401
13995 062754 104000
13996 062756 077404
13997 062760 000425
13998
13999 062762 000000
14000 062764 000000
14001 062766 000000
14002 062770 000000
14003
14004 062772 052525 052525 052525
14005 063000 063024
14006 063002 070707
14007 063004 070707
14008 063006 070707
14009 063010 070707
14010 063012 000001
14011 063014 177777
14012 063016 177777
14013 063020 177777
14014 063022 177777
14015
14016 063024 073567
14017 063026 004210
14018 063030 114631
14019 063032 125252
14020

TS452:
01:      SETD                      ;SET FD MODE
        MOV      #OPAT10,R0
        LDD      (R0),AC0          ;LOAD ACO WITH A
                                          ;DEFAULT PATTERN.
        MOV      #OPAT21,R0
        CLR      R3
        MOV      #OERR0,@#ERRVEC  ;IF A FAILURE
                                          ;OCCURS IN THE FSRC
                                          ;FLOWS AN ODD ADDR.
                                          ;TRAP TO 4 MAY OCCUR.
02:      LDD      @-(R0),AC0      ;TEST INSTRUCTION
03:      INC      R3
04:      INC      R3
        MOV      #ODAT00,R1
        STD      ACO,(R1)         ;GET THE DATA
        CMP      R0,#OPAT20
        BEQ      012              ;WAS R0 DECREMENTED
OERR0:
012:     EMT
        MOV      #ODAT00,R2        ;
        MOV      #ODAT10,R3        ;DATA CORRECT?
        MOV      #4,R4
013:     CMP      (R2)+,(R3)+
        BEQ      014
        EMT
014:     SOB     R4,013
        BR      ODONE
ODAT00: .WORD 0
ODAT01: .WORD 0
ODAT02: .WORD 0
ODAT03: .WORD 0
OPAT20: .WORD 52525,52525,52525
OPAT10: .WORD 0DAT10
OPAT21: 070707
OPAT22: 070707
OPAT23: 070707
OPAT24: 070707
        .WORD 1
OPAT10: .WORD -1
OPAT11: .WORD -1
OPAT12: .WORD -1
OPAT13: .WORD -1
ODAT10: .WORD 73567
ODAT11: .WORD 004210
ODAT12: .WORD 114631
ODAT13: .WORD 125252
  
```

14021 063034
14022 063034 004767 041532
14023
14024
14025
14026
14027
14028
14029
14030
14031
14032 063040
14033
14034 063040
14035 063040 170011
14036
14037 063042 012700 063126
14038 063046 172410
14039
14040 063050 012700 062675
14041
14042 063054 172460 000241
14043 063056
14044
14045 063060 012701 063146
14046 063064 174011
14047 063066 012703 000004
14048 063072 012702 063136
14049 063076 012701 063146
14050 063102 022221
14051 063104 001401
14052 063106 104000
14053 063110 077304
14054 063112 022700 062675
14055 063116 001401
14056 063120 104000
14057 063122 000137 063156
14058 063126 177777
14059 063130 177777
14060 063132 177777
14061 063134 177777
14062
14063 063136 010421
14064 063140 031463
14065 063142 052525
14066 063144 073567
14067
14068 063146 000000
14069 063150 000000
14070 063152 000000
14071 063154 000000
14072
14073 063156
14074 063156 004767 041410
14075
14076

ODONE: JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

:TEST 453 FSRC MODE 6 TEST

TS453:

P1: SETD ;SET FD MODE
MOV #PPAT10,R0
LDD (R0),AC0 ;LOAD A DEFAULT PATTERN
;INTO AC0
;COULD OCCUR.
MOV #PDAT10-241,R0

P2: LDD 241(R0),AC0
P3=P2+2

P4: MOV #PDAT00,R1
STD AC0,(R1) ;GET THE DATA
MOV #4,R3
MOV #PDAT10,R2

P5: MOV #PDAT00,R1
CMP (R2)+,(R1)+ ;CHECK THE DATA
BEQ 2\$

2\$: SOB R3,P5
CMP #PDAT10-241,R0 ;RO CORRECT?
BEQ 1\$

1\$: JMP @MPDONE
PPAT10: .WORD -1
PPAT11: -1
PPAT12: -1
PPAT13: -1

PDAT10: .WORD 010421
PDAT11: 031463
PDAT12: 052525
PDAT13: 073567

PDAT00: .WORD 0
PDAT01: 0
PDAT02: 0
PDAT03: 0

PDONE: JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE

;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

:TEST 454 FSRC MODE 7 TEST

TS454:

Q1: SETD
MOV #QPAT10,R0
LDD (R0),AC0 ;LOAD A DEFAULT
;PATTERN INTO ACO
MOV #QPAT20-241,R0
Q2: LDD @241(R0),AC0
Q3=Q2+2
Q4: MOV #QDAT00,R1
STD ACO,(R1) ;GET THE DATA
MOV #4,R3
MOV #QDAT00,R4
MOV #QDAT10,R5
Q5: CMP (R4)+,(R5)+ ;CHECK THE DATA
BEQ 2\$
EIT
2\$: SOB R3,Q5 ;
CMP #QPAT20-241,R0 ;CHECK R0.
BEQ 1\$
EIT
1\$: JMP @#QDONE ;

QPAT10: .WORD -1
QPAT11: -1
QPAT12: -1
QPAT13: -1
QPAT20: .WORD QDAT10
QPAT21: 52525
QPAT22: 52525
QPAT23: 52525
QDAT00: .WORD 0
QDAT01: 0
QDAT02: 0
QDAT03: 0
QDAT10: .WORD 073567
QDAT11: .WORD 052525
QDAT12: .WORD 031463
QDAT13: .WORD 010421

14077
14078
14079
14080
14081
14082
14083
14084 063162
14085
14086 063162
14087 063162 170011
14088
14089 063164 012700 063250
14090 063170 172410
14091
14092 063172 012700 063017
14093
14094 063176 172470 000241
14095 063200
14096
14097 063202 012701 063270
14098 063206 174011
14099
14100 063210 012703 000004
14101 063214 012704 063270
14102 063220 012705 063300
14103 063224 022425
14104 063226 001401
14105 063230 104000
14106 063232 077304
14107
14108 063234 022700 063017
14109 063240 001401
14110 063242 104000
14111 063244 000137 063310
14112
14113 063250 177777
14114 063252 177777
14115 063254 177777
14116 063256 177777
14117
14118 063260 063300
14119 063262 052525
14120 063264 052525
14121 063266 052525
14122
14123 063270 000000
14124 063272 000000
14125 063274 000000
14126 063276 000000
14127
14128 063300 073567
14129 063302 052525
14130 063304 031463
14131 063306 010421
14132

14133 063310
14134 063310 004767 041256
14135
14136
14137
14138
14139
14140
14141
14142
14143 063314
14144 063314 005037 064066
14145 063320 012700 064016
14146 063324 012701 000004
14147 063330 012720 177777
14148 063334 077103
14149
14150 063336 012737 000033 064070
14151 063344 012737 000023 064072
14152 063352 012737 063436 000244
14153 063360 012700 000200
14154 063364 170100
14155 063366 012700 064016
14156 063372 172410
14157 063374 013737 064070 064074
14158 063402 012737 000001 064076
14159 063410 012737 000254 064100
14160
14161 063416 012700 064026
14162 063422 172410
14163 063424 012704 000204
14164 063430 170205
14165
14166 063432 020405
14167 063434 001401
14168 063436
14169 063436 104000
14170 063440 012700 000200
14171 063444 170100
14172
14173 063446 012700 064016
14174 063452 172410
14175 063454 013737 064072 064074
14176 063462 012737 000003 064076
14177 063470 012737 000054 064100
14178
14179 063476 012700 064036
14180
14181 063502 172410
14182 063504 012704 000200
14183 063510 170205
14184
14185 063512 020405
14186 063514 001401
14187 063516 104000
14188 063520 012700 000200

QDONE: JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).
:*****
;TEST 455 (BUT EZBT Y8),(BUT ENBT) AND (BUT FIUV) TEST
:*****
TS455:
 CLR @#UFLAG
 MOV #UPAT00,R0 ;SET UP AC#0 DATA.
 MOV #4,R1
 U0: MOV #-1,(R0)+
 SOB R1,U0
 MOV #033,@#UTMP1
 MOV #023,@#UTMP2
 U1: MOV #UERR0,@#FPVECT ;IN CASE (BUT FIUV FAILS)
 MOV #200,R0
 LDFPS R0
 MOV #UPAT00,R0 ;LOAD ACO
 LDD (R0),AC0
 MOV @#UTMP1,@#UROM1
 MOV #001,@#UROM2
 MOV #254,@#UROM3
 U2: MOV #UPAT10,R0 ;LOAD 0 INTO ACO
 LDD (R0),AC0
 MOV #204,R4 ;SEE IF FPS IS CORRECT
 STFPS R5
 CMP R4,R5
 BEQ U3
 UERR0:
 U3: EMT ;
 MOV #200,R0
 LDFPS R0
 MOV #UPAT00,R0 ;LOAD ACO
 LDD (R0),AC0
 MOV @#UTMP2,@#UROM1
 MOV #003,@#UROM2
 MOV #054,@#UROM3
 U4: MOV #UPAT20,R0 ;LOAD A POSITIVE NUMBER
 ;INTO ACO
 LDD (R0),AC0
 MOV #200,R4 ;FPS CORRECT?
 STFPS R5
 CMP R4,R5
 BEQ ;
 EMT ;
 U5: MOV #200,R0

14189	063524	170100			LDFPS	R0		
14190	063526	012700	064016		MOV	#UPAT00,R0	:LOAD ACO	
14191	063532	172410			LDD	(R0),AC0		
14192	063534	013737	064072	064074	MOV	@#UTMP2,@#UROM1		
14193	063542	012737	000403	064076	MOV	#403,@#UROM2		
14194	063550	012737	000056	064100	MOV	#056,@#UROM3		
14195	063556	012700	064046		MOV	#UPAT30,R0	:LOAD A NEGATIVE :NUMBER INTO ALO	
14196								
14197	063562	172410			U6:	LDD	(R0),AC0	
14198	063564	012704	000210		MOV	#210,R4	:FPS CORRECT	
14199	063570	170205			STFPS	R5		
14200	063572	020405			CMP	R4,R5		
14201	063574	001401			BEQ	U7		
14202	063576	104000			EMT		:	
14203	063600	012700	000200		U7:	MOV	#200,R0	
14204	063604	170100			LDFPS	R0		
14205	063606	012700	064016		MOV	#UPAT00,R0	:LOAD ACO	
14206	063612	172410			LDD	(R0),AC0		
14207	063614	013737	064070	064074	MOV	@#UTMP1,@#UROM1		
14208	063622	012737	000401	064076	MOV	#401,@#UROM2		
14209	063630	012737	000256	064100	MOV	#256,@#UROM3		
14210	063636	012700	064056		MOV	#UPAT40,R0	:LOAD -0 INTO ACO	
14211	063642	172410			U10:	LDD	(R0),AC0	
14212	063644	000240			U11:	NOP	:TRAP FROM HERE IF	
14213	063646	012704	000214		MOV	#214,R4	:SEE IF FPS IS CORRECT.	
14214	063652	170205			STFPS	R5		
14215	063654	020405			CMP	R4,R5		
14216	063656	001401			BEQ	U12		
14217	063660	104000			EMT		:	
14218	063662	005737	064066		U12:	TST	@#UFLAG ;SEE IF ALL THE PATTERNS	
14219	063666	001021			BNE	U14	:HAVE BEEN TEST WITH	
14220							:BOTH AC NOT EQUAL TO 0 AND AC=0	
14221	063670	012700	064016		MOV	#UPAT00,R0	:IF NOT GO BACK AND	
14222	063674	012701	000004		MOV	#4,R1	:CHECK THEM WITH AC=0	
14223	063700	005020			U13:	CLR	(R0)+	
14224	063702	077102			SQB	R1,U13		
14225	063704	012737	177777	064066	MOV	#-1,@#UFLAG		
14226	063712	012737	000233	064070	MOV	#233,@#UTMP1		
14227	063720	012737	000223	064072	MOV	#223,@#UTMP2		
14228	063726	000137	063352		JMP	@#U1		
14229								
14230	063732	012737	063770	000244	U14:	MOV	#UERR3,@#FPVECT	:NOW SEE IF A TRAP CAN BE FORCED BY SETTING FIUV AND LOADING -0
14231	063740	012700	004200		MOV	#4200,R0	:SET FD AND FIUV	
14232	063744	170100			LDFPS	R0		
14233	063746	012700	064016		MOV	#UPAT00,R0	:SET UP ACO	
14234	063752	172410			LDD	(R0),AC0		
14235	063754	012700	064056		MOV	#UPAT40,R0	:LOAD -0	
14236	063760	172410			U15:	LDD	(R0),AC0	:SHOULD TRAP TO 244
14237	063762	170000			U16:	CFCC		
14238	063764	000240			NOP			
14239	063766	104000			EMT		:	
14240								
14241								
14242	063770	021627	063762		UERR3:	CMP	(SP),#U16	: INTERRUPT HERE WHEN FIUV SET AND ATTEMPTED TO LOAD-0
14243	063774	001401			BEQ	1\$		
14244	063776	104000			EMT		:	

14245	064000	022626	
14246	064002	005000	
14247	064004	170300	
14248	064006	022700	000014
14249	064012	001433	
14250	064014	104000	
14251	064016	000000	
14252	064020	000000	
14253	064022	000000	
14254	064024	000000	
14255			
14256	064026	000000	
14257	064030	000000	
14258	064032	000000	
14259	064034	000000	
14260			
14261	064036	010421	
14262	064040	114631	
14263	064042	125252	
14264	064044	177777	
14265			
14266	064046	114631	
14267	064050	135673	
14268	064052	146314	
14269	064054	167356	
14270			
14271	064056	100000	
14272	064060	000000	
14273	064062	000000	
14274	064064	000000	
14275			
14276	064066	000000	
14277	064070	000000	
14278	064072	000000	
14279	064074	000000	
14280	064076	000000	
14281	064100	000000	
14282	064102		
14283			
14284			
14285			
14286			
14287			
14288	064102		
14289	064102	012700	000200
14290	064106	170100	
14291	064110	012700	064432
14292	064114	172410	
14293	064116	012700	064432
14294	064122	172010	
14295	064124	170205	

```

1S:    CMP      (SP)+,(SP)+
        CLR      R0
        STST    R0          ;GET FEC.
        CMP     #14,R0      ;CORRECT
        BEQ     UDONE
        EMT
UPAT00: .WORD   0
UPAT01:      0
UPAT02:      0
UPAT03:      0
UPAT10: .WORD   0          ;0
UPAT11:      0
UPAT12:      0
UPAT13:      0
UPAT20: .WORD  010421     ;POS NUM
UPAT21:      114631
UPAT22:      125252
UPAT23:      177777
UPAT30:      114631     ;NEG NUM
UPAT31:      135673
UPAT32:      146314
UPAT33:      167356
UPAT40:      100000     ;NEG ZERO
UPAT41:      0
UPAT42:      0
UPAT43:      0
UFLAG: .WORD   0
UTMP1:      0
UTMP2:      0
UROM1:      0
UROM2:      0
UROM3:      0
UDONE:

:*****
:TEST 456      ADDF,ADD, SUBF AND SUBD WITH FSRC=AC=0 TEST
:*****
TS456:
        MOV     #200,R0
        LDFPS  R0          ;SET DOUBLE MODE
        MOV     #WPAT00,R0 ;LOAD AC0=
        LDD    (R0),AC0
        MOV     #WPAT00,R0
W2:    ADD     (R0),AC0     ;TEST INSTRUCTION. ADD ITSELF
        STFPS  R5          ;GET FPS
  
```

14296	064126	170011		SETD		;SET DOUBLE MODE
14297	064130	012700	064432	MOV	#WPAT00,R0	
14298	064134	174010		STD	AC0,(R0)	;GET THE RESULT
14299	064136	012701	064432	MOV	#WPAT00,R1	
14300	064142	012702	000004	MOV	#4,R2	
14301	064146	022021		W3: CMP	(R0)+,(R1)+	;IS RESULT CORRECT
14302	064150	001401		BEQ	W4	
14303	064152	104000		EMT		;
14304	064154	077204		W4: SOB	R2,W3	
14305	064156	022705	000204	CMP	#204,R5	;IS FPS CORRECT
14306	064162	001401		BEQ	W5	
14307	064164	104000		EMT		;
14308	064166	012700	000200	W5: MOV	#200,R0	
14309	064172	170100		LDFPS	R0	;SET DOUBLE MODE
14310	064174	012700	064432	MOV	#WPAT00,R0	;LOAD AC0=0
14311	064200	172410		LDD	(R0),AC0	
14312	064202	005000		CLR	R0	
14313	064204	170100		LDFPS	R0	;GO TO FLOATING MODE
14314	064206	012700	064432	W6: MOV	#WPAT00,R0	
14315	064212	172010		ADDF	(R0),AC0	;TEST INSTRUCTION
14316	064214	170205		STFPS	R5	;GET FPS
14317	064216	170011		SETD		;RESET TO DOUBLE MODE
14318	064220	012700	064432	MOV	#WPAT00,R0	
14319	064224	174010		STD	AC0,(R0)	;GET THE RESULT
14320	064226	012701	064432	MOV	#WPAT00,R1	
14321	064232	012702	000004	MOV	#4,R2	
14322	064236	022021		W7: CMP	(R0)+,(R1)+	;WAS THE RESULT
14323	064240	001401		BEQ	W10	
14324	064242	104000		EMT		;
14325	064244	077204		W10: SOB	R2,W7	
14326	064246	022705	000004	CMP	#4,R5	;WAS FPS CORRECT
14327	064252	001401		BEQ	W11	
14328	064254	104000		EMT		;
14329	064256	012700	000200	W11: MOV	#200,R0	
14330	064262	170100		LDFPS	R0	;SET DOUBLE MODE
14331	064264	012700	064432	MOV	#WPAT00,R0	;LOAD AC0=0
14332	064270	172410		LDD	(R0),AC0	
14333	064272	012700	064432	W12: MOV	#WPAT00,R0	
14334	064276	173010		SUBD	(R0),AC0	;TEST INSTRUCTION
14335	064300	170205		STFPS	R5	;GET FPS
14336	064302	170011		SETD		;SET DOUBLE MODE
14337	064304	012700	064432	MOV	#WPAT00,R0	
14338	064310	174010		STD	AC0,(R0)	;GET THE RESULT
14339	064312	012701	064432	MOV	#WPAT00,R1	
14340	064316	012702	000004	MOV	#4,R2	
14341	064322	022021		W13: CMP	(R0)+,(R1)+	;IS RESULT CORRECT?
14342	064324	001401		BEQ	W14	
14343	064326	104000		EMT		;
14344	064330	077204		W14: SOB	R2,W13	
14345	064332	022705	000204	CMP	#204,R5	;IS FPS CORRECT?
14346	064336	001401		BEQ	W15	
14347	064340	104000		EMT		;
14348	064342	012700	000200	W15: MOV	#200,R0	
14349	064346	170100		LDFPS	R0	;SET DOUBLE MODE
14350	064350	012700	064432	MOV	#WPAT00,R0	;LOAD AC0=0
14351	064354	172410		LDD	(R0),AC0	

14352	064356	005000	
14353	064360	170100	
14354	064362	012700	064432
14355	064366	173010	
14356	064370	170205	
14357	064372	170011	
14358	064374	012700	064432
14359	064400	174010	
14360	064402	012701	064432
14361	064406	012702	000004
14362	064412	022021	
14363	064414	001401	
14364	064416	104000	
14365	064420	077204	
14366	064422	022705	000004
14367	064426	001411	
14368	064430	104000	
14369			
14370	064432	000000	
14371	064434	000000	
14372	064436	000000	
14373	064440	000000	
14374			
14375	064442	000000	
14376	064444	000000	
14377	064446	000000	
14378	064450	000000	
14379			
14380	064452		
14381	064452	004767	040114
14382			
14383			
14384			
14385			
14386			
14387			
14388			
14389			
14390			
14391	064456		
14392	064456	012700	000200
14393	064462	170100	
14394	064464	012700	065016
14395	064470	172410	
14396	064472	012700	065026
14397	064476	172010	
14398	064500	170205	
14399	064502	170011	
14400	064504	012700	065006
14401	064510	174010	
14402	064512	012701	065016
14403	064516	012702	000004
14404	064522	022021	
14405	064524	001401	
14406	064526	104000	
14407	064530	077204	

```

CLR R0
LDFPS R0 ;ENTER FLOATING MODE.
MOV #WPAT00,R0
W16: SUBF (R0),AC0 ;TEST INSTRUCTION.
STFPS R5 ;GET FPS
SETD ;RESET TO DOUBLE MODE
MOV #WPAT00,R0 ;GET THE RESULT.
STD AC0,(R0)
MOV #WPAT00,R1
W17: MOV #4,R2
CMP (R0)+,(R1)+ ;IS RESULT CORRECT?
BEQ W20
EMT ;
W20: SOB R2,W17
CMP #4,R5 ;IS FPS CORRECT?
BEQ WDONE
EMT ;

WPAT00: .WORD 0
WPAT01: .WORD 0
WPAT02: .WORD 0
WPAT03: .WORD 0

WDA00: .WORD 0
WDA01: .WORD 0
WDA02: .WORD 0
WDA03: .WORD 0

WDONE: JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

```

```

:*****
:TEST 457 ADD AND SUB WITH FSRC=0
:*****
TS457:

```

```

MOV #200,R0
LDFPS R0 ;SET DOUBLE MODE
MOV #XPAT00,R0 ;SET AC0 TO POSITIVE
LDD (R0),AC0
W2: MOV #XPAT10,R0 ;FSRC=0
ADDD (R0),AC0 ;TEST INSTRUCTION
STFPS R5
SETD
MOV #XDAT00,R0 ;GET RESULT.
STD AC0,(R0)
MOV #XPAT00,R1
W3: MOV #4,R2
CMP (R0)+,(R1)+ ;IS RESULT CORRECT?
BEQ X4
EMT ;
W4: SOB R2,X3

```

14408	064532	012704	000200		MOV	#200,R4	
14409	064536	020405			CMP	R4,R5	;IS FPS CORRECT?
14410	064540	001401			BEQ	X5	
14411	064542	104000			EMT		;
14412	064544	012700	000200	X5:	MOV	#200,R0	
14413	064550	170100			LDFPS	R0	;SET DOUBLE MODE
14414	064552	012700	065036		MOV	#XPAT20,R0	;SET ACO TO
14415	064556	172410			LDD	(R0),AC0	
14416	064560	012700	065026		MOV	#XPAT10,R0	;FSRC=0
14417	064564	172010		X6:	ADDD	(R0),AC0	;TEST INSTRUCTION
14418	064566	170205			STFPS	R5	
14419	064570	170011			SETD		
14420	064572	012700	065006		MOV	#XDAT00,R0	;GET RESULT
14421	064576	174010			STD	AC0,(R0)	
14422	064600	012701	065036		MOV	#XPAT20,R1	
14423	064604	012702	000004		MOV	#4,R2	
14424	064610	022021		X7:	CMP	(R0)+,(R1)+	;IS RESULT CORRECT?
14425	064612	001401			BEQ	X10	
14426	064614	104000			EMT		;
14427	064616	077204		X10:	SOB	R2,X7	
14428	064620	012704	000210		MOV	#210,R4	
14429	064624	020405			CMP	R4,R5	;IS FPS CORRECT?
14430	064626	001401			BEQ	X11	
14431	064630	104000			EMT		;
14432	064632	012700	000200	X11:	MOV	#200,R0	
14433	064636	170100			LDFPS	R0	;SET DOUBLE MODE
14434	064640	012700	065016		MOV	#XPAT00,R0	;SET ACO TO NON-ZERO
14435	064644	172410			LDD	(R0),AC0	
14436	064646	012700	065026		MOV	#XPAT10,R0	;FSRC=0
14437	064652	173010		X12:	SUBD	(R0),AC0	;TEST INSTRUCTION
14438	064654	170205			STFPS	R5	
14439	064656	170011			SETD		
14440	064660	012700	065006		MOV	#XDAT00,R0	;GET RESULT
14441	064664	174010			STD	AC0,(R0)	
14442	064666	012701	065016		MOV	#XPAT00,R1	
14443	064672	012702	000004		MOV	#4,R2	
14444	064676	022021		X13:	CMP	(R0)+,(R1)+	;IS RESULT CORRECT?
14445	064700	001401			BEQ	X14	
14446	064702	104000			EMT		;
14447	064704	077204		X14:	SOB	R2,X13	
14448	064706	012704	000200		MOV	#200,R4	;IS FPS CORRECT?
14449	064712	020405			CMP	R4,R5	
14450	064714	001401			BEQ	X15	
14451	064716	104000			EMT		;
14452	064720	012700	000200	X15:	MOV	#200,R0	
14453	064724	170100			LDFPS	R0	;SET DOUBLE MODE
14454	064726	012700	065036		MOV	#XPAT20,R0	;SET ACO=A NEGATIVE
14455	064732	172410			LDD	(R0),AC0	
14456	064734	012700	065026		MOV	#XPAT10,R0	;FSRC=0
14457	064740	173010		X16:	SUBD	(R0),AC0	;TEST INSTRUCTION.
14458	064742	170205			STFPS	R5	
14459	064744	170011			SETD		
14460	064746	012700	065006		MOV	#XDAT00,R0	;GET RESULT
14461	064752	174010			STD	AC0,(R0)	
14462	064754	012701	065036		MOV	#XPAT20,R1	
14463	064760	012702	000004		MOV	#4,R2	

14464 064764 022021
 14465 064766 001401
 14466 064770 104000
 14467 064772 077204
 14468 064774 012704 000210
 14469 065000 020405
 14470 065002 001421
 14471 065004 104000
 14472 065006 000000
 14473 065010 000000
 14474 065012 000000
 14475 065014 000000
 14476
 14477 065016 010421
 14478 065020 021042
 14479 065022 031463
 14480 065024 042104
 14481
 14482 065026 000000
 14483 065030 000000
 14484 065032 000000
 14485 065034 000000
 14486 065036 104210
 14487 065040 114631
 14488 065042 125252
 14489 065044 135673
 14490
 14491 065046
 14492 065046 004767 037520
 14493
 14494
 14495
 14496
 14497
 14498
 14499
 14500
 14501 065052
 14502 065052 005037 065226
 14503 065056 012737 065246 065230
 14504 065064 012737 065256 065232
 14505 065072 012737 000210 065234
 14506 065100 012700 000200
 14507 065104 170100
 14508 065106 012700 065266
 14509 065112 172410
 14510 065114 013700 065230
 14511 065120 173010
 14512 065122 170205
 14513 065124 170011
 14514 065126 012700 065236
 14515 065132 174010
 14516 065134 012702 000004
 14517 065140 013701 065232
 14518 065144 022021
 14519 065146 001401

```

X17:  CMP      (R0)+,(R1)+      ;IS RESULT CORRECT?
      BEQ      X20
      EMT
X20:  SOB      R2,X17
      MOV      #210,R4          ;IS FPS CORRECT?
      CMP      R4,R5
      BEQ      XDONE
      EMT
XDAT00: .WORD 0
XDAT01: 0
XDAT02: 0
XDAT03: 0
XPAT00: .WORD 010421
XPAT01: 021042
XPAT02: 031463
XPAT03: 042104
XPAT10: .WORD 0
XAPT11: 0
XPAT12: 0
XPAT13: 0
XPAT20: .WORD 104210
XPAT21: 114631
XPAT22: 125252
XPAT23: 135673
XDONE: JSR      PC,,RSET      ;GO INITIALIZE THE FPS AND STACK; AND
      ;SEE IF THE USER HAS EXPRESSED
      ;THE DESIRE TO CHANGE THE SOFTWARE
      ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
      ;THE USER TYPED CONTROL G?).

;*****
;TEST 460 SUBD WITH AC=0 TEST
;*****
TS460: CLR      @#YFLAG
      MOV      #YPAT00,@#YTMP1 ;P
      MOV      #YPAT10,@#YTMP2 ;N
      MOV      #210,@#YTMP3
Y1:   MOV      #200,R0
      LDFPS   R0              ;SET DOUBLE MODE
      MOV      #YPAT20,R0     ;SET ACO=0
      LDD     (R0),AC0
      MOV      @#YTMP1,R0
Y2:   SUBD   (R0),AC0        ;TEST INSTRUCTION
      STFPS  R5
      SETD
      MOV      #YDAT00,R0     ;GET RESULT
      STD     ACO,(R0)
      MOV      #4,R2
      MOV      @#YTMP2,R1
Y3:   MOV      @#YTMP2,R1
      CMP      (R0)+,(R1)+
      BEQ      1$
    
```

14520 065150 104000
14521 065152 077204
14522 065154 023705 065234
14523 065160 001401
14524 065162 104000
14525 065164 005737 065226
14526 065170 001015
14527 065172 012737 177777 065226
14528 065200 012737 065256 065230
14529 065206 012737 065246 065232
14530 065214 012737 000200 065234
14531 065222 000726
14532 065224 000424
14533
14534 065226 000000
14535 065230 000000
14536 065232 000000
14537 065234 000000
14538
14539 065236 000000
14540 065240 000000
14541 065242 000000
14542 065244 000000
14543
14544 065246 063146
14545 065250 052525
14546 065252 042104
14547 065254 167356
14548
14549 065256 163146
14550 065260 052525
14551 065262 042104
14552 065264 167356
14553
14554 065266 000000
14555 065270 000000
14556 065272 000000
14557 065274 000000
14558
14559 065276
14560 065276 004767 037270
14561
14562
14563
14564
14565
14566
14567
14568 065302
14569 065302 005067 000134
14570 065306 012737 065460 065444
14571 065314 012737 000200 065446
14572 065322 012700 000200
14573 065326 170100
14574 065330 012700 065500
14575 065334 172410

```
EMT ;  
Y4: SOB R2,Y3 ;  
CMP @#YTMP3,R5 ;FPS CORRECT?  
BEQ Y4 ;  
EMT ;  
Y4: TST @#YFLAG ;FINISHED TEST?  
BNE Y5 ;  
MOV #-1,@#YFLAG  
MOV #YPAT10,@#YTMP1  
MOV #YPAT00,@#YTMP2  
MOV #200,@#YTMP3  
Y5: BR Y1  
BR YDONE  
YFLAG: .WORD 0  
YTMP1: 0  
YTMP2: 0  
YTMP3: 0  
YDAT00: .WORD 0  
YDAT01: 0  
YDAT02: 0  
YDAT03: 0  
YPAT00: 063146  
YPAT01: 052525  
YPAT02: 042104  
YPAT03: 167356  
YPAT10: 163146  
YPAT11: 052525  
YPAT12: 042104  
YPAT13: 167356  
YPAT20: 0  
YPAT21: 0  
YPAT22: 0  
YPAT23: 0  
YDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND  
;SEE IF THE USER HAS EXPRESSED  
;THE DESIRE TO CHANGE THE SOFTWARE  
;VIRTUAL CONSOLE SWITCH REGISTER (HAS  
;THE USER TYPED CONTROL G?).  
:*****  
:TEST 461 ADDD WITH AC=0 TEST  
:*****  
TS461:  
CLR ZFLAG  
MOV #ZPAT00,@#ZTMP1 ;P  
MOV #200,@#ZTMP2  
Z1: MOV #200,RO  
LDFPS RO ;SET DOUBLE MODE  
MOV #ZPAT20,RO ;SET ACO=0  
LDD (RO),AC0
```

```
14576 065336 013700 065444
14577 065342 172010
14578 065344 170205
14579 065346 170011
14580 065350 012700 065450
14581 065354 174010
14582 065356 012702 000004
14583 065362 013701 065444
14584 065366 022021
14585 065370 001401
14586 065372 104000
14587 065374 077204
14588 065376 023705 065446
14589 065402 001401
14590 065404 104000
14591 065406 005737 065442
14592 065412 001012
14593 065414 012737 177777 065442
14594 065422 012737 065470 065444
14595 065430 012737 000210 065446
14596 065436 000731
14597 065440 000423
14598
14599 065442 000000
14600 065444 000000
14601 065446 000000
14602
14603 065450 000000
14604 065452 000000
14605 065454 000000
14606 065456 000000
14607
14608 065460 031463
14609 065462 010421
14610 065464 146314
14611 065466 156735
14612
14613 065470 156735
14614 065472 167356
14615 065474 135673
14616 065476 146314
14617
14618 065500 000000
14619 065502 000000
14620 065504 000000
14621 065506 000000
14622
14623 065510
14624 065510 004767 037056
14625
14626
14627
14628
14629
14630
14631
```

```

Z2:  MOV @#ZTMP1,R0
      ADD (R0),AC0 ;TEST INSTRUCTION
      STFPS R5
      SETD
      MOV #ZDAT00,R0 ;GET RESULT
      STD AC0,(R0)
      MOV #4,R2
      MOV @#ZTMP1,R1 ;RESULT CORRECT?
Z3:  CMP (R0)+,(R1)+
      BEQ Z4
      EMT ;
Z4:  SOB R2,Z3
      CMP @#ZTMP2,R5 ;FPS CORRECT?
      BEQ Z5
      EMT ;
Z5:  TST @#ZFLAG ;FINISHED TEST?
      BNE Z6
      MOV #-1,@#ZFLAG
      MOV #ZPAT10,@#ZTMP1
      MOV #210,@#ZTMP2
      BR Z1
Z6:  BR ZDONE

ZFLAG: .WORD 0
ZTMP1: 0
ZTMP2: 0

ZDAT00: .WORD 0
ZDAT01: 0
ZDAT02: 0
ZDAT03: 0

ZPAT00: 031463
ZPAT01: 010421
ZPAT02: 146314
ZPAT03: 156735

ZPAT10: 156735
ZPAT11: 167356
ZPAT12: 135673
ZPAT13: 146314

ZPAT20: 0
ZPAT21: 0
ZPAT22: 0
ZPAT23: 0

ZDONE: JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;*****
```


14632
 14633
 14634 065514
 14635 065514 012700 003240
 14636 065520 170100
 14637 065522 012700 065724
 14638
 14639 065526 172410
 14640 065530 012700 065734
 14641 065534 172010
 14642
 14643 065536 012700 065714
 14644 065542 174010
 14645 065544 012701 065744
 14646 065550 012702 000004
 14647 065554 022021
 14648 065556 001401
 14649 065560 104000
 14650 065562 077204
 14651
 14652
 14653
 14654
 14655 065564 012700 003200
 14656 065570 170100
 14657 065572 012700 065724
 14658 065576 172410
 14659 065600 012700 065734
 14660 065604 172010
 14661
 14662 065606 012700 065714
 14663 065612 174010
 14664 065614 012701 065754
 14665 065620 012702 000004
 14666 065624 022021
 14667 065626 001401
 14668 065630 104000
 14669 065632 077204
 14670
 14671
 14672
 14673 065634 012700 003200
 14674 065640 170100
 14675 065642 012700 065724
 14676 065646 172410
 14677 065650 170001
 14678 065652 012700 065774
 14679 065656 172010
 14680
 14681 065660
 14682 065660 170011
 14683
 14684 065662 012700 065714
 14685 065666 174010
 14686 065670 012701 066004
 14687 065674 012702 000002

```

;TEST 462      ADDF AND ADDD WITH E(AC)=E(FSRC) TEST AND (BUT FT) TEST
;*****
TS462:
      MOV      #3240,R0
      LDFPS   R0
      MOV      #AAPAT0,R0      ;SET FIV FIV FD AND FT
                                ;FLOWS IN TRAP WILL
                                ;OCCUR
                                ;SET UP ACO
      LDD     (R0),AC0
      MOV      #AAPAT1,R0
AA2:  ADDD    (R0),AC0      ;TEST INSTRUCTION
                                ;SHOULD TRUNCATE
AA3:  MOV     #AADATO,R0
      STD     ACO,(R0)      ;GET THE RESULT
      MOV     #AAPAT2,R1
      MOV     #4,R2
AA4:  CMP     (R0)+,(R1)+  ;CORRECT?
      BEQ     AA7
      EMT
AA7:  SOB     R2,AA4

;NOW TEST DOUBLE FLOATING ROUND MODE.
;A 1 SHOULD BE ADDED TO THE LSB ON ROUND MODE.
      MOV     #3200,R0      ;SET FD FIV FIV. FT=0
      LDFPS   R0
      MOV     #AAPAT0,R0
      LDD     (R0),AC0      ;SET UP ACO OPERAND
      MOV     #AAPAT1,R0
AA11: ADDD    (R0),AC0      ;TEST INSTRUCTION
                                ;SHOULD ROUND
AA12: MOV     #AADATO,R0
      STD     ACO,(R0)      ;GET THE RESULT
      MOV     #AAPAT3,R1
      MOV     #4,R2
AA13: CMP     (R0)+,(R1)+  ;CORRECT?
      BEQ     AA20
      EMT
AA20: SOB     R2,AA13

;NOW TEST ADDF WITH FT=0, ROUND MODE
      MOV     #3200,R0      ;FIV=1, FIV=1, FT=0
      LDFPS   R0
      MOV     #AAPAT0,R0      ;LOAD ACO OPERAND
      LDD     (R0),AC0
      SETF
      MOV     #AAPAT5,R0      ;ENTER FLOATING MODE
AA22: ADDF    (R0),AC0      ;TEST INSTRUCTION
                                ;SHOULD ROUND
AA23: SETD
      ;RESET TO DOUBLE
      ;MODE
      MOV     #AADATO,R0      ;GET THE RESULT
      STD     ACO,(R0)
      MOV     #AAPAT6,R1
      MOV     #2,R2
      ;CORRECT?
  
```

14688 065700 022021
14689 065702 001401
14690 065704 104000
14691 065706 077204
14692 065710 000137 066014
14693 065714 000000
14694 065716 000000
14695 065720 000000
14696 065722 000000
14697 065724 000200
14698 065726 000000
14699 065730 000000
14700 065732 000000
14701 065734 000200
14702 065736 000000
14703 065740 000000
14704 065742 000001
14705 065744 000400
14706 065746 000000
14707 065750 000000
14708 065752 000000
14709 065754 000400
14710 065756 000000
14711 065760 000000
14712 065762 000001
14713 065764 000400
14714 065766 000000
14715 065770 100000
14716 065772 000000
14717 065774 000200
14718 065776 000001
14719 066000 000000
14720 066002 000000
14721 066004 000400
14722 066006 000001
14723 066010 000000
14724 066012 000000
14725 066014
14726 066014 004767 036552
14727
14728
14729
14730
14731
14732
14733
14734 066020
14735
14736 066020 012704 003200
14737 066024 170104
14738 066026 012700 066474
14739 066032 172410
14740 066034 012700 066514
14741 066040 172010
14742 066042 170205
14743 066044 012700 066464

```
AA24:  CMP      (R0)+,(R1)+  
        BEQ     AA27  
        EMT  
AA27:  SOB     R2,AA24  
        JMP     @AADONE  
AADAT0: 0  
        0  
        0  
        0  
AAPAT0: 200  
        0  
        0  
        0  
AAPAT1: 200  
        0  
        0  
        1  
AAPAT2: 400  
        0  
        0  
        0  
AAPAT3: 400  
        0  
        0  
        1  
AAPAT4: 400  
        0  
        100000  
        0  
AAPAT5: 200  
        1  
        0  
        0  
AAPAT6: 400  
        1  
        0  
        0  
AADONE: JSR     PC,.RSET      ;GO INITIALIZE THE FPS AND STACK; AND  
        ;SEE IF THE USER HAS EXPRESSED  
        ;THE DESIRE TO CHANGE THE SOFTWARE  
        ;VIRTUAL CONSOLE SWITCH REGISTER (HAS  
        ;THE USER TYPED CONTROL G?).  
;*****  
;TEST 463      ADDF AND ADDD WITH E(AC) LESS THAN E(FSRC) TEST  
;*****  
TS463:  
;EXPONENT DIFFERENCE=57=71 (OCT) FD=1  
        MOV     #3200,R4      ;SET FIV,FIV, AND FD  
        LDFPS  R4  
        MOV     #CCP0,R0      ;SET ACO OPERAND  
        LDD     (R0),ACO      ;ACO  
        MOV     #CCP2,R0  
CCX2:  ADDD    (R0),ACO      ;TEST INSTRUCTION  
        STFPS  R5            ;GET FPS  
        MOV     #CCDAT0,R0   ;GET THE RESULT
```

14744	066050	174010		STD	ACO,(R0)	
14745	066052	012701	066514	MOV	#CCP2,R1	;IS IT CORRECT
14746	066056	012702	000004	MOV	#4,R2	
14747	066062	022021		CCX3: CMP	(R0)+,(R1)+	
14748	066064	001401		BEQ	CCX6	
14749	066066	104000		EMT		:
14750	066070	077204		CCX6: SOB	R2,CCX3	
14751	066072	020405		CMP	R4,R5	;FPS CORRECT?
14752	066074	001401		BEQ	CCX7	
14753	066076	104000		EMT		:
14754				;EXPONENT DIFFERENCE=56=70 (OCT) FD=1		
14755	066100	012704	003200	CCX7: MOV	#3200,R4	;SET FIV,FIV, AND FD
14756	066104	170104		LDFPS	R4	
14757	066106	012700	066474	MOV	#CCP0,R0	;SET ACO OPERAND
14758	066112	172410		LDD	(R0),ACO	
14759	066114	012700	066504	MOV	#CCP1,R0	;FSRC
14760	066120	172010		CCX8: ADDD	(R0),ACO	;TEST INSTRUCTION
14761	066122	170205		STFPS	R5	;GET FPS
14762	066124	012700	066464	MOV	#CCDAT0,R0	;GET THE RESULT
14763	066130	174010		STD	ACO,(R0)	
14764	066132	012701	066564	MOV	#CCP7,R1	;IS IT CORRECT
14765	066136	012702	000004	MOV	#4,R2	
14766	066142	022021		CCX9: CMP	(R0)+,(R1)+	
14767	066144	001401		BEQ	CCX12	
14768	066146	104000		EMT		:
14769	066150	077204		CCX12: SOB	R2,CCX9	
14770	066152	020405		CMP	R4,R5	;FPS CORRECT?
14771	066154	001401		BEQ	CCX13	
14772	066156	104000		EMT		:
14773				;EXPONENT DIFFERENCE=25=31 (OCT) FD=0		
14774	066160	012700	066474	CCX13: MOV	#CCF0,R0	;SET UP ACO OPERAND.
14775	066164	172410		LDD	(R0),ACO	
14776	066166	012704	003000	MOV	#3000,R4	;SET FIV,FIV. CLEAR FD.
14777	066172	170104		LDFPS	R4	
14778	066174	012700	066554	MOV	#CCP6,R0	;FSRC
14779	066200	172010		CCX14: ADDF	(R0),ACO	;TEST INSTRUCTION
14780	066202	170205		STFPS	R5	
14781	066204	170011		SETD		;REENTER DOUBLE MOVE
14782	066206	012700	066464	MOV	#CCDAT0,R0	;GET THE RESULT
14783	066212	174010		STD	ACO,(R0)	
14784	066214	012701	066554	MOV	#CCP6,R1	;IS THE RESULT CORRECT?
14785	066220	012702	000002	MOV	#2,R2	
14786	066224	022021		CCX15: CMP	(R0)+,(R1)+	
14787	066226	001401		BEQ	CCX18	
14788	066230	104000		EMT		:
14789	066232	077204		CCX18: SOB	R2,CCX15	
14790	066234	020405		CMP	R4,R5	
14791	066236	001401		BEQ	CCX19	
14792	066240	104000		EMT		:
14793				;EXPONENT DIFFERENCE=24=30 (OCT) FD=0		
14794	066242	012700	066524	CCX19: MOV	#CCP3,R0	;SET UP ACO OPERAND.
14795	066246	172410		LDD	(R0),ACO	
14796	066250	012704	003000	MOV	#3000,R4	;SET FIV,FIV. CLEAR FD.
14797	066254	170104		LDFPS	R4	
14798	066256	012700	066544	MOV	#CCP5,R0	;FSRC
14799	066262	172010		CCX20: ADDF	(R0),ACO	;TEST INSTRUCTION

14800	066264	170205		STFPS	R5	
14801	066266	170011		SETD		:REENTER DOUBLE MOVE
14802	066270	012700	066464	MOV	#CCDAT0,R0	:GET THE RESULT
14803	066274	174010		STD	AC0,(R0)	
14804	066276	012701	066574	MOV	#CCP10,R1	:IS THE RESLT CORRECT?
14805	066302	012702	000002	MOV	#2,R2	
14806	066306	022021		CCX21:	CMP (R0)+,(R1)+	
14807	066310	001401			BEQ CCX24	
14808	066312	104000			EMT	:
14809	066314	077204		CCX24:	SOB R2,CCX21	
14810	066316	020405			CMP R4,R5	
14811	066320	001401			BEQ CCX25	
14812	066322	104000			EMT	:
14813					:EXPONENT DIFFERENCE=1 FD=1	
14814	066324	012704	003200	CCX25:	MOV #3200,R4	:SET FIV,FIV, AND FD
14815	066330	170104			LDFPS R4	
14816	066332	012700	066474		MOV #CCP0,R0	:SET AC0 OPERAND
14817	066336	172410			LDD (R0),AC0	
14818	066340	012700	066524		MOV #CCP3,R0	:FSRC
14819	066344	172010		CCX26:	ADDD (R0),AC0	:TEST INSTRUCTION
14820	066346	170205			STFPS R5	:GET FPS
14821	066350	012700	066464		MOV #CCDAT0,R0	:GET THE RESULT
14822	066354	174010			STD AC0,(R0)	
14823	066356	012701	066604		MOV #CCP11,R1	:IS IT CORRECT
14824	066362	012702	000004		MOV #4,R2	
14825	066366	022021		CCX27:	CMP (R0)+,(R1)+	
14826	066370	001401			BEQ CCX30	
14827	066372	104000			EMT	:
14828	066374	077204		CCX30:	SOB R2,CCX27	
14829	066376	020405			CMP R4,R5	:FPS CORRECT?
14830	066400	001401			BEQ CCX31	
14831	066402	104000			EMT	:
14832					:EXPONENT DIFFERENCE=100=144 (OCT) FD=1	
14833	066404	012704	003200	CCX31:	MOV #3200,R4	:SET FIV,FIV, AND FD
14834	066410	170104			LDFPS R4	
14835	066412	012700	066474		MOV #CCP0,R0	:SET AC0 OPERAND
14836	066416	172410			LDD (R0),AC0	
14837	066420	012700	066534		MOV #CCP4,R0	:FSRC
14838	066424	172010		CCX32:	ADDD (R0),AC0	:TEST INSTRUCTION
14839	066426	170205			STFPS R5	:GET FPS
14840	066430	012700	066464		MOV #CCDAT0,R0	:GET THE RESULT
14841	066434	174010			STD AC0,(R0)	
14842	066436	012701	066534		MOV #CCP4,R1	:IS IT CORRECT
14843	066442	012702	000004		MOV #4,R2	
14844	066446	022021		CCX33:	CMP (R0)+,(R1)+	
14845	066450	001401			BEQ CCX36	
14846	066452	104000			EMT	:
14847	066454	077204		CCX36:	SOB R2,CCX33	
14848	066456	020405			CMP R4,R5	:FPS CORRECT?
14849	066460	001461			BEQ CCXDONE	
14850	066462	104000			EMT	:
14851	066464	000000		CCDAT0:	0	
14852	066466	000000			0	
14853	066470	000000			0	
14854	066472	000000			0	
14855	066474	000200		CCP0:	200	:E(AC)=1

```
14856 066476 000000 0
14857 066500 000000 0
14858 066502 000000 0
14859 066504 016200 CCP1: 16200 ;E(FSRC)=E(AC)+56=57
14860 066506 000000 0 ; =71(OCT)
14861 066510 000000 0
14862 066512 000000 0
14863 066514 016400 CCP2: 16400 ;E(FSRC)=E(AC)+57=58
14864 066516 000000 0 ; =72(OCT)
14865 066520 000000 0
14866 066522 000000 0
14867 066524 000400 CCP3: 400 ;E(FSRC)=E(AC)+1=2
14868 066526 000000 0
14869 066530 000000 0
14870 066532 000000 0
14871 066534 031200 CCP4: 31200 ;E(FSRC)=E(AC)+100=101=145(OCT)
14872 066536 000000 0
14873 066540 000000 0
14874 066542 000000 0
14875 066544 006200 CCP5: 6200 ;E(FSRC)=E(AC)+24=25=31(OCT)
14876 066546 000000 0
14877 066550 000000 0
14878 066552 000000 0
14879 066554 006400 CCP6: 6400 ;E(FSRC)=E(AC)+25=26=32(OCT)
14880 066556 000000 0
14881 066560 000000 0
14882 066562 000000 0
14883 066564 016200 CCP7: 16200 ;CCP1 RES
14884 066566 000000 0
14885 066570 000000 0
14886 066572 000001 1
14887 066574 006200 CCP10: 6200 ;CCP5 RES
14888 066576 000001 1
14889 066600 000000 0
14890 066602 000000 0
14891 066604 000500 CCP11: 500 ;CCP3 RES
14892 066606 000000 0
14893 066610 000000 0
14894 066612 000000 0
14895 066614 000200 CCP12: 200 ;BAD CONSTANT
14896 066616 000000 0 ;RES CCP2,CCP4
14897 066620 000000 0
14898 066622 000000 0
14899
14900 066624
14901 066624 004767 035742 CCXDONE: JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND
14902 ;SEE IF THE USER HAS EXPRESSED
14903 ;THE DESIRE TO CHANGE THE SOFTWARE
14904 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
14905 ;THE USER TYPED CONTROL G?).
14906
14907
14908 ;*****
14909 ;TEST 464 ADDF AND ADDD WITH E(AC) GREATER THAN E(FSRC) TEST
14910 ;*****
14911 066630 TS464:
;EXPONENT DIFFERENCE=57=71 (OCT) FD=1
```

14912	066630	012704	003200	MOV	#3200,R4	;SET FIV FIV, AND FD
14913	066634	170104		LDFPS	R4	
14914	066636	012700	067324	MOV	#BBPAT2,R0	;SET ACO OPERAND.
14915	066642	172410		LDD	(R0),AC0	
14916	066644	012700	067314	MOV	#BBPAT1,R0	;FSRC
14917	066650	172010		ADDD	(R0),AC0	;TEST INSTRUCTION
14918	066652	170205		STFPS	R5	
14919	066654	012700	067274	MOV	#BBDAT0,R0	;GET THE RESULT
14920	066660	174010		STD	ACO,(R0)	
14921	066662	012701	067324	MOV	#BBPAT2,R1	;RESULT CORRECT?
14922	066666	012702	000004	MOV	#4,R2	
14923	066672	022021		CMP	(R0)+,(R1)+	
14924	066674	001401		BEQ	BB5	
14925	066676	104000		EMT		:
14926	066700	077204		SOB	R2,BB4	;WAS FPS CORRECT?
14927						
14928	066702	020405		CMP	R4,R5	
14929	066704	001401		BEQ	BB6	
14930	066706	104000		EMT		:
14931						;EXPONENT DIFFERENCE=56=70 (OCT) FD=1
14932	066710	012704	003200	MOV	#3200,R4	;SET FIV,FIV, AND FD
14933	066714	170104		LDFPS	R4	
14934	066716	012700	067344	MOV	#BBPAT4,R0	;SET ACO OPERAND
14935	066722	172410		LDD	(R0),AC0	
14936	066724	012700	067314	MOV	#BBPAT1,R0	;FSRC
14937	066730	172010		ADDD	(R0),AC0	;TEST INSTRUCTION
14938	066732	170205		STFPS	R5	;GET FPS
14939	066734	012700	067274	MOV	#BBDAT0,R0	;GET THE RESULT
14940	066740	174010		STD	ACO,(R0)	
14941	066742	012701	067404	MOV	#BBP10,R1	;IS IT CORRECT
14942	066746	012702	000004	MOV	#4,R2	
14943	066752	022021		CMP	(R0)+,(R1)+	
14944	066754	001401		BEQ	BB13	
14945	066756	104000		EMT		:
14946	066760	077204		SOB	R2,BB10	
14947	066762	020405		CMP	R4,R5	;FPS CORRECT?
14948	066764	001401		BEQ	BB14	
14949	066766	104000		EMT		:
14950						;EXPONENT DIFFERENCE=25=31 (OCT) FD=0
14951	066770	012700	067304	MOV	#BBPAT0,R0	;SET UP ACO OPERAND
14952	066774	172410		LDD	(R0),AC0	
14953	066776	012704	003000	MOV	#3000,R4	;SET FIV AND FIV
14954						;CLEAR FD
14955	067002	170104		LDFPS	R4	
14956	067004	012700	067314	MOV	#BBPAT1,R0	;FSRC
14957	067010	172010		ADDF	(R0),AC0	;TEST INSTRUCTION
14958	067012	170205		STFPS	R5	
14959	067014	170011		SETD		;REENTED DOUBLE MODE.
14960	067016	012700	067274	MOV	#BBDAT0,R0	;GET THE RESULT
14961	067022	174010		STD	ACO,(R0)	
14962	067024	012701	067304	MOV	#BBPAT0,R1	;IS THE RESULT
14963	067030	012702	000002	MOV	#2,R2	;CORRECT?
14964	067034	022021		CMP	(R0)+,(R1)+	
14965	067036	001401		BEQ	BB17	
14966	067040	104000		EMT		:
14967	067042	077204		SOB	R2,BB16	

14968	067044	020405		CMP	R4,R5	;IS FPS CORRECT?
14969	067046	001401		BEQ	BB20	
14970	067050	104000		EMT		
14971				;EXPONENT DIFFERENCE=24=30 (OCT)		
14972	067052	012700	067334	BB20:	MOV	#BBPAT3,R0 ;SET UP ACO OPERAND.
14973	067056	172410			LDD	(R0),AC0
14974	067060	012704	003000		MOV	#3000,R4 ;SET FIU,FIV. CLEAR FD.
14975	067064	170104			LDFPS	R4
14976	067066	012700	067314		MOV	#BBPAT1,R0 ;FSRC
14977	067072	172010		BB21:	ADDF	(R0),AC0 ;TEST INSTRUCTION
14978	067074	170205			STFPS	R5
14979	067076	170011			SETD	;REENTER DOUBLE MODE
14980	067100	012700	067274		MOV	#BBDATO,R0 ;GET THE RESULT
14981	067104	174010			STD	AC0,(R0)
14982	067106	012701	067374		MOV	#BBP7,R1 ;IS THE RESULT CORRECT?
14983	067112	012702	000002		MOV	#2,R2
14984	067116	022021		BB22:	CMP	(R0)+,(R1)+
14985	067120	001401			BEQ	BB25
14986	067122	104000			EMT	
14987	067124	077204		BB25:	SOB	R2,BB22
14988	067126	020405			CMP	R4,R5
14989	067130	001401			BEQ	BB26
14990	067132	104000			EMT	
14991				;EXPONENT DIFFERENCE=1		
14992	067134	012704	003200	BB26:	MOV	#3200,R4
14993	067140	170104			LDFPS	R4 ;SET UP ACO OPERAND
14994	067142	012700	067354		MOV	#BBPAT5,R0
14995	067146	172410			LDD	(R0),AC0
14996	067150	012700	067314		MOV	#BBPAT1,R0 ;FSRC
14997	067154	172010		BB27:	ADDD	(R0),AC0 ;TEST INSTRUCTION
14998	067156	170205			STFPS	R5
14999	067160	012700	067274		MOV	#BBDATO,R0 ;GET THE RESULT.
15000	067164	174010			STD	AC0,(R0)
15001	067166	012701	067414		MOV	#BBP11,R1 ;IS IT CORRECT?
15002	067172	012702	000004		MOV	#4,R2
15003	067176	022021		BB30:	CMP	(R0)+,(R1)+
15004	067200	001401			BEQ	BB31
15005	067202	104000			EMT	
15006	067204	077204		BB31:	SOB	R2,BB30
15007	067206	020405			CMP	R4,R5 ;IS FPS CORRECT
15008	067210	001401			BEQ	BB32
15009	067212	104000			EMT	
15010				;EXPONENT DIFFERENCE=100=144 (OCT)		
15011	067214	012704	003200	BB32:	MOV	#3200,R4
15012	067220	170104			LDFPS	R4 ;SET FIV,FIV AND FD
15013	067222	012700	067364		MOV	#BBPAT6,R0 ;SET UP ACO OPERAND.
15014	067226	172410			LDD	(R0),AC0
15015	067230	012700	067314		MOV	#BBPAT1,R0 ;FSRC
15016	067234	172010		BB33:	ADDD	(R0),AC0 ;TEST INSTRUCTION
15017	067236	170205			STFPS	R5
15018	067240	012700	067274		MOV	#BBDATO,R0 ;GET THE RESULT
15019	067244	174010			STD	AC0,(R0)
15020	067246	012701	067364		MOV	#BBPAT6,R1 ;IS IT CORRECT
15021	067252	012702	000004		MOV	#4,R2
15022	067256	022021		BB34:	CMP	(R0)+,(R1)+
15023	067260	001401			BEQ	BB35

15024 067262 04000
15025 067264 077204
15026 067266 020405
15027 067270 001455
15028 067272 104000
15029 067274 000000
15030 067276 000000
15031 067300 000000
15032 067302 000000
15033 067304 006400
15034 067306 000000
15035 067310 000000
15036 067312 000000
15037 067314 000200
15038 067316 000000
15039 067320 000000
15040 067322 000000
15041 067324 016400
15042 067326 000000
15043 067330 000000
15044 067332 000000
15045 067334 006200
15046 067336 000000
15047 067340 000000
15048 067342 000000
15049 067344 016200
15050 067346 000000
15051 067350 000000
15052 067352 000000
15053 067354 000400
15054 067356 000000
15055 067360 000000
15056 067362 000000
15057 067364 031200
15058 067366 000000
15059 067370 000000
15060 067372 000000
15061 067374 006200
15062 067376 000001
15063 067400 000000
15064 067402 000000
15065 067404 016200
15066 067406 000000
15067 067410 000000
15068 067412 000001
15069 067414 000500
15070 067416 000000
15071 067420 000000
15072 067422 000000
15073 067424
15074 067424 004767 035142
15075
15076
15077
15078
15079

```
EMT  
BB35: SOB R2, BB34 ;  
CMP R4, R5 ; IS FPS CORRECT  
BEQ BBDONE ;  
EMT ;  
BBDAT0: 0  
0  
0  
BBPAT0: 6400 ; F(AC)=E(FSRC)+25=26  
0 ; =32(OCT)  
0  
0  
BBPAT1: 200 ; E(FSRC)=1  
0  
0  
BBPAT2: 16400 ; E(AC)=E(FSRC)+57=58  
0 ; =72(OCT)  
0  
BBPAT3: 6200 ; E(AC)=E(FSRC)+24=25  
0 ; =31(OCT)  
0  
0  
BBPAT4: 16200 ; E(AC)=E(FSRC)+56=57  
0 ; =71(OCT)  
0  
0  
BBPAT5: 400 ; E(AC)=E(FSRC)+1=2  
0  
0  
BBPAT6: 31200 ; E(AC)=E(FSRC)+100=101  
0 ; =145(OCT)  
0  
0  
BBP7: 6200 ; BBPAT3 RES  
1  
0  
0  
BBP10: 16200 ; BBPAT4 RES  
0  
0  
1  
BBP11: 500 ; BBPAT5 RES  
0  
0  
0  
BBDONE: JSR PC, RSET ; GO INITIALIZE THE FPS AND STACK; AND  
; SEE IF THE USER HAS EXPRESSED  
; THE DESIRE TO CHANGE THE SOFTWARE  
; VIRTUAL CONSOLE SWITCH REGISTER (HAS  
; THE USER TYPED CONTROL G?).  
:*****
```


15080			
15081			
15082	067430		
15083			
15084	067430	012704	003200
15085	067434	170104	
15086	067436	012700	070214
15087	067442	172410	
15088	067444	012700	070214
15089	067450	172010	
15090	067452	170205	
15091	067454	012700	070174
15092	067460	174010	
15093	067462	012701	070314
15094	067466	012702	000004
15095	067472	022021	
15096	067474	001401	
15097	067476	104000	
15098	067500	077204	
15099	067502	052704	000010
15100	067506	020405	
15101	067510	001401	
15102	067512	104000	
15103			
15104	067514	012704	003200
15105	067520	170104	
15106	067522	012700	070224
15107	067526	172410	
15108	067530	012700	070214
15109	067534	172010	
15110	067536	170205	
15111	067540	012700	070174
15112	067544	174010	
15113	067546	012701	070204
15114	067552	012702	000004
15115	067556	022021	
15116	067560	001401	
15117	067562	104000	
15118	067564	077204	
15119	067566	052704	000004
15120	067572	020405	
15121	067574	001401	
15122	067576	104000	
15123			
15124	067600	012704	003200
15125	067604	170104	
15126	067606	012700	070214
15127	067612	172410	
15128	067614	012700	070224
15129	067620	172010	
15130	067622	170205	
15131	067624	012700	070174
15132	067630	174010	
15133	067632	012701	070204
15134	067636	012702	000004
15135	067642	022021	

```

:TEST 465      ADDD WITH NEGATIVE OPRANDS TEST
:*****
TS465:
: BOTH OPERANDS NEGATIVE
      MOV #3200,R4      ;SET FIO, FIV, AND FD
      LDFPS R4
      MOV #DDP1,R0      ;SET ACO OPERAND
      LDD (R0),ACO
      MOV #DDP1,R0      ;ESRC
      ADDD (R0),ACO     ;TEST INSTRUCTION
      STFPS R5          ;GET FPS
      MOV #DDDATO,R0    ;GET THE RESULT
      STD ACO,(R0)
      MOV #DDP9,R1      ;IS IT CORRECT
      DD2:  ADDD (R0),ACO
            STFPS R5
            MOV #DDDATO,R0
            STD ACO,(R0)
            MOV #DDP9,R1
            DD3:  CMP (R0)+,(R1)+
                  BEQ DD6
            EMT
            DD6:  SOB R2,DD3
                  BIS #10,R4
                  CMP R4,R5      ;FPS CORRECT?
                  BEQ DD7
            EMT
;AC POS FSRC NEG      AC=-FSRC
      DD7:  MOV #3200,R4      ;SET FIO, FIV, AND FD
            LDFPS R4
            MOV #DDP2,R0      ;SET ACO OPERAND
            LDD (R0),ACO
            MOV #DDP1,R0      ;FSRC
            ADDD (R0),ACO     ;TEST INSTRUCTION
            STFPS R5          ;GET FPS
            MOV #DDDATO,R0    ;GET THE RESULT
            STD ACO,(R0)
            MOV #DDP0,R1      ;IS IT CORRECT
            DD8:  ADDD (R0),ACO
                  STFPS R5
                  MOV #DDDATO,R0
                  STD ACO,(R0)
                  MOV #DDP0,R1
                  DD10:  CMP (R0)+,(R1)+
                        BEQ DD11
                        EMT
                        DD11:  SOB R2,DD10
                               BIS #4,R4
                               CMP R4,R5      ;FPS CORRECT?
                               BEQ DD12
                               EMT
;AC NEG FSRC POS      AC=-FSRC
      DD12:  MOV #3200,R4      ;SET FIO, FIV, AND FD
            LDFPS R4
            MOV #DDP1,R0      ;SET ACO OPERAND
            LDD (R0),ACO
            MOV #DDP2,R0      ;FSRC
            ADDD (R0),ACO     ;TEST INSTRUCTION
            STFPS R5          ;GET FPS
            MOV #DDDATO,R0    ;GET THE RESULT
            STD ACO,(R0)
            MOV #DDP0,R1      ;IS IT CORRECT
            DD13:  ADDD (R0),ACO
                  STFPS R5
                  MOV #DDDATO,R0
                  STD ACO,(R0)
                  MOV #DDP0,R1
                  DD14:  CMP (R0)+,(R1)+

```

15136	067644	001401		BEQ	DD15	
15137	067646	104000		EMT		:
15138	067650	077204		DD15: SOB	R2,DD14	
15139	067652	052704	000004	BIS	#4,R4	
15140	067656	020405		CMP	R4,R5	:EPS CORRECT?
15141	067660	001401		BEQ	DD16	
15142	067662	104000		EMT		:
15143				:ACO POC	FSRC NEG	/AC/ > /FSRC/
15144	067664	012704	003200	DD16: MOV	#3200,R4	:SET FIV, FIV AND FD
15145	067670	170104		LDFPS	R4	
15146	067672	012700	070234	MOV	#DDP3,R0	:SET ACO OPERAND
15147	067676	172410		LDD	(R0),ACO	
15148	067700	012700	070264	MOV	#DDP6,R0	:ESPC
15149	067704	172010		DD17: ADDD	(R0),ACO	:TEST INSTRUCTION
15150	067706	170205		STFPS	R5	:GET FPS
15151	067710	012700	070174	MOV	#DDDATO,R0	:GET THE RESULT
15152	067714	174010		STD	ACO,(R0)	
15153	067716	012701	070274	MOV	#DDP7,R1	:IS IT CORRECT
15154	067722	012702	000004	MOV	#4,R2	
15155	067726	022021		DD18: CMP	(R0)+,(R1)+	
15156	067730	001401		BEQ	DD21	
15157	067732	104000		EMT		:
15158	067734	077204		DD21: SOB	R2,DD18	
15159	067736	020405		CMP	R4,R5	:EPS CORRECT?
15160	067740	001401		BEQ	DD22	
15161	067742	104000		EMT		:
15162				:AC NEG	FSRC POS	/FSRC/ > /AC/
15163	067744	012704	003200	DD22: MOV	#3200,R4	:SET FIO,FIV, AND FD
15164	067750	170104		LDFPS	R4	
15165	067752	012700	070264	MOV	#DDP6,R0	:SET ACO OPERAND
15166	067756	172410		LDD	(R0),ACO	
15167	067760	012700	070234	MOV	#DDP3,R0	:FSPC
15168	067764	172010		DD23: ADDD	(R0),ACO	:TEST INSTRUCTION
15169	067766	170205		STFPS	R5	:GET FPS
15170	067770	012700	070174	MOV	#DDDATO,R0	:GET THE RESULT
15171	067774	174010		STD	ACO,(R0)	
15172	067776	012701	070274	MOV	#DDP7,R1	:IS IT CORRECT?
15173	070002	012702	000004	MOV	#4,R2	
15174	070006	022021		DD24: CMP	(R0)+,(R1)+	
15175	070010	001401		BEQ	DD27	
15176	070012	104000		EMT		:
15177	070014	077204		DD27: SOB	R2,DD24	
15178	070016	020405		CMP	R4,R5	:FPS CORRECT?
15179	070020	001401		BEQ	DD30	
15180	070022	104000		EMT		:
15181				:ACO POS	FSRC NEG	/AC/</FSRC/
15182	070024	012704	003200	DD30: MOV	#3200,R4	:SET FIO,FIV,AND FD
15183	070030	170104		LDFPS	R4	
15184	070032	012700	070244	MOV	#DDP4,R0	:SET ACO OPERAND
15185	070036	172410		LDD	(R0),ACO	
15186	070040	012700	070254	MOV	#DDP5,R0	:FSPC
15187	070044	172010		DD31: ADDD	(R0),ACO	:TEST INSTRUCTION
15188	070046	170205		STFPS	R5	:GET FPS
15189	070050	012700	070174	MOV	#DDDATO,R0	:GET THE RESULT
15190	070054	174010		STD	ACO,(R0)	
15191	070056	012701	070304	MOV	#DDP8,R1	:IS IT CORRECT

15192	070062	012702	000004
15193	070066	022021	
15194	070070	001401	
15195	070072	104000	
15196	070074	077204	
15197	070076	052704	000010
15198	070102	020405	
15199	070104	001401	
15200	070106	104000	
15201			
15202	070110	012704	003200
15203	070114	170104	
15204	070116	012700	070254
15205	070122	172410	
15206	070124	012700	070244
15207	070130	172010	
15208	070132	170205	
15209	070134	012700	070174
15210	070140	174010	
15211	070142	012701	070304
15212	070146	012702	000004
15213	070152	022021	
15214	070154	001401	
15215	070156	104000	
15216	070160	077204	
15217	070162	052704	000010
15218	070166	020405	
15219	070170	001455	
15220	070172	104000	
15221	070174	000000	
15222	070176	000000	
15223	070200	000000	
15224	070202	000000	
15225	070204	000000	
15226	070206	000000	
15227	070210	000000	
15228	070212	000000	
15229	070214	100200	
15230	070216	000000	
15231	070220	000000	
15232	070222	000000	
15233	070224	000200	
15234	070226	000000	
15235	070230	000000	
15236	070232	000000	
15237	070234	001100	
15238	070236	000000	
15239	070240	000000	
15240	070242	000000	
15241	070244	000600	
15242	070246	000000	
15243	070250	000000	
15244	070252	000000	
15245	070254	101100	
15246	070256	000000	
15247	070260	000000	

```

DD32:  MOV      #4,R2
        CMP      (R0)+,(R1)+
        BEQ      DD35
        EMT
DD35:  SOB      R2,DD32
        BIS      #10,R4
        CMP      R4,R5           ;FPS CORRECT?
        BEQ      DD36
        EMT
;ACO NEG      FSRC      POS      /FSRC/</AC/
DD36:  MOV      #3200,R4        ;SET F10, F1V, AND FD
        LDFPS   R4
        MOV      #DDP5,R0      ;SET ACO OPERAND
        LDD      (R0),ACO
        MOV      #DDP4,R0      ;FSPC
DD37:  ADDD     (R0),ACO        ;TEST INSTRUCTION
        STFPS   R5            ;GET FPS
        MOV      #DDDAT0,R0    ;GET THE RESULT
        STD     ACO,(R0)
        MOV      #DDP8,R1      ;IS IT CORRECT
        MOV      #4,R2
DD38:  CMP      (R0)+,(R1)+
        BEQ      DD41
        EMT
DD41:  SOB      R2,DD38
        BIS      #10,R4
        CMP      R4,R5           ;FPS CORRECT?
        BEQ      DDDONE
        EMT
DDDAT0: 0
         0
         0
DDP0:    0
         0
         0
DDP1:    100200          ;--DDP2
         0
         0
         0
DDP2:    200            ;--DDP1
         0
         0
         0
DDP3:    1100          ;EXP=4
         0            ;FRAC=...110...
         0
         0
         0
DDP4:    600           ;EXP=3
         0            ;FRAC=...100...
         0
         0
         0
DDP5:    101100        ;--DDP3
         0
         0
  
```

15248 070262 000000
15249 070264 100600
15250 070266 000000
15251 070270 000000
15252 070272 000000
15253 070274 001000
15254 070276 000000
15255 070300 000000
15256 070302 000000
15257 070304 101000
15258 070306 000000
15259 070310 000000
15260 070312 000000
15261 070314 100400
15262 070316 000000
15263 070320 000000
15264 070322 000000
15265 070324
15266 070324 004767 034242
15267
15268
15269
15270
15271
15272
15273
15274 070330
15275
15276 070330 012704 003200
15277 070334 170104
15278 070336 012700 070522
15279 070342 172410
15280 070344 012700 070522
15281 070350 173010
15282 070352 170205
15283 070354 012700 070500
15284 070360 174010
15285 070362 012701 070510
15286 070366 012702 000004
15287 070372 022021
15288 070374 001401
15289 070376 104000
15290 070400 077204
15291 070402 052704 000004
15292 070406 020405
15293 070410 001401
15294 070412 104000
15295
15296 070414 012704 003200
15297 070420 170104
15298 070422 012700 070542
15299 070426 172410
15300 070430 012700 070542
15301 070434 173010
15302 070436 170205
15303 070440 012700 070500

0
DDP6: 100600 ; -DDP4
0
0
DDP7: 1000 ; DDP3+DDP6
0
0
DDP8: 101000 ; DDP5+DDP4
0
0
DDP9: 100400 ; DDP1+DDP1
0
0
DDDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).
:*****
:TEST 466 SUBD TEST
:*****
TS466:
: USE POSITIVE OPERANDS
MOV #3200,R4 ;SET FIU, FIV, AND FD
LDFPS R4
MOV #EEP1,R0 ;SET ACO OPERAND
LDD (R0),ACO
MOV #EEP1,R0 ;FSPC
EE2: SUBD (R0),ACO ;TEST INSTRUCTION
STFPS R5 ;GET FPS
MOV #EEDATO,R0 ;GET THE RESULT
STD ACO,(R0)
MOV #EEP0,R1 ;IS IT CORRECT?
EE3: MOV #4,R2
CMP (R0)+,(R1)+
BEQ EE6
EMT ;
EE6: SOB R2,EE3
BIS #4,R4
CMP R4,R5 ;FPS CORRECT?
BEQ EE7
EMT ;
:USE NEGATIVE OPERANDS
EE7: MOV #3200,R4 ;SET FIO, FIV, AND FD
LDFPS R4
MOV #EEP3,R0 ;SET ACO OPERAND
LDD (R0),ACO
MOV #EEP3,R0 ;FSPC
EE8: SUBD (R0),ACO ;TEST INSTRUCTION
STFPS R5 ;GET FPS
MOV #EEDATO,R0 ;GET THE RESULT

15304 070444 174010
 15305 070446 012701 070510
 15306 070452 012702 000004
 15307 070456 022021
 15308 070460 001401
 15309 070462 104000
 15310 070464 077204
 15311 070466 052704 000004
 15312 070472 020405
 15313 070474 001432
 15314 070476 104000
 15315 070500 000000
 15316 070502 000000
 15317 070504 000000
 15318 070506 000000
 15319 070510 000000
 15320 070512 000000
 15321 070514 000000
 15322 070516 000000
 15323 070520 000000
 15324 070522 000200
 15325 070524 000000
 15326 070526 000000
 15327 070530 000000
 15328 070532 000400
 15329 070534 000000
 15330 070536 000000
 15331 070540 000000
 15332 070542 100200
 15333 070544 000000
 15334 070546 000000
 15335 070550 000000
 15336 070552 100400
 15337 070554 000000
 15338 070556 000000
 15339 070560 000000
 15340 070562
 15341 070562 004767 034004
 15342
 15343
 15344
 15345
 15346
 15347
 15348
 15349 070566
 15350
 15351 070566 012704 003200
 15352 070572 170104
 15353 070574 012700 070756
 15354 070600 172410
 15355 070602 012700 070766
 15356 070606 172010
 15357 070610 170205
 15358 070612 012700 070726
 15359 070616 174010

```

STD      ACO,(R0)
MOV      #EEP0,R1      ;IS IT CORRECT?
MOV      #4,R2
EE9:     CMP      (R0)+,(R1)+
BEQ      EE12
EMT
EE12:    SOB      R2,EE9
BIS      #4,R4
CMP      R4,R5      ;FPS CORRECT?
BEQ      EEDONE
EMT

EEDATO:  0
         0
         0
EEP0:    0
         0
00000
EEP1:    200
         0
         0
EEP2:    400
         0
         0
EEP3:    100200
         0
         0
EEP4:    100400
         0
         0
EEDONE:  JSR      PC,,RSET      ;GO INITIALIZE THE FPS AND STACK; AND
                                   ;SEE IF THE USER HAS EXPRESSED
                                   ;THE DESIRE TO CHANGE THE SOFTWARE
                                   ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
                                   ;THE USER TYPED CONTROL G?).
:*****
:TEST 467      NORMALIZE ALGORITHM TEST
:*****
TS467:
:USE DATA PATTERNS THAT REQUIRE ONLY ONE LEFT SHIFT TO NORMALIZE
MOV      #3200,R4      ;SET F10, F1V, AND FD
LDFPS   R4
MOV      #FFP2,R0      ;SET ACO OPERAND
LDD     (R0),ACO
MOV      #FFP3,R0      ;FSPC
ADD     (R0),ACO      ;TEST INSTRUCTION
STFPS   R5             ;GET FPS
MOV      #FFDATO,R0    ;GET THE RESULT
STD      ACO,(R0)
  
```

CJKDJB0 11/23-B CPU CLUSTER DIAG.
CJKDJB.P11 26-MAY-82 11:14

DNMAC X24.07-563 26-MAY-82^{K 7} 11:18 PAGE 295
T467 NORMALIZE ALGORITHM TEST

SEQ 0294

15360 070620 012701 070776

MOV #FFP4, R1

;IS IT CORRECT

15361 070624 012702 000004
15362 070630 022021
15363 070632 001401
15364 070634 104000
15365 070636 077204
15366 070640 020405
15367 070642 001401
15368 070644 104000
15369
15370
15371 070646 012704 003200
15372 070652 170104
15373 070654 012700 070736
15374 070660 172410
15375 070662 012700 070746
15376 070666 172010
15377 070670 170205
15378 070672 012700 070726
15379 070676 174010
15380 070700 012701 070776
15381 070704 012702 000004
15382 070710 022021
15383 070712 001401
15384 070714 104000
15385 070716 077204
15386 070720 020405
15387 070722 001431
15388 070724 104000
15389
15390
15391 070726 000000
15392 070730 000000
15393 070732 000000
15394 070734 000000
15395
15396 070736 016000
15397 070740 000000
15398 070742 000000
15399 070744 000001
15400 070746 116000
15401 070750 000000
15402 070752 000000
15403 070754 000000
15404 070756 000500
15405 070760 000000
15406 070762 000000
15407 070764 000000
15408 070766 100400
15409 070770 000000
15410 070772 000000
15411 070774 000000
15412 070776 000200
15413 071000 000000
15414 071002 000000
15415 071004 000000
15416

```
MOV #4,R2
FF3:  CMP (R0)+,(R1)+
      BEQ FF4
      EMT ;
FF4:  SOB R2,FF3 ;
      CMP R4,R5 ;FPS CORRECT?
      BEQ FF5
      EMT
;USE DATA PATTERNS WHICH REQUIRE 56 LEFT SHIFTS TO NORMALIZE
;THE RESULT
FF5:  MOV #3200,R4 ;SET FIU, FIV, AND FD
      LDFPS R4
      MOV #FFP0,R0 ;SET ACO OPERAND
      LDD (R0),ACO
      MOV #FFP1,R0 ;FSRC
FF6:  ADDD (R0),ACO ;TEST INSTRUCTION
      STFPS R5 ;GET FPS
      MOV #FFDAT0,R0 ;GET THE RESULT
      STD ACO,(R0)
      MOV #FFP4,R1 ;IS IT CORRECT
      MOV #4,R2
FF7:  CMP (R0)+,(R1)+
      BEQ FF10
      EMT ;
FF10: SOB R2,FF7 ;
      CMP R4,R5 ;FPS CORRECT?
      BEQ FFDONE
      EMT ;

FFDAT0: 0
        0
        0
        0

FFP0: 16000
        0
        0
        1

FFP1: 116000
        0
        0
        0

FFP2: 500
        0
        0
        0

FFP3: 100400
        0
        0
        0

FFP4: 200 ;FFP4=FFP0+FFP1
        0 ; =FFP3+FFP4
        0
        0
```

15417 071006
15418 071006 004767 033560
15419
15420
15421
15422
15423
15424
15425
15426
15427
15428
15429
15430
15431
15432
15433
15434
15435
15436 071012
15437
15438
15439
15440 071012 012704 003200
15441 071016 170104
15442 071020 012700 071422
15443 071024 172410
15444 071026 012700 071432
15445 071032 172010
15446 071034 170205
15447 071036 012700 071412
15448 071042 174010
15449 071044 012701 071442
15450 071050 012702 000004
15451 071054 022021
15452 071056 001401
15453 071060 104000
15454 071062 077204
15455 071064 020405
15456 071066 001401
15457 071070 104000
15458
15459
15460
15461
15462
15463 071072 012704 043200
15464
15465 071076 170104
15466 071100 012700 071472
15467 071104 172410
15468 071106 012700 071502
15469 071112 172010
15470 071114 170205
15471 071116 012700 071412
15472 071122 174010

```
FFDONE: JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK, AND  
;SEE IF THE USER HAS EXPRESSED  
;THE DESIRE TO CHANGE THE SOFTWARE  
;VIRTUAL CONSOLE SWITCH REGISTER (HAS  
;THE USER TYPED CONTROL G?).  
  
:*****  
:*****  
:FLOATING POINT SECOND PART  
:*****  
:*****  
  
:*****  
:TEST 470 ROUND\TRUNK TEST  
:*****  
TS470:  
  
;ROUND AND NORMALIZE TEST  
  
MOV #3200,R4 ;SET FIU, FIV, AND FD  
LDFPS R4  
MOV #HHP0,R0 ;SET ACO OPERAND  
LDD (R0),ACO  
MOV #HHP1,R0 ;FSPC  
HH2: ADDD (R0),ACO ;TEST INSTRUCTION  
STFPS R5 ;GET FPS  
MOV #HHDATO,R0 ;GET THE RESULT  
STD ACO,(R0)  
MOV #HHP2,R1 ;IS IT CORRECT  
MOV #4,R2  
HH3: CMP (R0)+,(R1)+  
BEQ HH6  
EMT ;  
HH6: SOB R2,HH3  
CMP R4,R5 ;FPS CORRECT?  
BEQ HH7  
EMT ;  
  
:THIS IS A TEST OF THE ABILITY  
:OF NORMALIZE TO PRODUCE A ZERO EXP. AND  
:OF THE R\T ALGORITHM TO PROPERLY SET THE FPS  
  
HH7: MOV #043200,R4 ;SET FIU,FIV,AND FD  
;FID  
LDFPS R4  
MOV #HHP5,R0 ;SET ACO OPERAND  
LDD (R0),ACO  
MOV #HHP6,R0 ;FSPC  
HH8: ADDD (R0),ACO ;TEST INSTRUCTION  
STFPS R5 ;GET FPS  
MOV #HHDATO,R0 ;GET THE RESULT  
STD ACO,(R0)
```


15473 071124 012701 071462
 15474 071130 012702 000004
 15475 071134 022021
 15476 071136 001401
 15477 071140 104000
 15478 071142 077204
 15479 071144 052704 100004
 15480 071150 020405
 15481 071152 001401
 15482 071154 104000
 15483
 15484
 15485
 15486 071156 012704 043200
 15487
 15488 071162 170104
 15489 071164 012700 071522
 15490 071170 172410
 15491 071172 012700 071532
 15492 071176 172010
 15493 071200 170205
 15494 071202 012700 071412
 15495 071206 174010
 15496 071210 012701 071512
 15497 071214 012702 000004
 15498 071220 022021
 15499 071222 001401
 15500 071224 104000
 15501 071226 077204
 15502 071230 052704 100014
 15503 071234 020405
 15504 071236 001401
 15505 071240 104000
 15506
 15507 071242 012704 000200
 15508 07.246 170104
 15509 071250 012700 071522
 15510 071254 172410
 15511 071256 012700 071522
 15512 071262 172010
 15513 071264 170205
 15514 071266 012700 071412
 15515 071272 174010
 15516 071274 012701 071542
 15517 071300 012702 000004
 15518 071304 022021
 15519 071306 001401
 15520 071310 104000
 15521 071312 077204
 15522 071314 052704 000000
 15523 071320 020405
 15524 071322 001401
 15525 071324 104000
 15526
 15527 071326 012704 003200
 15528 071332 170104

```

MOV #HHP4,R1 ;IS IT CORRECT
MOV #4,R2
HH9: CMP (R0)+,(R1)+
BEQ HH10
EMT ;
HH10: SOB R2,HH9
BIS #100004,R4
CMP R4,R5
BEQ HH11
EMT ;

;THIS IS A TEST OF THE R\T ALGORITHM'S
;ABILITY TO SET BOTH N AND Z ON A - 0 RESULT
HH11: MOV #043200,R4 ;SET FIV, FIV, AND FD

LDFPS R4
MOV #HHP8,R0 ;SET ACO OPERAND
LDD (R0),ACO
MOV #HHP9,R0 ;FSPC
HH12: ADD (R0),ACO ;TEST INSTRUCTION
STFPS R5 ;GET FPS
MOV #HHDATA,R0 ;GET THE RESULT
STD ACO,(R0)
MOV #HHP7,R1 ;IS IT CORRECT
HH13: MOV #4,R2
CMP (R0)+,(R1)+
BEQ HH16
EMT ;
HH16: SOB R2,HH13
BIS #100014,R4 ;FPS CORRECT?
CMP R4,R5
BEQ HH17
EMT

;TEST THAT CC ARE CLEARED BY R\T
HH17: MOV #00200,R4 ;SET FIV, FIV, AND FD
LDFPS R4
MOV #HHP8,R0 ;SET ACO OPERAND
LDD (R0),ACO
MOV #HHP8,R0 ;FSPC
HH18: ADD (R0),ACO ;TEST INSTRUCTION
STFPS R5 ;GET FPS
MOV #HHDATA,R0 ;GET THE RESULT
STD ACO,(R0)
MOV #HHP10,R1 ;IS IT CORRECT
HH19: MOV #4,R2
CMP (R0)+,(R1)+
BEQ HH20
EMT ;
HH20: SOB R2,HH19
BIS #00000,R4 ;FPS CORRECT?
CMP R4,R5
BEQ HH21
EMT

;TEST THAT N IS SET BY R\T
HH21: MOV #3200,R4 ;SET FIV, FIV, AND FD
LDFPS R4

```

15529	071334	012700	071472	MOV	#HHP5,R0	:SET ACO OPERAND
15530	071340	172410		LDD	(R0),ACO	
15531	071342	012700	071472	MOV	#HHP5,R0	:FSPC
15532	071346	172010		4H22: ADDD	(R0),ACO	:TEST INSTRUCTION
15533	071350	170205		STFPS	R5	:GET FPS
15534	071352	012700	071412	MOV	#HHDATO,R0	:GET THE RESULT
15535	071356	174010		STD	ACO,(R0)	
15536	071360	012701	071552	MOV	#HHP1,R1	:IS IT CORRECT
15537	071364	012702	000004	MOV	#4,R2	
15538	071370	022021		4H23: CMP	(R0)+,(R1)+	
15539	071372	001401		BEQ	4H24	
15540	071374	104000		EMT		:
15541	071376	077204		4H24: SOB	R2,4H23	
15542	071400	052704	000010	BIS	#10,R4	
15543	071404	020405		CMP	R4,R5	:FPS CORRECT?
15544	071406	001465		BEQ	HHDONE	
15545	071410	104000		EMT		:
15546	071412	000000		HHDATO: 0		
15547	071414	000000		0		
15548	071416	000000		0		
15549	071420	000000		0		
15550	071422	000452		HHP0: 452		
15551	071424	125252		125252		
15552	071426	125252		125252		
15553	071430	125253		125253		
15554	071432	000252		HHP1: 252		
15555	071434	125252		125252		
15556	071436	125252		125252		
15557	071440	125252		125252		
15558	071442	000600		HHP2: 600		:HHP0 + HHP1 WITH
15559	071444	000000		0		:PROPER NORMALIZATION
15560	071446	000000		0		
15561	071450	000000		0		
15562	071452	000400		HHP3: 400		:HHP0 + HHP1 WITH
15563	071454	000000		0		:BAD NORMALIZATION
15564	071456	000000		0		
15565	071460	000000		0		
15566	071462	000000		HHP4: 0		
15567	071464	000000		0		
15568	071466	000000		0		
15569	071470	000000		0		
15570	071472	100200		HHP5: 100200		
15571	071474	000000		0		
15572	071476	000000		0		
15573	071500	000000		0		
15574	071502	000300		HHP6: 300		
15575	071504	000000		0		
15576	071506	000000		0		
15577	071510	000000		0		
15578	071512	100000		HHP7: 100000		:HHP7 = HHP8 + HHP9
15579	071514	000000		0		: = HHP5 + HHP6
15580	071516	000000		0		
15581	071520	000000		0		
15582	071522	000200		HHP8: 200		
15583	071524	000000		0		
15584	071526	000000		0		

15585 071530 000000
15586 071532 100300
15587 071534 000000
15588 071536 000000
15589 071540 000000
15590 071542 000400
15591 071544 000000
15592 071546 000000
15593 071550 000000
15594 071552 100400
15595 071554 000000
15596 071556 000000
15597 071560 000000
15598 071562
15599 071562 004767 033004
15600
15601
15602
15603
15604
15605
15606
15607
15608
15609 071566
15610
15611
15612 071566 012704 000200
15613 071572 170104
15614 071574 012737 071712 000244
15615 071602 012700 072514
15616 071606 172410
15617 071610 012700 072514
15618 071614 172010
15619 071616 170205
15620 071620 012700 072444
15621 071624 174010
15622 071626 012701 072524
15623 071632 012702 000004
15624 071636 022021
15625 071640 001401
15626 071642 104000
15627 071644 077204
15628 071646 052704 000006
15629 071652 020405
15630 071654 001401
15631 071656 104000
15632
15633
15634 071660 012704 001200
15635 071664 170104
15636 071666 012737 071714 000244
15637 071674 012700 072514
15638 071700 172410
15639 071702 012700 072514
15640 071706 172010

HHP9: 0
100300
0
0
0
HHP10: 400 ;HHP10 = HHP8 + HHP8
0
0
0
HHP11: 100400 ;HHP11 = HHP5 + HHP5
0
0
0
HHDONE: JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;TEST 471 OVER\UNDER TEST

TS471:

;TEST OVERFLOW CONDITION WITH TRAP DISABLER FIV=0
MOV #200,R4 ;CLEAR FIU, FIV, AND SET FD
LDFPS R4
MOV #GGERO,@#FPVECT
MOV #GGP5,R0 ;SET ACO OPERAND
LDD (R0),ACO
MOV #GGP5,R0 ;FSRC
GG2: ADDD (R0),ACO ;TEST INSTRUCTION
STFPS R5 ;GET FPS
MOV #GGDATO,R0 ;GET THE RESULT
STD ACO,(R0)
MOV #GGP6,R1 ;IS IT CORRECT
GG3: CMP (R0)+,(R1)+
BEQ GG4
EMT ;
GG4: SOB R2,GG3
BIS #6,R4 ;FPS CORRECT?
CMP R4,R5
BEQ GG5
EMT ;
;TEST OVERFLOW WITH TRAPS ENABLED
;FIV = 1
GG5: MOV #1200,R4 ;CLEAR FIU, SET FIV, AND FD
LDFPS R4
MOV #GG7,@#FPVECT
MOV #GGP5,R0 ;SET ACO OPERAND
LDD (R0),ACO
MOV #GGP5,R0 ;FSPC
GG6: ADDD (R0),ACO ;TEST INSTRUCTION

15641	071710	170000			CFCC		;NO OVERFLOW TRAP OCCURED
15642	071712				GGERO:		
15643	071712	104000			EMT		:
15644	071714	012703	071710		GG7:	MOV #GG6+2,R3	
15645	071720	020316			CMP R3,(SP)		;CHECK STACK DATA
15646	071722	001401			BEQ 1\$		
15647	071724	104000			EMT		:
15648	071726	022626			1\$:	CMP (SP)+,(SP)+	
15649	071730	170205			STFPS R5		
15650	071732	012700	072444		MOV #GGDATO,R0		;GET THE RESULT
15651	071736	174010			STD ACO,(R0)		
15652	071740	012701	072524		MOV #GGP6,R1		;IS IT CORRECT
15653	071744	012702	000004		MOV #4,R2		
15654	071750	022021			GG8:	CMP (R0)+,(R1)+	
15655	071752	001401			BEQ GG9		
15656	071754	104000			EMT		:
15657	071756	077204			GG9:	SOB R2,GG8	
15658	071760	052704	100006		BIS #100006,R4		;EXACT ZERO RESULTED IF OVERFLOW
15659	071764	020405			CMP R4,R5		;FPS CORRECT?, CHECK FER, FZ, FV
15660	071766	001401			BEQ 1\$		
15661	071770	104000			EMT		:
15662	071772	012704	000010		1\$:	MOV #10,R4	
15663					;CHECK	FEC	
15664	071776	170305			STST R5		
15665	072000	020405			CMP R4,R5		
15666	072002	001401			BEQ GG10		
15667	072004	104000			EMT		:
15668					;CHECK	UNDER FLOW CONDITION WITH	
15669					;TRAPS	DISABLED (FIU = 0)	
15670	072006	012704	000200		GG10:	MOV #0200,R4	;SET FIU, FIV, AND FD
15671	072012	170104			LDFPS R4		
15672	072014	012737	071712	000244	MOV #GGERO,@#FPVECT		
15673	072022	012700	072464		MOV #GGP2,R0		;SET ACO OPERAND
15674	072026	172410			LDD (R0),ACO		;FSRC
15675	072030	012700	072474		MOV #GGP3,R0		
15676	072034	172010			GG11:	ADD (R0),ACO	;TEST INSTRUCTION
15677	072036	170205			STFPS R5		;GET FPS
15678	072040	012700	072444		MOV #GGDATO,R0		;GET THE RESULT
15679	072044	174010			STD ACO,(R0)		
15680	072046	012701	072524		MOV #GGP6,R1		;IS IT CORRECT
15681	072052	012702	000004		MOV #4,R2		
15682	072056	022021			GG12:	CMP (R0)+,(R1)+	
15683	072060	001401			BEQ GG13		
15684	072062	104000			EMT		:
15685	072064	077204			GG13:	SOB R2,GG12	
15686	072066	052704	000004		BIS #4,R4		;FPS CORRECT?
15687	072072	020405			CMP R4,R5		
15688	072074	001401			BEQ GG14		
15689	072076	104000			EMT		:
15690					;CHECK	UNDERFLOW CONDITION WITH	
15691					;TRAP	ENABLED (FIU = 1)	
15692	072100	012704	002200		GG14:	MOV #2200,R4	;SET FIU, FIV, AND FD
15693	072104	170104			LDFPS R4		
15694	072106	012737	072132	000244	MOV #GG16,@#FPVECT		
15695	072114	012700	072464		MOV #GGP2,R0		;SET ACO OPERAND
15696	072120	172410			LDD (R0),ACO		;FSPC

```

15697 072122 012700 072474
15698 072126 172010
15699 072130 170000
15700 072132 012703 072130
15701 072136 021603
15702 072140 001401
15703 072142 104000
15704 072144 022626
15705 072146 170205
15706 072150 012700 072444
15707 072154 174010
15708 072156 012701 072534
15709 072162 012702 000004
15710 072166 022021
15711 072170 001401
15712 072172 104000
15713 072174 077204
15714 072176 052704 100000
15715 072202 020405
15716 072204 001401
15717 072206 104000
15718 072210 012704 000012
15719
15720 072214 170305
15721 072216 020405
15722 072220 001401
15723 072222 104000
15724
15725
15726 072224 012704 000200
15727 072230 170104
15728 072232 012737 072350 000244
15729 072240 012700 072464
15730 072244 172410
15731 072246 012700 072544
15732 072252 172010
15733 072254 170205
15734 072256 012700 072444
15735 072262 174010
15736 072264 012701 072524
15737 072270 012702 000004
15738 072274 022021
15739 072276 001401
15740 072300 104000
15741 072302 077204
15742 072304 052704 000004
15743 072310 020405
15744 072312 001401
15745 072314 104000
15746
15747
15748 072316 012704 002200
15749 072322 170104
15750 072324 012737 072352 000244
15751 072332 012700 072464
15752 072336 172410

MOV #GGP3,R0
GG15: ADDD (R0),ACO ;TEST INSTRUCTION
CFCC
MOV #GG15+2,R3
GG16: CMP (SP),R3
BEQ 1$
EMT ;
1$: CMP (SP)+,(SP)+
STFPS R5 ;GET FPS
MOV #GGDATO,R0 ;GET THE RESULT
STD ACO,(R0)
MOV #GGP7,R1 ;IS IT CORRECT
GG17: MOV #4,R2
CMP (R0)+,(R1)+
BEQ GG18
EMT ;
GG18: SOB R2,GG17
BIS #100000,R4
CMP R4,R5 ;FPS CORRECT?
BEQ 1$
EMT ;
1$: MOV #12,R4
;CHECK FEC
STST R5
CMP R4,R5
BEQ GG19
EMT ;
;CHECK UNDERFLOW CONDITION WITH TRAPS
;DISABLED (FIU = 0)
GG19: MOV #0200,R4 ;SET FIU, FIV, AND FD
LDFPS R4
MOV #GGER14,#FPVECT
GG20: MOV #GGP2,R0 ;SET ACO OPERAND
LDD (R0),ACO
MOV #GGP8,R0 ;FSPC
ADDD (R0),ACO ;TEST INSTRUCTION
STFPS R5 ;GET FPS
MOV #GGDATO,R0 ;GET THE RESULT
STD ACO,(R0)
GG21: MOV #GGP6,R1 ;IS IT CORRECT
MOV #4,R2
CMP (R0)+,(R1)+
BEQ GG22
EMT ;
GG22: SOB R2,GG21
BIS #4,R4 ;FPS CORRECT?
CMP R4,R5
BEQ GG23
EMT ;
;CHECK UNDERFLOW CONDITION WITH TRAP
;ENABLED (FIU = 1)
GG23: MOV #2200,R4 ;SET FIU, FIV, AND FD
LDFPS R4
MOV #GG25,#FPVECT
MOV #GGP2,R0 ;SET ACO OPERAND
LDD (R0),ACO

```

15753 072340 012700 072544
 15754 072344 172010
 15755 072346 170000
 15756 072350
 15757 072350 104000
 15758 072352 012703 072346
 15759 072356 020316
 15760 072360 001401
 15761 072362 104000
 15762 072364 022626
 15763 072366 170205
 15764 072370 012700 072444
 15765 072374 174010
 15766 072376 012701 072554
 15767 072402 012702 000004
 15768 072406 022021
 15769 072410 001401
 15770 072412 104000
 15771 072414 077204
 15772 072416 052704 100004
 15773 072422 020405
 15774 072424 001401
 15775 072426 104000
 15776 072430 012704 000012
 15777
 15778 072434 170305
 15779 072436 020405
 15780 072440 001451
 15781 072442 104000
 15782 072444 000000
 15783 072446 000000
 15784 072450 000000
 15785 072452 000000
 15786
 15787 072454 000300
 15788 072456 000000
 15789 072460 000000
 15790 072462 000000
 15791 072464 100200
 15792 072466 000000
 15793 072470 000000
 15794 072472 000000
 15795 072474 000200
 15796 072476 000000
 15797 072500 000000
 15798 072502 000001
 15799 072504 010200
 15800 072506 000000
 15801 072510 000000
 15802 072512 000000
 15803 072514 077600
 15804 072516 000000
 15805 072520 000000
 15806 072522 000000
 15807 072524 000000
 15808 072526 000000

```

MOV #GGP8,R0 ;FSRC
GG24: ADDD (R0),ACO ;TEST INSTRUCTION
      CFCC
GGER14:
      EMT ;
GG25: MOV #GG24+2,R3 ;
      CMP R3,(SP) ;
      BEQ 1$ ;
1$: EMT ;
      CMP (SP)+,(SP)+ ;
      STFPS R5 ;GET FPS
      MOV #GGDATO,R0 ;GET THE RESULT
      STD ACO,(R0)
      MOV #GGP9,R1 ;IS IT CORRECT
GG26: CMP (R0)+,(R1)+
      BEQ GG27
      EMT ;
GG27: SOB R2,GG26
      BIS #100004,R4
      CMP R4,R5 ;FPS CORRECT?
      BEQ 1$
1$: MOV #12,R4
      :CHECK FEC
      STST R5
      CMP R4,R5
      BEQ GGDONE
      EMT ;
GGDATO: 0
        0
        0
        0
GGP1: 300
        0
        0
        0
GGP2: 100200
        0
        0
        0
GGP3: 200
        0
        0
        1
GGP4: 10200
        0
        0
        0
GGP5: 77600 ;OVER FLOW = GGP5 + GGP5
        0
        0
        0
GGP6: 0 ;OVERFLOW RESULT
        0 ;UNDERFLOW RESULT
  
```

```

15809 072530 000000
15810 072532 000000
15811
15812 072534 062400
15813 072536 000000
15814 072540 000000
15815 072542 000000
15816 072544 000340
15817 072546 000000
15818 072550 000000
15819 072552 000000
15820 072554 000100
15821 072556 000000
15822 072560 000000
15823 072562 000000
15824 072564
15825 072564 004767 032002
15826
15827
15828
15829
15830
15831
15832
15833
15834
15835 072570
15836
15837 072570 012704 000200
15838 072574 170104
15839 072576 012700 073326
15840 072602 172410
15841 072604 012700 073336
15842 072610 177420
15843 072612 020027 073342
15844 072616 001401
15845 072620 104000
15846 072622
15847 072622 170205
15848 072624 012700 073316
15849 072630 174010
15850 072632 012701 073406
15851 072636 012702 000004
15852 072642 022120
15853 072644 001401
15854 072646 104000
15855 072650 077204
15856 072652 012704 000200
15857 072656 020405
15858 072660 001401
15859 072662 104000
15860
15861 072664 012704 000200
15862 072670 170104
15863
15864 072672 012700 073326
  
```

```

0
0
GGP7: 62400
0
0
GGP8: 340
0
0
GGP9: 100
0
0
GGDONE: JSR PC,,RSET
;GGP6 = GGP4 + GGP5
; = GGP3 + GGP2 (FIU = 0)
; = GGP3 + GGP1
;GGP7 = GGP3 + GGP2 (FIU = 1)
;GO INITIALIZE THE FP. AND ACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).
  
```

```

;*****
;TEST 472 LDCFD AND LDCDF TEST
;*****
TS472:
;TEST FOR CORRECT AUTO INCREMENT CONSTANT.
MOV #200,R4 ;SET LONG INTEGER MODE
LDFPS R4
MOV #HXP1,R0
LDD (R0),ACO
MOV #HXP2,R0
HX2: LDCFD (R0)+,ACO ;IS R0 CORRECT
CMP R0,#HXP2+4
BEQ HX3
EMT ;
HX3: STFPS R5 ;GET FPS
MOV #HXDATA,R0
STD ACO,(R0) ;GET ACO
MOV #HXP7,R1 ;SEE IF RESULT IS
MOV #4,R2 ;CORRECT
HX4: CMP (R1)+,(R0)+
BEQ HX7
EMT ;
HX7: SOB R2,HX4 ;FPS CORRECT?
MOV #200,R4
CMP R4,R5
BEQ HX8
EMT ;
;NOW
HX8: TEST LDCDF
MOV #200,R4
LDFPS R4
MOV #HXP1,R0
  
```

15865	072676	172410			LDD	(R0),ACO	
15866							
15867	072700	012700	073336		MOV	#HXP2,R0	
15868	072704	170001			SETF		
15869	072706	177420			HX9: LDCFD	(R0)+,ACO	;TEST INSTRUCTION
15870	072710	020027	073346		CMP	R0,#HXP2+10	;WAS A GUOD
15871	072714	001401			BEQ	HX10	
15872	072716	104000			EMT		
15873							
15874	072720				HX10: STFPS	R5	
15875	072720	170205			MOV	#HXDATO,R0	
15876	072722	012700	073316		SETD		
15877	072726	170011			STD	ACO,(R0)	;GET RESULT
15878	072730	174010			MOV	#HXP8,R1	
15879	072732	012701	073416		MOV	#4,R2	
15880	072736	012702	000004		HX11: CMP	(R1)+,(R0)+	;IS IT CORRECT?
15881	072742	022120			BEQ	HX14	
15882	072744	001401			EMT		
15883	072746	104000			HX14: SOB	R2,HX11	
15884	072750	077204					
15885							
15886	072752	012704	000000		MOV	#0,R4	;FPS CORRECT?
15887	072756	020405			CMP	R4,R5	
15888	072760	001401			BEQ	HX15	
15889	072762	104000			EMT		
15890							
15891	072764	012704	000200		:TEST GR7 IMMEDIATE MODE CONSTANT		
15892	072770	170104			HX15: MOV	#200,R4	
15893	072772	012737	073022	000004	LDFPS	R4	;SET FD
15894	073000	005001			MOV	#HXER9,@#ERRVECT	
15895	073002	177427	043243		HX16: CLR	R1	
15896	073006	005201			HX165: LDCFD	#5201,ACO	
15897	073010	005201			INC	R1	
15898	073012	005201			INC	R1	
15899	073014	020127	000003		INC	R1	
15900	073020	001401			CMP	R1,#3	;SEE IF PC WAS
15901	073022				BEQ	HX17	
15902	073022	104000			HXER9: EMT		
15903	073024	012704	000200		HX17: MOV	#200,R4	
15904	073030	170104			LDFPS	R4	
15905	073032	012700	073376		MOV	#HXP6,R0	
15906	073036	172410			LDD	(R0),ACO	
15907	073040	012700	073336		MOV	#HXP2,R0	
15908	073044	177410			HX18: LDCFD	(R0),ACO	
15909							
15910	073046	012700	073316		MOV	#HXDATO,R0	
15911	073052	174010			STD	ACO,(R0)	;GET RESULT.
15912	073054	012701	073406		MOV	#HXP7,R1	
15913	073060	012702	000004		MOV	#4,R2	
15914	073064	022021			HX19: CMP	(R0)+,(R1)+	;IS RESULT CORRECT?
15915	073066	001401			BEQ	HX20	
15916	073070	104000			EMT		
15917	073072	077204			HX20: SOB	R2,HX19	
15918							
15919							
15920	073074	012704	000200		:TEST LDCFD WITH NEGATIVE OPERAND		
					MOV	#200,R4	

15921	073100	170104		LDFPS	R4	
15922	073102	012700	073376	MOV	#HXP6,R0	
15923	073106	172410		LDD	(R0),ACO	
15924	073110	012700	073356	MOV	#HXP4,R0	
15925	073114	177410		HX22: LDCFD	(R0),ACO	
15926						
15927	073116	012700	073316	MOV	#HXDATO,R0	
15928	073122	174010		STD	ACO,(R0)	;GET RESULT
15929						
15930	073124	012701	073366	MOV	#HXP5,R1	
15931	073130	012702	000004	MOV	#4,R2	
15932	073134	022120		HX23: CMP	(R1)+,(R0)+	
15933	073136	001401		BEQ	HX26	
15934	073140	104000		EMT		:
15935	073142	077204		HX26: SOB	R2,HX23	
15936						
15937				:TEST	LDCFD	0
15938						
15939	073144	012704	000200	MOV	#200,R4	
15940	073150	170104		LDFPS	R4	
15941						
15942	073152	012700	073326	MOV	#HXP1,R0	
15943	073156	172410		LDD	(R0),ACO	
15944	073160	172010		ADDD	(R0),ACO	
15945						
15946	073162	012700	073326	MOV	#HXP1,R0	
15947	073166	177410		HX28: LDCFD	(R0),ACO	
15948						
15949	073170	170205		STFPS	R5	
15950						
15951	073172	012700	073316	MOV	#HXDATO,R0	
15952	073176	174010		STD	ACO,(R0)	;GET RESULT
15953						
15954	073200	012701	073326	MOV	#HXP1,R1	
15955	073204	012702	000004	MOV	#4,R2	
15956	073210	022120		HX29: CMP	(R1)+,(R0)+	;IS IT 0?
15957	073212	001401		BEQ	HX30	
15958	073214	104000		EMT		:
15959	073216	077204		HX30: SOB	R2,HX29	
15960						
15961	073220	012704	000204	MOV	#204,R4	;FPS CORRECT
15962	073224	020405		CMP	R4,R5	
15963	073226	001401		BEQ	HX31	
15964	073230	104000		EMT		:
15965				:TEST	LDCFD	0
15966	073232	012704	000200	HX31: MOV	#200,R4	
15967	073236	170104		LDFPS	R4	
15968	073240	012700	073376	MOV	#HXP6,R0	
15969	073244	172410		LDD	(R0),ACO	
15970	073246	012700	073326	MOV	#HXP1,R0	
15971	073252	177410		HX32: LDCFD	(R0),ACO	
15972	073254	170205		STFPS	R5	
15973	073256	012700	073316	MOV	#HXDATO,R0	
15974	073262	174010		STD	ACO,(R0)	;GET RESULT
15975	073264	012701	073326	MOV	#HXP1,R1	
15976	073270	012702	000004	MOV	#4,R2	

15977 073274 022120
15978 073276 001401
15979 073300 104000
15980 073302 077204
15981
15982 073304 012704 000204
15983 073310 020405
15984 073312 001445
15985 073314 104000
15986
15987 073316 000000
15988 073320 000000
15989 073322 000000
15990 073324 000000
15991
15992 073326 000000
15993 073330 000000
15994 073332 000000
15995 073334 000000
15996
15997 073336 000577
15998 073340 177776
15999 073342 177777
16000 073344 177776
16001 073346 005201
16002 073350 000000
16003 073352 000000
16004 073354 000000
16005 073356 100577
16006 073360 177776
16007 073362 177777
16008 073364 177776
16009 073366 100577
16010 073370 177776
16011 073372 000000
16012 073374 000000
16013 073376 000252
16014 073400 125252
16015 073402 125252
16016 073404 125252
16017
16018 073406 000577
16019 073410 177776
16020 073412 000000
16021 073414 000000
16022 073416 000577
16023 073420 177777
16024 073422 000000
16025 073424 000000
16026
16027 073426
16028 073426 004767 031140
16029
16030
16031
16032

HX33: CMP (R1)+,(R0)+ ;IS IT ZERO?
BEQ HX34
EMT ;
HX34: SOB R2,HX33
MOV #204,R4 ;FPS CORRECT?
CMP R4,R5
BEQ HXDONE
EMT ;

HXDATO: 0
0
0
0

HXP1: 0
0
0
0

HXP2: 577
177776
177777
177776

HXP3: 5201
0
0
0

HXP4: 100577
177776
177777
177776

HXP5: 100577
177776
0
0

HXP6: 252
125252
125252
125252

HXP7: 577
177776
0
0

HXP8: 577
177777
0
0

HADONE: JSR PC,,RSET

;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

16033
16034
16035
16036
16037
16038
16039
16040
16041
16042
16043
16044
16045
16046
16047
16048
16049
16050
16051
16052
16053
16054
16055
16056
16057
16058
16059
16060
16061
16062
16063
16064
16065
16066
16067
16068
16069
16070
16071
16072
16073
16074
16075
16076
16077
16078
16079
16080
16081
16082
16083
16084
16085
16086
16087
16088

073432

073432 004737 074126
073436 000000 000000 000000
073444 000000
073446 000000 000000 000000
073454 000000
073456 000200
073460 000204

073462 004737 074126
073466 000000 000000 000000
073474 000000
073476 025252
073500 052525
073502 125252
073504 052525
073506 000200
073510 000200

073512 004737 074126
073516 000000 000000 000000
073524 000000
073526 125252
073530 125252
073532 052525
073534 125252
073536 000200
073540 000210

073542 004737 074126
073546 025252
073550 052525
073552 125252
073554 052525
073556 000000 000000 000000
073564 000000
073566 000200
073570 000210

073572 004737 074126
073576 125252

```
*****
:TEST 473      CMPD TEST
*****
TS473:

:TEST THE CMPD INSTRUCTION WITH (FSRC=AC=0)
AAA1:  JSR      PC,@#CMPSUB
1$:     .WORD   0,0,0,0          ;AC0
2$:     .WORD   0,0,0,0          ;FSRC
3$:     200                ;FPS BEFORE EXECUTION
        204                ;FPS AFTER EXECUTION

:TEST CMPD WITH (AC=0) AND FSRC POSITIVE.
AAA2:  JSR      PC,@#CMPSUB
1$:     .WORD   0,0,0,0          ;AC
2$:     25252                ;FSRC
        52525
        125252
3$:     52525                ;FPS BEFORE EXECUTION
        200                ;FPS AFTER EXECUTION
        200

:TEST CMPD WITH (AC=0) AND FSRC NEGATIVE
AAA3:  JSR      PC,@#CMPSUB
1$:     .WORD   0,0,0,0          ;AC
2$:     125252                ;FSRC
        125252
        52525
3$:     125252                ;FPS BEFORE EXECUTION
        200                ;FPS AFTER EXECUTION
        210

:TEST CMPD WITH (FSRC=0) AND AC POSITIVE
AAA4:  JSR      PC,@#CMPSUB
1$:     25252                ;AC
        52525
        125252
2$:     .WORD   0,0,0,0          ;FSRC
3$:     200                ;FPS BEFORE EXECUTION
        210                ;FPS AFTER EXECUTION

:TEST CMPD WITH (FSRC=0) AND AC NEGATIVE
AAA5:  JSR      PC,@#CMPSUB
1$:     125252                ;AC
```

```
16089 073600 125252 125252
16090 073602 052525 52525
16091 073604 125252 125252
16092 073606 000000 000000 000000 2$: .WORD 0.0.0.0 ;FSRC
16093 073614 000000 000000 3$: 200 ;FPS BEFORE EXECUTION
16094 073616 000200 000200 ;FPS AFTER EXECUTION
16095 073620 000200
16096
16097 ;TEST CMPD WITH AC POSITIVE AND FSRC NEGATIVE
16098 073622 004737 074126 AAA6: JSR PC,@#CMPSUB
16099 073626 052525 1$: 52525 ;AC
16100 073630 125252 125252
16101 073632 052525 52525
16102 073634 125252 125252
16103 073636 125252 2$: 125252 ;:FSRC
16104 073640 052525 52525
16105 073642 125252 125252
16106 073644 052525 52525
16107 073646 000200 3$: 200 ;FPS BEFORE EXECUTION
16108 073650 000210 210 ;FPS AFTER EXECUTION
16109
16110
16111 ;TEST CMPD WITH AC NEGATIVE AND FSRC POSITIVE
16112 073652 004737 074126 AAA7: JSR PC,@#CMPSUB
16113 073656 125252 1$: 125252 ;AC
16114 073660 052525 52525
16115 073662 125252 125252
16116 073664 052525 52525
16117 073666 052525 2$: 52525 ;FSRC
16118 073670 125252 125252
16119 073672 052525 52525
16120 073674 125252 125252
16121 073676 000200 3$: 200 ;FPS BEFORE EXECUTION
16122 073700 000200 200 ;FPS AFTER EXECUTION
16123
16124 ;TEST CMPD WITH AC POSITIVE AND FSRC POSITIVE
16125 ;AND EAC LESS THAN EFSRC.
16126 073702 004737 074126 AAA8: JSR PC,@#CMPSUB
16127 073706 012345 1$: 12345 ;AC
16128 073710 067654 67654
16129 073712 032101 32101
16130 073714 023456 23456
16131 073716 023456 2$: 23456 ;FSRC
16132 073720 076543 76543
16133 073722 021012 21012
16134 073724 034567 34567
16135 073726 000200 3$: 200 ;FPS BEFORE EXECUTION
16136 073730 000200 200 ;FPS AFTER EXECUTION
16137
16138
16139 ;TEST CMPD WITH AC POSITIVE, FSRC POSITIVE AND EAC GREATER THAN EFSRC
16140 073732 004737 074126 AAA9: JSR PC,@#CMPSUB
16141 073736 045676 1$: 45676 ;AC
16142 073740 054321 54321
16143 073742 012345 12345
16144 073744 067654 67654
```

16145	073746	034567		2\$:	34567	;FSRC
16146	073750	065432			65432	
16147	073752	101234			101234	
16148	073754	056765			56765	
16149	073756	000200		3\$:	200	;FPS BEFORE EXECUTION
16150	073760	000210			210	;FPS AFTER EXECUTION
16151						
16152						
16153	073762	004737	074126			
16154	073766	012345				
16155	073770	067012				
16156	073772	034567				
16157	073774	012345				
16158	073776	012345		2\$:	12345	;FSRC
16159	074000	067012			67012	
16160	074002	034567			34567	
16161	074004	012345			012345	
16162	074006	000200		3\$:	200	;FPS BEFORE EXECUTION
16163	074010	000204			204	;FPS AFTER EXECUTION
16164						
16165						
16166						
16167	074012	004737	074126			
16168	074016	012345				
16169	074020	067012				
16170	074022	034567				
16171	074024	012345				
16172	074026	012345		2\$:	12345	;FSRC
16173	074030	070123			70123	
16174	074032	045670			45670	
16175	074034	123456			123456	
16176	074036	000200		3\$:	200	;FPS BEFORE EXECUTION
16177	074040	000200			200	;FPS AFTER EXECUTION
16178						
16179						
16180						
16181	074042	004737	074126			
16182	074046	054321				
16183	074050	076543				
16184	074052	021076				
16185	074054	054321				
16186	074056	054321		2\$:	54321	;FSRC
16187	074060	065432			65432	
16188	074062	107654			107654	
16189	074064	032107			32107	
16190	074066	000200		3\$:	200	;FPS BEFORE EXECUTION
16191	074070	000210			210	;FPS AFTER EXECUTION
16192						
16193						
16194						
16195	074072	004737	074126			
16196	074076	112345				
16197	074100	043210				
16198	074102	076543				
16199	074104	021076				
16200	074106	112345		2\$:	112345	;FSRC

16201 074110 054321
 16202 074112 007654
 16203 074114 032107
 16204 074116 000200
 16205 074120 000210
 16206
 16207
 16208 074122 000137 074232
 16209
 16210
 16211
 16212
 16213
 16214
 16215
 16216
 16217
 16218
 16219
 16220
 16221
 16222
 16223
 16224
 16225
 16226
 16227
 16228
 16229
 16230
 16231
 16232
 16233
 16234 074126 012601
 16235
 16236 074130 016100 000020
 16237 074134 170100
 16238
 16239 074136 010100
 16240 074140 172410
 16241
 16242 074142 010100
 16243 074144 062700 000010
 16244
 16245 074150 000240
 16246 074152 173410
 16247
 16248 074154 170205
 16249
 16250 074156 016104 000022
 16251 074162 020405
 16252 074164 020405
 16253 074166 001401
 16254 074170 104000
 16255 074172 012700 074222
 16256 074176 174010

```

54321
07654
32107
3$: 200 ;FPS BEFORE EXECUTION
    210 ;FPS AFTER EXECUTION

JMP @#AAADONE ;FINISHED CMPD TEST.

;THIS SUBROUTINE, CMPSUB, IS CALLED TO SET UP, EXECUTE
;AND CHECK THE RESULTS OF A CMPD INSTRUCTION.
;IT IS CALLED THUS:
:
:       JSR      PC,@#CMPSUB
:       ACARG: .WORD X,X,X,X ;AC OPERAND
:       FSRCARG: .WORD X,X,X,X ;FSRC OPERAND
:       FPSB: .WORD X ;FPS BEFORE EXECUTION
:       FPSA: .WORD X ;FPS AFTER EXECUTION
:       FPSE: .WORD X ;ERROR FPS
:       ERR: ERROR X ;FPS ERROR
:       CONT: ;RETURN ADDRESS

;THE OPERANDS ARE SET UP (USING ACO FOR THE AC OPERAND). THEN
;FPSB IS LOADED INTO THE FPS. THE INSTRUCTION, CMPD, IS EXECUTED.
;AFTER THE EXECUTION THE FPS IS CHECKED AGAINST FPSA. IF IT IS A MATCH
;THEN THERE WAS NO ERROR AND CONTROL IS RETURNED TO CONT. IF
;THE FPS IS INCORRECT IT IS COMPARED WITH FPSE IN AN ATTEMPT TO ANALYSE
;THE FAILURE. IF THE FPS IS THE SAME AS FPSE THEN CONTROL IS
;RETURNED TO THE ERROR CALL AT LOCATION ERR. IF THE FPS WAS
;NOT CORRECT BUT DIDN'T MATCH FPSE A GENERAL ERROR IS REPORTED
;AND CONTROL IS PASSED TO CONT.

CMPSUB: MOV      (SP)+,R1 ;PICK UP A POINTER TO THE
;ARGUMENTS.
        MOV      20(R1),R0 ;GET THE FPS BEFORE EXECUTION.
        LDFPS   R0 ;LOAD IT INTO THE FPS.

        MOV      R1,R0 ;GET ADDRESS OF AC OPERAND.
        LDD     (R0),ACO ;LOAD ACO OPERAND

        MOV      R1,R0 ;COMPUTE FSRC OPERAND
        ADD     #10,R0 ;ADDRESS

        NOP ;FOR SCOPING.
1$: CMPD      (R0),ACO ;EXECUTE THE TEST INSTRUCTION.

        STFPS   R5 ;SAVE FPS AFTER INSTRUCTION.

        MOV      22(R1),R4 ;GET EXPECTED FPS.
        CMP     R4,R5 ;WAS FPS CORRECT?
        CMP     R4,R5 ;WAS FPS CORRECT?
        BEQ     3$

        EMT

3$: MOV      #CMPTMP,R0 ;IF FPS WAS CORRECT MAKE SURE
        STD     ACO,(R0) ;ACO WAS NOT AFFECTED BY CMPD.
  
```

```
16257 074200 010102          MOV      R1,R2
16258 074202 012703 000004    MOV      #4,R3
16259 074206 022220          4$:     CMP      (R2)+,(R0)+
16260 074210 001401          BEQ     5$
16261 074212 104000          EMT
16262 074214 077304          5$:     SOB     R3,4$
16263
16264 074216 000161 000024    JMP     24(R1)          ;RETURN
16265
16266 074222 000000 000000 000000  CMPTMP: .WORD 0,0,0,0
16267 074230 000000
16268
16269
16270
16271 074232
16272 074232 004767 030334    AAADONE: JSR     PC,.RSET          ;GO INITIALIZE THE FPS AND STACK; AND
16273                                     ;SEE IF THE USER HAS EXPRESSED
16274                                     ;THE DESIRE TO CHANGE THE SOFTWARE
16275                                     ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
16276                                     ;THE USER TYPED CONTROL G?).
16277
16278
16279
16280
16281
16282
16283
16284 074236
16285
16286 074236 012704 040200    BBB0:   MOV     #40200,R4          ;SET UP FPS
16287                                     ;WITH INTERRUPTS
16288                                     ;DISABLED.
16289 074242 170104          LDFPS  R4
16290 074244 012737 074302 000244    MOV     #BBB1,R4          ;SET UP FOR ANY FP INTERRUPTS.
16291 074252 012700 074466    MOV     #BBB1,R0          ;SET UP ACO = 0
16292 074256 172410          LDD    (R0),ACO
16293 074260 012701 074466    MOV     #BBB1,R1          ;FSRC = 0
16294
16295 074264 174411          BBB1:   DIVD   (R1),ACO          ;TEST INSTRUCTION
16296
16297 074266 170205          STFPS  R5          ;GET FPS
16298 074270 170303          STST   R3          ;GET FEC
16299
16300 074272 012704 140204          MOV     #140204,R4        ;EXPECTED FPS.
16301 074276 020405          CMP     R4,R5          ;IS FPS CORRECT.
16302 074300 001401          BEQ     BBB7
16303
16304 074302 104000          BBBER1: EMT
16305 074304 012702 000004    BBB7:   MOV     #4,R2          ;EXPECTED FEC.
16306 074310 020203          CMP     R2,R3          ;IS FEC CORRECT?
16307 074312 001401          BEQ     BBB2
16308 074314 104000          EMT
16309
16310
16311 074316 012704 040200    ;TEST DIVD WITH (FSRC=0) AND TRAPS DISABLED.
16312 074322 170104          BBB2:   MOV     #40200,R4        ;LOAD FPS WITH TRAPS DISABLED.
16313                                     LDFPS  R4
```

```

16313
16314 074324 012700 074476      MOV    #BBBP2,R0      ;SET UP ACO OPERAND (NON ZERO).
16315 074330 172410              LDD    (R0),ACO
16316 074332 012700 074466      MOV    #BBBP1,R0      ;FSRC=0
16317 074336 174410      BBB3:  DIVD   (R0),ACO
16318
16319 074340 170205              STFPS  R5              ;GET FPS.
16320 074342 170303              STST   R3              ;GET FEC.
16321
16322 074344 012704 140200      MOV    #140200,R4      ;EXPECTED FPS.
16323 074350 020405              CMP    R4,R5           ;IS FPS CORRECT?
16324 074352 001401              BEQ    1$
16325 074354 104000              EMT
16326 074356 012702 000004      1$:   MOV    #4,R2         ;EXPECTED FEC.
16327 074362 020203              CMP    R2,R3           ;WAS FEC CORRECT?
16328 074364 001401              BEQ    BBB4
16329 074366 104000              EMT
16330
16331      ;TEST DIVD WITH FSRC=0) AND TRAPS ENABLED.
16332 074370 012704 000200      BBB4:  MOV    #200,R4      ;SET UP FPS. TRAP ENABLED.
16333 074374 170104              LDFPS  R4
16334
16335 074376 012700 074476      MOV    #BBBP2,R0      ;SET UP ACO OPERAND (NON ZERO).
16336 074402 172410              LDD    (R0),ACO
16337
16338 074404 012737 074424 000244      MOV    #BBB6,#FPVECT  ;SET UP FOR THE EXPECTED INTERRUPT.
16339 074412 012700 074466      MOV    #BBBP1,R0      ;FSRC=0
16340
16341 074416 174410              BBB5:  DIVD   (R0),ACO      ;TEST INSTRUCTION (SHOULD RESULT IN TRAP).
16342 074420 170000              CFCC
16343 074422 104000              EMT
16344 074424 022716 074420      BBB6:  CMP    #BBB5+2,(SP) ;TRAP TO HERE WHEN THE DIVISION BY 0
16345      ;OCCURS. FIRST SEE IF THE ADDRESS OF
16346      ;THE TRAP IS 2+THE ADDRESS OF THE TEST
16347      ;DIVD INSTRUCTION.
16348 074430 001401              BEQ    1$
16349 074432 104000              EMT
16350 074434 170205              1$:   STFPS  R5              ;GET FPS.
16351 074436 170303              STST   R3              ;GET FEC.
16352 074440 022626              CMP    (SP)+,(SP)+    ;RESET THE STACK.
16353
16354 074442 012704 100200      MOV    #100200,R4      ;EXPECTED FPS.
16355 074446 020405              CMP    R4,R5           ;IS FPS CORRECT?
16356 074450 001401              BEQ    2$
16357 074452 104000              EMT
16358 074454 012702 000004      2$:   MOV    #4,R2         ;EXPECTED FEC.
16359 074460 020203              CMP    R2,R3           ;IS FEC CORRECT?
16360 074462 001411              BEQ    BBBDONE
16361 074464 104000              EMT
16362
16363 074466 000000 000000 000000  BBBP1:  .WORD  0,0,0,0
16364 074474 000000
16365 074476 012345 054321 023456  BBBP2:  .WORD  12345,54321,23456,76543
16366 074504 076543
16367
16368
  
```


16369
16370 074506
16371 074506 004767 030060
16372
16373
16374
16375
16376
16377
16378
16379
16380
16381
16382 074512
16383
16384
16385 074512 004767 000404
16386 074516 000000 000000
16387 074522 012345 067012
16388 074526 000000 000000
16389 074532 000000
16390 074534 000004
16391
16392
16393 074536 004737 075122
16394 074542 065652 125252
16395 074546 065600 000000
16396 074552 040252 125252
16397 074556 003000
16398 074560 003000
16399
16400
16401 074562 004767 000334
16402 074566 076400 000000
16403 074572 076400 000000
16404 074576 040200 000000
16405 074602 001000
16406 074604 001000
16407
16408 074606 004737 075122
16409 074612 056777 177777
16410 074616 054200 000000
16411 074622 042777 177777
16412 074626 000000
16413 074630 000000
16414
16415
16416 074632 004737 075122
16417 074636 012377 177777
16418 074642 012300 000000
16419 074646 040252 125252
16420 074652 000000
16421 074654 000000
16422
16423
16424 074656 004737 075122

BBBDONE:
JSR PC, .RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

:TEST 475 DIVF TEST

TS475:
:CHECK DIVF WITH (AC=0).
CCC1: JSR PC, DIVFSUB
1\$: .WORD 0,0 ;AC
2\$: .WORD 12345,67012 ;FSRC
3\$: .WORD 0,0 ;RES
4\$: 0 ;FPS BEFORE EXECUTION
4 ;FPS AFTER EXECUTION
:TEST DIVF WITH AC POSITIVE, FSRC POSITIVE AND IN ROUND MODE.
CCC2: JSR PC, DIVFSUB
1\$: .WORD 65652,125252 ;AC
2\$: .WORD 65600,0 ;FSRC
3\$: .WORD 40252,125252 ;RES
4\$: 3000 ;FPS BEFORE EXECUTION.
3000 ;FPS AFTER EXECUTION.
:TEST DIVF WITH AC POSITIVE, FSRC POSITIVE.
CCC3: JSR PC, DIVFSUB
1\$: .WORD 76400,0 ;AC
2\$: .WORD 76400,0 ;FSRC
3\$: .WORD 40200,0 ;RES
4\$: 1000 ;FPS BEFORE EXECUTION.
1000 ;FPS AFTER EXECUTION.
:TEST DIVF WITH BOTH OPERANDS POSITIVE.
CCC4: JSR PC, DIVFSUB
1\$: .WORD 56777,177777 ;AC
2\$: .WORD 54200,0 ;FSRC
3\$: .WORD 42777,177777 ;RES
4\$: 0 ;FPS BEFORE EXECUTION.
0 ;FPS AFTER EXECUTION.
:TEST THE DIVF INSTRUCTION:
CCC5: JSR PC, DIVFSUB
1\$: .WORD 12377,177777 ;AC
2\$: .WORD 12300,0 ;FSRC
3\$: .WORD 40252,125252 ;RES
4\$: 0 ;FPS BEFORE EXECUTION.
0 ;FPS AFTER EXECUTION.
:TEST DIVIDE ALGORITHM. TEST ROUND CONSTANT.
CCC6: JSR PC, DIVFSUB

CJKDJB0 11/23-B CPU CLUSTER DIAG.
CJKDJB.P11 26-MAY-82 11:14

DNMAC X24.07-563 26-MAY-82^{E 9} 11:18 PAGE 315
T475 DIVF TEST

SEQ 0314

16425 074662 064600 000001

18: .WORD 64600,1 ;AC

16426 074666 066600 000000
16427 074672 036200 000001
16428 074676 000000
16429 074700 000000
16430
16431
16432 074702 004737 075122
16433 074706 034577 177776
16434 074712 023400 000000
16435 074716 051377 177776
16436 074722 000017
16437 074724 000000
16438
16439
16440
16441 074726 004737 075122
16442 074732 067652 125252
16443 074736 056500 000000
16444 074742 051343 107070
16445 074746 000000
16446 074750 000000
16447
16448
16449 074752 004737 075122
16450 074756 140400 000000
16451 074762 140500 000000
16452 074766 040052 125253
16453 074772 000000
16454 074774 000000
16455
16456
16457 074776 004737 075122
16458 075002 160077 000000
16459 075006 040277 000000
16460 075012 160000 000000
16461 075016 000007
16462 075020 000010
16463
16464
16465 075022 004737 075122
16466 075026 040400 000000
16467 075032 140500 000000
16468 075036 140052 125253
16469 075042 000017
16470 075044 000010
16471
16472
16473
16474 075046 004737 075122
16475 075052 060100 000001
16476 075056 040300 000000
16477 075062 060000 000000
16478 075066 000052
16479 075070 000040
16480
16481

2\$: .WORD 66600,0 ;FSRC
3\$: .WORD 36200,1 ;RES
4\$: 0 ;FPS BEFORE EXECUTION.
;FPS AFTER EXECUTION.
:TEST DIVF.
CCC7: JSR PC,@DIVFSUB
1\$: .WORD 34577,177776 ;AC
2\$: .WORD 23400,0 ;FSRC
3\$: .WORD 51377,177776 ;RES
4\$: 17 ;FPS BEFORE EXECUTION.
0 ;FPS AFTER EXECUTION.
:DIVF TEST.
CCC8: JSR PC,@DIVFSUB
1\$: .WORD 67652,125252 ;AC
2\$: .WORD 56500,0 ;FSRC
3\$: .WORD 51343,107070 ;RES
4\$: 0 ;FPS BEFORE EXECUTION.
0 ;FPS AFTER EXECUTION.
:DIVF WITH AC NEGATIVE, FSRC NEGATIVE.
CCC9: JSR PC,@DIVFSUB
1\$: .WORD 140400,0 ;AC
2\$: .WORD 140500,0 ;FSRC
3\$: .WORD 040052,125253 ;RES
4\$: 0 ;FPS BEFORE EXECUTION.
0 ;FPS AFTER EXECUTION.
:DIVF WITH AC NEGATIVE AND FSRC POSITIVE.
CCC10: JSR PC,@DIVFSUB
1\$: .WORD 160077,0 ;AC
2\$: .WORD 40277,0 ;FSRC
3\$: .WORD 160000,0 ;RES
4\$: 7 ;FPS BEFORE EXECUTION.
10 ;FPS AFTER EXECUTION.
:DIVF WITH AC POSITIVE AND FSRC NEGATIVE.
CCC11: JSR PC,@DIVFSUB
1\$: .WORD 40400,0 ;AC
2\$: .WORD 140500,0 ;FSRC
3\$: .WORD 140052,125253 ;RES
4\$: 17 ;FPS BEFORE EXECUTION.
10 ;FPS AFTER EXECUTION.
:TEST DIVF BOTH OPERANDS POSITIVE AND TRUNCATE MODE.
CCC12: JSR PC,@DIVFSUB
1\$: .WORD 60100,1 ;AC
2\$: .WORD 40300,0 ;FSRC
3\$: .WORD 60000,0 ;RES
4\$: 52 ;FPS BEFORE EXECUTION.
40 ;FPS AFTER EXECUTION.
:DIVF WITH POSITIVE OPERANDS AND ROUND MODE.

```

16482 075072 004767 000024
16483 075076 060100 000001
16484 075102 040300 000000
16485 075106 060000 000001
16486 075112 000005 000001
16487 075114 000000
16488
16489 075116 000137 075240
16490
16491
16492
16493
16494
16495
16496
16497
16498
16499
16500
16501
16502
16503
16504
16505
16506
16507
16508
16509
16510
16511
16512
16513
16514
16515
16516
16517 075122 012601
16518 075124 012700 000200
16519 075130 170100
16520 075132 010100
16521 075134 172410
16522 075136 016100 000014
16523 075142 170100
16524 075144 010100
16525 075146 062700 000004
16526
16527 075152 174410
16528
16529 075154 170204
16530 075156 012700 000200
16531 075162 170100
16532
16533 075164 012700 075230
16534 075170 174010
16535 075172 021061 000010
16536 075176 001401
16537 075200 104000
  
```

```

CCC13: JSR      PC,DIVFSUB
1$:     .WORD   60100,1      ;AC
2$:     .WORD   40300,0      ;FSRC
3$:     .WORD   60000,1      ;RES
4$:     5
        0                    ;FPS BEFORE EXECUTION.
                               ;FPS AFTER EXECUTION.

        JMP      @#CCCCDONE   ;GO TO NEXT TEST.
  
```

:THIS SUBROUTINE, DIVFSUB, IS CALLED TO SET UP, EXECUTE
 :AND CHECK THE RESULT OF A DIVF INSTRUCTION. IT IS CALLED THUS:

```

:
:      JSR      PC,@#DIVFSUB
:
:      ACARG:  .WORD   X,X      ;AC OPERAND
:      FSRCARG: .WORD   X,X      ;FSRC OPERAND
:      RES:    .WORD   X,X      ;EXPECTED RESULT
:      FPSB:   .WORD   X        ;FPS BEFORE EXECUTION
:      FPSA:   .WORD   X        ;FPS AFTER EXECUTION
:      ERRES:  .WORD   X,X      ;ERROR RESULT
:      ERR:    ERROR  X        ;RESULT ERROR
:      CONT:   ;RETURN ADDRESS
  
```

:THE OPERANDS ARE SET UP (USING ACO FOR THE AC OPERAND). THEN
 :FPSB IS LOADED INTO THE FPS. THE INSTRUCTION, DIVF IS EXECUTED.
 :AFTER THE EXECUTION THE RESULT IS CHECKED AGAINST THE
 :EXPECTED CORRECT RESULT, RES. IF IT IS CORRECT THEN THE FPS
 :IS CHECKED WITH THE EXPECTED CORRECT FPS, FPSA. IF THE FPS WAS
 :INCORRECT THEN IT IS REPORTED. IF THE RESULT WAS INCORRECT IT
 :IS COMPARED WITH ERRES IN AN ATTEMPT TO ANALYSE THE ERROR. IF
 :THE INCORRECT RESULT MATCHED ERRES THEN CONTROL IS PASSED TO
 :THE ERROR CALL AT ERR. IF THE INCORRECT RESULT DID NOT MATCH ERRES
 :THEN THE FAILURE IS REPORTED IN DIVFSUB AND CONTROL IS PASSED TO
 :CONT. IF NO ERRORS ARE DETECTED THEN DIVFSUB RETURNS CONTROL
 :TO CONT.

```

DIVFSUB: MOV      (SP)+,R1      ;GET A POINTER TO THE ARGUMENTS.
        MOV      #200,R0   ;SET FD MODE.
        LDFPS   R0
        MOV      R1,R0     ;LOAD THE AC OPERAND.
        LDD     (R0),ACO
        MOV      14(R1),R0 ;LOAD THE FPS
        LDFPS   R0
        MOV      R1,R0
        ADD     #4,R0      ;ESTABLISH A POINTER TO FSRC.

1$:     DIVF    (R0),ACO    ;TEST INSTRUCTION.

        STFPS   R4        ;GET THE FPS.
        MOV      #200,R0   ;SET FD MODE
        LDFPS   R0

        MOV      #DIVFT,R0 ;GET THE RESULT OF THE DIVF.
        STD     ACO,(R0)
        CMP     (R0),10(R1) ;IS THE RESULT CORRECT?
        BEQ    2$
        EMT
  
```

```
16538 075202 026061 000002 000012 2$: CMP 2(R0),12(R1)
16539 075210 001401 BEQ 3$
16540 075212 104000 EMT
16541 075214 026104 000016 3$: CMP 16(R1),R4 ;IS FPS CORRECT?
16542 075220 001401 BEQ 4$
16543 075222 104000 EMT
16544 075224 000161 000020 4$: JMP 20(R1) ;IF NO ERRORS OCCURRED RETURN.
16545
16546 075230 000000 000000 000000 DIVFT: .WORD 0,0,0,0
16547 075236 000000
16548
16549 075240
16550 075240 004767 027326 CCCDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
16551 ;SEE IF THE USER HAS EXPRESSED
16552 ;THE DESIRE TO CHANGE THE SOFTWARE
16553 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
16554 ;THE USER TYPED CONTROL G?).
16555
16556
16557
16558
16559 ;*****
16560 ;TEST 476 DIVD TEST
16561 ;*****
16562 TS476:
16563 ;DIVD TEST WITH POSITIVE OPERANDS AND IN ROUND MODE.
16564 075244 004737 075610 DDD1: JSR PC,@DIVDSUB
16565 075250 034277 000000 000000 1$: .WORD 34277,0,0,0 ;AC
16566 075256 000000
16567 075260 040277 000000 000000 2$: .WORD 40277,0,0,0 ;FSRC
16568 075266 000000
16569 075270 034200 000000 000000 3$: .WORD 34200,0,0,0 ;RES
16570 075276 000000
16571 075300 000200 4$: 200 ;FPS BEFORE EXECUTION.
16572 075302 000200 200 ;FPS AFTER EXECUTION.
16573
16574 ;DIVD WITH AC NEGATIVE AND FSRC POSITIVE IN TRUNCATE MODE.
16575 075304 004737 075610 DDD2: JSR PC,@DIVDSUB
16576 075310 134277 000000 000000 1$: .WORD 134277,0,0,0 ;AC
16577 075316 000000
16578 075320 040277 000000 000000 2$: .WORD 40277,0,0,0 ;FSRC
16579 075326 000000
16580 075330 134200 000000 000000 3$: .WORD 134200,0,0,0 ;RES
16581 075336 000000
16582 075340 000207 4$: 207 ;FPS BEFORE EXECUTION.
16583 075342 000210 210 ;FPS AFTER EXECUTION.
16584
16585 ;DIVD TEST WITH OPERANDS BOTH NEGATIVE AND IN TRUNCATE MODE.
16586 075344 004767 000240 DDD3: JSR PC,DIVDSUB
16587 075350 134300 000000 000000 1$: .WORD 134300,0,0,1 ;AC
16588 075356 000001
16589 075360 140300 000000 000000 2$: .WORD 140300,0,0,0 ;FSRC
16590 075366 000000
16591 075370 034200 000000 000000 3$: .WORD 34200,0,0,0 ;RES
16592 075376 000000
16593 075400 000250 4$: 250 ;FPS BEFORE EXECUTION.
```

```
16594 075402 000240          240          ;FPS AFTER EXECUTION.
16595
16596          ;DIVD WITH AC POSITIVE AND FSRC NEGATIVE IN ROUND MODE.
16597 075404 004737 075610   DDD4: JSR PC, @DIVDSUB
16598 075410 034300 000000 000000 1$: .WORD 34300,0,0,1 ;AC
16599 075416 000001
16600 075420 140300 000000 000000 2$: .WORD 140300,0,0,0 ;FSRC
16601 075426 000000
16602 075430 134200 000000 000000 3$: .WORD 134200,0,0,1 ;RES
16603 075436 000001
16604 075440 000207          4$: 207          ;FPS BEFORE EXECUTION.
16605 075442 000210          210          ;FPS AFTER EXECUTION.
16606
16607          ;DIVD TEST.
16608 075444 004737 075610   DDD5: JSR PC, @DIVDSUB
16609 075450 100400 000000 000000 1$: .WORD 100400,0,0,0 ;AC
16610 075456 000000
16611 075460 000500 000000 000000 2$: .WORD 500,0,0,0 ;FSRC
16612 075466 000000
16613 075470 140052 125252          3$: .WORD 140052,125252 ;RES
16614 075474 125252 125252          .WORD 125252,125252
16615 075500 007647          4$: 7647          ;FPS BEFORE EXECUTION.
16616 075502 007650          7650          ;FPS AFTER EXECUTION.
16617
16618          ;DIVD TEST WITH AC POSITIVE AND FSRC NEGATIVE IN ROUND MODE.
16619
16620 075504 004737 075610   DDD6: JSR PC, @DIVDSUB
16621 075510 000400 000000 000000 1$: .WORD 400,0,0,0 ;AC
16622 075516 000000
16623 075520 100500 000000 000000 2$: .WORD 100500,0,0,0 ;FSRC
16624 075526 000000
16625 075530 140052 125252          3$: .WORD 140052,125252 ;RES
16626 075534 125252 125253          .WORD 125252,125253
16627 075540 007707          4$: 7707          ;FPS BEFORE EXECUTION.
16628 075542 007710          7710          ;FPS AFTER EXECUTION.
16629
16630          ;DIVD TEST.
16631 075544 004737 075610   DDD7: JSR PC, @DIVDSUB
16632 075550 170360 170360          1$: .WORD 170360,170360 ;AC
16633 075554 170360 170360          .WORD 170360,170360
16634 075560 170360 170360          2$: .WORD 170360,170360 ;FSRC
16635 075564 170360 170360          .WORD 170360,170360
16636 075570 040200 000000 000000 3$: .WORD 40200,0,0,0 ;RES
16637 075576 000000
16638 075600 007717          4$: 7717          ;FPS BEFORE EXECUTION.
16639 075602 007700          7700          ;FPS AFTER EXECUTION.
16640
16641 075604 000137 075732          JMP @DDDDONE ;GO TO NEXT TEST.
16642
16643
16644          ;THIS SUBROUTINE, DIVDSUB, IS CALLED TO SET UP, EXECUTE
16645          ;AND CHECK THE RESULT OF A DIVD INSTRUCTION. IT IS CALLED THUS:
16646          ;
16647          ;
16648          ;          JSR PC, @DIVDSUB
16649          ;          ACARG: .WORD X,X,X,X ;AC OPERAND
          ;          FSRCARG: .WORD X,X,X,X ;FSRC OPERAND
```

16650
 16651
 16652
 16653
 16654
 16655
 16656
 16657
 16658
 16659
 16660
 16661
 16662
 16663
 16664
 16665
 16666
 16667
 16668
 16669
 16670 075610 012601
 16671 075612 012700 000200
 16672 075616 170100
 16673
 16674 075620 010100
 16675 075622 172410
 16676 075624 016100 000030
 16677 075630 170100
 16678
 16679 075632 010100
 16680 075634 062700 000010
 16681
 16682 075640 174410
 16683 075642 170204
 16684 075644 012700 000200
 16685 075650 170100
 16686 075652 012700 075722
 16687 075656 174010
 16688 075660 010102
 16689 075662 062702 000020
 16690 075666 012703 075722
 16691 075672 012705 000004
 16692 075676 022223
 16693 075700 001401
 16694 075702 104000
 16695 075704 077504
 16696
 16697 075706 026104 000032
 16698 075712 001401
 16699 075714 104000
 16700 075716 000161 000034
 16701 075722 000000 000000 000000
 16702 075730 000000
 16703
 16704 075732
 16705 075732 004767 026634

```

: RES: .WORD X,X,X,X ;EXPECTED RESULT
: FPSB: .WORD X ;FPS BEFORE EXECUTION
: FPSA: .WORD X ;FPS AFTER EXECUTION
: ERRES: .WORD X,X,X,X ;ERROR RESULT
: ERR: ERROR X ;RESULT ERROR
: CONT: ;RETURN ADDRESS

```

```

:THE OPERANDS ARE SET UP (USING ACO FOR THE AC OPERAND). THEN
:FPSB IS LOADED INTO THE FPS. THE INSTRUCTION, DIVD IS EXECUTED.
:AFTER THE EXECUTION THE RESULT IS CHECKED AGAINST THE
:EXPECTED CORRECT RESULT, RES. IF IT IS CORRECT THEN THE FPS
:IS CHECKED WITH THE EXPECTED CORRECT FPS, FPSA. IF THE FPS WAS
:INCORRECT THEN IT IS REPORTED. IF THE RESULT WAS INCORRECT IT
:IS COMPARED WITH ERRES IN AN ATTEMPT TO ANALYSE THE ERROR. IF
:THE INCORRECT RESULT MATCHED ERRES THEN CONTROL IS PASSED TO
:THE ERROR CALL AT ERR. IF THE INCORRECT RESULT DID NOT MATCH ERRES
:THEN THE FAILURE IS REPORTED IN DIVDSUB AND CONTROL IS PASSED TO
:CONT. IF NO ERRORS ARE DETECTED THEN DIVDSUB RETURNS CONTROL
:TO CONT.

```

```

DIVDSUB: MOV (SP)+,R1 ;GET A POINTER TO THE ARGUMENTS.
MOV #200,R0 ;SET FD MODE.
LDFPS R0
MOV R1,R0 ;SET UP THE ACO OPERAND.
LDD (R0),ACO
MOV 30(R1),R0 ;LOAD THE FPS.
LDFPS R0
MOV R1,R0 ;ESTABLISH A POINTER TO FSRC.
ADD #10,R0
1$: DIVD (R0),ACO ;EXECUTE THE TEST INSTRUCTION.
STFPS R4 ;GET THE FPS.
MOV #200,R0 ;SET FD MODE.
LDFPS R0
MOV #DIVDT,R0 ;GET THE RESULT.
STD ACO,(R0)
MOV R1,R2 ;CHECK THE RESULT.
ADD #20,R2
MOV #DIVDT,R3
MOV #4,R5
2$: CMP (R2)+,(R3)+
BEQ 3$
EMT
3$: SOB R5,2$
CMP 32(R1),R4 ;IS FPS CORRECT?
BEQ 4$
EMT
4$: JMP 34(R1) ;RETURN.
DIVDT: .WORD 0,0,0,0
DDDDONE: JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND

```

16706
16707
16708
16709
16710
16711
16712
16713
16714
16715
16716 075736
16717
16718
16719 075736 004737 076346
16720 075742 000000 000000
16721 075746 000000 000000
16722 075752 000000 000000
16723 075756 007517
16724 075760 007504
16725
16726
16727 075762 004737 076346
16728 075766 071625 034435
16729 075772 000000 000000
16730 075776 000000 000000
16731 076002 000013
16732 076004 000004
16733
16734
16735 076006 004737 076346
16736 076012 000000 000000
16737 076016 071625 153443
16738 076022 000000 000000
16739 076026 007500
16740 076030 007504
16741
16742
16743 076032 004737 076346
16744 076036 040200 000000
16745 076042 040177 177777
16746 076046 040177 177777
16747 076052 000017
16748 076054 000000
16749
16750
16751 076056 004767 000264
16752 076062 040177 177777
16753 076066 040200 000000
16754 076072 040177 177777
16755 076076 000040
16756 076100 000040
16757
16758
16759 076102 004737 076346
16760 076106 040100 000000
16761 076112 040100 000000

:SEE IF THE USER HAS EXPRESSED
:THE DESIRE TO CHANGE THE SOFTWARE
:VIRTUAL CONSOLE SWITCH REGISTER (HAS
:THE USER TYPED CONTROL G?).

:TEST 477 MULF TEST

TS477:

:MULF WITH (FSRC=AC=0)
EEE1: JSR PC,@MULFSUB
1\$: .WORD 0,0 :AC
2\$: .WORD 0,0 :FSRC
3\$: .WORD 0,0 :RES
4\$: 7517 :FPS BEFORE EXECUTION.
7504 :FPS AFTER EXECUTION.

:MULF WITH (FSRC=0).
EEE2: JSR PC,@MULFSUB
1\$: .WORD 71625,34435 :AC
2\$: .WORD 0,0 :FSRC
3\$: .WORD 0,0 :RES
4\$: 13 :FPS BEFORE EXECUTION.
4 :FPS AFTER EXECUTION.

:MULF WITH (AC=0)
EEE3: JSR PC,@MULFSUB
1\$: .WORD 0,0 :AC
2\$: .WORD 071625,153443 :FSRC
3\$: .WORD 0,0 :RES
4\$: 7500 :FPS BEFORE EXECUTION.
7504 :FPS AFTER EXECUTION.

:MULF WITH AC POSITIVE AND FSRC POSITIVE IN ROUND MODE.
EEE4: JSR PC,@MULFSUB
1\$: .WORD 40200,0 :AC
2\$: .WORD 40177,-1 :FSRC
3\$: .WORD 40177,-1 :RES
4\$: 17 :FPS BEFORE EXECUTION.
0 :FPS AFTER EXECUTION.

:MULF WITH AC POSITIVE AND FSRC POSITIVE IN TRUNCATE MODE.
EEE5: JSR PC,MULFSUB
1\$: .WORD 40177,-1 :AC
2\$: .WORD 40200,0 :FSRC
3\$: .WORD 40177,-1 :RES
4\$: 40 :FPS BEFORE EXECUTION.
40 :FPS AFTER EXECUTION.

:MULF WITH BOTH OPERANDS POSITIVE NORMALIZE TEST.
EEE6: JSR PC,@MULFSUB
1\$: .WORD 40100,0 :AC
2\$: .WORD 40100,0 :FSRC

16762 076116 040020 000000
16763 076122 000012
16764 076124 000000
16765
16766
16767 076126 004737 076346
16768 076132 017500 000000
16769 076136 023652 125252
16770 076142 003177 177777
16771 076146 007417
16772 076150 007400
16773
16774
16775 076152 004737 076346
16776 076156 040342 000000
16777 076162 176542 000000
16778 076166 176707 102000
16779 076172 000007
16780 076174 000010
16781
16782
16783 076176 004737 076346
16784 076202 140200 000000
16785 076206 007417 007417
16786 076212 107417 007417
16787 076216 000000
16788 076220 000010
16789
16790
16791 076222 004737 076346
16792 076226 144600 000000
16793 076232 154000 000000
16794 076236 060400 000000
16795 076242 000017
16796 076244 000000
16797
16798
16799 076246 004737 076346
16800 076252 140300 000000
16801 076256 160000 000001
16802 076262 060100 000002
16803 076266 000010
16804 076270 000000
16805
16806
16807 076272 004737 076346
16808 076276 060000 000001
16809 076302 140300 000000
16810 076306 160100 000001
16811 076312 007547
16812 076314 007550
16813
16814
16815 076316 004737 076346
16816 076322 040277 000000
16817 076326 060000 000001

3\$: .WORD 40020,0 ;RES
4\$: 12 ;FPS BEFORE EXECUTION.
0 ;FPS AFTER EXECUTION.

:MULF WITH BOTH OPERANDS POSITIVE IN ROUND MODE.
EEE7: JSR PC,@MULFSUB
1\$: .WORD 17500,0 ;AC
2\$: .WORD 23652,125252 ;FSRC
3\$: .WORD 3177,-1 ;RES
4\$: 7417 ;FPS BEFORE EXECUTION.
7400 ;FPS AFTER EXECUTION.

:MULF WITH AC POSITIVE AND FSRC NEGATIVE IN ROUND MODE.
EEE8: JSR PC,@MULFSUB
1\$: .WORD 40342,0 ;AC
2\$: .WORD 176542,0 ;FSRC
3\$: .WORD 176707,102000 ;RES
4\$: 7 ;FPS BEFORE EXECUTION.
10 ;FPS AFTER EXECUTION.

:MULF WITH AC NEGATIVE AND FSRC POSITIVE IN ROUND MODE.
EEE9: JSR PC,@MULFSUB
1\$: .WORD 140200,0 ;AC
2\$: .WORD 7417,7417 ;FSRC
3\$: .WORD 107417,7417 ;RES
4\$: 0 ;FPS BEFORE EXECUTION.
10 ;FPS AFTER EXECUTION.

:MULF WITH BOTH OPERANDS NEGATIVE IN ROUND MODE.
EEE10: JSR PC,@MULFSUB
1\$: .WORD 144600,0 ;AC
2\$: .WORD 154000,0 ;FSRC
3\$: .WORD 60400,0 ;RES
4\$: 17 ;FPS BEFORE EXECUTION.
0 ;FPS AFTER EXECUTION.

:MULF BOTH OPERANDS NEGATIVE IN ROUND MODE.
EEE11: JSR PC,@MULFSUB
1\$: .WORD 140300,0 ;AC
2\$: .WORD 160000,1 ;FSRC
3\$: .WORD 60100,2 ;RES
4\$: 10 ;FPS BEFORE EXECUTION.
0 ;FPS AFTER EXECUTION.

:MULF WITH AC POSITIVE AND FSRC NEGATIVE IN TRUNCATE MODE.
EEE12: JSR PC,@MULFSUB
1\$: .WORD 60000,1 ;AC
2\$: .WORD 140300,0 ;FSRC
3\$: .WORD 160100,1 ;RES
4\$: 7547 ;FPS BEFORE EXECUTION.
7550 ;FPS AFTER EXECUTION.

:MULF WITH AC POSITIVE AND FSRC POSITIVE IN ROUND MODE.
EEE13: JSR PC,@MULFSUB
1\$: .WORD 40277,0 ;AC
2\$: .WORD 60000,1 ;FSRC

16818 076332 060077 000001
 16819 076336 000014
 16820 076340 000000
 16821
 16822 076342 000167 000116
 16823
 16824
 16825
 16826
 16827
 16828
 16829
 16830
 16831
 16832
 16833
 16834
 16835
 16836
 16837
 16838
 16839
 16840
 16841
 16842
 16843
 16844
 16845
 16846
 16847
 16848
 16849
 16850 076346 012601
 16851 076350 012700 000200
 16852 076354 170100
 16853 076356 010100
 16854 076360 172410
 16855 076362 016100 000014
 16856 076366 170100
 16857 076370 010100
 16858 076372 062700 000004
 16859
 16860 076376 171010
 16861
 16862 076400 170204
 16863 076402 012700 000200
 16864 076406 170100
 16865
 16866 076410 012700 076454
 16867 076414 174010
 16868 076416 021061 000010
 16869 076422 001401
 16870 076424 104000
 16871 076426 026061 000002 000012 2\$:
 16872 076434 001401
 16873 076436 104000

3\$: .WORD 60077,1 ;RES
 4\$: 14 ;FPS BEFORE EXECUTION.
 0 ;FPS AFTER EXECUTION.
 JMP EEEDONE ;GO TO THE NEXT TEST.

: THIS SUBROUTINE, MULFSUB, IS CALLED TO SET UP, EXECUTE
 : AND CHECK THE RESULT OF A MULF INSTRUCTION. IT IS CALLED THUS:

```

:
:          JSR      PC,@MULFSUB
:          ACARG:  .WORD  X,X          ;AC OPERAND
:          FSRCARG: .WORD  X,X          ;FSRC OPERAND
:          RES:    .WORD  X,X          ;EXPECTED RESULT
:          FPSB:   .WORD  X           ;FPS BEFORE EXECUTION
:          FPSA:   .WORD  X           ;FPS AFTER EXECUTION
:          ERRES:  .WORD  X,X         ;ERRJR RESULT
:          ERR:    ERROR              ;RESULT ERROR
:          CONT:   ;RETURN ADDRESS
  
```

: THE OPERANDS ARE SET UP (USING ACO FOR THE AC OPERAND). THEN
 : FPSB IS LOADED INTO THE FPS. THE INSTRUCTION, MULF IS EXECUTED.
 : AFTER THE EXECUTION THE RESULT IS CHECKED AGAINST THE
 : EXPECTED CORRECT RESULT, RES. IF IT IS CORRECT THEN THE FPS
 : IS CHECKED WITH THE EXPECTED CORRECT FPS, FPSA. IF THE FPS WAS
 : INCORRECT THEN IT IS REPORTED. IF THE RESULT WAS INCORRECT IT
 : IS COMPARED WITH ERRES IN AN ATTEMPT TO ANALYSE THE ERROR. IF
 : THE INCORRECT RESULT MATCHED ERRES THEN CONTROL IS PASSED TO
 : THE ERROR CALL AT ERR. IF THE INCORRECT RESULT DID NOT MATCH ERRES
 : THEN THE FAILURE IS REPORTED IN MULFSUB AND CONTROL IS PASSED TO
 : CONT. IF NO ERRORS ARE DETECTED THEN MULFSUB RETURNS CONTROL
 : TO CONT.

```

MULFSUB:  MOV      (SP)+,R1          ;GET A POINTER TO THE ARGUMENTS.
:          MOV      #200,R0        ;SET FD MODE.
:          LDFPS   R0
:          MOV      R1,R0          ;LOAD THE AC OPERAND.
:          LDD     (R0),ACO
:          MOV      14(R1),R0      ;LOAD THE FPS
:          LDFPS   R0
:          MOV      R1,R0
:          ADD     #4,R0           ;ESTABLISH A POINTER TO FSRC.
1$:      MULF     (R0),ACO        ;TEST INSTRUCTION.
:          STFPS   R4             ;GET THE FPS.
:          MOV      #200,R0        ;SET FD MODE
:          LDFPS   R0
:          MOV      #MULFT,R0     ;GET THE RESULT OF THE MULF.
:          STD     ACO,(R0)
:          CMP     (R0),10(R1)    ;IS THE RESULT CORRECT?
:          BEQ     2$
:          EMT
:          CMP     2(R0),12(R1)
:          BEQ     3$
:          EMT
  
```

```
16874 076440 026104 000016 3$: CMP 16(R1),R4 ;IS FPS CORRECT?
16875 076444 001401 4$
16876 076446 104000 EMT ;
16877 076450 000161 000020 4$: JMP 20(R1) ;IF NO ERRORS OCCURRED RETURN.
16878
16879 076454 000000 000000 000000 MULFT: .WORD 0,0,0,0
16880 076462 000000
16881
16882 076464 EEEDONE:
16883 076464 004767 026102 JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
16884 ;SEE IF THE USER HAS EXPRESSED
16885 ;THE DESIRE TO CHANGE THE SOFTWARE
16886 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
16887 ;THE USER TYPED CONTROL G?).
16888
16889
16890
16891
16892
16893
16894 076470 ;*****
16895 ;TEST 500 MULD TEST
16896 ;*****
16897 ;MULD TEST WITH AC POSITIVE AND FSRC POSITIVE.
16898 076470 004737 076674 FFF1: JSR PC,MULDSUB
16899 076474 040200 000000 000000 1$: .WORD 40200,0,0,0 ;AC
16900 076504 023777 177777 177777 2$: .WORD 23777,-1,-1,-1 ;FSRC
16901 076512 177777
16902 076514 023777 177777 177777 3$: .WORD 23777,-1,-1,-1 ;RES
16903 076522 177777
16904 076524 000217 4$: 217 ;FPS BEFORE EXECUTION.
16905 076526 000200 200 ;FPS AFTER EXECUTION.
16906
16907 ;MULD TEST WITH BOTH OPERANDS POSITIVE TRUNCATION TEST.
16908 076530 004767 000140 FFF2: JSR PC,MULDSUB
16909 076534 065400 000000 000000 1$: .WORD 65400,0,0,1 ;AC
16910 076542 000001
16911 076544 037577 177777 177777 2$: .WORD 37577,-1,-1,-2 ;FSRC
16912 076552 177776
16913 076554 064777 177777 177777 3$: .WORD 64777,-1,-1,-1 ;RES
16914 076562 177777
16915 076564 000247 4$: 247 ;FPS BEFORE EXECUTION.
16916 076566 000240 240 ;FPS AFTER EXECUTION.
16917
16918 ;MULD TEST WITH BOTH OPERANDS NEGATIVE IN ROUND MODE.
16919 076570 004737 076674 FFF3: JSR PC,MULDSUB
16920 076574 137577 177777 177777 1$: .WORD 137577,-1,-1,-2 ;AC
16921 076602 177776
16922 076604 165400 000000 000000 2$: .WORD 165400,0,0,1 ;FSRC
16923 076612 000001
16924 076614 065000 000000 000000 3$: .WORD 65000,0,0,0 ;RES
16925 076622 000000
16926 076624 007717 4$: 7717 ;FPS BEFORE EXECUTION.
16927 076626 007700 7700 ;FPS AFTER EXECUTION.
16928
16929 ;MULD TEST WITH AC POSITIVE AND FSRC NEGATIVE IN ROUND MODE.
```

16930 076630 004737 076674
 16931 076634 017500 000000 000000
 16932 076642 00^000
 16933 076644 12,652 125252
 16934 076650 125252 125252
 16935 076654 103177 177777 177777
 16936 07666 177777
 16937 076664 000200
 16938 076666 000210

FFF4: JSR PC,@MULDSUB
 1\$: .WORD 17500,0,0,0 ;AC
 2\$: .WORD 123652,125252 ;FSRC
 .WORD 125252,125252
 3\$: .WORD 103177,-1,-1,-1 ;RES
 4\$: 200 ;FPS BEFORE EXECUTION.
 210 ;FPS AFTER EXECUTION.

16939
 16940 076670 000167 000122

JMP FFFDONE

; THIS SUBROUTINE, MULDSUB, IS CALLED TO SET UP, EXECUTE
 ; AND CHECK THE RESULT OF A MULDT INSTRUCTION. IT IS CALLED THUS::

16941
 16942
 16943
 16944
 16945
 16946
 16947
 16948
 16949
 16950
 16951
 16952
 16953
 16954
 16955
 16956
 16957
 16958
 16959
 16960
 16961
 16962
 16963
 16964
 16965
 16966

```

:
:      JSR      PC,@MULDSUB
:      ACARG:  .WORD  X,X,X,X      ;AC OPERAND
:      FSRCARG: .WORD  X,X,X,X      ;FSRC OPERAND
:      RES:    .WORD  X,X,X,X      ;EXPECTED RESULT
:      FPSB:   .WORD  X              ;FPS BEFORE EXECUTION
:      FPSA:   .WORD  X              ;FPS AFTER EXECUTION
:      ERRES:  .WORD  X,X,X,X      ;ERROR RESULT
:      ERR:    ERROR  X              ;RESULT ERROR
:      CONT:   ;RETURN ADDRESS
  
```

; THE OPERANDS ARE SET UP (USING ACO FOR THE AC OPERAND). THEN
 ; FPSB IS LOADED INTO THE FPS. THE INSTRUCTION, MULDT IS EXECUTED.
 ; AFTER THE EXECUTION THE RESULT IS CHECKED AGAINST THE
 ; EXPECTED CORRECT RESULT, RES. IF IT IS CORRECT THEN THE FPS
 ; IS CHECKED WITH THE EXPECTED CORRECT FPS, FPSA. IF THE FPS WAS
 ; INCORRECT THEN IT IS REPORTED. IF THE RESULT WAS INCORRECT IT
 ; IS COMPARED WITH ERRES IN AN ATTEMPT TO ANALYSE THE ERROR. IF
 ; THE INCORRECT RESULT MATCHED ERRES THEN CONTROL IS PASSED TO
 ; THE ERROR CALL AT ERR. IF THE INCORRECT RESULT DID NOT MATCH ERRES
 ; THEN THE FAILURE IS REPORTED IN MULDSUB AND CONTROL IS PASSED TO
 ; CONT. IF NO ERRORS ARE DETECTED THEN MULDSUB RETURNS CONTROL
 ; TO CONT.

16967 076674 012601
 16968 076676 012700 000200
 16969 076702 170100
 16970
 16971 076704 010100
 16972 076706 172410
 16973 076710 016100 000030
 16974 076714 170100
 16975
 16976 076716 010100
 16977 076720 062700 000010
 16978
 16979 076724 171010
 16980
 16981 076726 170204
 16982 076730 012700 000200
 16983 076734 170100
 16984
 16985 076736 012700 077006

```

MULDSUB:  MOV      (SP)+,R1      ;GET A POINTER TO THE ARGUMENTS.
          MOV      #200,R0      ;SET FD MODE.
          LDFPS   R0
          MOV      R1,R0      ;SET UP THE ACO OPERAND.
          LDD     (R0),ACO
          MOV      30(R1),R0    ;LOAD THE FPS.
          LDFPS   R0
          MOV      R1,R0      ;ESTABLISH A POINTER TO FSRC.
          ADD     #10,R0
          1$:  MULDT  (R0),ACO    ;EXECUTE THE TEST INSTRUCTION.
          STFPS   R4          ;GET THE FPS.
          MOV      #200,R0      ;SET FD MODE.
          LDFPS   R0
          MOV      #MULDT,R0    ;GET THE RESULT.
  
```

16986 076742 174010
16987 076744 010102
16988 076746 062702 000020
16989 076752 012703 077006
16990 076756 012705 000004
16991 076762 022223
16992 076764 001401
16993 076766 104000
16994 076770 077504
16995
16996 076772 026104 000032
16997 076776 001401
16998 077000 104000
16999 077002 000161 000034
17000
17001 077006 000000 000000 000000
17002 077014 000000
17003 077016
17004 077016 004767 025550
17005
17006
17007
17008
17009
17010
17011
17012
17013
17014
17015 077022
17016
17017
17018 077022 004737 077166
17019 077026 020200 000000
17020 077032 020000 000000
17021 077036 000000 000000
17022 077042 000000
17023 077044 000004
17024 077046 000012
17025 077050 177777
17026
17027
17028 077052 004737 077166
17029 077056 010200 000000
17030 077062 010000 000000
17031 077066 000000 000000
17032 077072 005013
17033 077074 005004
17034 077076 000012
17035 077100 177777
17036
17037 077102 004737 077166
17038 077106 060200 000000
17039 077112 060000 000000
17040 077116 000000 000000
17041 077122 000000

STD ACO,(R0)
MOV R1,R2 ;CHECK THE RESULT.
ADD #20,R2
MOV #MULDT,R3
MOV #4,R5
2\$: CMP (R2)+,(R3)+
BEQ 3\$
EMT ;
3\$: SOB R5,2\$
CMP 32(R1),R4 ;IS FPS CORRECT?
BEQ 4\$
EMT ;
4\$: JMP 34(R1) ;RETURN.
MULDT: .WORD 0,0,0,0
FFFDONE:
JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;TEST 501 UNDER\OVER FLOW, USING MULF WITH TRAPS DISABLED, TEST

TS501:

;UNDERFLOW, WITH EXPONENT OF RESULT = -129
III1: JSR PC,@#OVUNFNT
1\$: .WORD 20200,0 ;AC
2\$: .WORD 20000,0 ;FSRC
3\$: .WORD 0,0 ;RES
5\$: 0 ;FPS BEFORE EXECUTION.
4 ;FPS AFTER EXECUTION.
6\$: 12 ;FEC
-1 ;FLAG

;UNDERFLOW, WITH EXPONENT OF RESULT = -193
III2: JSR PC,@#OVUNFNT
1\$: .WORD 10200,0 ;AC
2\$: .WORD 10000,0 ;FSRC
3\$: .WORD 0,0 ;RES
5\$: 5013 ;FPS BEFORE EXECUTION.
5004 ;FPS AFTER EXECUTION.
6\$: 12 ;FEC
-1 ;FLAG

;OVERFLOW, EXPONENT OF RESULT = 128
III3: JSR PC,@#OVUNFNT
1\$: .WORD 60200,0 ;AC
2\$: .WORD 60000,0 ;FSRC
3\$: .WORD 0,0 ;RES
5\$: 0 ;FPS BEFORE EXECUTION.

17042 077124 000006
17043 077126 000010
17044 077130 000000
17045
17046 077132 004737 077166
17047 077136 060200 000000
17048 077142 060200 000000
17049 077146 000000 000000
17050 077152 006011
17051 077154 006006
17052 077156 000010
17053 077160 000000
17054 077162 000167 000132
17055
17056
17057
17058
17059
17060
17061
17062
17063
17064
17065
17066
17067
17068
17069
17070
17071
17072
17073
17074
17075
17076
17077
17078
17079
17080
17081
17082
17083
17084
17085
17086
17087
17088
17089
17090
17091
17092
17093
17094
17095
17096 077166 012601
17097 077170 012700 000200

6 ;FPS AFTER EXECUTION.
6S: 10 ;FEC
0 ;FLAG
;OVERFLOW, EXPONENT OF RESULT = 130
1114: JSR PC,@OVUNFNT
1S: .WORD 60200,0 ;AC
2S: .WORD 60200,0 ;FSRC
3S: .WORD 0,0 ;RES
5S: 6011 ;FPS BEFORE EXECUTION.
6006 ;FPS AFTER EXECUTION.
6S: 10 ;FEC
0 ;FLAG
8S: JMP IIIDONE ;GO TO NEXT TEST.

;THIS SUBROUTINE, OVUNFNT, IS USED TO SET UP THE OPERANDS, EXECUTE
;THE MULF INSTRUCTION AND CHECK THE RESULTS OF AN INSTRUCTION WITH
;OPERANDS WHICH SHOULD RESULT IN EITHER OVERFLOW OR UNDERFLOW. A CALL
;TO IT IS MADE THUS:

.....
ACARG: .WORD X,X ;AC OPERAND
FSRCARG: .WORD X,X ;FSRC OPERAND
RES: .WORD X,X ;EXPECTED RESULT
ERRES: .WORD X,X ;ERROR RESULT
FPSB: .WORD X ;FPS BEFORE EXECUTION
FPSA: .WORD X ;FPS AFTER EXECUTION
FEC: .WORD X ;EXPECTED FEC
FLAG: .WORD X ;0/-1,OVER/UNDER FLOW FLAG
ERR1: ERROR X ;TRAP ERROR.
BR CONT
ERR2: ERROR X ;DATA, RESULT ERROR
CONT: ;RETURN ADDRESS
.....

;THE OPERANDS ARE SET UP (USING ACO AS THE ACCUMULATOR). THEN
;THE MULF INSTRUCTION IS EXECUTED. IF NO TRAP OCCURS THEN THE
;RESULT IS CHECKED AGAINST RES. IF THE RESULT IS CORRECT THEN THE FPS IS
;COMPARED WITH FPSA IF THIS TOO IS CORRECT OVUNFNT RETURNS CONTROL
;TO THE CALLING ROUTINE AT CONT. IF THE FPS IS BAD OVUNFNT
;REPORTS THIS FAILURE AND THEN RETURNS TO CONT. IF THE RESULT OF THE
;MULF IS INCORRECT, THE INCORRECT RESULT IS COMPARED WITH THE
;ANTICIPATED FAILING DATA PATTERN, ERRES. IF THE FAILURE IN
;THE RESULT WAS ANTICIPATED CORRECTLY TO BE ERRES THEN OVUNFNT
;WILL TRANSFER CONTROL TO THE ERROR CALL AT ERR2. OTHERWISE THE
;RESULT WAS INCORRECT BUT WAS NOT ANTICIPATED AND OVUNFNT WILL
;REPORT THE FAILURE AFTER WHICH CONTROL WILL BE PASSED TO CONT.
;IF A TRAP OCCURS (IT SHOULD NOT) THEN OVUNFNT WILL READ THE FEC.
;SHOULD THE FEC MATCH THE ANTICIPATED FEC OVUNFNT WILL
;STORE ALL DATA AND TRANSFER CONTROL TO THE ERROR CALL AT ERR1. IF THE
;FEC IS NOT THE SAME AS THE ANTICIPATED FEC OVUNFNT WILL REPORT
;THE ERROR AND RETURN TO CONT. NOTE THAT OVUNFNT USES THE FLAG
;TO TELL WHETHER OR NOT THESE PARTICULAR OPERANDS WILL RESULT IN
;UNDERFLOW (FLAG=-1) OR OVERFLOW (FLAG=0).

OVUNFNT: MOV (SP)+,R1 ;GET A POINTER TO THE ARGUMENTS.
MOV #200,R0 ;SET FD MODE.

```

17098 077174 170100          LDFPS  R0
17099
17100 077176 010100          MOV    R1,R0          ;LOAD ACO, OPERAND.
17101 077200 172410          LDD   (R0),ACO
17102 077202 016100 000014    MOV    14(R1),R0     ;LOAD THE FPS
17103 077206 170100          LDFPS  R0
17104 077210 012737 077270 000244  MOV    #25$,@#FPVECT ;SET UP THE FP TRAP VECTOR IN CASE
17105                                     ;OF ERROR.
17106 077216 010100          MOV    R1,R0          ;COMPUTE THE ADDRESS OF FSRC.
17107 077220 062700 000004    ADD   #4,R0
17108
17109 077224 171010          1$:   MULF  (R0),ACO   ;TEST INSTRUCTION.
17110
17111 077226 170204          2$:   STFPS R4          ;GET FPS.
17112 077230 170305          STST  R5              ;GET FEC.
17113 077232 012700 000200    MOV    #200,R0       ;SET FD MODE.
17114 077236 170100          LDFPS  R0
17115 077240 012700 077310    MOV    #OVFNNT,R0    ;GET THE RESULT.
17116 077244 174010          STD   ACO,(R0)
17117 077246 012700 077310    MOV    #OVFNNT,R0    ;CHECK THE RESULT.
17118 077252 010102          MOV    R1,R2
17119 077254 062702 000010    ADD   #10,R2
17120 077260 012703 000002    MOV    #2,R3
17121 077264 022022          3$:   CMP   (R0)+,(R2)+
17122 077266 001401          BEQ   5$
17123 077270
17124 077270 104000          25$:  EMT
17125 077272 077304          5$:   SOB   R3,3$
17126
17127 077274 026104 000016    CMP   16(R1),R4      ;WAS FPS CORRECT?
17128 077300 001401          BEQ   4$
17129 077302 104000          EMT
17130 077304 000161 000024          4$:   JMP   24(R1)      ;RETURN, TEST COMPLETED.
17131
17132 077310 000000 000000 000000  OVFNNT: .WORD 0,0,0,0
17133 077316 000000
17134
17135 077320
17136 077320 004767 025246          IIIDONE: JSR   PC,.RSET      ;GO INITIALIZE THE FPS AND STACK; AND
17137                                     ;SEE IF THE USER HAS EXPRESSED
17138                                     ;THE DESIRE TO CHANGE THE SOFTWARE
17139                                     ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
17140                                     ;THE USER TYPED CONTROL G?).
17141
17142
17143
17144
17145
17146
17147
17148 077324
17149
17150
17151 077324 004737 077550          ;UNDERFLOW, EXPONENT OF RESULT=-129
17152 077330 020200 000000          JJJ1: JSR   PC,@#OVUNDNT
17153 077334 127272 000000          1$:   .WORD 20200,0      ;AC
                                     .WORD 127272,0

```

```

:*****
:TEST 502 UNDER\OVER FLOW, USING MULF WITH TRAP DISABLED, TEST
:*****
TS502:

```

```

17154 077340 020000 000000 000000 2$: .WORD 20000,0,0,0 :FSRC
17155 077346 000000
17156 077350 000000 000000 000000 3$: .WORD 0,0,0,0 :RES
17157 077356 000000
17158 077360 000200 5$: 200 :FPS BEFORE EXECUTION.
17159 077362 000204 :FPS AFTER EXECUTION.
17160 077364 000012 6$: 12 :FEC
17161 077366 177777 -1 :FLAG

```

```

17162
17163 ;UNDERFLOW, EXPONENT OF RESULT = -193
17164 077370 004737 077550 JJJ2: JSR PC, @OVUNDNT
17165 077374 010200 000000 1$: .WORD 10200,0 :AC
17166 077400 123456 000000 .WORD 123456,0
17167 077404 010000 000000 000000 2$: .WORD 10000,0,0,0 :FSRC
17168 077412 000000
17169 077414 000000 000000 000000 3$: .WORD 0,0,0,0 :RES
17170 077422 000000
17171 077424 005213 5$: 5213 :FPS BEFORE EXECUTION.
17172 077426 005204 :FPS AFTER EXECUTION.
17173 077430 000012 6$: 12 :FEC
17174 077432 177777 -1 :FLAG

```

```

17175
17176 ;OVERFLOW, EXPONENT OF RESULT = 128
17177 077434 004737 077550 JJJ3: JSR PC, @OVUNDNT
17178 077440 060200 000000 1$: .WORD 60200,0 :AC
17179 077444 065432 000000 .WORD 65432,0
17180 077450 060000 000000 000000 2$: .WORD 60000,0,0,0 :FSRC
17181 077456 000000
17182 077460 000000 000000 000000 3$: .WORD 0,0,0,0 :RES
17183 077466 000000
17184 077470 000200 5$: 200 :FPS BEFORE EXECUTION.
17185 077472 000206 :FPS AFTER EXECUTION.
17186 077474 000010 6$: 10 :FEC
17187 077476 000000 0 :FLAG

```

```

17188
17189 ;OVERFLOW, EXPONENT OF RESULT = 130
17190 077500 004737 077550 JJJ4: JSR PC, @OVUNDNT
17191 077504 060200 000000 1$: .WORD 60200,0 :AC
17192 077510 125252 000000 .WORD 125252,0
17193 077514 060200 000000 000000 2$: .WORD 60200,0,0,0 :FSRC
17194 077522 000000
17195 077524 000000 000000 000000 3$: .WORD 0,0,0,0 :RES
17196 077532 000000
17197 077534 006211 5$: 6211 :FPS BEFORE EXECUTION.
17198 077536 006206 :FPS AFTER EXECUTION.
17199 077540 000010 6$: 10 :FEC
17200 077542 000000 0 :FLAG
17201 077544 000137 077702 8$: JMP @JJJJDONE :GO TO NEXT TEST

```

```

17202
17203 ;THIS SUBROUTINE, OVUNDNT, IS USED TO SET UP THE OPERANDS, EXECUTE
17204 ;THE MULD INSTRUCTION AND CHECK THE RESULTS OF AN INSTRUCTION WITH
17205 ;OPERANDS WHICH SHOULD RESULT IN EITHER OVERFLOW OR UNDERFLOW. A CALL
17206 ;TO IT IS MADE THUS:

```

```

17207
17208 ;
17209 ; ACARG: .WORD X,X,X,X ;AC OPERAND
; FSRCARG: .WORD X,X,X,X ;FSRC OPERAND

```


17210	:	RES:	.WORD	X,X,X,X	:	EXPECTED RESULT
17211	:	ERRES:	.WORD	X,X,X,X	:	ERROR RESULT
17212	:	FPSB:	.WORD	X	:	FPS BEFORE EXECUTION
17213	:	FPSA:	.WORD	X	:	FPS AFTER EXECUTION
17214	:	FEC:	.WORD	X	:	EXPECTED FEC
17215	:	FLAG:	.WORD	X	:	0/-1,OVER/UNDER FLOW FLAG
17216	:	ERR1:	ERROR	X	:	TRAP ERROR.
17217	:	BR	CONT			
17218	:	ERR2:	ERROR	X	:	DATA, RESULT ERROR
17219	:	LONT:			:	RETURN ADDRESS

17220
 17221 : THE OPERANDS ARE SET UP (USING ACO AS THE ACCUMULATOR). THEN
 17222 : THE MULD INSTRUCTION IS EXECUTED. IF NO TRAP OCCURS THEN THE
 17223 : RESULT IS CHECKED AGAINST RES. IF THE RESULT IS CORRECT THEN THE FPS IS
 17224 : COMPARED WITH FPSA IF THIS TOO IS CORRECT OVUNDNT RETURNS CONTROL
 17225 : TO THE CALLING ROUTINE AT CONT. IF THE FPS IS BAD OVUNDNT
 17226 : REPORTS THIS FAILURE AND THEN RETURNS TO CONT. IF THE RESULT OF THE
 17227 : MULD IS INCORRECT, THE INCORRECT RESULT IS COMPARED WITH THE
 17228 : ANTICIPATED FAILING DATA PATTERN, ERRES. IF THE FAILURE IN
 17229 : THE RESULT WAS ANTICIPATED CORRECTLY TO BE ERRES THEN OVUNDNT
 17230 : WILL TRANSFER CONTROL TO THE ERROR CALL AT ERR2. OTHERWISE THE
 17231 : RESULT WAS INCORRECT BUT WAS NOT ANTICIPATED AND OVUNDNT WILL
 17232 : REPORT THE FAILURE AFTER WHICH CONTROL WILL BE PASSED TO CONT.
 17233 : IF A TRAP OCCURS (IT SHOULD NOT) THEN OVUNDNT WILL READ THE FEC.
 17234 : SHOULD THE FEC MATCH THE ANTICIPATED FEC OVUNDNT WILL
 17235 : STORE ALL DATA AND TRANSFER CONTROL TO THE ERROR CALL AT ERR1. IF THE
 17236 : FEC IS NOT THE SAME AS THE ANTICIPATED FEC OVUNDNT WILL REPORT
 17237 : THE ERROR AND RETURN TO CONT. NOTE THAT OVUNDNT USES THE FLAG
 17238 : TO TELL WHETHER OR NOT THESE PARTICULAR OPERANDS WILL RESULT IN
 17239 : UNDERFLOW (FLAG=-1) OR OVERFLOW (FLAG=0).

17240						
17241	077550	012601		OVUNDNT:	MOV	(SP)+,R1 ;GET A POINTER TO THE ARGUMENTS.
17242	077552	012700	000200		MOV	#200,R0 ;SET FD MODE.
17243	077556	170100			LDFPS	R0
17244						
17245	077560	010100			MOV	R1,R0 ;LOAD ACO, OPERAND.
17246	077562	172410			LDD	(R0),ACO
17247	077564	016100	000030		MOV	30(R1),R0 ;LOAD THE FPS.
17248	077570	170100			LDFPS	R0
17249	077572	012737	077652 000244		MOV	#25\$,@#FPVECT ;SET UP THE FP TRAP VECTOR IN CASE ;OF ERROR.
17250						
17251	077600	010100			MOV	R1,R0 ;COMPUTE THE ADDRESS OF FSRC.
17252	077602	062700	000010		ADD	#10,R0
17253						
17254	077606	171010		1\$:	MULD	(R0),ACO ;TEST INSTRUCTION.
17255						
17256	077610	170204		2\$:	STFPS	R4 ;GET FPS.
17257	077612	170305			STST	R5 ;GET FEC.
17258	077614	012700	000200		MOV	#200,R0 ;SET FD MODE.
17259	077620	170100			LDFPS	R0
17260	077622	012700	077672		MOV	#OVDNTT,R0 ;GET THE RESULT.
17261	077626	174010			STD	ACO,(R0)
17262	077630	012700	077672		MOV	#OVDNTT,R0 ;CHECK THE RESULT.
17263	077634	010102			MOV	R1,R2
17264	077636	062702	000020		ADD	#20,R2
17265	077642	012703	000004		MOV	#4,R3

17266 077646 022022
17267 077650 001401
17268 077652
17269 077652 104000
17270 077654 077304
17271
17272 077656 026104 000032
17273 077662 001401
17274 077664 104000
17275 077666 000161 000040
17276
17277 077672 000000 000000 000000
17278 077700 000000
17279
17280 077702
17281 077702 004767 024664
17282
17283
17284
17285
17286
17287
17288
17289
17290
17291
17292
17293 077706
17294
17295
17296 077706 004737 100052
17297 077712 020123 045676
17298 077716 020200 000000
17299 077722 000123 045676
17300 077726 002000
17301 077730 102004
17302 077732 000012
17303 077734 177777
17304
17305
17306 077736 004737 100052
17307 077742 010127 127272
17308 077746 010200 000000
17309 077752 060127 127272
17310 077756 007017
17311 077760 107000
17312 077762 000012
17313 077764 177777
17314
17315
17316 077766 004737 100052
17317 077772 060252 125252
17318 077776 060000 000000
17319 100002 000052 125252
17320 100006 001000
17321 100010 101006

3\$: CMP (R0)+,(R2)+
BEQ 5\$
25\$:
5\$: EMT
SOB R3,3\$;
CMP 32(R1),R4 ;WAS FPS CORRECT?
BEQ 4\$
EMT
4\$: JMP 40(R1) ;RETURN, TEST COMPLETED.
OVDNNT: .WORD 0,0,0,0
JJJDONE:
JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

:TEST 503 UNDER\OVER FLOW, USING MULF WITH TRAPS ENABLED, TEST

TS503:

:UNDERFLOW, EXPONENT OF RESULT = -129
KKK1: JSR PC,@#OVUNFT
1\$: .WORD 20123,45676 ;AC
2\$: .WORD 20200,0 ;FSRC
3\$: .WORD 123,45676 ;RES
5\$: 2000 ;FPS BEFORE EXECUTION.
102004 ;FPS AFTER EXECUTION.
6\$: 12 ;FEC
-1 ;FLAG

:UNDERFLOW, EXPONENT OF THE RESULT = -193
KKK3: JSR PC,@#OVUNFT
1\$: .WORD 10127,127272 ;AC
2\$: .WORD 10200,0 ;FSRC
3\$: .WORD 60127,127272 ;RES
5\$: 7017 ;FPS BEFORE EXECUTION.
107000 ;FPS AFTER EXECUTION.
6\$: 12 ;FEC
-1

:OVERFLOW, EXPONENT OF THE RESULT = 128
KKK4: JSR PC,@#OVUNFT
1\$: .WORD 60252,125252 ;AC
2\$: .WORD 60000,0 ;FSRC
3\$: .WORD 000052,125252 ;RES
5\$: 1000 ;FPS BEFORE EXECUTION.
101006 ;FPS AFTER EXECUTION.

```

17322 100012 000010
17323 100014 000000
17324
17325
17326 100016 004737 100052
17327 100022 060345 067654
17328 100026 060200 000000
17329 100032 000345 067654
17330 100036 007015
17331 100040 107002
17332 100042 000010
17333 100044 000000
17334 100046 000167 000162
17335
17336
17337
17338
17339
17340
17341
17342
17343
17344
17345
17346
17347
17348
17349
17350
17351
17352
17353
17354
17355
17356
17357
17358
17359
17360
17361
17362
17363
17364
17365
17366
17367
17368
17369
17370
17371
17372 100052 012601
17373 100054 012700 000200
17374 100060 170100
17375 100062 010100
17376 100064 172410
17377 100066 016100 000014
  
```

```

6S: 10 ;FEC
0 ;FLAG

:OVERFLOW, EXPONENT OF RESULT = 130
KKK5: JSR PC,@OVUNFT
1S: .WORD 60345,67654 ;AC
2S: .WORD 60200,0 ;FSRC
3S: .WORD 345,67654 ;RES
5S: 7015 ;FPS BEFORE EXECUTION.
107002 ;FPS AFTER EXECUTION.
6S: 10 ;FEC
0 ;FLAG
8S: JMP KKKDONE
  
```

:THIS SUBROUTINE, OVUNFT, IS USED TO SET UP THE OPERANDS, EXECUTE THE MULF INSTRUCTION AND CHECK THE RESULTS OF AN INSTRUCTION WITH OPERANDS WHICH SHOULD RESULT IN EITHER OVERFLOW OR UNDERFLOW. A CALL TO IT IS MADE THUS:

```

:
: ACARG: .WORD X,X ;AC OPERAND
: FSRCARG: .WORD X,X ;FSRC OPERAND
: RES: .WORD X,X ;EXPECTED RESULT
: ERRES: .WORD X,X ;ERROR RESULT
: FPSB: .WORD X ;FPS BEFORE EXECUTION
: FPSA: .WORD X ;FPS AFTER EXECUTION
: FEC: .WORD X ;EXPECTED FEC
: FLAG: .WORD X ;0/-1,OVER/UNDER FLOW FLAG
: ERR1: ERROR X ;TRAP ERROR.
: BR CONT
: ERR2: ERROR X ;DATA, RESULT ERROR
: CONT: ;RETURN ADDRESS
  
```

:THE OPERANDS ARE SET UP (USING ACO AS THE ACCUMULATOR). THEN THE MULF INSTRUCTION IS EXECUTED. IF THE TRAP OCCURS THEN THE RESULT IS CHECKED AGAINST RES. IF THE RESULT IS CORRECT THEN THE FPS IS COMPARED WITH FPSA IF THIS TOO IS CORRECT OVUNFT RETURNS CONTROL TO THE CALLING ROUTINE AT CONT. IF THE FPS IS BAD OVUNFT REPORTS THIS FAILURE AND THEN RETURNS TO CONT. THE FEC IS TREATED IN THE SAME WAY. IF THE RESULT OF THE MULF IS INCORRECT, THE INCORRECT RESULT IS COMPARED WITH THE ANTICIPATED FAILING DATA PATTERN, ERRES. IF THE FAILURE IN THE RESULT WAS ANTICIPATED CORRECTLY TO BE ERRES THEN OVUNFT WILL TRANSFER CONTROL TO THE ERROR CALL AT ERR2. OTHERWISE THE RESULT WAS INCORRECT BUT WAS NOT ANTICIPATED AND OVUNFT WILL REPORT THE FAILURE AFTER WHICH CONTROL WILL BE PASSED TO CONT. IF NO TRAP OCCURS CONTROL IS PASSED TO ERR1. NOTE THAT OVUNFT USES THE FLAG TO TELL WHETHER OR NOT THESE PARTICULAR OPERANDS WILL RESULT IN UNDERFLOW (FLAG=-1) OR OVERFLOW (FLAG=0).

```

OVUNFT: MOV (SP)+,R1 ;GET A POINTER TO THE ARGUMENTS.
MOV #200,R0 ;SET FD MODE.
LDFPS R0
MOV R1,R0 ;LOAD ACO, OPERAND.
LDD (R0),ACO
MOV 14(R1),R0 ;LOAD THE FPS.
  
```

```
17378 100072 170100 LDFPS R0
17379 100074 012737 100116 000244 MOV #50$,@#FPVECT ;SET UP THE FP TRAP VECTOR IN CASE
17380 ;OF ERROR.
17381 100102 010100 MOV R1,R0 ;COMPUTE THE ADDRESS OF FSRC.
17382 100104 062700 000004 ADD #4,R0
17383
17384 100110 171010 1$: MULF (R0),ACO ;TEST INSTRUCTION. SHOULD CAUSE TRAP.
17385 100112 170000 2$: CFCC
17386 100114 104000 EMT ;
17387 100116 011602 50$: MOV (SP),R2 ;TRAP TO HERE AND SEE IF THE PC OF THE
17388 100120 020227 100112 CMP R2,#2$ ;TRAP WAS THAT OF THE MULF INSTRUCTION.
17389 100124 001401 BEQ 51$
17390 100126 104000 EMT ;
17391 100130 022626 51$: CMP (SP)+,(SP)+ ;RESET THE STACK
17392 100132 170204 STFPS R4 ;GET FPS.
17393 100134 170305 STST R5 ;GET FEC.
17394 100136 012700 000200 MOV #200,R0 ;SET FD MODE.
17395 100142 170100 LDFPS R0
17396 100144 012700 100224 MOV #OVFTT,R0 ;GET THE RESULT.
17397 100150 174010 STD ACO,(R0)
17398 100152 012700 100224 MOV #OVFTT,R0 ;CHECK THE RESULT.
17399 100156 010102 MOV R1,R2
17400 100160 062702 000010 ADD #10,R2
17401 100164 012703 000002 MOV #2,R3
17402 100170 022022 3$: CMP (R0)+,(R2)+
17403 100172 001401 BEQ 5$
17404 100174 104000 EMT ;
17405 100176 077304 5$: SOB R3,3$
17406
17407 100200 026104 000016 CMP 16(R1),R4 ;WAS FPS CORRECT?
17408 100204 001401 BEQ 6$
17409 100206 104000 EMT ;
17410 100210 026105 000020 6$: CMP 20(R1),R5 ;IS FEC CORRECT?
17411 100214 001401 BEQ 4$
17412 100216 104000 EMT ;
17413 100220 000161 000024 4$: JMP 24(R1) ;RETURN, TEST COMPLETED.
17414
17415 100224 000000 000000 000000 OVFTT: .WORD 0,0,0,0
17416 100232 000000
17417
17418 100234
17419 100234 004767 024332 KKKDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
17420 ;SEE IF THE USER HAS EXPRESSED
17421 ;THE DESIRE TO CHANGE THE SOFTWARE
17422 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
17423 ;THE USER TYPED CONTROL G?).
17424
17425
17426
17427
17428
17429 *****
17430 :TEST 504 UNDER\OVER FLOW, USING MULF WITH TRAPS ENABLED, TEST
17431 *****
17432 TS504:
17433
17433 :UNDERFLOW, EXPONENT OF RESULT = -129
```

17434 100240 004737 100464
17435 100244 020052 125252
17436 100250 125252 125252
17437 100254 020300 000000 000000
17438 100262 000000
17439 100264 000177 177777 177777
17440 100272 177777
17441 100274 002200
17442 100276 102204
17443 100300 000012
17444 100302 177777
17445

LLL1: JSR PC,@OVUNDT
1\$: .WORD 20052,125252 ;AC
.WORD 125252,125252
2\$: .WORD 20300,0,0,0 ;FSRC
3\$: .WORD 177,-1,-1,-1 ;RES
5\$: 2200 ;FPS BEFORE EXECUTION.
102204 ;FPS AFTER EXECUTION.
6\$: 12 ;FEC
-1 ;FLAG

:UNDERFLOW, EXPONENT OF THE RESULT = -193

17446
17447 100304 004737 100464
17448 100310 010327 127272
17449 100314 036363 045454
17450 100320 010000 000000 000000
17451 100326 000000
17452 100330 060127 127272
17453 100334 036363 045454
17454 100340 007217
17455 100342 107200
17456 100344 000012
17457 100346 177777
17458

LLL2: JSR PC,@OVUNDT
1\$: .WORD 10327,127272 ;AC
.WORD 36363,45454
2\$: .WORD 10000,0,0,0 ;FSRC
3\$: .WORD 60127,127272 ;RES
.WORD 36363,45454
5\$: 7217 ;FPS BEFORE EXECUTION.
107200 ;FPS AFTER EXECUTION.
6\$: 12 ;FEC
-1 ;FLAG

:OVERFLOW, EXPONENT OF THE RESULT = 128

17459
17460 100350 004737 100464
17461 100354 060252 125252
17462 100360 125252 125252
17463 100364 160100 000000 000000
17464 100372 000000
17465 100374 100177 177777 177777
17466 100402 177777
17467 100404 001200
17468 100406 101216
17469 100410 000010
17470 100412 000000
17471

LLL3: JSR PC,@OVUNDT
1\$: .WORD 60252,125252 ;AC
.WORD 125252,125252 ;FSRC
2\$: .WORD 160100,0,0,0 ;FSRC
3\$: .WORD 100177,-1,-1,-1 ;RES
5\$: 1200 ;FPS BEFORE EXECUTION.
101216 ;FPS AFTER EXECUTION.
6\$: 10 ;FEC
0 ;FLAG

:OVERFLOW, EXPONENT OF THE RESULT = 130

17472
17473 100414 004737 100464
17474 100420 060345 067654
17475 100424 056765 045676
17476 100430 060200 000000 000000
17477 100436 000000
17478 100440 000345 067654
17479 100444 056765 045676
17480 100450 007215
17481 100452 107202
17482 100454 000010
17483 100456 000000
17484 100460 000137 100646
17485

LLL4: JSR PC,@OVUNDT
1\$: .WORD 60345,67654 ;AC
.WORD 56765,45676
2\$: .WORD 60200,0,0,0 ;FSRC
3\$: .WORD 345,67654 ;RES
.WORD 56765,45676
5\$: 7215 ;FPS BEFORE EXECUTION.
107202 ;FPS AFTER EXECUTION.
6\$: 10 ;FEC
0 ;FLAG
8\$: JMP @LLLLDONE

:THIS SUBROUTINE, OVUNDT, IS USED TO SET UP THE OPERANDS, EXECUTE
:THE MULD INSTRUCTION AND CHECK THE RESULTS OF AN INSTRUCTION WITH
:OPERANDS WHICH SHOULD RESULT IN EITHER OVERFLOW OR UNDERFLOW. A CALL
:TO IT IS MADE THUS:

17486
17487
17488
17489

CJKDJBO 11/23-B CPU CLUSTER DIAG.
CJKDJB.P11 26-MAY-82 11:14

L 10
DNMAC X24.07-563 26-MAY-82 11:18 PAGE 335
T504 UNDER\OVER FLOW, USING MULD WITH TRAPS ENABLED, TEST

SEQ 0334

17490	:	ACARG: .WORD	X,X,X,X	:AC OPERAND
17491	:	FSRCARG: .WORD	X,X,X,X	:FSRC OPERAND
17492	:	RES: .WORD	X,X,X,X	:EXPECTED RESULT
17493	:	ERRES: .WORD	X,X,X,X	:ERROR RESULT
17494	:	FPSB: .WORD	X	:FPS BEFORE EXECUTION
17495	:			

17496
 17497
 17498
 17499
 17500
 17501
 17502
 17503
 17504
 17505
 17506
 17507
 17508
 17509
 17510
 17511
 17512
 17513
 17514
 17515
 17516
 17517
 17518
 17519
 17520
 17521
 17522 100464 012601
 17523 100466 012700 000200
 17524 100472 170100
 17525
 17526 100474 010100
 17527 100476 172410
 17528 100500 016100 000030
 17529 100504 170100
 17530 100506 012737 100530 000244
 17531
 17532 100514 010100
 17533 100516 062700 000010
 17534
 17535 100522 171010
 17536 100524 170000
 17537 100526 104000
 17538 100530 011602
 17539 100532 020227 100524
 17540 100536 001401
 17541 100540 104000
 17542 100542 022626
 17543 100544 170204
 17544 100546 170305
 17545 100550 012700 000200
 17546 100554 170100
 17547 100556 012700 100636
 17548 100562 174010
 17549 100564 012700 100636
 17550 100570 010102
 17551 100572 062702 000020

```

:          FPSA: .WORD X          :FPS AFTER EXECUTION
:          FEC:  .WORD X          :EXPECTED FEC
:          FLAG: .WORD X          :0/-1,OVER/UNDER FLOW FLAG
:          ERR1: ERROR X         :TRAP ERROR.
:          BR   : CONT
:          ERR2: ERROR X         :DATA, RESULT ERROR
:          CONT:                  :RETURN ADDRESS
:
:THE OPERANDS ARE SET UP (USING ACO AS THE ACCUMULATOR). THEN
:THE MULD INSTRUCTION IS EXECUTED. IF THE TRAP OCCURS THEN THE
:RESULT IS CHECKED AGAINST RES. IF THE RESULT IS CORRECT THEN THE FPS IS
:COMPARED WITH FPSA IF THIS TOO IS CORRECT OVUNDT RETURNS CONTROL
:TO THE CALLING ROUTINE AT CONT. IF THE FPS IS BAD OVUNDT
:REPORTS THIS FAILURE AND THEN RETURNS TO CONT. THE FEC IS TREATED
:IN THE SAME WAY. IF THE RESULT OF THE
:MULF IS INCORRECT, THE INCORRECT RESULT IS COMPARED WITH THE
:ANTICIPATED FAILING DATA PATTERN, ERRES. IF THE FAILURE IN
:THE RESULT WAS ANTICIPATED CORRECTLY TO BE ERRES THEN OVUNDT
:WILL TRANSFER CONTROL TO THE ERROR CALL AT ERR2. OTHERWISE THE
:RESULT WAS INCORRECT BUT WAS NOT ANTICIPATED AND OVUNDT WILL
:REPORT THE FAILURE AFTER WHICH CONTROL WILL BE PASSED TO CONT.
:IF NO TRAP OCCURS CONTROL IS PASSED TO ERR1.
:NOTE THAT OVUNDT USES THE FLAG
:TO TELL WHETHER OR NOT THESE PARTICULAR OPERANDS WILL RESULT IN
:UNDERFLOW (FLAG=-1) OR OVERFLOW (FLAG=0).
OVUNDT: MOV (SP)+,R1          :GET A POINTER TO THE ARGUMENTS.
        MOV #200,R0          :SET FD MODE.
        LDFPS R0
        MOV R1,R0           :LOAD ACO, OPERAND.
        LDD (R0),ACO
        MOV 30(R1),R0        :LOAD THE FPS.
        LDFPS R0
        MOV #50$,@#FPVECT    :SET UP THE FP TRAP VECTOR IN CASE
                               :OF ERROR.
        MOV R1,R0           :COMPUTE THE ADDRESS OF FSRC.
        ADD #10,R0
1$: MULD (R0),ACO          :TEST INSTRUCTION. SHOULD CAUSE TRAP.
2$: CFCC
   EMT
50$: MOV (SP),R2           :TRAP TO HERE AND SEE IF THE PC OF THE
   CMP R2,#2$              :TRAP WAS THAT OF THE MULF INSTRUCTION.
   BEQ 51$                 :BRANCH IF YES.
   EMT
51$: CMP (SP)+,(SP)+       :RESET THE STACK
   STFPS R4                :GET FPS.
   STST R5                 :GET FEC.
   MOV #200,R0             :SET FD MODE.
   LDFPS R0
   MOV #OVDTT,R0          :GET THE RESULT.
   STD ACO,(R0)
   MOV #OVDTT,R0          :CHECK THE RESULT.
   MOV R1,R2
   ADD #20,R2

```

17552 100576 012703 000004
17553 100602 022022
17554 100604 001401
17555 100606 104000
17556 100610 077304
17557 100612 026104 000032
17558 100616 001401
17559 100620 104000
17560 100622 026105 000034
17561 100626 001401
17562 100630 104000
17563 100632 000161 000040
17564
17565 100636 000000 000000 000000
17566 100644 000000
17567
17568 100646
17569 100646 004767 023720
17570
17571
17572
17573
17574
17575
17576
17577
17578
17579
17580
17581
17582 100652
17583
17584
17585 100652 004737 101376
17586 100656 000000 000000
17587 100662 000000 000000
17588 100666 000000 000000
17589 100672 000000 000000
17590 100676 000013
17591 100700 000004
17592
17593
17594 100702 004737 101376
17595 100706 123456 076543
17596 100712 000000 000000
17597 100716 000000 000000
17598 100722 000000 000000
17599 100726 000000
17600 100730 000004
17601
17602
17603 100732 004737 101376
17604 100736 000000 000000
17605 100742 076543 021234
17606 100746 000000 000000
17607 100752 000000 000000

MOV #4,R3
3\$: CMP (R0)+,(R2)+
BEQ 5\$
EIT ;
5\$: SOB R3,3\$;
CMP 32(R1),R4 ;WAS FPS CORRECT?
BEQ 6\$
EIT ;
6\$: CMP 34(R1),R5 ;IS FEC CORRECT?
BEQ 4\$
EIT ;
4\$: JMP 40(R1) ;RETURN, TEST COMPLETED.
OVDTT: .WORD 0,0,0,0
LLLDONE:
JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;TEST 505 MODF TEST

TS505:

;MODF WITH (FSRC=AC=0)
GGG1: JSR PC,@#MODFSUB
1\$: .WORD 0,0 ;AC
2\$: .WORD 0,0 ;FSRC
3\$: .WORD 0,0 ;FRACTIONAL RES.
4\$: .WORD 0,0 ;INTEGER RES.
7\$: 13 ;FPS BEFORE EXECUTION.
4 ;FPS AFTER EXECUTION.

;MODF TEST, WITH (FSRC=0)
GGG2: JSR PC,@#MODFSUB
1\$: .WORD 123456,76543 ;AC
2\$: .WORD 0,0 ;FSRC
3\$: .WORD 0,0 ;FRACTIONAL RES.
4\$: .WORD 0,0 ;INTEGER RESULT.
7\$: 0 ;FPS BEFORE EXECUTION.
4 ;FPS AFTER EXECUTION.

;MODF TEST WITH (AC=0)
GGG3: JSR PC,@#MODFSUB
1\$: .WORD 0,0 ;AC
2\$: .WORD 76543,21234 ;FSRC
3\$: .WORD 0,0 ;FRACTIONAL RES.
4\$: .WORD 0,0 ;INTEGER RES.

17608 100756 000003
17609 100760 000004
17610
17611
17612 100762 004737 101376
17613 100766 046252 125252
17614 100772 040300 000000
17615 100776 000000 000000
17616 101002 046377 177777
17617 101006 000013
17618 101010 000004
17619
17620
17621 101012 004737 101376
17622 101016 077652 125252
17623 101022 040300 000000
17624 101026 000000 000000
17625 101032 077777 177777
17626 101036 000000
17627 101040 000004
17628
17629
17630 101042 004737 101376
17631 101046 046200 000001
17632 101052 040340 000000
17633 101056 000000 000000
17634 101062 046340 000001
17635 101066 000013
17636 101070 000004
17637
17638
17639 101072 004737 101376
17640 101076 046000 000001
17641 101102 040340 000000
17642 101106 040100 000000
17643 101112 046140 000001
17644 101116 000000
17645 101120 000000
17646
17647
17648 101122 004737 101376
17649 101126 042577 177777
17650 101132 040200 000000
17651 101136 040177 176000
17652 101142 042577 140000
17653 101146 000000
17654 101150 000000
17655
17656
17657 101152 004737 101376
17658 101156 042577 140001
17659 101162 040200 000000
17660 101166 046000 000000
17661 101172 042577 140000
17662 101176 000000
17663 101200 000000

7\$: 3 :FPS BEFORE EXECUTION.
4 :FPS AFTER EXECUTION.

:MODF TEST WITH EXPONENT OF THE RESULT = 25
GGG4: JSR PC,@#MODFSUB
1\$: .WORD 46252,125252 :AC
2\$: .WORD 40300,0 :FSRC
3\$: .WORD 0,0 :FRACTIONAL RES.
4\$: .WORD 46377,-1 :INTEGER RES.
7\$: 13 :FPS BEFORE EXECUTION.
4 :FPS AFTER EXECUTION.

:MODF TEST WITH EXPONENT OF THE RESULT = 127
GGG5: JSR PC,@#MODFSUB
1\$: .WORD 77652,125252 :AC
2\$: .WORD 40300,0 :FSRC
3\$: .WORD 0,0 :FRACTIONAL RES.
4\$: .WORD 77777,-1 :INTEGER RES.
7\$: 0 :FPS BEFORE EXECUTION.
4 :FPS AFTER EXECUTION.

:MODF TEST WITH EXPONENT OF RESULT = 25
GGG6: JSR PC,@#MODFSUB
1\$: .WORD 46200,1 :AC
2\$: .WORD 40340,0 :FSRC
3\$: .WORD 0,0 :FRACTIONAL RES.
4\$: .WORD 46340,1 :INTEGER RES.
7\$: 13 :FPS BEFORE EXECUTION.
4 :FPS AFTER EXECUTION.

:MODF TEST WITH EXPONENT OF THE RESULT = 24
GGG7: JSR PC,@#MODFSUB
1\$: .WORD 46000,1 :AC
2\$: .WORD 40340,0 :FSRC
3\$: .WORD 40100,0 :FRACTIONAL RES.
4\$: .WORD 46140,1 :INTEGER RESULT.
7\$: 0 :FPS BEFORE EXECUTION.
0 :FPS AFTER EXECUTION.

:MODF TEST WITH EXPONENT OF THE RESULT = 10
GGG8: JSR PC,@#MODFSUB
1\$: .WORD 42577,-1 :AC
2\$: .WORD 40200,0 :FSRC
3\$: .WORD 40177,176000 :FRACTIONAL RES.
4\$: .WORD 42577,140000 :INTEGER RES.
7\$: 0 :FPS BEFORE EXECUTION.
0 :FPS AFTER EXECUTION.

:MODF TEST WITH THE EXPONENT OF THE RESULT = 10
GGG9: JSR PC,@#MODFSUB
1\$: .WORD 42577,140001 :AC
2\$: .WORD 40200,0 :FSRC
3\$: .WORD 34600,0 :FRACTIONAL RES.
4\$: .WORD 42577,140000 :INTEGER RES.
7\$: 0 :FPS BEFORE EXECUTION.
0 :FPS AFTER EXECUTION.

17664
17665
17666 101202 004737 101376
17667 101206 042377 100000
17668 101212 040200 000000
17669 101216 000000 000000
17670 101222 042377 100000
17671 101226 000013
17672 101230 000004

;MODF TEST WITH EXPONENT OF THE RESULT = 9
GGG10: JSR PC,@#MODFSUB
1\$: .WORD 42377,100000 ;AC
2\$: .WORD 40200,0 ;FSRC
3\$: .WORD 0,0 ;FRACTIONAL RES.
4\$: .WORD 42377,100000 ;INTEGER RES.
7\$: 13 ;FPS BEFORE EXECUTION.
4 ;FPS AFTER EXECUTION.

17673
17674
17675 101232 004737 101376
17676 101236 040177 177777
17677 101242 040200 000000
17678 101246 040177 177777
17679 101252 000000 000000
17680 101256 000017
17681 101260 000000

;MODF TEST WITH EXPONENT OF THE RESULT = 0
GGG11: JSR PC,@#MODFSUB
1\$: .WORD 40177,-1 ;AC
2\$: .WORD 40200,0 ;FSRC
3\$: .WORD 40177,-1 ;FRACTIONAL RES.
4\$: .WORD 0,0 ;INTEGER RES.
7\$: 17 ;FPS BEFORE EXECUTION.
0 ;FPS AFTER EXECUTION.

17682
17683
17684 101262 004737 101376
17685 101266 034377 177777
17686 101272 040200 000000
17687 101276 034377 177777
17688 101302 000000 000000
17689 101306 000000
17690 101310 000000

;MODF TEST WITH EXPONENT OF THE RESULT = -15
GGG12: JSR PC,@#MODFSUB
1\$: .WORD 34377,-1 ;AC
2\$: .WORD 40200,0 ;FSRC
3\$: .WORD 17377,-1 ;FRACTIONAL RES.
4\$: .WORD 0,0 ;INTEGER RES.
7\$: 0 ;FPS BEFORE EXECUTION.
0 ;FPS AFTER EXECUTION.

17691
17692
17693 101312 004737 101376
17694 101316 020000 000001
17695 101322 040300 000000
17696 101326 020100 000002
17697 101332 000000 000000
17698 101336 000000
17699 101340 000000

;MODF TEST WITH EXPONENT OF RESULT = -64, IN ROUND MODE
GGG13: JSR PC,@#MODFSUB
1\$: .WORD 20000,1 ;AC
2\$: .WORD 40300,0 ;FSRC
3\$: .WORD 20100,2 ;FRACTIONAL RES.
4\$: .WORD 0,0 ;INTEGER RES.
7\$: 0 ;FPS BEFORE EXECUTION.
0 ;FPS AFTER EXECUTION.

17700
17701
17702 101342 004737 101376
17703 101346 142777 170000
17704 101352 040200 000000
17705 101356 140000 000000
17706 101362 142777 160000
17707 101366 000007
17708 101370 000010
17709 101372 000167 000204

;MODF TEST WITH EXPONENT OF RESULT = 11
GGG14: JSR PC,@#MODFSUB
1\$: .WORD 142777,170000 ;AC
2\$: .WORD 40200,0 ;FSRC
3\$: .WORD 140000,0 ;FRACTIONAL RES.
4\$: .WORD 142777,160000 ;INTEGER RES.
7\$: 7 ;FPS BEFORE EXECUTION.
10 ;FPS AFTER EXECUTION.
9\$: JMP GGGDONE ;GO TO NEXT TEST.

17710
17711
17712
17713
17714
17715
17716
17717
17718
17719

;THIS SUBROUTINE, MODFSUB, IS CALLED TO SETUP THE
;OPERANDS, EXECUTE THE MODF INSTRUCTION AND CHECK THE RESULTS.
;IT IS CALLED THUS:
;
; ACARG: .WORD X,X ;AC OPERAND
; FSRCARG: .WORD X,X ;FSRC OPERAND
; FRES: .WORD X,X ;FRACTIONAL RESULT
; INTRES: .WORD X,X ;INTEGER RESULT

```

17720 : ERFRES: .WORD X,X ;ERROR FRACTION RESULT
17721 : ERINTRES: .WORD X,X ;ERROR INTEGER RESULT
17722 : FPSB: .WORD X ;FPS BEFORE EXECUTION
17723 : FPSA: .WORD X ;FPS AFTER EXECUTION
17724 : ERR1: ERROR X ;FRACTION ERROR
17725 : BR CONT
17726 : ERR2: ERROR X ;INTEGER ERROR
17727 : CONT: ;RETURN ADDRESS
17728 :
17729 : THE OPERANDS ARE SET UP (USING ACO FOR THE AC ARGUMENT). THE MODF
17730 : INSTRUCTION IS EXECUTED. THEN THE RESULTS ARE RETRIEVED.
17731 : THE FRACTION PART OF THE RESULT IS COMPARED WITH FRES. IF THIS IS CORRECT
17732 : THEN THE INTEGER PART IS COMPARED WITH INTRES. IF BOTH OF THESE ARE CORRECT
17733 : THEN THE FPS IS COMPARED WITH FPSA. AFTER EXECUTION IF NO ERRORS OCCURRED
17734 : THEN MODFSUB WILL RETURN TO CONT. IF THE FPS WAS INCORRECT
17735 : IT IS REPORTED HERE. IF THE FRACTION IS INCORRECT IT IS COMPARED WITH
17736 : THE ANTICIPATED BAD FRACTION, ERFRES. IF THIS DOESN'T MATCH
17737 : THE TRUE RESULT THEN THE ERROR IS REPORTED HERE. IF THE ANTICIPATED
17738 : FAILURE MATCHES THE TRUE RESULT THEN MODFSUB PASSES CONTROL TO THE
17739 : ERROR CALL AT ERR1. LIKEWISE IF THE INTEGER PART OF THE RESULT IS
17740 : NOT CORRECT THEN IT IS COMPARED WITH THE ANTICIPATED INTEGER
17741 : FAILURE. IF THIS DOESN'T MATCH THEN THE ERROR IS REPORTED HERE.
17742 : IF A MATCH IS MADE HOWEVER, MODFSUB WILL RETURN CONTROL TO THE ERROR
17743 : CALL AT ERR2.
17744 :
17745 101376 012601 MODFSUB: MOV (SP)+,R1 ;GET A POINTER TO THE ARGUMENTS
17746 101400 012700 000200 MOV #200,R0 ;SET FD MODE.
17747 101404 170100 LDFPS R0
17748 101406 010100 MOV R1,R0 ;SET UP ACO
17749 101410 172410 LDD (R0),ACO
17750 101412 012700 101572 MOV #MODP1,R0 ;PUT A BACKGROUND PATTERN INTO AC1.
17751 101416 172510 LDD (R0),AC1
17752 101420 016100 000020 MOV 20(R1),R0 ;SET UP THE FPS.
17753 101424 170100 LDFPS R0
17754 101426 010100 MOV R1,R0 ;COMPUTE THE ADDRESS OF THE FSRC.
17755 101430 062700 000004 ADD #4,R0
17756 :
17757 101434 171410 1$: MODF (R0),ACO ;EXECUTE THE TEST INSTRUCTION.
17758 :
17759 101436 170204 STFPS R4 ;GET THE FPS.
17760 101440 012700 000200 MOV #200,R0 ;SET FD MODE.
17761 101444 170100 LDFPS R0
17762 101446 012700 101552 MOV #MODFT0,R0 ;GET THE FRACTIONAL RESULT.
17763 101452 174010 STD ACO,(R0)
17764 101454 012700 101562 MOV #MODFT1,R0 ;GET THE INTEGER RESULT.
17765 101460 174110 STD AC1,(R0)
17766 101462 012702 101552 MOV #MODFT0,R2 ;CHECK THE FRACTIONAL RESULT.
17767 101466 026112 000010 CMP 10(R1),(R2)
17768 101472 001401 BEQ 2$
17769 101474 104000 EMT ;
17770 101476 026162 000012 000002 2$: CMP 12(R1),2(R2)
17771 101504 001401 BEQ 3$
17772 101506 104000 EMT ;
17773 101510 012702 101562 3$: MOV #MODFT1,R2 ;CHECK THE INTEGER RESULT.
17774 101514 026112 000014 CMP 14(R1),(R2)
17775 101520 001401 BEQ 4$

```

```
17776 101522 104000          EMT          ;
17777 101524 026162 000016 000002 4$:  CMP      16(R1),2(R2) ;
17778 101532 001401          BEQ      5$          ;
17779 101534 104000          EMT          ;
17780 101536 026104 000022          5$:  CMP      22(R1),R4    ;CHECK THE FPS.
17781 101542 001401          BEQ      9$          ;
17782 101544 104000          EMT          ;
17783 101546 000161 000024          9$:  JMP      24(R1)    ;RETURN.
17784
17785 101552 000000 000000 000000 MODFT0: .WORD 0,0,0,0
17786 101560 000000
17787
17788 101562 000000 000000 000000 MODFT1: .WORD 0,0,0,0
17789 101570 000000
17790
17791 101572 177777 177777 177777 MODP1: .WORD -1,-1,-1,-1
17792 101600 177777
17793
17794 101602          GGGDONE:
17795 101602 004767 022764          JSR      PC,,RSET    ;GO INITIALIZE THE FPS AND STACK; AND
17796                                     ;SEE IF THE USER HAS EXPRESSED
17797                                     ;THE DESIRE TO CHANGE THE SOFTWARE
17798                                     ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
17799                                     ;THE USER TYPED CONTROL G?).
17800
17801
17802
17803
17804
17805 ;*****
17806 ;TEST 506      MODD TEST
17807 ;*****
17808 TS506:
17809 ;MODD WITH (FSRC=AC=0)
17810 101606 004737 102622 HHH1:  JSR      PC, @#MODDSUB
17811 101612 000000 000000 000000 1$:  .WORD 0,0,0,0      ;AC
17812 101620 000000
17813 101622 000000 000000 000000 2$:  .WORD 0,0,0,0      ;FSRC
17814 101630 000000
17815 101632 000000 000000 000000 3$:  .WORD 0,0,0,0      ;FRACTIONAL RES.
17816 101640 000000
17817 101642 000000 000000 000000 4$:  .WORD 0,0,0,0      ;INTEGER RES.
17818 101650 000000
17819 101652 000200          7$:  200              ;FPS BEFORE EXECUTION.
17820 101654 000204          204              ;FPS AFTER EXECUTION.
17821
17822 ;MODD TEST WITH FSRC=0
17823 101656 004737 102622 HHH2:  JSR      PC, @#MODDSUB
17824 101662 012345 067012 1$:  .WORD 012345,67012  ;AC
17825 101666 034567 012345          .WORD 34567,012345
17826 101672 000000 000000 000000 2$:  .WORD 0,0,0,0      ;FSRC
17827 101700 000000
17828 101702 000000 000000 000000 3$:  .WORD 0,0,0,0      ;FRACTIONAL RES.
17829 101710 000000
17830 101712 000000 000000 000000 4$:  .WORD 0,0,0,0      ;INTEGER RES.
17831 101720 000000
```

```
17832 101722 000213 7$: 213 ;FPS BEFORE EXECUTION.  
17833 101724 000204 204 ;FPS AFTER EXECUTION.  
17834  
17835 ;MODD TEST WITH (AC=0)  
17836 101726 004737 102622 HHH3: JSR PC, @#MODDSUB  
17837 101732 000000 000000 000000 1$: .WORD 0,0,0,0 ;AC  
17838 101740 000000  
17839 101742 072727 127272 2$: .WORD 72727,127272 ;FSRC  
17840 101746 072727 127272 .WORD 72727,127272  
17841 101752 000000 000000 000000 3$: .WORD 0,0,0,0 ;FRACTIONAL RES.  
17842 101760 000000  
17843 101762 000000 000000 000000 4$: .WORD 0,0,0,0 ;INTEGER RES.  
17844 101770 000000  
17845 101772 000213 7$: 213 ;FPS BEFORE EXECUTION.  
17846 101774 000204 204 ;FPS AFTER EXECUTION.  
17847  
17848 ;MODD TEST WITH EXPONENT OF THE RESULT = 57  
17849 101776 004737 102622 HHH4: JSR PC, @#MODDSUB  
17850 102002 056252 125252 1$: .WORD 56252,125252 ;AC  
17851 102006 125252 125250 .WORD 125252,125250  
17852 102012 040300 000000 000000 2$: .WORD 40300,0,0,0 ;FSRC  
17853 102020 000000  
17854 102022 000000 000000 000000 3$: .WORD 0,0,0,0 ;FRACTIONAL RES.  
17855 102030 000000  
17856 102032 056377 177777 177777 4$: .WORD 56377,-1,-1,-4 ;INTEGER RES.  
17857 102040 177774  
17858 102042 000213 7$: 213 ;FPS BEFORE EXECUTION.  
17859 102044 000204 204 ;FPS AFTER EXECUTION.  
17860  
17861 ;MODD TEST WITH EXPONENT OF THE RESULT = 79  
17862 102046 004737 102622 HHH5: JSR PC, @#MODDSUB  
17863 102052 140240 000000 000000 1$: .WORD 140240,0,0,0 ;AC  
17864 102060 000000  
17865 102062 063714 146314 2$: .WORD 63714,146314 ;FSRC  
17866 102066 133572 167737 .WORD 133572,167737  
17867 102072 000000 000000 000000 3$: .WORD 0,0,0,0 ;FRACTIONAL RES.  
17868 102100 000000  
17869 102102 163777 177777 4$: .WORD 163777,-1 ;INTEGER RES.  
17870 102106 162531 125726 .WORD 162531,125726  
17871 102112 000210 7$: 210 ;FPS BEFORE EXECUTION.  
17872 102114 000204 204 ;FPS AFTER EXECUTION.  
17873  
17874 ;MODD TEST WITH EXPONENT OF THE RESULT = 57  
17875 102116 004737 102622 HHH6: JSR PC, @#MODDSUB  
17876 102122 056200 000000 000000 1$: .WORD 56200,0,0,1 ;AC  
17877 102130 000001  
17878 102132 040340 000000 000000 2$: .WORD 40340,0,0,0 ;FSRC  
17879 102140 000000  
17880 102142 000000 000000 000000 3$: .WORD 0,0,0,0 ;FRACTIONAL RES.  
17881 102150 000000  
17882 102152 056340 000000 000000 4$: .WORD 56340,0,0,1 ;INTEGER RES.  
17883 102160 000001  
17884 102162 000213 7$: 213 ;FPS BEFORE EXECUTION.  
17885 102164 000204 204 ;FPS AFTER EXECUTION.  
17886  
17887 ;MODD TEST WITH EXPONENT OF THE RESULT = 56
```

```
17888 102166 004737 102622 HHH7: JSR PC,2#MODDSUB
17889 102172 056000 000000 000000 1$: .WORD 56000,0,0,1 ;AC
17890 102200 000001 17891 102202 040340 000000 000000 2$: .WORD 40340,0,0,0 ;FSRC
17892 102210 000000 17893 102212 040100 000000 000000 3$: .WORD 40100,0,0,0 ;FRACTIONAL RES.
17894 102220 000000 17895 102222 056140 000000 000000 4$: .WORD 56140,0,0,1 ;INTEGER RES.
17896 102230 000001 17897 102232 000213 7$: 213 ;FPS BEFORE EXECUTION.
17898 102234 000200 200 ;FPS AFTER EXECUTION.
17899
17900 ;MODD TEST WITH EXPONENT OF THE RESULT = 36
17901 102236 004737 102622 HHH8: JSR PC,2#MODDSUB
17902 102242 051177 177777 177777 1$: .WORD 51177,-1,-1,-1 ;AC
17903 102250 177777 17904 102252 040200 000000 000000 2$: .WORD 40200,0,0,0 ;FSRC
17905 102260 000000 17906 102262 040177 177760 000000 3$: .WORD 40177,-20,0,0 ;FRACTIONAL RES.
17907 102270 000000 17908 102272 051177 177777 177760 4$: .WORD 51177,-1,-20,0 ;INTEGER RES.
17909 102300 000000 17910 102302 000217 7$: 217 ;FPS BEFORE EXECUTION.
17911 102304 000200 200 ;FPS AFTER EXECUTION.
17912
17913 ;MODD TEST WITH EXPONENT OF THE RESULT = 30
17914 102306 004737 102622 HHH9: JSR PC,2#MODDSUB
17915 102312 040200 000000 000000 1$: .WORD 40200,0,0,0 ;AC
17916 102320 000000 17917 102322 047577 177777 2$: .WORD 47577,-1 ;FSRC
17918 102326 176000 000001 17919 102332 031600 000000 000000 3$: .WORD 176000,1 ;FRACTIONAL RES.
17920 102340 000000 17921 102342 047577 177777 4$: .WORD 31600,0,0,0 ;INTEGER RES.
17922 102346 176000 000000 .WORD 47577,-1
17923 102352 000200 7$: 200 ;FPS BEFORE EXECUTION.
17924 102354 000200 200 ;FPS AFTER EXECUTION.
17925
17926 ;MODD TEST WITH EXPONENT OF THE RESULT = 31
17927 102356 004737 102622 HHH10: JSR PC,2#MODDSUB
17928 102362 047777 177777 1$: .WORD 47777,-1 ;AC
17929 102366 177000 000000 17930 102372 040200 000000 000000 2$: .WORD 177000,0 ;FSRC
17931 102400 000000 17932 102402 000000 000000 000000 3$: .WORD 40200,0,0,0 ;FRACTIONAL RES.
17933 102410 000000 17934 102412 047777 177777 4$: .WORD 0,0,0,0 ;INTEGER RES.
17935 102416 177000 000000 .WORD 47777,-1
17936 102422 000213 7$: 213 ;FPS BEFORE EXECUTION.
17937 102424 000204 204 ;FPS AFTER EXECUTION.
17938
17939 ;MODD TEST WITH EXPONENT OF THE RESULT = 0
17940 102426 004737 102622 HHH11: JSR PC,2#MODDSUB
17941 102432 040200 000000 000000 1$: .WORD 40200,0,0,0 ;AC
17942 102440 000000 17943 102442 040177 072727 2$: .WORD 40177,72727 ;FSRC
```

```

17944 102446 127272 072727          .WORD 127272,72727
17945 102452 040177 072727          3$: .WORD 40177,72727 ;FRACTIONAL RES.
17946 102456 127272 072727          .WORD 127272,72727
17947 102462 000000 000000 000000 4$: .WORD 0,0,0,0 ;INTEGER RES.
17948 102470 000000
17949 102472 000200          7$: 200 ;FPS BEFORE EXECUTION.
17950 102474 000200          200 ;FPS AFTER EXECUTION.
  
```

```

17951
17952 ;MODD TEST WITH EXPONENT OF THE RESULT = -115
  
```

```

17953 102476 004737 102622 HHH12: JSR PC,@#MODDSUB
17954 102502 003777 177777 1$: .WORD 3377,-1 ;AC
17955 102506 177777 052525          .WORD -1,52525
17956 102512 040200 000000 000000 2$: .WORD 40200,0,0,0 ;FSRC
17957 102520 000000
17958 102522 003377 177777 3$: .WORD 3377,-1 ;FRACTIONAL RES.
17959 102526 177777 052525          .WORD -1,52525
17960 102532 000000 000000 000000 4$: .WORD 0,0,0,0 ;INTEGER RES.
17961 102540 000000
17962 102542 000200          7$: 200 ;FPS BEFORE EXECUTION.
17963 102544 000200          200 ;FPS AFTER EXECUTION.
  
```

```

17964
17965 ;MODD TEST WITH EXPONENT OF THE RESULT = -63, IN ROUND MODE.
  
```

```

17966 102546 004737 102622 HHH13: JSR PC,@#MODDSUB
17967 102552 040300 000000 000000 1$: .WORD 40300,0,0,0 ;AC
17968 102560 000000
17969 102562 020200 000000 000000 2$: .WORD 20200,0,0,1 ;FSRC
17970 102570 000001
17971 102572 020300 000000 000000 3$: .WORD 20300,0,0,2 ;FRACTIONAL RES.
17972 102600 000002
17973 102602 000000 000000 000000 4$: .WORD 0,0,0,0 ;INTEGER RES.
17974 102610 000000
17975 102612 000200          7$: 200 ;FPS BEFORE EXECUTION.
17976 102614 000200          200 ;FPS AFTER EXECUTION.
17977 102616 000137 103016 9$: JMP @#HHHDONE ;GO TO THE NEXT TEST.
  
```

```

17978
17979 ;THIS SUBROUTINE, MODDSUB, IS CALLED TO SETUP THE
17980 ;OPERANDS, EXECUTE THE MODD INSTRUCTION AND CHECK THE RESULTS.
17981 ;IT IS CALLED THUS:
  
```

```

17982 :
17983 :          ACARG: .WORD X,X,X,X          ;AC OPERAND
17984 :          FSRCARG: .WORD X,X,X,X        ;FSRC OPERAND
17985 :          FRES: .WORD X,X,X,X           ;FRACTIONAL RESULT
17986 :          INTRES: .WORD X,X,X,X         ;INTEGER RESULT
17987 :          ERFRES: .WORD X,X,X,X         ;ERROR FRACTION RESULT
17988 :          ERINTRES: .WORD X,X,X,X       ;ERROR INTEGER RESULT
17989 :          FPSB: .WORD X                  ;FPS BEFORE EXECUTION
17990 :          FPSA: .WORD X                  ;FPS AFTER EXECUTION
17991 :          ERR1: ERROR X                  ;FRACTION ERROR
17992 :          BR CONT                        ;
17993 :          ERR2: ERROR X                  ;INTEGER ERROR
17994 :          CONT:                          ;RETURN ADDRESS
  
```

```

17995
17996 ;THE OPERANDS ARE SET UP (USING ACO FOR THE AC ARGUMENT). THE MODD
17997 ;INSTRUCTION IS EXECUTED. THEN THE RESULTS ARE RETRIEVED.
17998 ;THE FRACTION PART OF THE RESULT IS COMPARED WITH FRES. IF THIS IS CORRECT
17999 ;THEN THE INTEGER PART IS COMPARED WITH INTRES. IF BOTH OF THESE ARE CORRECT
  
```

18000
 18001
 18002
 18003
 18004
 18005
 18006
 18007
 18008
 18009
 18010
 18011
 18012 102622 012601
 18013 102624 012700 000200
 18014 102630 170100
 18015 102632 010100
 18016 102634 172410
 18017 102636 012700 101572
 18018 102642 172510
 18019 102644 016100 000040
 18020 102650 170100
 18021 102652 010100
 18022 102654 062700 000010
 18023
 18024 102660 171410
 18025
 18026 102662 170204
 18027 102664 012700 000200
 18028 102670 170100
 18029 102672 012700 102776
 18030 102676 174010
 18031 102700 012700 103006
 18032 102704 174110
 18033 102706 012702 102776
 18034 102712 010103
 18035 102714 062703 000020
 18036 102720 012705 000004
 18037 102724 022223
 18038 102726 001401
 18039 102730 104000
 18040 102732 077504
 18041 102734 012702 103006
 18042 102740 010103
 18043 102742 062703 000030
 18044 102746 012705 000004
 18045 102752 022223
 18046 102754 001401
 18047 102756 104000
 18048 102760 077504
 18049 102762 026104 000042
 18050 102766 001401
 18051 102770 104000
 18052 102772 000161 000044
 18053
 18054 102776 000000 000000 000000
 18055 103004 000000

; THEN THE FPS IS COMPARED WITH FPSA. AFTER EXECUTION IF NO ERRORS OCCURRED
 ; THEN MODDSUB WILL RETURN TO CONT. IF THE FPS WAS INCORRECT
 ; IT IS REPORTED HERE. IF THE FRACTION IS INCORRECT IT IS COMPARED WITH
 ; THE ANTICIPATED BAD FRACTION, ERFRES. IF THIS DOESN'T MATCH
 ; THE TRUE RESULT THEN THE ERROR IS REPORTED HERE. IF THE ANTICIPATED
 ; FAILURE MATCHES THE TRUE RESULT THEN MODDSUB PASSES CONTROL TO THE
 ; ERROR CALL AT ERR1. LIKEWISE IF THE INTEGER PART OF THE RESULT IS
 ; NOT CORRECT THEN IT IS COMPARED WITH THE ANTICIPATED INTEGER
 ; FAILURE. IF THIS DOESN'T MATCH THEN THE ERROR IS REPORTED HERE.
 ; IF A MATCH IS MADE HOWEVER, MODDSUB WILL RETURN CONTROL TO THE ERROR
 ; CALL AT ERR2.

MODDSUB: MOV (SP)+,R1 ;GET A POINTER TO THE ARGUMENTS
 MOV #200,R0 ;SET FD MODE.
 LDFPS R0
 MOV R1,R0 ;SET UP ACO
 LDD (R0),ACO
 MOV #MODPT1,R0 ;PUT A BACKGROUND PATTERN INTO AC1.
 LDD (R0),AC1
 MOV 40(R1),R0 ;SET UP THE FPS.
 LDFPS R0
 MOV R1,R0 ;COMPUTE THE ADDRESS OF THE FSRC.
 ADD #10,R0
 1\$: MODD (R0),ACO ;EXECUTE THE TEST INSTRUCTION.
 STFPS R4 ;GET THE FPS.
 MOV #200,R0 ;SET FD MODE.
 LDFPS R0
 MOV #MODDT0,R0 ;GET THE FRACTIONAL RESULT.
 STD ACO,(R0)
 MOV #MODDT1,R0 ;GET THE INTEGER RESULT.
 STD AC1,(R0)
 MOV #MODDT0,R2 ;CHECK THE FRACTIONAL RESULT.
 MOV R1,R3
 ADD #20,R3
 MOV #4,R5
 2\$: CMP (R2)+,(R3)+
 BEQ 4\$
 EMT ;
 4\$: SOB R5,2\$;CHECK THE INTEGER RESULT.
 MOV R1,R3
 ADD #30,R3
 MOV #4,R5
 3\$: CMP (R2)+,(R3)+
 BEQ 5\$
 EMT ;
 5\$: SOB R5,3\$;CHECK THE FPS.
 CMP 42(R1),R4
 BEQ 9\$
 EMT ;
 9\$: JMP 44(R1) ;RETURN.

MODDT0: .WORD 0,0,0,0

18056
18057 103006 000000 000000 000000 MODDT1: .WORD 0,0,0,0
18058 103014 000000
18059
18060 103016
18061 103016 004767 021550 HHHDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
18062 ;SEE IF THE USER HAS EXPRESSED
18063 ;THE DESIRE TO CHANGE THE SOFTWARE
18064 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
18065 ;THE USER TYPED CONTROL G?).
18066
18067
18068
18069

18070 ;*****
18071 ;TEST 507 UNDER/OVER FLOW, USING MODF WITH TRAPS DISABLED, TEST
18072 ;*****
18073 103022 TS507:

18074 ;UNDERFLOW TEST, WITH EXPONENT OF THE RESULT = -129, FIU = 1, FID = 1
18075 MPM1: JSR PC,MODFOV
18076 103022 004767 000214 1\$: .WORD 20123,45676 ;AC
18077 103026 020123 045676 2\$: .WORD 20200,0 ;FSRC
18078 103032 020200 000000 3\$: .WORD 123,45676 ;FRACTIONAL RES.
18079 103036 000123 045676 4\$: .WORD 0,0 ;INTEGER RES.
18080 103042 000000 000000 7\$: 42000 ;FPS BEFORE EXECUTION.
18081 103046 042000 ;FPS AFTER EXECUTION.
18082 103050 142004 ;FEC
18083 103052 000012 ;
18084 103054 104000 EMT ;
18085 ;

18086 ;UNDERFLOW EXP OF RESULT = -193, FIU = 0, FID = 1
18087 103056 004737 103242 MPM2: JSR PC,MODFOV
18088 103062 010200 000000 1\$: .WORD 10200,0 ;AC
18089 103066 010000 000000 2\$: .WORD 10000,0 ;FSRC
18090 103072 000000 000000 3\$: .WORD 0,0 ;FRACTIONAL RES.
18091 103076 000000 000000 4\$: .WORD 0,0 ;INTEGER RES.
18092 103102 005013 7\$: 5013 ;FPS BEFORE EXECUTION.
18093 103104 005004 ;FPS AFTER EXECUTION.
18094 103106 000012 ;FEC
18095 103110 000240 NOP ;
18096 ;

18097 ;OVERFLOW TEST WITH EXPONENT OF THE RESULT = 128, FIV = 1, FID = 1
18098 103112 004737 103242 MPM3: JSR PC,MODFOV
18099 103116 060052 125252 1\$: .WORD 60052,125252 ;AC
18100 103122 060200 000000 2\$: .WORD 60200,0 ;FSRC
18101 103126 000000 000000 3\$: .WORD 0,0 ;FRACTIONAL RES.
18102 103132 000052 125252 4\$: .WORD 52,125252 ;INTEGER RES.
18103 103136 041000 7\$: 41000 ;FPS BEFORE EXECUTION.
18104 103140 141006 ;FPS AFTER EXECUTION.
18105 103142 000010 ;FEC
18106 103144 8\$: 10 ;
18107 103144 104000 EMT ;
18108 ;

18109 ;OVERFLOW TEST WITH EXPONENT OF THE RESULT = 130, FIV = 0, FID = 1
18110 103146 004737 103242 MPM4: JSR PC,MODFOV
18111 103152 060345 067654 1\$: .WORD 60345,67654 ;AC
2\$: .WORD 60200,0 ;FSRC

```

18112 103162 000000 000000
18113 103166 000000 000000
18114 103172 006011
18115 103174 006006
18116 103176 000010
18117 103200 000240
18118
18119
18120 103202 004737 103242
18121 103206 160252 125252
18122 103212 060000 000000
18123 103216 000000 000000
18124 103222 100052 125252
18125 103226 041000
18126 103230 141006
18127 103232 000010
18128 103234
18129 103234 104000
18130 103236 000137 103456
18131
18132
18133
18134
18135
18136
18137
18138
18139
18140
18141
18142
18143
18144
18145
18146
18147
18148
18149
18150
18151
18152
18153
18154
18155
18156
18157
18158
18159
18160
18161
18162
18163
18164
18165
18166 103242 012601
18167 103244 012700 000200
  
```

```

3$: .WORD 0,0 ;FRACTIONAL RES.
4$: .WORD 0,0 ;INTEGER RES.
7$: 6011 ;FPS BEFORE EXECUTION.
6006 ;FPS AFTER EXECUTION.
10 ;FEC
8$: NOP
;OVERFLOW TEST WITH EXPONENT OF THE RESULT = 128, RESULT NEGATIVE
;AND FIV = 1, FID = 1
MMMS: JSR PC,@MODFOV
1$: .WORD 160252,125252 ;AC
2$: .WORD 60000,0 ;FSRC
3$: .WORD 0,0 ;FRACTIONAL RES.
4$: .WORD 100052,125252 ;INTEGER RES.
7$: 41000 ;FPS BEFORE EXECUTION.
141006 ;FPS AFTER EXECUTION.
10 ;FEC
8$:
9$: EMT ;
JMP @MMMMDONE ;GO TO THE NEXT TEST.
  
```

```

;THIS SUBROUTINE, MODFOV, IS CALLED TO SETUP THE
;OPERANDS, EXECUTE THE MODF INSTRUCTION AND CHECK THE RESULTS.
;IT IS CALLED THUS:
  
```

```

:
: ACARG: .WORD X,X ;AC OPERAND
: FSRCARG: .WORD X,X ;FSRC OPERAND
: FRES: .WORD X,X ;FRACTIONAL RESULT
: INTRES: .WORD X,X ;INTEGER RESULT
: ERFRES: .WORD X,X ;ERROR FRACTION RESULT
: ERINTRES: .WORD X,X ;ERROR INTEGER RESULT
: FPSB: .WORD X ;FPS BEFORE EXECUTION
: FPSA: .WORD X ;FPS AFTER EXECUTION
: FEC: .WORD X ;FEC
: ERR1: ERROR X ;FEC ERROR
: BR CONT
: ERR2: ERROR X ;INTEGER ERROR
: CONT: ;RETURN ADDRESS
  
```

```

;THE OPERANDS ARE SET UP (USING ACO FOR THE AC ARGUMENT). THE MODF
;INSTRUCTION IS EXECUTED. THEN THE RESULTS ARE RETRIEVED.
;THE FRACTION PART OF THE RESULT IS COMPARED WITH FRES. IF THIS IS CORRECT
;THEN THE INTEGER PART IS COMPARED WITH INTRES. IF BOTH OF THESE ARE CORRECT
;THEN THE FPS IS COMPARED WITH FPSA. AFTER EXECUTION IF NO ERRORS OCCURRED
;THEN MODFOV WILL RETURN TO CONT. IF THE FPS WAS INCORRECT
;IT IS REPORTED HERE. IF THE FRACTION IS INCORRECT IT IS COMPARED WITH
;THE ANTICIPATED BAD FRACTION, ERFRES. IF THIS DOESN'T MATCH
;THE TRUE RESULT THEN THE ERROR IS REPORTED HERE. IF THE ANTICIPATED
;FAILURE MATCHES THE TRUE RESULT THEN MODFOV PASSES CONTROL TO THE
;ERROR CALL AT ERR1. LIKEWISE IF THE INTEGER PART OF THE RESULT IS
;NOT CORRECT THEN IT IS COMPARED WITH THE ANTICIPATED INTEGER
;FAILURE. IF THIS DOESN'T MATCH THEN THE ERROR IS REPORTED HERE.
;IF A MATCH IS MADE HOWEVER, MODFOV WILL RETURN CONTROL TO THE ERROR
;CALL AT ERR2.
  
```

```

MODFOV: MOV (SP)+,R1 ;GET A POINTER TO THE ARGUMENTS
MOV #200,R0 ;SET FD MODE.
  
```

18168	103250	170100				LDFPS	R0		
18169	103252	010100				MOV	R1,R0	;SET UP ACO	
18170	103254	172410				LDD	(R0),ACO		
18171	103256	012700	101572			MOV	#MODP1,R0	;PUT A BACKGROUND PATTERN INTO AC1.	
18172	103262	172510				LDD	(R0),AC1		
18173	103264	016100	000020			MOV	20(R1),R0	;SET UP THE FPS.	
18174	103270	170100				LDFPS	R0		
18175	103272	010100				MOV	R1,R0	;COMPUTE THE ADDRESS OF THE FSRC.	
18176	103274	062700	000004			ADD	#4,R0		
18177									
18178	103300	171410			1\$:	MODF	(R0),ACO	;EXECUTE THE TEST INSTRUCTION.	
18179									
18180	103302	170204				STFPS	R4	;GET THE FPS.	
18181	103304	170305				STST	R5	;GET FEC.	
18182	103306	012700	000200			MOV	#200,R0	;SET FD MODE.	
18183	103312	170100				LDFPS	R0		
18184	103314	012700	103436			MOV	#MODF0,R0	;GET THE FRACTIONAL RESULT.	
18185	103320	174010				STD	ACO,(R0)		
18186	103322	012700	103446			MOV	#MODF1,R0	;GET THE INTEGER RESULT.	
18187	103326	174110				STD	AC1,(R0)		
18188	103330	012702	103436			MOV	#MODF0,R2	;CHECK THE FRACTIONAL RESULT.	
18189	103334	026112	000010			CMP	10(R1),(R2)		
18190	103340	001401				BEQ	2\$		
18191	103342	104000				EMT			
18192	103344	026162	000012	000002	2\$:	CMP	12(R1),2(R2)		
18193	103352	001401				BEQ	3\$		
18194	103354	104000				EMT			
18195	103356	012702	103446		3\$:	MOV	#MODF1,R2	;CHECK THE INTEGER RESULT.	
18196	103362	026112	000014			CMP	14(R1),(R2)		
18197	103366	001401				BEQ	4\$		
18198	103370	104000				EMT			
18199	103372	026162	000016	000002	4\$:	CMP	16(R1),2(R2)		
18200	103400	001401				BEQ	5\$		
18201	103402	104000				EMT			
18202	103404	026104	000022		5\$:	CMP	22(R1),R4	;CHECK THE FPS.	
18203	103410	001401				BEQ	6\$		
18204	103412	104000				EMT			
18205	103414	026105	000024		6\$:	CMP	24(R1),R5	;CHECK THE FEC.	
18206	103420	001002				BNE	25\$;BRANCH IF INCORRECT.	
18207									
18208	103422	000161	000030		9\$:	JMP	30(R1)	;RETURN.	
18209						;REPORT	FEC ERROR.		
18210	103426	010102			25\$:	MOV	R1,R2		
18211	103430	062702	000026			ADD	#26,R2		
18212	103434	000112				JMP	(R2)		
18213									
18214	103436	000000	000000	000000		MODF0:	.WORD	0,0,0,0	
18215	103444	000000							
18216									
18217	103446	000000	000000	000000		MODF1:	.WORD	0,0,0,0	
18218	103454	000000							
18219									
18220	103456					MMMDONE:			
18221	103456	004767	021110			JSR	PC,.RSET	;GO INITIALIZE THE FPS AND STACK; AND	
18222								;SEE IF THE USER HAS EXPRESSED	
18223								;THE DESIRE TO CHANGE THE SOFTWARE	

18224
18225
18226
18227
18228
18229
18230
18231
18232
18233 103462
18234
18235
18236 103462 004737 103746
18237 103466 020252 125252
18238 103472 125252 125252
18239 103476 020100 000000 000000
18240 103504 000000
18241 103506 000177 177777 177777
18242 103514 177777
18243 103516 000000 000000 000000
18244 103524 000000
18245 103526 042200
18246 103530 142204
18247 103532 000012
18248 103534
18249 103534 104000
18250
18251 103536 004737 103746
18252 103542 010000 000000
18253 103546 123456 000000
18254 103552 010200 000000 000000
18255 103560 000000
18256 103562 000000 000000 000000
18257 103570 000000
18258 103572 000000 000000 000000
18259 103600 000000
18260 103602 005213
18261 103604 005204
18262 103606 000012
18263 103610 000240
18264
18265 103612 004737 103746
18266 103616 060252 125252
18267 103622 125252 125252
18268 103626 060100 000000 000000
18269 103634 000000
18270 103636 000000 000000 000000
18271 103644 000000
18272 103646 000177 177777 177777
18273 103654 177777
18274 103656 041200
18275 103660 141206
18276 103662 000010
18277 103664
18278 103664 104000
18279

;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;TEST 510 UNDER/OVER FLOW, USING MODD WITH TRAPS DISABLED, TEST

TS510:

;UNDERFLOW TEST WITH EXPONENT OF THE RESULT = -129, FIU = 1, FID = 1

NNN1: JSR PC, @#MODDOV
1\$: .WORD 20252, 125252 ;AC
.WORD 125252, 125252
2\$: .WORD 20100, 0, 0, 0 ;FSRC
3\$: .WORD 177, -1, -1, -1 ;FRACTIONAL RES.
4\$: .WORD 0, 0, 0, 0 ;INTEGER RES.
7\$: 42200 ;FPS BEFORE EXECUTION.
142204 ;FPS AFTER EXECUTION.
12 ;FEC
8\$:

EMT
;UNDERFLOW TEST WITH EXPONENT OF THE RESULT = -193, FIU = 0, FID = 1

NNN2: JSR PC, @#MODDOV
1\$: .WORD 10000, 0 ;AC
.WORD 123456, 0
2\$: .WORD 10200, 0, 0, 0 ;FSRC
3\$: .WORD 0, 0, 0, 0 ;FRACTIONAL RES.
4\$: .WORD 0, 0, 0, 0 ;INTEGER RES.
7\$: 5213 ;FPS BEFORE EXECUTION.
5204 ;FPS AFTER EXECUTION.
12
8\$: NOP

;OVERFLOW TEST WITH EXPONENT OF THE RESULT = 128, FIV = 1, FID = 1

NNN3: JSR PC, @#MODDOV
1\$: .WORD 60252, 125252 ;AC
.WORD 125252, 125252
2\$: .WORD 60100, 0, 0, 0 ;FSRC
3\$: .WORD 0, 0, 0, 0 ;FRACTIONAL RES.
4\$: .WORD 177, -1, -1, -1 ;INTEGER RES.
7\$: 41200 ;FPS BEFORE EXECUTION.
141206 ;FPS AFTER EXECUTION.
10 ;FEC
8\$:

EMT
;OVERFLOW TEST WITH EXPONENT OF THE RESULT = 130, FIV = 0, FID = 1

```

18280 103666 004737 103746
18281 103672 060200 000000
18282 103676 125252 000000
18283 103702 060200 000000 000000
18284 103710 000000
18285 103712 000000 000000 000000
18286 103720 000000
18287 103722 000000 000000 000000
18288 103730 000000
18289 103732 006211
18290 103734 006206
18291 103736 000010
18292 103740 000240
18293 103742 000137 104162
  
```

```

NNN4: JSR PC,@#MODDOV
1$: .WORD 60200,0 ;AC
      .WORD 125252,0
2$: .WORD 60200,0,0,0 ;FSRC
3$: .WORD 0,0,0,0 ;FRACTIONAL RES.
4$: .WORD 0,0,0,0 ;INTEGER RES.
7$: 6211 ;FPS BEFORE EXECUTION.
     6206 ;FPS AFTER EXECUTION.
     10 ;FEC
8$: NOP
9$: JMP @#NNNDONE ;GO TO NEXT TEST.
  
```

```

:THIS SUBROUTINE, MODDOV, IS CALLED TO SETUP THE
:OPERANDS, EXECUTE THE MODD INSTRUCTION AND CHECK THE RESULTS.
:IT IS CALLED THUS:
  
```

```

18294
18295
18296
18297
18298
18299
18300
18301
18302
18303
18304
18305
18306
18307
18308
18309
18310
18311
18312
18313
18314
18315
18316
18317
18318
18319
18320
18321
18322
18323
18324
18325
18326
18327
  
```

```

ACARG: .WORD X,X,X,X ;AC OPERAND
FSRCARG: .WORD X,X,X,X ;FSRC OPERAND
FRES: .WORD X,X,X,X ;FRACTIONAL RESULT
INTRES: .WORD X,X,X,X ;INTEGER RESULT
ERFRES: .WORD X,X,X,X ;ERROR FRACTION RESULT
ERINTRES: .WORD X,X,X,X ;ERROR INTEGER RESULT
FPSB: .WORD X ;FPS BEFORE EXECUTION
FPSA: .WORD X ;FPS AFTER EXECUTION
ERR1: ERROR X ;FRACTION ERROR
      BR CONT
ERR2: ERROR X ;INTEGER ERROR
CONT: ;RETURN ADDRESS
  
```

```

:THE OPERANDS ARE SET UP (USING ACO FOR THE AC ARGUMENT). THE MODD
:INSTRUCTION IS EXECUTED. THEN THE RESULTS ARE RETRIEVED.
:THE FRACTION PART OF THE RESULT IS COMPARED WITH FRES. IF THIS IS CORRECT
:THEN THE INTEGER PART IS COMPARED WITH INTRES. IF BOTH OF THESE ARE CORRECT
:THEN THE FPS IS COMPARED WITH FPSA. AFTER EXECUTION IF NO ERRORS OCCURRED
:THEN MODDOV WILL RETURN TO CONT. IF THE FPS WAS INCORRECT
:IT IS REPORTED HERE. IF THE FRACTION IS INCORRECT IT IS COMPARED WITH
:THE ANTICIPATED BAD FRACTION, ERFRES. IF THIS DOESN'T MATCH
:THE TRUE RESULT THEN THE ERROR IS REPORTED HERE. IF THE ANTICIPATED
:FAILURE MATCHES THE TRUE RESULT THEN MODDOV PASSES CONTROL TO THE
:ERROR CALL AT ERR1. LIKEWISE IF THE INTEGER PART OF THE RESULT IS
:NOT CORRECT THEN IT IS COMPARED WITH THE ANTICIPATED INTEGER
:FAILURE. IF THIS DOESN'T MATCH THEN THE ERROR IS REPORTED HERE.
:IF A MATCH IS MADE HOWEVER, MODDOV WILL RETURN CONTROL TO THE ERROR
:CALL AT ERR2.
  
```

```

18328 103746 012601
18329 103750 012700 000200
18330 103754 170100
18331 103756 010100
18332 103760 172410
18333 103762 012700 101572
18334 103766 172510
18335 103770 016100 000040
  
```

```

MODDOV: MOV (SP)+,R1 ;GET A POINTER TO THE ARGUMENTS
        MOV #200,R0 ;SET FD MODE.
        LDFPS R0
        MOV R1,R0 ;SET UP ACO
        LDD (R0),ACO
        MOV #MODP1,R0 ;PUT A BACKGROUND PATTERN INTO AC1.
        LDD (R0),AC1
        MOV 40(R1),R0 ;SET UP THE FPS.
  
```

:TEST 511 MORE MICROCODES COVERAGE

```
TS511:
XT1:  MOV  #XT1A,@#244
      STFPS #0
      BR   XT2
XT1A:  EMT ;
XT2:  MOV  #-1,R0
      LDFPS #0
      STFPS R0
      TST  R0
      BEQ  XT2A
XT2A:  MOV  #XPAT0,R0
      LDF  -(R0),ACO
      CMP  #XPAT0-4,R0
      BEQ  XT2B
XT2B:  STFPS R0
      CMP  #4,R0 ;CHECK IF FZ IS SET?
      BEQ  XT3
XT3:  LDFPS #0
      MOV  #XPAT0,R0
      STF  ACO,-(R0)
      CMP  #XPAT0-4,R0
      BEQ  XT3A
XT3A:  STFPS R0
      TST  R0
      BEQ  XT4
XT4:  LDFPS #0
      MOV  #XT4A,@#244 ;INTRPT ON UNDEFINED VARIABLE
      LDFPS #04000
      LDF  @#XPAT0,ACO
      DIVF @#XPAT3,ACO ;GET UNDEFINED VARIABLE, _0
      EMT ;
XT4A:  STFPS R0
      CMP  #104004,R0 ;CHECK: FER,FIUV,FZ ARE SET?
      BEQ  XT4B
XT4B:  MOV  #XBUF,R0
      STF  ACO,(R0)
      TST  @#XBUF
      BEQ  XT5
XT5:  MOV  #XT5A,@#244 ;INTRPT ON UNDEFINED VARIBALE
      LDFPS #04000
      LDCDF @#XPAT3,ACO ;GET UNDEFINED VARIABLE, _0
      EMT ;
```

```
18392
18393
18394
18395 104166
18396 104166 012737 104202 000244
18397 104174 170227 000000
18398 104200 000401
18399 104202
18400 104202 104000
18401
18402 104204 012700 177777
18403 104210 170127 000000
18404 104214 170200
18405 104216 005700
18406 104220 001401
18407 104222 104000
18408 104224 012700 104760
18409 104230 172440
18410 104232 022700 104754
18411 104236 001401
18412 104240 104000
18413 104242 170200
18414 104244 022700 000004
18415 104250 001401
18416 104252 104000
18417 104254 170127 000000
18418 104260 012700 104760
18419 104264 174040
18420 104266 022700 104754
18421 104272 001401
18422 104274 104000
18423 104276 170200
18424 104300 005700
18425 104302 001401
18426 104304 104000
18427
18428 104306 170127 000000
18429 104312 012737 104336 000244
18430 104320 170127 004000
18431 104324 172437 104760
18432 104330 174437 105010
18433 104334 104000
18434 104336 170200
18435 104340 022700 104004
18436 104344 001401
18437 104346 104000
18438 104350 012700 104750
18439 104354 174010
18440 104356 005737 104750
18441 104362 001401
18442 104364 104000
18443
18444 104366 012737 104406 000244
18445 104374 170127 004000
18446 104400 177437 105010
18447 104404 104000
```


18504 104640 170127 000000
18505 104644 172437 105020
18506 104650 173437 105020
18507 104654 170200
18508 104656 022700 000004
18509 104662 001401
18510 104664 104000
18511
18512 104666 170127 000000
18513 104672 172437 105020
18514 104676 174437 105000
18515 104702 012700 104750
18516 104706 174010
18517 104710 022737 040176 104750
18518 104716 001401
18519 104720 104000
18520
18521 104722 170127 000000
18522 104726 172437 105030
18523 104732 174437 105040
18524 104736 170200
18525 104740 022700 000004
18526 104744 001445
18527 104746 104000
18528
18529
18530 104750 000000 000000 000000
18531 104756 000000
18532 104760 000000 000000 000000
18533 104766 000000
18534 104770 000001 000001 000001
18535 104776 000001
18536 105000 040401 000000 000000
18537 105006 000000
18538 105010 100000 000000 000000
18539 105016 000000
18540 105020 040400 000000 000000
18541 105026 000000
18542 105030 000207 000000 000000
18543 105036 000000
18544 105040 077007 000000 000000
18545 105046 000000
18546 105050 000000 000000 000000
18547 105056 000000
18548
18549 105060
18550 105060 004767 017506
18551
18552
18553
18554
18555
18556
18557
18558
18559

XT11: LDFPS #0
LDF @#XPAT4,ACO
CMPF @#XPAT4,ACO
STFPS R0
CMP #4,R0 ;CHECK IF FZ IS SET?
BEQ XT12
EMT ;
XT12: LDFPS #0
LDF @#XPAT4,ACO
DIVF @#XPAT2,ACO
MOV #XBUF,R0
STF ACO,(R0)
CMP #040176,@#XBUF ;CHECK DATA
BEQ XT13
EMT ;
XT13: LDFPS #0
LDF @#XPAT5,ACO
DIVF @#XPAT6,ACO
STFPS R0
CMP #4,R0
BEQ XTDONE
EMT ;

XBUF: .WORD 0,0,0,0
XPAT0: .WORD 0,0,0,0
XPAT1: .WORD 1,1,1,1
XPAT2: .WORD 40401,0,0,0
XPAT3: .WORD 100000,0,0,0
XPAT4: .WORD 040400,0,0,0
XPAT5: .WORD 207,0,0,0
XPAT6: .WORD 77007,0,0,0
XPAT0: .WORD 0,0,0,0

XTDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;*****

```
18560 ;TEST 512 STF WITH ILLEGAL ACCUMULATOR TEST
18561 :*****
18562 105064 TS512:
18563
18564 105064 005000 CLR R0 ;SET THE FPS.
18565 105066 170100 LDFPS R0
18566
18567 105070 012737 105110 000244 MOV #000T,@#FPVECT ;SET UP FOR FP TRAPS.
18568 105076 012737 105104 037364 MOV #1$,@#STMP2
18569
18570 105104 174007 1$: STF ACO,AC7 ;THIS TEST INSTRUCTION SHOULD
18571 ;CAUSE A TRAP.
18572
18573 ;REPORT FAILURE OF USE OF ILLEGAL ACCUMULATOR 7 TO CAUSE AN FPP TRAP.
18574 105106 0002:
18575 105106 104000 EMT ;INSTRUCTION DID NOT TRAP
18576
18577 ;TRAP TO 000T, HERE, WHEN THE EXPECTED ERROR OCCURS.
18578 105110 011600 000T: MOV (SP),R0 ;MAKE SURE THE ERROR OCCURRED
18579 105112 022700 105106 CMP #0002,R0 ;AT THE CORRECT ADDRESS.
18580 105116 001420 BEQ TS513
18581 105120 104000 EMT ;FLOATING POINT TRAP DID NOT OPERATE RIGHT
18582
18583 105122 170204 0003: STFPS R4 ;GET FPS.
18584 105124 170305 STST R5 ;GET FEC.
18585 105126 012702 100000 MOV #100000,R2 ;EXPECTED FPS
18586 105132 012703 000002 MOV #2,R3 ;EXPECTED FEC
18587 105136 022626 CMP (SP)+,(SP)+ ;RESET THE STACK.
18588
18589 105140 020204 CMP R2,R4 ;WAS FPS CORRECT?
18590 105142 001401 BEQ 0004
18591 105144 104000 EMT ;FPS INCORRECTLY SET AFTER USE OF ILLEGAL ACC
18592 105146 020305 0004: CMP R3,R5 ;WAS THE FEC CORRECT?
18593 105150 001401 BEQ 000DONE
18594 105152 104000 EMT ;INCORRECT FEC AFTER USE OF ILLEGAL ACC
18595
18596 105154 000DONE:
18597 105154 004767 017412 JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND
18598 ;SEE IF THE USER HAS EXPRESSED
18599 ;THE DESIRE TO CHANGE THE SOFTWARE
18600 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
18601 ;THE USER TYPED CONTROL G?).
18602
18603
18604
18605
18606 :*****
18607 ;TEST 513 FDST MODE 1, FLOATING MODE, TEST
18608 :*****
18609 105160 TS513:
18610
18611
18612 105160 012700 177777 MOV #-1,R0 ;SET UP A BACKGROUND PATTERN IN THE
18613 105164 012701 105274 MOV #PPPBF0,R1 ;INPUT BUFFER.
18614 105170 012702 000014 MOV #14,R2
18615 105174 010021 PPP2: MOV R0,(R1)+
```

CJKDJ80 11/23-B CPU CLUSTER DIAG.
CJKDJB.P11 26-MAY-82 11:14

G 12

DNMAC X24.07-563 26-MAY-82 11:18 PAGE 356
T513 FDST MODE 1, FLOATING MODE, TEST

SEQ 0355

18616 105176 077202
18617
18618 105200 012700 000200
18619 105204 170100
18620 105206 012700 105324
18621 105212 172410
18622

SOB R2,PPP2
MOV #200,R0 ;SET FD MODE.
LDFPS R0
MOV #PPPTP1,R0 ;PUT TEST DATA INTO ACO.
LDD (R0),ACO

18623 105214 012700 105310
18624 105220 005002
18625 105222 170102
18626
18627 105224 174010
18628
18629 105226 022700 105310
18630 105232 001401
18631 105234 104000
18632
18633 105236 012700 105310
18634 105242 012701 105324
18635 105246 022021
18636 105250 001031
18637 105252 022011
18638 105254 001027
18639 105256 022720 177777
18640 105262 001024
18641 105264 022710 177777
18642 105270 001021
18643 105272 000421
18644
18645 105274 177777 177777 177777
18646 105302 177777 177777 177777
18647
18648 105310 177777 177777 177777
18649 105316 177777 177777 177777
18650
18651 105324 123456 023456
18652 105330 034567 045671
18653
18654 105334
18655 105334 104000
18656 105336
18657 105336 004767 017230
18658
18659
18660
18661
18662
18663
18664
18665
18666
18667
18668
18669 105342
18670
18671
18672
18673 105342 012700 177777
18674 105346 012701 105456
18675 105352 012702 000014
18676 105356 010021
18677 105360 077202
18678

MOV #PPPBF1,R0 ;FDST ADDRESS.
CLR R2 ;CLEAR THE FPS.
LDFPS R2
PPP3: STF ACO,(R0) ;TEST INSTRUCTION.
CMP #PPPBF1,R0 ;WAS RO MODIFIED DURING EXECUTION?
BEQ PPP4
EMT ;RO MODIFIED
PPP4: MOV #PPPBF1,R0 ;CHECK THE DATA IN THE OUTPUT BUFFER.
MOV #PPPTP1,R1
CMP (R0)+,(R1)+
BNE PPP10 ;BRANCH IF INCORRECT.
CMP (R0)+,(R1)
BNE PPP10 ;BRANCH IF INCORRECT.
CMP #-1,(R0)+ ;WAS FLOATING MODE USED?
BNE PPP10 ;BRANCH IF NOT.
CMP #-1,(R0)
BNE PPP10
BR PPPDONE ;GO TO NEXT TEST.
PPPBF0: .WORD -1,-1,-1,-1,-1,-1
PPPBF1: .WORD -1,-1,-1,-1,-1,-1
PPPTP1: .WORD 123456,23456
.WORD 34567,45671
PPP10: EMT ;
PPPDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

:TEST 514 FDST MODE 2 TEST

TS514:

:FIRST TEST STF.
MOV #-1,R0 ;SET UP THE OUTPUT BUFFER.
MOV #QQQBF0,R1
MOV #14,R2
QQQ2: MOV R0,(R1)+
SOB R2,QQQ2

18679	105362	012700	000200		MOV	#200,R0		;SET FD MODE.
18680	105366	170100			LDFPS	R0		
18681	105370	012700	105506		MOV	#QQQTP1,R0		;SETUP ACO.
18682	105374	172410			LDD	(R0),ACO		
18683								
18684	105376	012700	105472		MOV	#QQQBF1,R0		;FDST ADDRESS.
18685	105402	005002			CLR	R2		
18686	105404	170102			LDFPS	R2		;SET FPS.
18687	105406	174020		QQQ3:	STF	ACO,(R0)+		;TEST INSTRUCTION.
18688								
18689	105410	022700	105476		CMP	#QQQBF1+4,R0		;WAS R0 INCREMENTED BY 4 PROPERLY?
18690								
18691	105414	001401			BEQ	QQQ4		
18692	105416	104000			EMT			;REPORT R0 INCORRECT AFTER FDST MODE 2
18693	105420	012700	105472	QQQ4:	MOV	#QQQBF1,R0		;WAS THE OUTPUT DATA CORRECT?
18694	105424	012701	105506		MOV	#QQQTP1,R1		
18695	105430	022021			CMP	(R0)+,(R1)+		
18696	105432	001031			BNE	QQQ10		;BRANCH IF INCORRECT.
18697	105434	022021			CMP	(R0)+,(R1)+		
18698	105436	001027			BNE	QQQ10		;BRANCH IF INCORRECT.
18699	105440	022027	177777		CMP	(R0)+,#-1		;SEE IF ANY OTHER DATA BUFFER WORDS WERE MODIFIED.
18700	105444	001024			BNE	QQQ10		;BRANCH IF INCORRECT.
18701	105446	022027	177777		CMP	(R0)+,#-1		
18702	105452	001021			BNE	QQQ10		;BRANCH IF INCORRECT.
18703	105454	000421			BR	QQQ20		
18704	105456	177777	177777	177777	QQQBF0:	.WORD	-1,-1,-1,-1,-1,-1	
18705	105464	177777	177777	177777				
18706	105472	177777	177777	177777	QQQBF1:	.WORD	-1,-1,-1,-1,-1,-1	
18707	105500	177777	177777	177777				
18708	105506	076543			QQQTP1:	76543		
18709	105510	065432				65432		
18710	105512	054321				54321		
18711	105514	043210				43210		
18712								;REPORT OUTPUT DATA INCORRECT:
18713	105516				QQQ10:	EMT		:
18714	105516	104000						
18715								
18716								;NOW TEST STD MODE 2.
18717								
18718	105520	012700	105456		QQQ20:	MOV	#QQQBF0,R0	;SET UP DEFAULT INPUT DATA BUFFER.
18719	105524	010001				MOV	R0,R1	
18720	105526	012702	000014			MOV	#14,R2	
18721	105532	010021			QQQ22:	MOV	R0,(R1)+	
18722	105534	077202				SQB	R2,QQQ22	
18723	105536	012700	000200			MOV	#200,R0	;ENTER FLOATING DOUBLE MODE.
18724	105542	170100				LDFPS	R0	
18725	105544	012700	105506			MOV	#QQQTP1,R0	;LOAD ACO.
18726	105550	172410				LDD	(R0),ACO	
18727	105552	012700	105472			MOV	#QQQBF1,R0	;SET DESTINATION ADDRESS.
18728	105556	012737	105564	037364		MOV	#QQQ23,@#STMP2	
18729	105564	174020			QQQ23:	STD	ACO,(R0)+	;TEST INSTRUCTION.
18730	105566	022700	105502			CMP	#QQQBF1+10,R0	;WAS R0 INCREMENTED BY 10 CORRECTLY?
18731	105572	001401				BEQ	QQQ24	
18732	105574	104000				EMT		;REPORT R0 INCORRECTLY INCREMENTED
18733	105576	012700	105472		QQQ24:	MOV	#QQQBF1,R0	;DID THE DATA REACH THE OUTPUT BUFFER CORRECTLY?
18734	105602	012701	105506			MOV	#QQQTP1,R1	

18735	105606	012702	000004
18736	105612	022021	
18737	105614	001002	
18738	105616	077203	
18739	105620	000401	
18740			
18741	105622		
18742	105622	104000	
18743	105624		
18744	105624	004767	016742
18745			
18746			
18747			
18748			
18749			
18750			
18751			
18752			
18753	105630		
18754			
18755	105630	012700	105700
18756	105634	012701	105746
18757	105640	012702	000004
18758	105644	012021	
18759	105646	077202	
18760	105650	012700	000200
18761	105654	170100	
18762	105656	012700	105756
18763	105662	172410	
18764	105664	012737	105744 000004
18765	105672	005001	
18766	105674	005004	
18767			
18768			
18769			
18770			
18771			
18772			
18773	105676	174027	
18774	105700	005201	
18775	105702	005201	
18776	105704	005201	
18777	105706	005201	
18778	105710	012700	105766
18779	105714	012702	105700
18780	105720	012703	000004
18781	105724	022022	
18782	105726	001006	
18783	105730	077303	
18784	105732	005704	
18785	105734	001003	
18786	105736	022701	000003
18787	105742	001415	
18788	105744		
18789	105744	104000	
18790			

```

MOV #4,R2
1$: CMP (R0)+,(R1)+
   BNE QQQ25 ;BRANCH IF INCORRECT.
   SOB R2,1$
   BR QQQDONE
;REPORT DATA INCORRECT.
QQQ25: EMT
QQQDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;*****
;TEST 515 FDST MODE 2, WITH GR7, TEST
;*****
TS515:
MOV #RRR3,R0 ;SET UP THE DATA BUFFER FOLLOWING THE TEST INSTRUCTION.
MOV #RRRTP1,R1
MOV #4,R2
1$: MOV (R0)+,(R1)+
   SOB R2,1$
   MOV #200,R0 ;ENTER FLOATING DOUBLE MODE.
   LDFPS R0
   MOV #RRRTP2,R0 ;SET UP ACO.
   LDD (R0),AC0
   MOV #RRR10,@WERRVECT ;SET UP FOR AN ODD ADDRESS.
   CLR R1
   CLR R4
;THIS IS THE TEST INSTRUCTION. IT SHOULD MODIFY THE FIRST LOCATION
;AFTER IT TO BE AN INCREMENT R4, INC R4, INSTRUCTION INSTEAD
;OF AN INCREMENT R1 INSTRUCTION. THE INCREMENT R4 SHOULD NOT BE
;EXECUTED SINCE THE PC SHOULD BE INCREMENTED BY TWO DURING IMMEDIATE
;MODE ADDRESSING. THUS AFTER THE EXECUTION OF THE NEXT 5 INSTRUCTIONS
;R1 SHOULD CONTAIN 3 AND R4 SHOULD CONTAIN 0.
RRR2: STD ACO,(R7)+ ;TEST INSTRUCTION.
RRR3: INC R1 ;THE STD INSTRUCTION SHOULD CHANGE THIS TO INC R4.
      INC R1
      INC R1
      INC R1
RRR4: MOV #RRREXP,R0 ;SEE IF THE DATA WAS OUTPUT CORRECTLY.
      MOV #RRR3,R2
      MOV #4,R3
RRR4: CMP (R0)+,(R2)+
      BNE RRR10 ;BRANCH IF INCORRECT.
      SOB R3,RRR4
      TST R4 ;MAKE SURE R4 IS 0.
      BNE RRR10 ;BRANCH IF R4 IS INCORRECT.
      CMP #3,R1 ;SEE IF R1 IS CORRECT.
      BEQ RRRDONE
RRR10: EMT
;THESE ARE TEST DATA PATTERNS USED TO SET UP THE OUTPUT BUFFER AT RRR3.

```

18791 105746 005201
 18792 105750 005201
 18793 105752 005201
 18794 105754 005201
 18795
 18796 105756 005204
 18797 105760 005204
 18798 105762 005204
 18799 105764 005204
 18800
 18801 105766 005204
 18802 105770 005201
 18803 105772 005201
 18804 105774 005201
 18805 105776
 18806 105776 004767 016570
 18807
 18808
 18809
 18810
 18811
 18812
 18813
 18814
 18815 106002
 18816
 18817 106002 012700 177777
 18818 106006 012701 106130
 18819 106012 012702 000010
 18820 106016 010021
 18821 106020 077202
 18822 106022 012700 000200
 18823 106026 170100
 18824 106030 012700 106150
 18825 106034 172410
 18826 106036 012737 106160 000004
 8827 106044 012700 106140
 18828
 18829 106050 174040
 18830 106052 005201
 18831 106054 020027 106130
 18832 106060 001037
 18833 106062 012700 106130
 18834 106066 012701 106150
 18835 106072 012702 000004
 18836 106076 022021
 18837 106100 001027
 18838 106102 077203
 18839 106104 012700 177777
 18840 106110 012701 106140
 18841 106114 012702 000004
 18842 106120 020021
 18843 106122 001016
 18844 106124 077203
 18845 106126 000415
 18846

```

RRRTP1: INC R1
        INC R1
        INC R1
        INC R1
;THIS IS THE DATA PUT IN ACO BEFORE EXECUTION OF THE STD.
RRRTP2: INC R4
        INC R4
        INC R4
        INC R4
;THIS IS THE EXPECTED DATA AT RRR3 AFTER EXECUTION OF THE STD.
RRREXP: INC R4
        INC R1
        INC R1
        INC R1
RRRDONE: JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;*****
;TEST 516 FDST MODE 4 TEST
;*****
TS516:
        MOV #-1,R0 ;SET UP THE OUTPUT BUFFER.
        MOV #SSSBF0,R1
        MOV #10,R2
1$: MOV R0,(R1)+
        SOB R2,1$
        MOV #200,R0 ;ENTER FLOATING DOUBLE MODE.
        LDFPS R0
        MOV #SSSTP1,R0 ;SET UP ACO.
        LDD (R0),ACO
        MOV #SSS10,#ERRVECT ;SET UP FOR A TRAP TO 4.
        MOV #SSSA1,R0 ;SET UP THE DESTINATION ADDRESS.

SSS2: STD ACO,-(R0) ;TEST INSTRUCTION.
      INC R1
      CMP R0,#SSSBF0 ;SEE IF R0 WAS DECREMENTED PROPERLY.
      BNE SSS10 ;BRANCH IF R0 IS INCORRECT.
      MOV #SSSBF0,R0 ;WAS THE OUTPUT DATA CORRECT?
      MOV #SSSTP1,R1
      MOV #4,R2
1$: CMP (R0)+,(R1)+
      BNE SSS10 ;BRANCH IF INCORRECT.
      SOB R2,1$
      MOV #-1,R0 ;IS THE REST OF THE OUTPUT BUFFER CORRECT, -1?
      MOV #SSSA1,R1
2$: MOV #4,R2
      CMP R0,(R1)+
      BNE SSS10 ;BRANCH IF INCORRECT.
      SOB R2,2$
      BR SSSDONE
  
```

18847
18848 106130 177777
18849 106132 177777
18850 106134 177777
18851 106136 177777
18852 106140 177777
18853 106142 177777
18854 106144 177777
18855 106146 177777
18856
18857
18858 106150 147250
18859 106152 036147
18860 106154 025036
18861 106156 147250
18862
18863 106160
18864 106160 104000
18865 106162
18866 106162 004767 016404
18867
18868
18869
18870
18871
18872
18873
18874
18875 106166
18876
18877 106166 012701 106276
18878 106172 012700 177777
18879 106176 012702 000013
18880 106202 010021
18881 106204 077202
18882 106206 012737 106276 106312
18883 106214 012700 000200
18884 106220 170100
18885 106222 012700 106314
18886 106226 172410
18887 106230 012737 106324 000004
18888 106236 012700 106312
18889
18890 106242 174030
18891
18892 106244 020027 106314
18893 106250 001025
18894 106252 012701 106276
18895 106256 012702 106314
18896 106262 012703 000004
18897 106266 022122
18898 106270 001015
18899 106272 077303
18900 106274 000414
18901
18902

:THIS IS THE OUTPUT DATA BUFFER.

SSSBFO: -1
-1
-1
-1
SSSA1: -1
-1
-1
-1

:THIS IS THE TEST DATA LOADED INTO ACO:

SSSTP1: 147250
36147
25036
147250

SSS10:

SSSDONE: EMT ;

JSR PC,.RSET

:GO INITIALIZE THE FPS AND STACK; AND
:SEE IF THE USER HAS EXPRESSED
:THE DESIRE TO CHANGE THE SOFTWARE
:VIRTUAL CONSOLE SWITCH REGISTER (HAS
:THE USER TYPED CONTROL G?).

:*****

:TEST 517 FDST MODE 3 TEST

:*****

TS517:

MOV #TTTBFO,R1 ;SET UP THE OUTPUT DATA BUFFER.

MOV #-1,R0

MOV #13,R2

1\$: MOV R0,(R1)+

SOB R2,1\$

MOV #TTTBFO,@#TTTA2

MOV #200,R0 ;ENTER DOUBLE FLOATING MODE.

LDFPS R0

MOV #TTTTP1,R0 ;SET UP ACO.

LDD (R0),ACO

MOV #TTT10,@#ERRVECT ;SET UP FOR TRAPS TO 4.

MOV #TTTA2,R0 ;SET UP THE DESTINATION ADDRESS.

TTT2: STD ACO,@(R0)+ ;TEST INSTRUCTION.

CMP R0,#TTTA2+2 ;SEE IF R0 WAS INCREMENTED CORRECTLY.

BNE TTT10 ;BRANCH IF INCORRECT.

MOV #TTTBFO,R1 ;CHECK THE OUTPUT DATA BUFFER.

MOV #TTTTP1,R2

MOV #4,R3

TTT3: CMP (R1)+,(R2)+

BNE TTT10 ;BRANCH IF NOT CORRECT.

SOB R3,TTT3

BR TTTDONE

:THIS IS THE OUTPUT DATA BUFFER:

18903	106276	177777	
18904	106300	177777	
18905	106302	177777	
18906	106304	177777	
18907	106306	177777	
18908	106310	177777	
18909	106312	106276	
18910	106314	101213	
18911	106316	141516	
18912	106320	071727	
18913	106322	037475	
18914			
18915	106324		
18916	106324	104000	
18917			
18918	106326		
18919	106326	004767	016240
18920			
18921			
18922			
18923			
18924			
18925			
18926			
18927			
18928	106332		
18929			
18930	106332	012701	106442
18931	106336	012700	177777
18932	106342	012702	000013
18933	106346	010021	
18934	106350	077202	
18935	106352	012737	106442 106454
18936	106360	012700	000200
18937	106364	170100	
18938	106366	012700	106460
18939	106372	172410	
18940	106374	012737	106470 000004
18941	106402	012700	106456
18942	106406	174050	
18943	106410	020027	106454
18944	106414	001025	
18945	106416	012701	106442
18946	106422	012702	106460
18947	106426	012703	000004
18948	106432	022122	
18949	106434	001015	
18950	106436	077303	
18951	106440	000414	
18952			
18953			
18954	106442	177777	
18955	106444	177777	
18956	106446	177777	
18957	106450	177777	
18958	106452	177777	

```

TTTBFO: -1
        -1
        -1
        -1
        -1
TTTA1:  -1
TTTA2:  TTTBFO
TTTTP1: 101213
        141516
        71727
        37475

TTT10:

        EMT ;

TTTDONE: JSR    PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND
        ;SEE IF THE USER HAS EXPRESSED
        ;THE DESIRE TO CHANGE THE SOFTWARE
        ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
        ;THE USER TYPED CONTROL G?).

```

```

:*****
:TEST 520      FDST MODE 5 TEST
:*****
TS520:

```

```

        MOV    #UUUBFO,R1      ;SET UP THE OUTPUT DATA BUFFER.
        MOV    #-1,R0
        MOV    #13,R2
1$:     MOV    R0,(R1)+
        SOB    R2,1$
        MOV    #UUUBFO,@#UUUA1
        MOV    #200,R0        ;ENTER DOUBLE FLOATING MODE.
        LDFPS R0
        MOV    #UUUTP1,R0     ;SET UP ACO.
        LDD    (R0),ACO
        MOV    #UUU10,@#ERRVECT ;GET READY FOR ANY TRAPS TO 4.
        MOV    #UUUA2,R0     ;SET UP THE DESTINATION ADDRESS.
UUU2:  STD    ACO,@-(R0)     ;TEST INSTRUCTION.
        CMP    R0,#UUUA2-2   ;WAS R0 DECRIMENTED PROPERLY?
        BNE   UUU10         ;BRANCH IF R0 IS INCORRECT.
        MOV    #UUUBFO,R1   ;WAS THE DATA OUTPUT CORRECTLY?
        MOV    #UUUTP1,R2
        MOV    #4,R3
UUU3:  CMP    (R1)+,(R2)+
        BNE   UUU10         ;BRANCH IF DATA IS INCORRECT.
        SOB   R3,UUU3
        BR    UUUDONE

```

```

;THIS IS THE OUTPUT DATA BUFFER
UUUBFO: -1
        -1
        -1
        -1
        -1

```

18959 106454 106442
18960 106456 177777
18961 106460 020212
18962 106462 023242
18963 106464 026273
18964 106466 031323
18965
18966 106470
18967 106470 104000
18968 106472
18969 106472 004767 016074
18970
18971
18972
18973
18974
18975
18976
18977
18978 106476
18979
18980 106476 012700 000200
18981 106502 170100
18982 106504 012701 106606
18983 106510 012700 177777
18984 106514 012702 000004
18985 106520 010021
18986 106522 077202
18987 106524 012737 106626 000004
18988 106532 012700 106616
18989 106536 172410
18990 106540 012700 100705
18991 106544 012701 000001
18992 106550 174060 005701
18993
18994 106554 020027 100705
18995 106560 001022
18996 106562 012702 106606
18997 106566 012703 106616
18998 106572 012704 000004
18999 106576 022223
19000 106600 001012
19001 106602 077403
19002 106604 000411
19003 106606 177777
19004 106610 177777
19005 106612 177777
19006 106614 177777
19007 106616 030313
19008 106620 023334
19009 106622 035363
19010 106624 074041
19011
19012 106626
19013 106626 104000
19014 106630

UUUA1: UUUBF0
UUUA2: -1
UUUTP1: 20212
23242
26273
031323

UUU10:

EMT

UUUDONE:

JSR PC,.RSET

:

:GO INITIALIZE THE FPS AND STACK; AND
:SEE IF THE USER HAS EXPRESSED
:THE DESIRE TO CHANGE THE SOFTWARE
:VIRTUAL CONSOLE SWITCH REGISTER (HAS
:THE USER TYPED CONTROL G?).

:*****
:TEST 521 FDST MODE 6, INDEX MODE, TEST
:*****
TS521:

MOV #200,R0 ;ENTER DOUBLE FLOATING MODE.
LDFPS R0
MOV #VVVBF0,R1 ;SET UP THE OUT PUT DATA BUFFER.
MOV #-1,R0
MOV #4,R2
1\$: MOV R0,(R1)+
SOB R2,1\$
MOV #VVV10,#ERRVECT ;SET UP VECTOR 4 INCASE OF ERROR.
MOV #VVVTP1,R0 ;SET UP ACO.
LDD (R0),ACO
MOV #VVVBF0-5701,R0 ;SET UP THE DESTINATION ADDRESS.
MOV #1,R1
VVV2: STD ACO,5701(R0) ;TEST INSTRUCTION.

CMP R0,#VVVBF0-5701 ;SEE IF R0 WAS MODIFIED.
BNE VVV10 ;BRANCH IF INCORRECT.
MOV #VVVBF0,R2 ;WAS THE OUTPUT DATA CORRECT.
MOV #VVVTP1,R3
MOV #4,R4
1\$: CMP (R2)+,(R3)+
BNE VVV10 ;BRANCH IF INCORRECT DATA.
SOB R4,1\$
BR VVVDONE
VVVBF0: -1
-1
-1
-1
VVVTP1: 30313
23334
35363
74041
VVV10:
EMT
VVVDONE:

19015 106630 004767 015736
19016
19017
19018
19019
19020
19021
19022
19023
19024 106634
19025
19026 106634 012700 000200
19027 106640 170100
19028 106642 012701 106752
19029 106646 012700 177777
19030 106652 012702 000004
19031 106656 010021
19032 106660 077202
19033 106662 012737 107002 000004
19034 106670 012700 106762
19035 106674 172410
19036 106676 012700 101071
19037 106702 012701 000001
19038 106706 012737 106752 106772
19039 106714 174070 005701
19040
19041 106720 020027 101071
19042 106724 001026
19043 106726 012702 106752
19044 106732 012703 106762
19045 106736 012704 000004
19046 106742 022223
19047 106744 001016
19048 106746 077403
19049 106750 000415
19050 106752 177777
19051 106754 177777
19052 106756 177777
19053 106760 177777
19054 106762 041424
19055 106764 034445
19056 106766 046475
19057 106770 051525
19058 106772 177777
19059 106774 177777
19060 106776 177777
19061 107000 177777
19062
19063 107002
19064 107002 104000
19065 107004
19066 107004 004767 015562
19067
19068
19069
19070

```
JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND  
;SEE IF THE USER HAS EXPRESSED  
;THE DESIRE TO CHANGE THE SOFTWARE  
;VIRTUAL CONSOLE SWITCH REGISTER (HAS  
;THE USER TYPED CONTROL G?).  
:*****  
:TEST 522 FDST MODE 7, INDEX DEFERRED MODE, TEST  
:*****  
T522:  
MOV #200,R0 ;ENTER DOUBLE FLOATING MODE.  
LDFPS R0  
MOV #WWWBFO,R1 ;SET UP THE OUTPUT DATA BUFFER.  
MOV #-1,R0  
MOV #4,R2  
1$: MOV R0,(R1)+  
SOB R2,1$  
MOV #WWW10,@#ERRVECT ;SET UP FOR TRAPS TO 4.  
MOV #WWWTP1,R0 ;SET UP ACO.  
LDD (R0),ACO  
MOV #WWWBF1-5701,R0 ;SET UP THE DESTINATION ADDRESS.  
MOV #1,R1  
106772 WWW2: MOV #WWWBFO,@#WWWBF1  
STD ACO,@5701(R0) ;TEST INSTRUCTION.  
CMP R0,#WWWBF1-5701 ;IS R0 CORRECT?  
BNE WWW10 ;BRANCH IF INCORRECT.  
MOV #WWWBFO,R2 ;WAS THE DATA OUTPUT CORRECTLY?  
MOV #WWWTP1,R3  
MOV #4,R4  
1$: CMP (R2)+,(R3)+  
BNE WWW10 ;BRANCH IF DATA IS INCORRECT.  
SOB R4,1$  
BR WWWDONE  
WWWBFO: -1  
-1  
-1  
-1  
WWWTP1: 41424  
34445  
46475  
051525  
WWWBF1: -1  
-1  
-1  
-1  
WWW10:  
WWWDONE: EMT ;  
JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND  
;SEE IF THE USER HAS EXPRESSED  
;THE DESIRE TO CHANGE THE SOFTWARE  
;VIRTUAL CONSOLE SWITCH REGISTER (HAS  
;THE USER TYPED CONTROL G?).
```

19071
19072
19073
19074
19075 107010
19076
19077
19078 107010 004767 000262
19079 107014 000000
19080 107016 000000
19081 107020 000000
19082 107022 000000
19083 107024 000000
19084 107026 000000
19085 107030 000000
19086 107032 000000
19087 107034 000000
19088 107036 000000
19089 107040 177777
19090 107042 177777
19091 107044 047000
19092 107046 047004
19093 107050 177777
19094 107052 147004
19095
19096
19097 107054 004767 000216
19098 107060 017203
19099 107062 142536
19100 107064 047506
19101 107066 172031
19102 107070 017203
19103 107072 142536
19104 107074 000000
19105 107076 000000
19106 107100 017203
19107 107102 142536
19108 107104 047506
19109 107106 172031
19110 107110 040000
19111 107112 040000
19112 107114 177777
19113 107116 177777
19114
19115
19116 107120 004767 000152
19117 107124 050717
19118 107126 027374
19119 107130 075767
19120 107132 077071
19121 107134 050717
19122 107136 027374
19123 107140 000000
19124 107142 000000
19125 107144 000000
19126 107146 000000

```
*****  
:TEST 523 STCFD TEST  
*****  
TS523:  
  
;AC=0  
XXX1: JSR PC,STCFDS  
1$: 0 ;AC  
0  
0  
0  
2$: 0 ;RES  
0  
0  
3$: 0 ;ERROR RES.  
0  
-1  
-1  
4$: 47000 ;FPS BEFORE EXECUTION.  
47004 ;FPS AFTER EXECUTION.  
-1 ;FEC  
147004 ;ERROR FPS.  
  
XXX2: JSR PC,STCFDS  
1$: 17203 ;AC  
142536  
47506  
2$: 172031 ;RES  
17203  
142536  
0  
0  
3$: 17203 ;ERROR RES.  
142536  
47506  
172031  
4$: 40000 ;FPS BEFORE EXECUTION.  
40000 ;FPS AFTER EXECUTION.  
-1 ;FEC  
-1 ;ERROR FPS.  
  
XXX3: JSR PC,STCFDS  
1$: 50717 ;AC  
27374  
75767  
77071  
2$: 50717 ;RES  
27374  
0  
0  
3$: 0 ;ERROR RES.  
0
```

```
19127 107150 000000 0
19128 107152 000000 0
19129 107154 047000 4$: 47000 ;FPS BEFORE EXECUTION.
19130 107156 047000 47000 ;FPS AFTER EXECUTION.
19131 107160 177777 -1 ;FEC
19132 107162 174002 174002 ;ERROR FPS.
19133
19134
19135 107164 004767 000106 XXX4: JSR PC,STCFDS
19136 107170 020212 1$: 20212 ;AC
19137 107172 032425 32425
19138 107174 026272 26272
19139 107176 002123 02123
19140 107200 020212 2$: 20212 ;RES
19141 107202 032425 32425
19142 107204 000000 0
19143 107206 000000 0
19144 107210 020212 3$: 20212 ;ERROR RES.
19145 107212 032425 32425
19146 107214 100000 100000
19147 107216 000000 0
19148 107220 040000 4$: 40000 ;FPS BEFORE EXECUTION.
19149 107222 040000 40000 ;FPS AFTER EXECUTION.
19150 107224 177777 -1 ;FEC
19151 107226 177777 -1 ;ERROR FPS.
```

```
19152
19153
19154 107230 004767 000042 XXX5: JSR PC,STCFDS
19155 107234 121314 1$: 121314 ;AC
19156 107236 151617 151617
19157 107240 101112 101112
19158 107242 131415 131415
19159 107244 121314 2$: 121314 ;RES
19160 107246 151617 151617
19161 107250 000000 0
19162 107252 000000 0
19163 107254 021314 3$: 21314 ;ERROR RES.
19164 107256 151617 151617
19165 107260 000000 0
19166 107262 000000 0
19167 107264 040000 4$: 40000 ;FPS BEFORE EXECUTION.
19168 107266 040010 40010 ;FPS AFTER EXECUTION.
19169 107270 177777 -1 ;FEC
19170 107272 177777 -1 ;ERROR FPS.
19171 107274 000460 6$: BR XXXDONE
```

```
19172
19173
19174
19175
19176 ;THIS SUBROUTINE, STCFDS, IS USED TO SET UP THE OPERANDS, EXECUTE
19177 ;THE STCFD INSTRUCTION AND CHECK THE RESULTS. A CALL
19178 ;TO IT IS MADE THUS:
19179
19180
19181
19182
```

```

:
:
: JSR PC,@#STCFDS
: ACARG: .WORD X,X,X,X ;AC OPERAND
: RES: .WORD X,X,X,X ;EXPECTED RESULT
```

19183
 19184
 19185
 19186
 19187
 19188
 19189
 19190
 19191
 19192
 19193
 19194
 19195
 19196
 19197
 19198
 19199
 19200
 19201
 19202
 19203
 19204
 19205
 19206
 19207
 19208 107276 012601
 19209 107300 012700 000200
 19210 107304 170100
 19211 107306 010100
 19212 107310 172410
 19213 107312 012700 177777
 19214 107316 012702 107426
 19215 107322 012703 000004
 19216 107326 010022
 19217 107330 077302
 19218 107332 016100 000030
 19219 107336 170100
 19220 107340 012700 107426
 19221 107344 176010
 19222
 19223 107346 170204
 19224 107350 170305
 19225 107352 010102
 19226 107354 062702 000010
 19227 107360 012703 107426
 19228 107364 012700 000004
 19229 107370 022223
 19230 107372 001014
 19231 107374 077003
 19232
 19233 107376 016102 000032
 19234 107402 020204
 19235 107404 001007
 19236 107406 005702
 19237 107410 100003
 19238 107412 026105 000036

```

ERRES: .WORD X,X,X,X ;ERROR RESULT
FPSB: .WORD X ;FPS BEFORE EXECUTION
FPSA: .WORD X ;FPS AFTER EXECUTION
FEC: .WORD X ;EXPECTED FEC
ERFPS: .WORD X ;ERROR FPS.
ERR1: ERROR X ;DATA ERROR.
BR CONT
ERR2: ERROR X ;FPS ERROR.
CONT: ;RETURN ADDRESS

:THE OPERANDS ARE SET UP (USING ACO AS THE ACCUMULATOR). THEN
:THE STCFD INSTRUCTION IS EXECUTED.
:THE RESULT IS CHECKED AGAINST RES. IF THE RESULT IS CORRECT THEN THE FPS IS
:COMPARED WITH FPSA IF THIS TOO IS CORRECT STCFDS RETURNS CONTROL
:TO THE CALLING ROUTINE AT CONT. IF THE FPS IS BAD STCFDS
:COMPARE IT TO ERROR FPS. IF THIS MATCHES THEN STCFDS WILL RETURN
:TO THE ERROR CALL AT ERR2, OTHERWISE STCFDS ITSELF
:REPORTS THIS FAILURE AND THEN RETURNS TO CONT. IF THE RESULT OF THE
:STCFD IS INCORRECT, THE INCORRECT RESULT IS COMPARED WITH THE
:ANTICIPATED FAILING DATA PATTERN, ERRES. IF THE FAILURE IN
:THE RESULT WAS ANTICIPATED CORRECTLY TO BE ERRES THEN STCFDS
:WILL TRANSFER CONTROL TO THE ERROR CALL AT ERR1. OTHERWISE THE
:RESULT WAS INCORRECT BUT WAS NOT ANTICIPATED AND STCFDS WILL
:REPORT THE FAILURE AFTER WHICH CONTROL WILL BE PASSED TO CONT.

STCFDS: MOV (SP)+,R1 ;PICK UP THE POINTER TO THE OPERANDS.
MOV #200,R0 ;ENTER DOUBLE FLOATING MODE.
LDFPS R0
MOV R1,R0 ;LOAD ACO.
LDD (R0),ACO
MOV #-1,R0 ;FILL THE OUTPUT BUFFER WITH -1'S.
MOV #STCFT,R2
MOV #4,R3
1$: MOV R0,(R2)+
SOB R3,1$
MOV 30(R1),R0 ;LOAD THE FPS.
LDFPS R0
MOV #STCFT,R0 ;SET UP THE DESTINATION ADDRESS.
2$: STCFD ACO,(R0) ;TEST INSTRUCTION.

STFPS R4 ;GET THE FPS.
STST R5 ;GET THE FEC.
MOV R1,R2 ;CHECK THE RESULT.
ADD #10,R2
MOV #STCFT,R3
MOV #4,R0
3$: CMP (R2)+,(R3)+
BNE 10$ ;BRANCH IF INCORRECT.
SOB R0,3$

MOV 32(R1),R2
CMP R2,R4 ;IS THE FPS CORRECT?
BNE 10$ ;BRANCH IF FPS INCORRECT.
TST R2 ;IF EXPECTED FPS IS NEGATIVE, THEN
BPL 4$ ;GO AHEAD AND CHECK THE FEC.
CMP 36(R1),R5

```

19239	107416	001002				BNE	10\$:BRANCH IF FEC IS INCORRECT.
19240	107420	000161	000040			4\$: JMP	40(R1)		:RETURN.
19241	107424					10\$:			
19242	107424	104000				EMT			:
19243	107426	177777	177777	177777	STCFT:	-1,-1,-1,-1			
19244	107434	177777							
19245	107436				XXXDONE:				
19246	107436	004767	015130			JSR	PC,.RSET		:GO INITIALIZE THE FPS AND STACK; AND
19247									:SEE IF THE USER HAS EXPRESSED
19248									:THE DESIRE TO CHANGE THE SOFTWARE
19249									:VIRTUAL CONSOLE SWITCH REGISTER (HAS
19250									:THE USER TYPED CONTROL G?).

19251
19252
19253
19254
19255 107442
19256
19257
19258 107442 004767 000262
19259 107446 000000
19260 107450 000000
19261 107452 000000
19262 107454 000000
19263 107456 000000
19264 107460 000000
19265 107462 177777
19266 107464 177777
19267 107466 000000
19268 107470 000000
19269 107472 000000
19270 107474 000000
19271 107476 047200
19272 107500 047204
19273 107502 177777
19274 107504 177777
19275
19276
19277 107506 004767 000216
19278 107512 067574
19279 107514 073727
19280 107516 170777
19281 107520 067574
19282 107522 067574
19283 107524 073730
19284 107526 177777
19285 107530 177777
19286 107532 067574
19287 107534 073727
19288 107536 177777
19289 107540 177777
19290 107542 040200
19291 107544 040200
19292 107546 177777
19293 107550 177777
19294

```
*****  
:TEST 524 STCDF TEST  
*****  
TS524:  
:AC=0  
YYY1: JSR PC,STCDF5  
1$: 0 :AC  
0  
0  
0  
2$: 0 :RES  
0  
-1  
-1  
3$: 0 :ERROR RES.  
0  
0  
0  
4$: 47200 :FPS BEFORE EXECUTION.  
47204 :FPS AFTER EXECUTION.  
-1 :FEC  
-1 :ERROR FPS.
```

```
YYY2: JSR PC,STCDF5  
1$: 67574 :ACO  
73727  
170777  
2$: 67574 :RES  
73730  
-1  
-1  
3$: 67574 :ERROR RES.  
73727  
-1  
-1  
4$: 40200 :FPS BEFORE EXECUTION.  
40200 :FPS AFTER EXECUTION.  
-1 :FEC  
-1 :ERROR FPS.
```

19295							
19296	107552	004767	000152	YYY3:	JSR	PC,STCDF5	
19297	107556	077777		1\$:	77777		:ACO
19298	107560	177777			-1		
19299	107562	100000			100000		
19300	107564	000000			0		
19301	107566	000000		2\$:	0		:RES
19302	107570	000000			0		
19303	107572	177777			-1		
19304	107574	177777			-1		
19305	107576	077777		3\$:	77777		:ERROR RES.
19306	107600	177777			-1		
19307	107602	177777			-1		
19308	107604	177777			-1		
19309	107606	040200		4\$:	40200		:FPS BEFORE EXECUTION.
19310	107610	040206			40206		:FPS AFTER EXECUTION.
19311	107612	177777			-1		:FEC
19312	107614	040204			40204		:ERROR FPS.
19313							
19314							
19315	107616	004767	000106	YYY4:	JSR	PC,STCDF5	
19316	107622	077777		1\$:	77777		:ACO
19317	107624	177777			-1		
19318	107626	100000			100000		
19319	107630	000000			0		
19320	107632	000000		2\$:	0		:RES
19321	107634	000000			0		
19322	107636	177777			-1		
19323	107640	177777			-1		
19324	107642	077777		3\$:	77777		:ERROR RES.
19325	107644	177777			-1		
19326	107646	177777			-1		
19327	107650	177777			-1		
19328	107652	040200		4\$:	40200		:FPS BEFORE EXECUTION.
19329	107654	040206			40206		:FPS AFTER EXECUTION.
19330	107656	177777			-1		:FEC
19331	107660	140206			140206		:ERROR FPS.
19332							
19333							
19334	107662	004767	000042	YYY5:	JSR	PC,STCDF5	
19335	107666	177777		1\$:	177777		:ACO
19336	107670	177777			-1		
19337	107672	100000			100000		
19338	107674	000000			0		
19339	107676	100000		2\$:	100000		:RES
19340	107700	000000			0		
19341	107702	177777			-1		
19342	107704	177777			-1		
19343	107706	000000		3\$:	0		:ERROR RES.
19344	107710	000000			0		
19345	107712	177777			-1		
19346	107714	177777			-1		
19347	107716	047200		4\$:	47200		:FPS BEFORE EXECUTION.
19348	107720	147216			147216		:FPS AFTER EXECUTION.
19349	107722	000010			10		:FEC
19350	107724	047206			47206		:ERROR FPS.

19351 107726 000460
19352
19353
19354
19355
19356
19357
19358
19359
19360
19361
19362
19363
19364
19365
19366
19367
19368
19369
19370
19371
19372
19373
19374
19375
19376
19377
19378
19379
19380
19381
19382
19383
19384 107730 012601
19385 107732 012700 000200
19386 107736 170100
19387 107740 010100
19388 107742 172410
19389 107744 012700 177777
19390 107750 012702 110060
19391 107754 012703 000004
19392 107760 010022
19393 107762 077702
19394 10 64 016100 000030
19395 107770 170100
19396 107772 012700 110060
19397 107776 176010
19398
19399 110000 170204
19400 110002 170305
19401 110004 010102
19402 110006 062702 000010
19403 110012 012703 110060
19404 110016 012700 000004
19405 110022 022223
19406 110024 001014

6\$: BR YYDONE
:THIS SUBROUTINE, STCFDS, IS USED TO SET UP THE OPERANDS, EXECUTE
:THE STCDF INSTRUCTION AND CHECK THE RESULTS. A CALL
:TO IT IS MADE THUS:
:
: JSR PC,@#STCFDS
: ACARG: .WORD X,X,X,X ;AC OPERAND
: RES: .WORD X,X,X,X ;EXPECTED RESULT
: ERRES: .WORD X,X,X,X ;ERROR RESULT
: FPSB: .WORD X ;FPS BEFORE EXECUTION
: FPSA: .WORD X ;FPS AFTER EXECUTION
: FEC: .WORD X ;EXPECTED FEC
: ERFPS: .WORD X ;ERROR FPS.
: ERR1: ERROR X ;DATA ERROR.
: BR CONT
: ERR2: ERROR X ;FPS ERROR.
: CONT: ;RETURN ADDRESS
:
:THE OPERANDS ARE SET UP (USING ACO AS THE ACCUMULATOR). THEN
:THE STCFD INSTRUCTION IS EXECUTED.
:THE RESULT IS CHECKED AGAINST RES. IF THE RESULT IS CORRECT THEN THE FPS IS
:COMPARED WITH FPSA IF THIS TOO IS CORRECT STCFDS RETURNS CONTROL
:TO THE CALLING ROUTINE AT CONT. IF THE FPS IS BAD STCFDS
:COMPARE IT TO ERROR FPS. IF THIS MATCHES THEN STCFDS WILL RETURN
:TO THE ERROR CALL AT ERR2, OTHERWISE STCFDS ITSELF
:REPORTS THIS FAILURE AND THEN RETURNS TO CONT. IF THE RESULT OF THE
:STCFD IS INCORRECT, THE INCORRECT RESULT IS COMPARED WITH THE
:ANTICIPATED FAILING DATA PATTERN, ERRES. IF THE FAILURE IN
:THE RESULT WAS ANTICIPATED CORRECTLY TO BE ERRES THEN STCFDS
:WILL TRANSFER CONTROL TO THE ERROR CALL AT ERR1. OTHERWISE THE
:RESULT WAS INCORRECT BUT WAS NOT ANTICIPATED AND STCFDS WILL
:REPORT THE FAILURE AFTER WHICH CONTROL WILL BE PASSED TO CONT.
:
STCFDS: MOV (SP)+,R1 ;PICK UP THE POINTER TO THE OPERANDS.
MOV #200,R0 ;ENTER DOUBLE FLOATING MODE.
LDFPS R0
MOV R1,R0 ;LOAD ACO.
LDD (R0),ACO
MOV #-1,R0 ;FILL THE OUTPUT BUFFER WITH -1'S.
MOV #STCDT,R2
MOV #4,R3
1\$: MOV R0,(R2)+
SOB R3,1\$
MOV 30(R1),R0 ;LOAD THE FPS.
LDFPS R0
MOV #STCDT,R0 ;SET UP THE DESTINATION ADDRESS.
2\$: STCDF ACO,(R0) ;TEST INSTRUCTION.
:
STFPS R4 ;GET THE FPS.
STST R5 ;GET THE FEC.
MOV R1,R2 ;CHECK THE RESULT.
ADD #10,R2
MOV #STCDT,R3
MOV #4,R0
3\$: CMP (R2)+,(R3)+
BNE 10\$;BRANCH IF INCORRECT.

```
19407 110026 077003          SOB      R0,3$
19408
19409 110030 016102 000032    MOV      32(R1),R2
19410 110034 020204          CMP      R2,R4          ;IS THE FPS CORRECT?
19411 110036 001007          BNE     10$           ;BRANCH IF FPS INCORRECT.
19412 110040 005702          TST     R2           ;IF EXPECTED FPS IS NEGATIVE, THEN
19413 110042 100003          BPL     4$           ;GO AHEAD AND CHECK THE FEC.
19414 110044 026105 000034    CMP      34(R1),R5
19415 110050 001002          BNE     10$           ;BRANCH IF FEC IS INCORRECT.
19416 110052 000161 000040    4$:     JMP      40(R1)  ;RETURN.
19417 110056          10$:
19418 110056 104000          EMT
19419 110060 177777 177777  STCDT:  -1,-1,-1,-1
19420 110066 177777
19421 110070
19422 110070 004767 014476    YYYDONE: JSR     PC,.RSET   ;GO INITIALIZE THE FPS AND STACK; AND
19423                                     ;SEE IF THE USER HAS EXPRESSED
19424                                     ;THE DESIRE TO CHANGE THE SOFTWARE
19425                                     ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
19426                                     ;THE USER TYPED CONTROL G?).
19427
19428 ;*****
19429 ;TEST 525      STCFD WITH ILLEGAL ACCUMULATOR TEST
19430 ;*****
19431 TS525:
19432 110074          MOV      #40000,R0    ;DISSABLE INTERRUPTS.
19433 110100          LDFPS   R0
19434 110102 176006    ZZZ2:   STCFD   AC0,AC6 ;THIS TEST INSTRUCTION SHOULD CAUSE AN ERROR.
19435
19436 110104 170204          STFPS   R4           ;GET FPS.
19437 110106 170305          STST    R5           ;GET FEC.
19438 110110 020427 140000    CMP      R4,#140000  ;IS FPS CORRECT?
19439 110114 001004          BNE     ZZZ10        ;BRANCH IF INCORRECT FPS.
19440 110116 022705 000002    CMP      #2,R5       ;IS FEC CORRECT?
19441 110122 001001          BNE     ZZZ10        ;BRANCH IF INCORRECT.
19442 110124 000401          BR      ZZZDONE
19443
19444 110126          ZZZ10:
19445 110126 104000          EMT
19446
19447 110130          ZZZDONE:
19448 110130 004767 014436    JSR     PC,.RSET   ;GO INITIALIZE THE FPS AND STACK; AND
19449                                     ;SEE IF THE USER HAS EXPRESSED
19450                                     ;THE DESIRE TO CHANGE THE SOFTWARE
19451                                     ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
19452                                     ;THE USER TYPED CONTROL G?).
19453
19454 ;*****
19455 ;TEST 526      CLRD TEST
19456 ;*****
19457 TS526:
19458 110134          MOV      #AABTP1,R0  ;SET UP OUTPUT BUFFER
19459 110140          MOV      #AABBFO,R1
19460 110144 012702 000004    MOV      #4,R2
19461 110150 012021          MOV      (R0)+,(R1)+
19462 110152 077202          SOB      R2,1$
```

19463 110154 012700 110230
19464 110160 012701 000213
19465 110164 170101
19466 110166 170410
19467
19468 110170 170205
19469 110172 012702 000004
19470 110176 012701 110230
19471 110202 005721
19472 110204 001010
19473 110206 077203
19474 110210 022705 000204
19475 110214 001004
19476 110216 020027 110230
19477 110222 001001
19478 110224 000411
19479
19480
19481 110226
19482 110226 104000
19483
19484
19485 110230 073475
19486 110232 067707
19487 110234 127347
19488 110236 056770
19489
19490 110240 073475
19491 110242 067707
19492 110244 127347
19493 110246 056770
19494 110250
19495 110250 004767 014316
19496
19497
19498
19499
19500
19501
19502
19503
19504 110254
19505 110254 012700 040200
19506 110260 170100
19507 110262 170407
19508
19509 110264 170204
19510 110266 170305
19511 110270 020427 140200
19512 110274 001004
19513 110276 022705 000002
19514 110302 001001
19515 110304 000401
19516
19517 110306
19518 110306 104000

MOV #AABBFO,R0 ;SET UP DESTINATION OPERAND ADDRESS.
MOV #213,R1 ;SET UP FPS.
LDFPS R1
2\$: CLRD (R0) ;TEST INSTRUCTION.
STFPS R5 ;GET FPS.
MOV #4,R2 ;SEE IF RESULT CLEAR, 0.
MOV #AABBFO,R1
3\$: TST (R1)+
BNE AAB2 ;BRANCH IF RESULT INCORRECT, NOT 0.
SOB R2,3\$
CMP #204,R5 ;SEE IF FPS IS CORRECT.
BNE AAB2 ;BRANCH IF INCORRECT.
CMP R0,#AABBFO ;SEE IF R0 IS CORRECT.
BNE AAB2 ;BRANCH IF R0 IS INCORRECT.
BR AABDONE
AAB2: EMT ;
;THIS IS THE TEST DATA BUFFER, OUTPUT DATA BUFFER.
AABBFO: 73475
67707
127347
56770
;THIS IS THE DATA USED TO SET UP THE OUTPUT BUFFER.
AABTP1: 73475
67707
127347
56770
AABDONE: JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).
;*****
;TEST 527 CLRD WITH ILLEGAL ACCUMULATOR TEST
;*****
T527: MOV #40200,R0 ;SET UP THE FPS, NO INTERRUPTS AND FD=1.
LDFPS R0
CCB2: CLRD AC7 ;TEST INSTRUCTION.
STFPS R4 ;GET FPS.
STST R5 ;GET FEC.
CMP R4,#140200 ;IS THE FPS CORRECT?
BNE CCB10 ;BRANCH IF FPS IS INCORRECT.
CMP #2,R5 ;IS THE FEC CORRECT?
BNE CCB10 ;BRANCH IF FEC IS INCORRECT.
BR CCBDONE
CCB10: EMT ;

19519 110310
19520 110310 004767 014256
19521
19522
19523
19524
19525
19526
19527
19528
19529 110314
19530
19531 110314 012700 040200
19532 110320 170100
19533 110322 170707
19534
19535 110324 170204
19536 110326 170305
19537
19538 110330 022704 140200
19539 110334 001004
19540 110336 022705 000002
19541 110342 001001
19542 110344 000401
19543 110346
19544 110346 104000
19545
19546 110350
19547 110350 004767 014216
19548
19549
19550
19551
19552
19553
19554
19555
19556 110354
19557
19558 110354 012700 000200
19559 110360 170100
19560 110362 012700 110454
19561 110366 172410
19562 110370 005000
19563 110372 170100
19564 110374 012700 110464
19565 110400 172410
19566
19567 110402 012700 000201
19568 110406 170100
19569 110410 170700
19570
19571 110412 170205
19572 110414 012700 000200
19573 110420 170100
19574 110422 012700 110474

```
CCBDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK AND  
;SEE IF THE USER HAS EXPRESSED  
;THE DESIRE TO CHANGE THE SOFTWARE  
;VIRTUAL CONSOLE SWITCH REGISTER (HAS  
;THE USER TYPED CONTROL G?).  
  
:*****  
:TEST 530 NEGf, ABSF AND TSTF SOURCE MODE 0 WITH ILLEGAL AC7, TEST  
:*****  
TS530:  
      MOV #40200,R0 ;SET UP THE FPS, FID=1 AND FD=1.  
      LDFPS R0  
VVB2: NEGd AC7 ;TEST INSTRUCTION.  
  
      STFPS R4 ;GET FPS.  
      STST R5 ;GET FEC.  
  
      CMP #140200,R4 ;IS FPS CORRECT?  
      BNE VVB10 ;BRANCH IF FPS IS INCORRECT.  
      CMP #2,R5 ;IS FEC CORRECT?  
      BNE VVB10 ;BRANCH IF FEC IS INCORRECT.  
      BR VVBDONE  
VVB10: EMT ;  
VVBDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND  
;SEE IF THE USER HAS EXPRESSED  
;THE DESIRE TO CHANGE THE SOFTWARE  
;VIRTUAL CONSOLE SWITCH REGISTER (HAS  
;THE USER TYPED CONTROL G?).  
  
:*****  
:TEST 531 NEGf, ABSF AND TSTF SOURCE MODE 0 TEST  
:*****  
TS531:  
      MOV #200,R0 ;SET FD MODE.  
      LDFPS R0  
      MOV #DDBTP1,R0 ;SET UP ACO.  
      LDD (R0),AC0 ;SET ACO = 0  
      CLR R0 ;CLEAR THE FPS.  
      LDFPS R0  
      MOV #DDBTP2,R0 ;LOAD ACO TO BE A FLOATING 0.  
      LDF (R0),AC0 ;SET ACO=ZERO  
;FLOAT  
;SET FD MODE.  
DDB2: NEGd ACO ;TEST INSTRUCTION  
  
      STFPS R5 ;GET FPS.  
      MOV #200,R0 ;SET FD MODE.  
      LDFPS R0  
      MOV #DDBBF0,R0 ;GET THE RESULT OUT OF ACO.
```

19575 110426 174010
19576
19577 110430 012701 000004
19578 110434 005720
19579 110436 001005
19580 110440 077103
19581 110442 022705 000204
19582 110446 001001
19583 110450 000415
19584 110452
19585 110452 104000
19586
19587
19588 110454 101112
19589 110456 131415
19590 110460 161710
19591 110462 111213
19592 110464 000000
19593 110466 000000
19594 110470 000000
19595 110472 000000
19596
19597 110474 177777
19598 110476 177777
19599 110500 177777
19600 110502 177777
19601
19602 110504
19603 110504 004767 014062
19604
19605
19606
19607
19608
19609
19610
19611
19612 110510
19613
19614 110510 012700 110610
19615 110514 012701 110630
19616 110520 012702 000004
19617 110524 012021
19618 110526 077202
19619 110530 012700 000200
19620 110534 170100
19621 110536 012700 110630
19622 110542 012737 110640 000004
19623 110550 170710
19624
19625 110552 170205
19626 110554 012701 110630
19627 110560 012702 000004
19628 110564 005721
19629 110566 001024
19630 110570 077203

```
STD ACO,(R0) ;SEE IF THE RESULT IS CORRECT.
1$: MOV #4,R1
TST (R0)+
BNE DDB5 ;BRANCH IF THE RESULT IS INCORRECT.
SOB R1,1$
CMP #204,R5 ;IS THE FPS CORRECT?
BNE DDB5 ;BRANCH IF THE FPS IS INCORRECT.
BR DDBDONE
DDB5: EMT ;
;THESE ARE TEST DATA TABLES AND AN OUTPUT BUFFER.
DDBTP1: 101112
131415
161710
111213
DDBTP2: 0
0
0
0
DDBBF0: -1
-1
-1
-1
DDBDONE: JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).
```

;TEST 532 NEGF, ABSF AND TSTF SOURCE MODE 1 TEST

TS532:

```
MOV #EEBTP1,R0 ;SET UP THE DATA BUFFER.
MOV #EEBBF1,R1
1$: MOV #4,R2
MOV (R0)+,(R1)+
SOB R2,1$
MOV #200,R0 ;SET FD MODE.
LDFPS R0
MOV #EEBBF1,R0 ;SET UP THE OPERAND ADDRESS.
MOV #EEB10,@#ERRVECT ;SET UP VECTOR 4 IN CASE OF ERROR.
EEB2: NEG D (R0) ;TEST INSTRUCTION.
STFPS R5 ;GET FPS.
MOV #EEBBF1,R1 ;SEE IF RESULT IS CORRECT.
1$: MOV #4,R2
TST (R1)+
BNE EEB10 ;BRANCH IF NOT CORRECT.
SOB R2,1$
```

19631
19632 110572 020027 110630
19633 110576 001020
19634 110600 022705 000204
19635 110604 001015
19636 110606 000415
19637
19638
19639 110610 000177
19640 110612 167574
19641 110614 137271
19642 110616 107675
19643 110620 177777
19644 110622 177777
19645 110624 177777
19646 110626 177777
19647 110630 177777
19648 110632 177777
19649 110634 177777
19650 110636 177777
19651 110640
19652 110640 104000
19653 110642
19654 110642 004767 013724
19655
19656
19657
19658
19659
19660
19661
19662
19663 110646
19664
19665 110646 012700 110746
19666 110652 012701 110756
19667 110656 012702 000004
19668 110662 012021
19669 110664 077202
19670 110666 012700 000200
19671 110672 170100
19672 110674 012700 110756
19673 110700 012737 110766 000004
19674
19675 110706 170620
19676
19677 110710 170205
19678 110712 012701 110756
19679 110716 012702 000004
19680 110722 005721
19681 110724 001020
19682 110726 077203
19683
19684 110730 020027 110766
19685 110734 001014
19686 110736 022705 000204

```
CMP R0,#EEBBF1 ;IS R0 CORRECT?  
BNE EEB10 ;BRANCH IF NOT CORRECT.  
CMP #204,R5 ;IS THE FPS CORRECT?  
BNE EEB10 ;BRANCH IF NOT CORRECT.  
BR EEBDONE
```

;THESE ARE TEST DATA TABLES AND A BUFFER.

EEBTP1: 177

167574

137271

107675

EEBBF0: -1

-1

-1

-1

EEBBF1: -1

-1

-1

-1

EEB10:

EEBDONE: EMT ;

JSR PC,,RSET

;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;TEST 533 NEG, ABSF AND TSTF SOURCE MODE 2 TEST

TS533:

MOV #FFBTP1,R0 ;SET UP THE DATA BUFFER.

MOV #FFBBF1,R1

MOV #4,R2

1\$: MOV (R0)+,(R1)+

SOB R2,1\$

MOV #200,R0 ;SET FD.

LDFPS R0

MOV #FFBBF1,R0 ;SET UP THE OPERAND ADDRESS.

MOV #FFB10,#ERRVECT ;SET UP VECTOR 4 IN CASE OF AN ERROR.

FFB2: ABSD (R0)+ ;TEST INSTRUCTION.

STFPS R5 ;GET FPS.

MOV #FFBBF1,R1 ;CHECK RESULT.

MOV #4,R2

1\$: TST (R1)+

BNE FFB10 ;BRANCH IF INCORRECT.

SOB R2,1\$

CMP R0,#FFBBF1+10 ;IS R0 CORRECT?

BNE FFB10 ;BRANCH IF INCORRECT.

CMP #204,R5 ;IS THE FPS CORRECT?

19687 110742 001011
19688 110744 000411
19689
19690
19691 110746 000177
19692 110750 167574
19693 110752 137271
19694 110754 107675
19695 110756 177777
19696 110760 177777
19697 110762 177777
19698 110764 177777
19699 110766
19700 110766 104000
19701 110770
19702 110770 004767 013576
19703
19704
19705
19706
19707
19708
19709
19710 110774

BNE FFB10 ;BRANCH IF INCORRECT.
BR FFBDONE

;THESE ARE TEST DATA TABLES AND DATA BUFFER.

FFBTP1: 177
167574
137271
107675

FFBBF1: -1
-1
-1
-1

FFB10: EMT ;

FFBDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;TEST 534 NEGF, ABSF AND TSTF SOURCE MODE 4 TEST

TS534:

```
19711
19712 110774 012700 111074      MOV      #GGBTP1,R0      ;SET UP THE DATA BUFFER.
19713 111000 012701 111104      MOV      #GGBBF0,R1
19714 111004 012702 000004      MOV      #4,R2
19715 111010 012021              1$: MOV      (R0)+,(R1)+
19716 111012 077202              SOB      R2,1$
19717 111014 012700 000200      MOV      #200,R0        ;SET FD.
19718 111020 170100              LDFPS   R0
19719 111022 012700 111114      MOV      #GGBBF1,R0     ;SET UP THE OPERAND ADDRESS.
19720 111026 012737 111124 000004  MOV      #GGB10,#ERRVECT ;SET UP VECTOR 4 IN CASE OF AN ERROR.
19721
19722 111034 170640              GGB2:  ABSD  -(R0)      ;TEST INSTRUCTION.
19723
19724 111036 170205              STFPS   R5              ;GET FPS.
19725 111040 012701 111104      MOV      #GGBBF0,R1     ;CHECK RESULT.
19726 111044 012702 000004      MOV      #4,R2
19727 111050 005721              1$:  TST      (R1)+
19728 111052 001024              BNE     GGB10           ;BRANCH IF INCORRECT.
19729 111054 077203              SOB      R2,1$
19730
19731 111056 020027 111104      CMP      R0,#GGBBF0     ;IS R0 CORRECT?
19732 111062 001020              BNE     GGB10           ;BRANCH IF INCORRECT.
19733 111064 022705 000204      CMP      #204,R5        ;IS THE FPS CORRECT?
19734 111070 001015              BNE     GGB10           ;BRANCH IF INCORRECT.
19735 111072 000415              BR      GGBDONE
19736
19737 :THESE ARE TEST DATA TABLES AND DATA BUFFER.
19738 111074 000177      GGBTP1: 177
19739 111076 117273              117273
19740 111100 147576              147576
19741 111102 177071              177071
19742 111104 177777      GGBBF0: -1
19743 111106 177777              -1
19744 111110 177777              -1
19745 111112 177777              -1
19746 111114 177777      GGBBF1: -1
19747 111116 177777              -1
19748 111120 177777              -1
19749 111122 177777              -1
19750 111124              GGB10:
19751 111124 104000      EMT      ;
19752 111126              GGBDONE:
19753 111126 004767 013440      JSR      PC,.RSET      ;GO INITIALIZE THE FPS AND STACK; AND
19754 ;SEE IF THE USER HAS EXPRESSED
19755 ;THE DESIRE TO CHANGE THE SOFTWARE
19756 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
19757 ;THE USER TYPED CONTROL G?).
19758 :*****
19759 :TEST 535      NEG, ABSF AND TSTF SOURCE MODE 3 TEST
19760 :*****
19761 111132      TS535:
19762
19763 111132 012700 111232      MOV      #HHBTP1,R0     ;SET UP THE DATA BUFFER.
19764 111136 012701 111252      MOV      #HHBBF0,R1
19765 111142 012702 000010      MOV      #10,R2
19766 111146 012021              1$:  MOV      (R0)+,(R1)+
```



```
19767 111150 077202          SOB      R2,1$
19768 111152 012700 000200  MOV      #200,R0          ;SET FD.
19769 111156 170100          LDFPS   R0
19770 111160 012700 111262  MOV      #HHBBF1,R0      ;SET UP THE OPERAND ADDRESS.
19771 111164 012737 111272 000004  MOV      #HMB10,@#ERRVECT ;SET UP VECTOR 4 IN CASE OF AN ERROR.
19772
19773 111172 170630          HMB2:   ABSD   @(R0)+      ;TEST INSTRUCTION.
19774
19775 111174 170205          STFPS   R5              ;GET FPS.
19776 111176 012701 111252  MOV      #HHBBF0,R1      ;CHECK RESULT.
19777 111202 012702 000004  MOV      #4,R2
19778 111206 005721          1$:    TST      (R1)+
19779 111210 001030          BNE     HMB10          ;BRANCH IF INCORRECT.
19780 111212 077203          SOB     R2,1$
19781 111214 020027 111264  CMP     R0,#HHBBF1+2    ;IS R0 CORRECT?
19782 111220 001024          BNE     HMB10          ;BRANCH IF INCORRECT.
19783 111222 022705 000204  CMP     #204,R5        ;IS THE FPS CORRECT?
19784 111226 001021          BNE     HMB10          ;BRANCH IF INCORRECT.
19785 111230 000421          BR      HMBDONE
```

;THESE ARE TEST DATA TABLES AND DATA BUFFER.

```
19787
19788 111232 000177          HMBTBP1: 177
19789 111234 147576          147576
19790 111236 177071          177071
19791 111240 107576 111252 177777 107576,HMBBF0,-1,-1,-1
19792 111246 177777 177777
```

HMBBF0: -1

-1

-1

HMBBF1: -1

-1

-1

HMB10: -1

HMBDONE: EMT ;

```
19802 111272 104000          JSR     PC,.RSET      ;GO INITIALIZE THE FPS AND STACK; AND
19803 111274          ;SEE IF THE USER HAS EXPRESSED
19804 111274 004767 013272          ;THE DESIRE TO CHANGE THE SOFTWARE
19805          ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
19806          ;THE USER TYPED CONTROL G?).
```

;TEST 536 NEGF, ABSF AND TSTF SOURCE MODE 5 TEST

```
19812 111300          T5536:
19813
19814 111300 012700 111400          MOV      #IIBTP1,R0      ;SET UP THE DATA BUFFER.
19815 111304 012701 111420          MOV      #IIBBF0,R1
19816 111310 012702 000010          MOV      #10,R2
19817 111314 012021          1$:    MOV      (R0)+,(R1)+
19818 111316 077202          SOB     R2,1$
19819 111320 012700 000200  MOV      #200,R0          ;SET FD.
19820 111324 170100          LDFPS   R0
19821 111326 012700 111432  MOV      #IIBBF1+2,R0    ;SET UP THE OPERAND ADDRESS.
19822 111332 012737 111440 000004  MOV      #IIB10,@#ERRVECT ;SET UP VECTOR 4 IN CASE OF AN ERROR.
```

19823
19824 111340 170750
19825
19826 111342 170205
19827 111344 012701 111420
19828 111350 012702 000004
19829 111354 005721
19830 111356 001030
19831 111360 077203
19832 111362 020027 111430
19833 111366 001024
19834 111370 022705 000204
19835 111374 001021
19836 111376 000421
19837
19838
19839 111400 000176
19840 111402 177074
19841 111404 127374
19842 111406 157677 111420 177777
19843 111414 177777 177777
19844 111420 177777
19845 111422 177777
19846 111424 177777
19847 111426 177777
19848 111430 177777
19849 111432 177777
19850 111434 177777
19851 111436 177777
19852
19853 111440
19854 111440 104000
19855 111442
19856 111442 004767 013124
19857
19858
19859
19860
19861
19862
19863
19864 111446
19865
19866 111446 012700 111550
19867 111452 012701 111562
19868 111456 012702 000004
19869 111462 012021
19870 111464 077202
19871 111466 012700 000200
19872 111472 170100
19873 111474 012700 111553
19874 111500 012737 111572 000004
19875
19876 111506 170660 000007
19877
19878 111512 170205

```
IIB2:  NEG  @-(R0)          ;TEST INSTRUCTION.  
      STFPS R5              ;GET FPS.  
      MOV  #IIBBF0,R1      ;CHECK RESULT.  
1$:   MOV  #4,R2  
      TST  (R1)+  
      BNE  IIB10           ;BRANCH IF INCORRECT.  
      SOB  R2,1$  
      CMP  R0,#IIBBF1     ;IS R0 CORRECT?  
      BNE  IIB10           ;BRANCH IF INCORRECT.  
      CMP  #204,R5        ;IS THE FPS CORRECT?  
      BNE  IIB10           ;BRANCH IF INCORRECT.  
      BR   IIBDONE
```

;THESE ARE TEST DATA TABLES AND DATA BUFFER.

```
IIBTP1: 176  
        177074  
        127374  
        157677,IIBBF0,-1,-1,-1
```

```
IIBBF0: -1  
        -1  
        -1  
        -1  
IIBBF1: -1  
        -1  
        -1  
        -1
```

```
IIB10:  
IIBDONE: EMT  
        JSR  PC,.RSET
```

;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;TEST 537 NEG, ABSF AND TSTF SOURCE MODE 6 TEST

```
T5537:  
      MOV  #JJBTP1,R0      ;SET UP THE DATA BUFFER.  
      MOV  #JJBBF0,R1  
      MOV  #4,R2  
1$:   MOV  (R0)+,(R1)+  
      SOB  R2,1$  
      MOV  #200,R0        ;SET FD.  
      LDFPS R0  
      MOV  #JJBBF0-7,R0   ;SET UP THE OPERAND ADDRESS.  
      MOV  #JJB10,@#ERRVECT ;SET UP VECTOR 4 IN CASE OF AN ERROR.  
JJB2:  ABSD  7(R0)        ;TEST INSTRUCTION.  
      STFPS R5              ;GET FPS.
```

19879 111514 012701 111562
19880 111520 012702 000004
19881 111524 005721
19882 111526 001021
19883 111530 077203
19884 111532 020027 111553
19885 111536 001015
19886 111540 022705 000204
19887 111544 001012
19888 111546 000412
19889
19890
19891 111550 000177
19892 111552 161524
19893 111554 131273
19894 111556 107174 000000
19895 111562 177777
19896 111564 177777
19897 111566 177777
19898 111570 177777
19899 111572
19900 111572 104000
19901 111574
19902 111574 004767 012772
19903
19904
19905
19906
19907
19908
19909
19910 111600
19911
19912 111600 012700 111702
19913 111604 012701 111722
19914 111610 012702 000010
19915 111614 012021
19916 111616 077202
19917 111620 012700 000200
19918 111624 170100
19919 111626 012700 111723
19920 111632 012737 111742 000004
19921
19922 111640 170770 000007
19923
19924 111644 170205
19925 111646 012701 111722
19926 111652 012702 000004
19927 111656 005721
19928 111660 001030
19929 111662 077203
19930 111664 020027 111723
19931 111670 001024
19932 111672 022705 000204
19933 111676 001021
19934 111700 000421

```
MOV #JJBFF0,R1 ;CHECK RESULT.  
MOV #4,R2  
1$: TST (R1)+  
BNE JJB10 ;BRANCH IF INCORRECT.  
SOB R2,1$  
CMP R0,#JJBFF0-7 ;IS R0 CORRECT?  
BNE JJB10 ;BRANCH IF INCORRECT.  
CMP #204,R5 ;IS THE FPS CORRECT?  
BNE JJB10 ;BRANCH IF INCORRECT.  
BR JJB DONE
```

;THESE ARE TEST DATA TABLES AND DATA BUFFER.

```
JJBTP1: 177  
161524  
131273  
107174,  
JJBFF0: -1  
-1  
-1  
-1  
JJB10:  
JJB DONE:
```

```
EMT ;  
JSR PC, .RSET ;GO INITIALIZE THE FPS AND STACK; AND  
;SEE IF THE USER HAS EXPRESSED  
;THE DESIRE TO CHANGE THE SOFTWARE  
;VIRTUAL CONSOLE SWITCH REGISTER (HAS  
;THE USER TYPED CONTROL G?).
```

```
*****  
:TEST 540 NEGf, ABSF AND TSTF SOURCE MODE 7 TEST  
*****  
T5540:
```

```
MOV #KKBTP1,R0 ;SET UP THE DATA BUFFER.  
MOV #KKBBFF0,R1  
MOV #10,R2  
1$: MOV (R0)+,(R1)+  
SOB R2,1$  
MOV #200,R0 ;SET FD.  
LDFPS R0  
MOV #KKBBF1-7,R0 ;SET UP THE OPERAND ADDRESS.  
MOV #KKB10,@#ERRVECT ;SET UP VECTOR 4 IN CASE OF AN ERROR.  
KKB2: NEGd @7(R0) ;TEST INSTRUCTION.  
STFPS R5 ;GET FPS.  
MOV #KKBBFF0,R1 ;CHECK RESULT.  
MOV #4,R2  
1$: TST (R1)+  
BNE KKB10 ;BRANCH IF INCORRECT.  
SOB R2,1$  
CMP R0,#KKBBF1-7 ;IS R0 CORRECT?  
BNE KKB10 ;BRANCH IF INCORRECT.  
CMP #204,R5 ;IS THE FPS CORRECT?  
BNE KKB10 ;BRANCH IF INCORRECT.  
BR KKB DONE
```

19935
19936
19937 111702 000177
19938 111704 167574
19939 111706 137271
19940 111710 107675 111722 177777
19941 111716 177777 177777
19942 111722 177777
19943 111724 177777
19944 111726 177777
19945 111730 177777
19946 111732 177777
19947 111734 177777
19948 111736 177777
19949 111740 177777
19950 111742
19951 111742 104000
19952 111744
19953 111744 004767 012622
19954
19955
19956
19957
19958
19959
19960
19961 111750
19962 111750 012700 112040
19963 111754 012701 112050
19964 111760 012702 000004
19965 111764 012021
19966 111766 077202
19967 111770 012700 000200
19968 111774 170100
19969 111776 012737 112060 000004
19970
19971 112004 170767 000040
19972
19973 112010 170205
19974 112012 012701 112050
19975 112016 012702 000004
19976 112022 005721
19977 112024 001015
19978 112026 077203
19979 112030 022705 000204
19980 112034 001011
19981 112036 000411
19982
19983
19984 112040 000127
19985 112042 137475
19986 112044 147372
19987 112046 117057
19988 112050 177777
19989 112052 177777
19990 112054 177777

;THESE ARE TEST DATA TABLES AND DATA BUFFER.

KKBTP1: 177
167574
137271
107675, KKBBF0, -1, -1, -1

KKBBF0: -1
-1
-1
-1
KKBBF1: -1
-1
-1
-1
KKB10: -1

EMT

KKBDONE: JSR PC, .RSET

;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;TEST 541 NEG, ABSF AND TSTF SOURCE MODE 6, GR7, TEST

T5541:

MOV #LLBTP1, R0 ;SET UP THE DATA BUFFER.
MOV #LLBBF0, R1
MOV #4, R2
1\$: MOV (R0)+, (R1)+
SOB R2, 1\$
MOV #200, R0 ;SET FD.
LDFPS R0
MOV #LLB10, @#ERRVECT ;SET UP VECTOR 4 IN CASE OF AN ERROR.
LLB2: NEG D LLBBF0 ;TEST INSTRUCTION.
STFPS R5 ;GET FPS.
MOV #LLBBF0, R1 ;CHECK RESULT.
MOV #4, R2
1\$: TST (R1)+
BNE LLB10 ;BRANCH IF INCORRECT.
SOB R2, 1\$
CMP #204, R5 ;IS THE FPS CORRECT?
BNE LLB10 ;BRANCH IF INCORRECT.
BR LLBDONE

;THESE ARE TEST DATA TABLES AND DATA BUFFER.

LLBTP1: 127
137475
147372
117057

LLBBF0: -1
-1
-1

19991 112056 177777
19992
19993 112060
19994 112060 104000
19995 112062
19996 112062 004767 012504
19997
19998
19999
20000
20001
20002
20003
20004 112066
20005
20006 112066 012700 112156
20007 112072 012701 112176
20008 112076 012702 000010
20009 112102 012021
20010 112104 077202
20011 112106 012700 000200
20012 112112 170100
20013 112114 012737 112216 000004
20014
20015 112122 170677 000060
20016
20017 112126 170205
20018 112130 012701 112176
20019 112134 012702 000004
20020 112140 005721
20021 112142 001025
20022 112144 077203
20023 112146 022705 000204
20024 112152 001021
20025 112154 000421
20026
20027
20028 112156 000137
20029 112160 045607
20030 112162 101230
20031 112164 045607 112176 177777
20032 112172 177777 177777
20033 112176 177777
20034 112200 177777
20035 112202 177777
20036 112204 177777
20037 112206 177777
20038 112210 177777
20039 112212 177777
20040 112214 177777
20041
20042 112216
20043 112216 104000
20044 112220
20045 112220 004767 012346
20046

-1
LLB10:
EMT
LLBDONE:
JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).
:*****
:TEST 542 NEG, ABSF AND TSTF SOURCE MODE 7, GR7, TEST
:*****
TS542:
MOV #MMBTP1,R0 ;SET UP THE DATA BUFFER.
MOV #MMBBF0,R1
MOV #10,R2
1\$: MOV (R0)+,(R1)+
SOB R2,1\$
MOV #200,R0 ;SET FD.
LDFPS R0
MOV #MMB10,@#ERRVECT ;SET UP VECTOR 4 IN CASE OF AN ERROR.
MMB2: ABSD @MMBBF1 ;TEST INSTRUCTION.
STFPS R5 ;GET FPS.
MOV #MMBBF0,R1 ;CHECK RESULT.
MOV #4,R2
1\$: TST (R1)+
BNE MMB10 ;BRANCH IF INCORRECT.
SOB R2,1\$
CMP #204,R5 ;IS THE FPS CORRECT?
BNE MMB10 ;BRANCH IF INCORRECT.
BR MMBDONE
:THESE ARE TEST DATA TABLES AND DATA BUFFER.
MMBTP1: 137
045607
101230
45607,MMBBF0,-1,-1,-1
MMBBF0: -1
-1
-1
-1
MMBBF1: -1
-1
-1
-1
MMB10:
EMT
MMBDONE:
JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED

20047
20048
20049
20050
20051
20052
20053 112224
20054
20055 112224 012700 000200
20056 112230 170100
20057 112232 012700 112312
20058 112236 172410
20059 112240 170700
20060
20061 112242 170205
20062 112244 012700 000200
20063 112250 170100
20064 112252 012700 112332
20065 112256 174010
20066 112260 012700 112332
20067 112264 012701 112322
20068 112270 012702 000004
20069 112274 022021
20070 112276 001021
20071 112300 077203
20072 112302 022705 000210
20073 112306 001015
20074 112310 000415
20075
20076
20077 112312 013572
20078 112314 046013
20079 112316 057246
20080 112320 013570
20081 112322 113572
20082 112324 046013
20083 112326 057246
20084 112330 013570
20085 112332 000000
20086 112334 000000
20087 112336 000000
20088 112340 000000
20089
20090 112342
20091 112342 104000
20092 112344
20093 112344 004767 012222
20094
20095
20096
20097
20098
20099
20100
20101 112350
20102

```

;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).
:*****
:TEST 543 SPECIAL DEST, MODE 0, TEST
:*****
TS543:
      MOV #200,R0 ;SET FD.
      LDFPS R0
      MOV #NNBTP1,R0 ;SET UP ACO.
      LDD (R0),ACO
NNB2: NEG D ACO ;TEST INSTRUCTION.

      STFPS R5 ;GET FPS.
      MOV #200,R0 ;SET FD.
      LDFPS R0
      MOV #NNBBF0,R0 ;GET THE RESULT.
      STD ACO,(R0)
      MOV #NNBBF0,R0 ;IS THE RESULT CORRECT?
      MOV #NNBTP2,R1
      MOV #4,R2
1$: CMP (R0)+,(R1)+
      BNE NNB10 ;BRANCH IF INCORRECT.
      SOB R2,1$
      CMP #210,R5 ;IS THE FPS CORRECT?
      BNE NNB10 ;BRANCH IF INCORRECT.
      BR NNB DONE

;THESE ARE DATA TABLES AND A DATA BUFFER.
NNBTP1: 013572
        46013
        57246
        013570
NNBTP2: 113572
        46013
        57246
        013570
NNBBF0: 0
        0
        0
        0
NNB10:
NNB DONE:
      JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).
:*****
:TEST 544 SPECIAL DEST, MODE 1, TEST
:*****
TS544:
```

20103 112350 012701 112452
20104 112354 012700 112462
20105 112360 012702 000004
20106 112364 012021
20107 112366 077202
20108 112370 012700 112452
20109 112374 042710 100000
20110 112400 012701 000200
20111 112404 170101
20112
20113 112406 170710
20114 112410 170205
20115 112412 012701 112452
20116 112416 012702 112462
20117 112422 012703 000004
20118 112426 022122
20119 112430 001020
20120 112432 077303
20121 112434 022700 112452
20122 112440 001014
20123 112442 022705 000210
20124 112446 001011
20125 112450 000411
20126
20127
20128 112452 023245
20129 112454 026720
20130 112456 122324
20131 112460 052672
20132 112462 123245
20133 112464 026720
20134 112466 122324
20135 112470 052672
20136
20137 112472
20138 112472 104000
20139 112474
20140 112474 004767 012072
20141
20142
20143
20144
20145
20146
20147
20148 112500
20149
20150 112500 012701 112602
20151 112504 012700 112612
20152 112510 012702 000004
20153 112514 012021
20154 112516 077202
20155 112520 012700 112602
20156 112524 042710 100000
20157 112530 012701 000200
20158 112534 170101

MOV #00BTP1,R1 ;SET UP THE DATA BUFFER.
MOV #00BTP2,R0
MOV #4,R2
1\$: MOV (R0)+,(R1)+
SOB R2,1\$
MOV #00BTP1,R0
BIC #100000,(R0) ;MAKE OPERAND POSITIVE.
MOV #200,R1 ;SET FD.
LDFPS R1

00B2: NEG (R0) ;TEST INSTRUCTION.
STFPS R5 ;GET FPS.
MOV #00BTP1,R1 ;IS THE RESULT CORRECT.
MOV #00BTP2,R2
MOV #4,R3
1\$: CMP (R1)+,(R2)+
BNE 00B10 ;BRANCH IF INCORRECT.
SOB R3,1\$
CMP #00BTP1,R0 ;IS R0 CORRECT.
BNE 00B10 ;BRANCH IF INCORRECT.
CMP #210,R5 ;IS THE FPS CORRECT?
BNE 00B10 ;BRANCH IF INCORRECT.
BR 00BDONE

;THESE ARE DATA TABLES AND A DATA BUFFER.
00BTP1: 023245
26720
122324
52672
00BTP2: 123245
26720
122324
52672

00B10:
00BDONE: EMT ;
JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).
:*****
;TEST 545 SPECIAL DEST, MODE 2, TEST
:*****
TS545:
MOV #PPBTP1,R1 ;SET UP THE DATA BUFFER.
MOV #PPBTP2,R0
MOV #4,R2
1\$: MOV (R0)+,(R1)+
SOB R2,1\$
MOV #PPBTP1,R0
BIC #100000,(R0) ;MAKE OPERAND POSITIVE.
MOV #200,R1 ;SET FD.
LDFPS R1

```

20159
20160 112536 170720 PPB2:  NEG D      (R0)+      ;TEST INSTRUCTION.
20161
20162 112540 170205      STFPS   R5      ;GET FPS.
20163 112542 012701 112602  MOV    #PPBTP1,R1      ;IS THE RESULT CORRECT.
20164 112546 012702 112612  MOV    #PPBTP2,R2
20165 112552 012703 000004  MOV    #4,R3
20166 112556 022122      1$:    CMP    (R1)+,(R2)+
20167 112560 001020      BNE    PPB10      ;BRANCH IF INCORRECT.
20168 112562 077303      SOB    R3,1$
20169 112564 022700 112612  CMP    #PPBTP1+10,R0  ;IS R0 CORRECT.
20170 112570 001014      BNE    PPB10      ;BRANCH IF INCORRECT.
20171 112572 022705 000210  CMP    #210,R5      ;IS THE FPS CORRECT?
20172 112576 001011      BNE    PPB10      ;BRANCH IF INCORRECT.
20173 112600 000411      BR     PPBDONE
20174
20175      ;THESE ARE DATA TABLES AND A DATA BUFFER.
20176 112602 023245  PPBTP1: 023245
20177 112604 026720      26720
20178 112606 122324      122324
20179 112610 052672      52672
20180 112612 123245  PPBTP2: 123245
20181 112614 026720      26720
20182 112616 122324      122324
20183 112620 052672      52672
20184
20185 112622      PPB10:
20186 112622 104000      EMT      ;
20187 112624      PPBDONE:
20188 112624 004767 011742  JSR    PC,,RSET    ;GO INITIALIZE THE FPS AND STACK; AND
20189      ;SEE IF THE USER HAS EXPRESSED
20190      ;THE DESIRE TO CHANGE THE SOFTWARE
20191      ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
20192      ;THE USER TYPED CONTROL G?).
20193
20194      ;*****
20195      ;TEST 546      SPECIAL DEST, MODE 4, TEST
20196      ;*****
20196 112630      TS546:
20197 112630 012701 112734  MOV    #QQBTP1,R1    ;SET UP THE DATA BUFFER.
20198 112634 012700 112754  MOV    #QQBTP2,R0
20199 112640 012702 000004  MOV    #4,R2
20200 112644 012021      1$:    MOV    (R0)+,(R1)+
20201 112646 077202      SOB    R2,1$
20202 112650 012700 112744  MOV    #QQBTP1+10,R0
20203 112654 042760 100000 177770  BIC    #100000,-10(R0) ;MAKE OPERAND POSITIVE.
20204 112662 012701 000200  MOV    #200,R1      ;SET FD.
20205 112666 170101      LDFPS  R1
20206
20207 112670 170740  QQB2:  NEG D      -(R0)      ;TEST INSTRUCTION.
20208
20209 112672 170205      STFPS   R5      ;GET FPS.
20210 112674 012701 112734  MOV    #QQBTP1,R1      ;IS THE RESULT CORRECT.
20211 112700 012702 112754  MOV    #QQBTP2,R2
20212 112704 012703 000004  MOV    #4,R3
20213 112710 022122      1$:    CMP    (R1)+,(R2)+
20214 112712 001024      BNE    QQB10      ;BRANCH IF INCORRECT.

```


20215 112714 077303
20216 112716 022700 112734
20217 112722 001020
20218 112724 022705 000210
20219 112730 001015
20220 112732 000415
20221
20222
20223 112734 023245
20224 112736 026720
20225 112740 122324
20226 112742 052672
20227 112744 177777 177777 177777
20228 112752 177777
20229 112754 123245
20230 112756 026720
20231 112760 122324
20232 112762 052672
20233
20234 112764
20235 112764 104000
20236 112766
20237 112766 004767 011600
20238
20239
20240
20241
20242
20243
20244
20245
20246 112772
20247
20248 112772 012701 113102
20249 112776 012700 113112
20250 113002 012702 000004
20251 113006 012021
20252 113010 077202
20253 113012 012700 113122
20254 113016 012710 113102
20255 113022 042737 100000 113102
20256 113030 012701 000200
20257 113034 170101
20258
20259 113036 170730
20260
20261 113040 170205
20262 113042 012701 113102
20263 113046 012702 113112
20264 113052 012703 000004
20265 113056 022122
20266 113060 001021
20267 113062 077303
20268 113064 022700 113124
20269 113070 001015
20270 113072 022705 000210

SOB R3,1\$
CMP #QOBTP1,R0 ;IS R0 CORRECT.
BNE QOB10 ;BRANCH IF INCORRECT.
CMP #210,R5 ;IS THE FPS CORRECT?
BNE QOB10 ;BRANCH IF INCORRECT.
BR QOBDONE
;THESE ARE DATA TABLES AND A DATA BUFFER.
QOBTP1: 023245
26720
122324
52672
.WORD -1,-1,-1,-1
QOBTP2: 123245
26720
122324
52672
QOB10:
EMT ;
QOBDONE: JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;TEST 547 SPECIAL DEST, MODE 3, TEST

T547:

MOV #RRBTP1,R1 ;SET UP THE DATA BUFFER.
MOV #RRBTP2,R0
MOV #4,R2
1\$: MOV (R0)+,(R1)+
SOB R2,1\$
MOV #RRBTP3,R0
MOV #RRBTP1,(R0)
20255: BIC #100000,#RRBTP1 ;MAKE THE OPERAND POSITIVE.
MOV #200,R1 ;SET FD.
LDFPS R1
RRB2: NEG D @ (R0)+ ;TEST INSTRUCTION.
STFPS R5 ;GET FPS.
MOV #RRBTP1,R1 ;IS THE RESULT CORRECT.
MOV #RRBTP2,R2
MOV #4,R3
1\$: CMP (R1)+,(R2)+
BNE RRB10 ;BRANCH IF INCORRECT.
SOB R3,1\$
CMP #RRBTP3+2,R0 ;IS R0 CORRECT.
BNE RRB10 ;BRANCH IF INCORRECT.
CMP #210,R5 ;IS THE FPS CORRECT?

20271 113076 001012
20272 113100 000412
20273
20274
20275 113102 023245
20276 113104 026720
20277 113106 122324
20278 113110 052672
20279 113112 123245
20280 113114 026720
20281 113116 123324
20282 113120 052672
20283 113122 113102
20284
20285 113124
20286 113124 104000
20287 113126
20288 113126 004767 011440
20289
20290
20291
20292
20293
20294
20295
20296
20297 113132
20298 113132 012701 113244
20299 113136 012700 113254
20300 113142 012702 000004
20301 113146 012021
20302 113150 077202
20303 113152 012700 113266
20304 113156 012760 113244 177776
20305 113164 042737 100000 113244
20306 113172 012701 000200
20307 113176 170101
20308
20309 113200 170750
20310
20311 113202 170205
20312 113204 012701 113244
20313 113210 012702 113254
20314 113214 012703 000004
20315 113220 022122
20316 113222 001021
20317 113224 077303
20318 113226 022700 113264
20319 113232 001015
20320 113234 022705 000210
20321 113240 001012
20322 113242 000412
20323
20324
20325 113244 023245
20326 113246 026720

BNE RRB10 ;BRANCH IF INCORRECT.
BR RRBDONE

;THESE ARE DATA TABLES AND A DATA BUFFER.

RRBTP1: 023245

26720

122324

52672

RRBTP2: 123245

26720

123324

52672

RRBTP3: RRBTP1

RRB10:

EMT ;

RRBDONE:

JSR PC,,RSET

;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;TEST 550 SPECIAL DEST, MODE 5, TEST

T550:

MOV #SSBTP1,R1 ;SET UP THE DATA BUFFER.

MOV #SSBTP2,R0

MOV #4,R2

1\$: MOV (R0)+,(R1)+

SOB R2,1\$

MOV #SSBTP3+2,R0

MOV #SSBTP1,-2(R0)

BIC #100000,@#SSBTP1 ;MAKE THE OPERAND POSITIVE.

MOV #200,R1 ;SET FD.

LDFPS R1

SSB2: NEG D @-(R0) ;TEST INSTRUCTION.

STFPS R5 ;GET FPS.

MOV #SSBTP1,R1 ;IS THE RESULT CORRECT.

MOV #SSBTP2,R2

MOV #4,R3

1\$: CMP (R1)+,(R2)+

BNE SSB10 ;BRANCH IF INCORRECT.

SOB R3,1\$

CMP #SSBTP3,R0 ;IS R0 CORRECT.

BNE SSB10 ;BRANCH IF INCORRECT.

CMP #210,R5 ;IS THE FPS CORRECT?

BNE SSB10 ;BRANCH IF INCORRECT.

BR SSBDONE

;THESE ARE DATA TABLES AND A DATA BUFFER.

SSBTP1: 023245

26720

20327 113250 122324
20328 113252 052672
20329 113254 123245
20330 113256 026270
20331 113260 122324
20332 113262 052672
20333 113264 113244
20334
20335 113266
20336 113266 104000
20337 113270
20338 113270 004767 011276
20339
20340
20341
20342
20343
20344
20345
20346 113274
20347 113274 012701 113376
20348 113300 012700 113406
20349 113304 012702 000004
20350 113310 012021
20351 113312 077202
20352 113314 012700 113376
20353 113320 042710 100000
20354 113324 012701 000000
20355 113330 170101
20356
20357 113332 170720
20358
20359 113334 170205
20360 113336 012701 113376
20361 113342 012702 113406
20362 113346 012703 000004
20363 113352 022122
20364 113354 001020
20365 113356 077303
20366 113360 022700 113402
20367 113364 001014
20368 113366 022705 000010
20369 113372 001011
20370 113374 000411
20371
20372
20373 113376 023245
20374 113400 026720
20375 113402 122324
20376 113404 052672
20377 113406 123245
20378 113410 026720
20379 113412 122324
20380 113414 052672
20381
20382 113416

122324
52672
SSBTP2: 123245
26270
122324
52672
SSBTP3: SSBTP1
SSB10:
SSBDONE: EMT ;
JSR PC, .RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).
:*****
:TEST 551 SPECIAL DEST, FLOATING MODE 2, TEST
:*****
T5551:
MOV #TTBTP1,R1 ;SET UP THE DATA BUFFER.
MOV #TTBTP2,R0
MOV #4,R2
1\$: MOV (R0)+,(R1)+
SOB R2,1\$
MOV #TTBTP1,R0
BIC #100000,(R0) ;MAKE OPERAND POSITIVE.
MOV #000,R1 ;SET FD.
LDFPS R1
TTB2: NEG (R0)+ ;TEST INSTRUCTION.
STFPS R5 ;GET FPS.
MOV #TTBTP1,R1 ;IS THE RESULT CORRECT.
MOV #TTBTP2,R2
MOV #4,R3
1\$: CMP (R1)+,(R2)+
BNE TTB10 ;BRANCH IF INCORRECT.
SOB R3,1\$
CMP #TTBTP1+4,R0 ;IS R0 CORRECT.
BNE TTB10 ;BRANCH IF INCORRECT.
CMP #010,R5 ;IS THE FPS CORRECT?
BNE TTB10 ;BRANCH IF INCORRECT.
BR TTBDONE
:THESE ARE DATA TABLES AND A DATA BUFFER.
TTBTP1: 023245
26720
122324
52672
TTBTP2: 123245
26720
122324
52672
TTB10:

20383 113416 104000
20384 113420
20385 113420 004767 011146
20386
20387
20388
20389
20390
20391
20392
20393 113424
20394 113424 012700 113542
20395 113430 012701 113470
20396 113434 012702 000004
20397 113440 012021
20398 113442 077202
20399 113444 012700 113470
20400 113450 042737 100000 113470
20401 113456 012701 000200
20402 113462 170101
20403 113464 005001
20404
20405 113466 170727
20406 113470 005201 005201 005201
20407 113476 005201
20408
20409 113500 170205
20410 113502 012707 113470
20411 113506 012707 113542
20412 113512 012704 000004
20413 113516 022322
20414 113520 001014
20415 113522 077403
20416 113524 022701 000003
20417 113530 001010
20418 113532 022705 000210
20419 113536 001005
20420 113540 000405
20421
20422
20423 113542 105201
20424 113544 005201
20425 113546 005201
20426 113550 005201
20427
20428 113552
20429 113552 104000
20430 113554
20431 113554 004767 011012
20432
20433
20434
20435
20436
20437
20438

```
EMT ;
TTBDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).
;*****
;TEST 552 SPECIAL DEST, MODE2, GR7 (IMMEDIATE), TEST
;*****
TS552:
MOV #UUBTP2,R0
MOV #UUBTP1,R1 ;SET UP THE DATA BUFFER.
MOV #4,R2
1$: MOV (R0)+,(R1)+
SOB R2,1$
MOV #UUBTP1,R0
BIC #100000,@#UUBTP1 ;MAKE THE OPERAND POSITIVE.
MOV #200,R1 ;SET FD.
LDFPS R1
CLR R1
UUB2: NEG (R7)+ ;TEST INSTRUCTION.
UUBTP1: 5201,5201,5201,5201
;NOTE THAT AFTER EXECUTING THIS INSTRUCTION R1 SHOULD CONTAIN 3.
STFPS R5 ;GET FPS.
MOV #UUBTP1,R3 ;IS THE RESULT CORRECT.
MOV #UUBTP2,R2
MOV #4,R4
1$: CMP (R3)+,(R2)+
BNE UUB10 ;BRANCH IF INCORRECT.
SOB R4,1$
CMP #3,R1 ;WAS R1 INCREMENTED CORRECTLY.
BNE UUB10 ;BRANCH IF INCORRECT.
CMP #210,R5 ;IS THE FPS CORRECT?
BNE UUB10 ;BRANCH IF INCORRECT.
BR UUBDONE
;THESE ARE DATA TABLE.
UUBTP2: 105201
5201
5201
5201
UUB10:
EMT ;
UUBDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).
;*****
;TEST 553 SPECIAL DEST, MODE 6, TEST
;*****
```

20439 113560
20440 113560 012701 113674
20441 113564 012700 113704
20442 113570 012702 000004
20443 113574 012021
20444 113576 077202
20445 113600 012700 106473
20446 113604 042737 100000 113674
20447 113612 012701 000200
20448 113616 170101
20449
20450 113620 005001
20451 113622 170760 005201
20452
20453 113626 170205
20454 113630 005701
20455 113632 001030
20456 113634 012701 113674
20457 113640 012702 113704
20458 113644 012703 000004
20459 113650 022122
20460 113652 001020
20461 113654 077303
20462 113656 022700 106473
20463 113662 001014
20464 113664 022705 000210
20465 113670 001011
20466 113672 000411
20467
20468
20469 113674 023245
20470 113676 026720
20471 113700 122324
20472 113702 052672
20473 113704 123245
20474 113706 026720
20475 113710 122324
20476 113712 052672
20477
20478
20479 113714
20480 113714 104000
20481 113716
20482 113716 004767 010650
20483
20484
20485
20486
20487
20488
20489
20490
20491 113722
20492
20493 113722 012701 114044
20494 113726 012700 114054

TS553:
MOV #XXBTP1,R1 ;SET UP THE DATA BUFFER.
MOV #XXBTP2,R0
MOV #4,R2
1\$: MOV (R0)+,(R1)+
SOB R2,1\$
MOV #XXBTP1-5201,R0
BIC #100000,@#XXBTP1;MAKE OPERAND POSITIVE.
MOV #200,R1 ;SET FD.
LDFPS R1

XXB2: CLR R1
NEGD 5201(R0) ;TEST INSTRUCTION.

STFPS R5 ;GET FPS.
TST R1
BNE XXB10 ;WAS THE PC CORRECT AFTER EXECUTION?
MOV #XXBTP1,R1 ;IS THE RESULT CORRECT.
MOV #XXBTP2,R2
MOV #4,R3
1\$: CMP (R1)+,(R2)+
BNE XXB10 ;BRANCH IF INCORRECT.
SOB R3,1\$
CMP #XXBTP1-5201,R0 ;IS R0 CORRECT.
BNE XXB10 ;BRANCH IF INCORRECT.
CMP #210,R5 ;IS THE FPS CORRECT?
BNE XXB10 ;BRANCH IF INCORRECT.
BR XXBDONE

;THESE ARE DATA TABLES AND A DATA BUFFER.
XXBTP1: 023245
26720
122324
52672
XXBTP2: 123245
26720
122324
52672

XXB10:
EMT ;
XXBDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;*****
;TEST 554 SPECIAL DEST, MODE 7, TEST
;*****
TS554:
MOV #YYBTP1,R1 ;SET UP THE DATA BUFFER.
MOV #YYBTP2,R0

20495	113732	012702	000004	
20496	113736	012021		
20497	113740	077202		
20498	113742	012700	106663	
20499	113746	012760	114044	005201
20500	113754	042737	100000	114044
20501	113762	012701	000200	
20502	113766	170101		
20503				
20504	113770	005001		
20505	113772	170770	005201	
20506				
20507	113776	170205		
20508	114000	005701		
20509	114002	001031		
20510	114004	012701	114044	
20511	114010	012702	114054	
20512	114014	012703	000004	
20513	114020	022122		
20514	114022	001021		
20515	114024	077303		
20516	114026	022700	106663	
20517	114032	001015		
20518	114034	022705	000210	
20519	114040	001012		
20520	114042	000412		
20521				
20522				
20523	114044	023245		
20524	114046	026720		
20525	114050	122324		
20526	114052	052672		
20527	114054	123245		
20528	114056	026720		
20529	114060	123324		
20530	114062	052672		
20531	114064	114044		
20532	114066			
20533	114066	104000		
20534				
20535	114070			
20536	114070	004767	010476	
20537				
20538				
20539				
20540				
20541				
20542				
20543				
20544	114074			
20545				
20546	114074	004767	000526	
20547	114100	000000		
20548	114102	016341		
20549	114104	055772		
20550	114106	021133		

```

MOV #4,R2
1$: MOV (R0)+,(R1)+
SOB R2,1$
MOV #YYBTP3-5201,R0
MOV #YYBTP1,5201(R0)
BIC #100000,@#YYBTP1 ;MAKE THE OPERAND POSITIVE.
MOV #200,R1 ;SET FD.
LDFPS R1

YB2: CLR R1
NEG D @5201(R0) ;TEST INSTRUCTION.

STFPS R5 ;GET FPS.
TST R1 ;WAS THE PC CORRECT AFTER EXECUTION?
BNE YYB10
MOV #YYBTP1,R1 ;IS THE RESULT CORRECT.
MOV #YYBTP2,R2
1$: MOV #4,R3
CMP (R1)+,(R2)+
BNE YYB10 ;BRANCH IF INCORRECT.
SOB R3,1$
CMP #YYBTP3-5201,R0 ;IS R0 CORRECT.
BNE YYB10 ;BRANCH IF INCORRECT.
CMP #210,R5 ;IS THE FPS CORRECT?
BNE YYB10 ;BRANCH IF INCORRECT.
BR YYBDONE

;THESE ARE DATA TABLES AND A DATA BUFFER.
YYBTP1: 023245
26720
122324
52672
YYBTP2: 123245
26720
123324
52672
YYBTP3: YYBTP1
YYB10:
EMT ;

YYBDONE:
JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;*****
;TEST 555 NEG D, ABS D AND TST D TEST
;*****
TS555:
;TEST NEG D WITH POS NONZERO OPERAND
WOB1: JSR PC,NATSUB
1$: 0 ;FLAG=NEG D.
2$: 16341 ;OPERAND.
55772
21133

```

20551	114110	055447			55447	
20552	114112	116341		3\$:	116341	;RESULT.
20553	114114	055772			55772	
20554	114116	021133			21133	
20555	114120	055447			55447	
20556	114122	016341		4\$:	16341	;ERROR RES.
20557	114124	055772			55772	
20558	114126	021133			21133	
20559	114130	055447			55447	
20560	114132	000207		5\$:	207	;FPS BEFORE EXECUTION.
20561	114134	000210			210	;FPS AFTER EXECUTION.
20562	114136	000200			200	;ERROR FPS.
20563	114140	177777			-1	;FEC
20564						
20565	114142	004767	000460			
20566	114146	000000		WMB2:	JSR PC,NATSUB	
20567	114150	152525		1\$:	0	;FLAG=NEGD.
20568	114152	053545		2\$:	152525	;OPERAND.
20569	114154	055565			53545	
20570	114156	057505			55565	
20571	114160	052525		3\$:	52525	;RESULT.
20572	114162	053545			53545	
20573	114164	055565			55565	
20574	114166	057505			57505	
20575	114170	152525		4\$:	152525	;ERROR RES.
20576	114172	053545			53545	
20577	114174	055565			55565	
20578	114176	057505			57505	
20579	114200	000217		5\$:	217	;FPS BEFORE EXECUTION.
20580	114202	000200			200	;FPS AFTER EXECUTION.
20581	114204	000210			210	;ERROR FPS.
20582	114206	177777			-1	;FEC
20583						
20584	114210	004767	000412			
20585	114214	000001		WMB3:	JSR PC,NATSUB	
20586	114216	060705		1\$:	1	;FLAG=ABSD.
20587	114220	124735		2\$:	60705	;OPERAND.
20588	114222	060124			124735	
20589	114224	073560			60124	
20590	114226	060705		3\$:	73560	;RESULT.
20591	114230	124735			60705	
20592	114232	060124			124735	
20593	114234	073560			60124	
20594	114236	160705		4\$:	73560	;ERROR RES.
20595	114240	124735			160705	
20596	114242	060124			124735	
20597	114244	073560			60124	
20598	114246	000217		5\$:	73560	;FPS BEFORE EXECUTION.
20599	114250	000200			217	;FPS AFTER EXECUTION.
20600	114252	000210			200	;ERROR FPS.
20601	114254	177777			210	;EITHER BUT OP1B
20602					-1	
20603	114256	004767	000344			
20604	114262	000001		WMB4:	JSR PC,NATSUB	
20605	114264	154345		1\$:	1	;FLAG=ABSD.
20606	114266	076567		2\$:	154345	;OPERAND.
					76567	

20607	114270	032123		32123	
20608	114272	043234		43234	
20609	114274	054345		54345	;RESULT.
20610	114276	076567		76567	
20611	114300	032123		32123	
20612	114302	043234		43234	
20613	114304	154345		154345	;ERROR RES.
20614	114306	076567		76567	
20615	114310	032123		32123	
20616	114312	043234		43234	
20617	114314	000217		217	;FPS BEFORE EXECUTION.
20618	114316	000200		200	;FPS AFTER EXECUTION.
20619	114320	177777		-1	;ERROR FPS.
20620	114322	177777		-1	
20621					;TEST WITH POSITIVE OP
20622	114324	004767	000276	WWS5: JSR PC,NATSUB	
20623	114330	000002		1\$: 2	;FLAG=TSTD.
20624	114332	012321		2\$: 12321	;OPERAND.
20625	114334	045654		45654	
20626	114336	070107		70107	
20627	114340	034543		34543	
20628	114342	012321		12321	;RESULT.
20629	114344	045654		45654	
20630	114346	070107		70107	
20631	114350	034543		34543	
20632	114352	112321		112321	;ERROR RES.
20633	114354	045654		45654	
20634	114356	070107		70107	
20635	114360	034543		34543	
20636	114362	000217		217	;FPS BEFORE EXECUTION.
20637	114364	000200		200	;FPS AFTER EXECUTION.
20638	114366	000210		210	;ERROR FPS.
20639	114370	177777		-1	
20640					;TEST TSTD WITH NEG OP
20641	114372	004767	000230	WWS6: JSR PC,NATSUB	
20642	114376	000002		1\$: 2	;FLAG=TSTD.
20643	114400	123765		2\$: 123765	;OPERAND.
20644	114402	023407		23407	
20645	114404	034510		34510	
20646	114406	045621		45621	
20647	114410	123765		123765	;RESULT.
20648	114412	023407		23407	
20649	114414	034510		34510	
20650	114416	045621		45621	
20651	114420	023765		23765	;ERROR RES.
20652	114422	023407		23407	
20653	114424	034510		34510	
20654	114426	045621		45621	
20655	114430	000207		207	;FPS BEFORE EXECUTION.
20656	114432	000210		210	;FPS AFTER EXECUTION.
20657	114434	000200		200	;ERROR FPS.
20658	114436	177777		-1	
20659					;TEST TSTD 0 OP
20660	114440	004767	000162	WWS7: JSR PC,NATSUB	
20661	114444	000002		1\$: 2	;FLAG=TSTD.
20662	114446	000175		2\$: 175	;OPERAND.


```
20663 114450 176737
20664 114452 071727
20665 114454 037574
20666 114456 000175 3S: 175 ;RESULT.
20667 114460 176737
20668 114462 071727
20669 114464 037574
20670 114466 000000 4S: 0 ;ERROR RES.
20671 114470 000000
20672 114472 000000
20673 114474 000000
20674 114476 000200 5S: 200 ;FPS BEFORE EXECUTION.
20675 114500 000204 ;FPS AFTER EXECUTION.
20676 114502 000214 ;ERROR FPS.
20677 114504 177777
20678
20679 114506 004767 000114 ;TEST TSTD -0 OP FIUV=0
WUB10: JSR PC,NATSUB
20680 114512 000002 1S: 2 ;FLAG=TSTD.
20681 114514 100123 2S: 100123 ;OPERAND.
20682 114516 021012
20683 114520 034565
20684 114522 043210
20685 114524 100123 3S: 100123 ;RESULT.
20686 114526 021012
20687 114530 034565
20688 114532 043210
20689 114534 000000 4S: 0 ;ERROR RES.
20690 114536 000000
20691 114540 000000
20692 114542 000000
20693 114544 040203 5S: 40203 ;FPS BEFORE EXECUTION.
20694 114546 040214 ;FPS AFTER EXECUTION.
20695 114550 140214 ;ERROR FPS.
20696 114552 177777
20697
20698 114554 004767 000046 ;TEST TSTD -0 OP FIUV=1
WUB11: JSR PC,NATSUB
20699 114560 000002 1S: 2 ;FLAG=TSTD.
20700 114562 100137 2S: 100137 ;OPERAND.
20701 114564 024613
20702 114566 057024
20703 114570 060137
20704 114572 100137 3S: 100137 ;RESULT.
20705 114574 024613
20706 114576 057024
20707 114600 060137
20708 114602 000000 4S: 0 ;ERROR RES.
20709 114604 000000
20710 114606 000000
20711 114610 000000
20712 114612 044200 5S: 44200 ;FPS BEFORE EXECUTION.
20713 114614 144214 ;FPS AFTER EXECUTION.
20714 114616 044214 ;ERROR FPS.
20715 114620 000014
20716 114622 000167 000162 JMP WUBDONE
20717
20718 ;THIS SUBROUTINE, NATSUB, IS USED TO SET UP THE OPERANDS. EXECUTE
```

20719
 20720
 20721
 20722
 20723
 20724
 20725
 20726
 20727
 20728
 20729
 20730
 20731
 20732
 20733
 20734
 20735
 20736
 20737
 20738
 20739
 20740
 20741
 20742
 20743
 20744
 20745
 20746
 20747
 20748
 20749
 20750
 20751
 20752
 20753
 20754
 20755
 20756
 20757
 20758
 20759
 20760
 20761
 20762
 20763
 20764
 20765
 20766
 20767
 20768
 20769
 20770
 20771
 20772
 20773
 20774

114626 012601
 114630 010102
 114632 062702 000002
 114636 012703 114776
 114642 012704 000004
 114646 012223
 114650 077402
 114652 016100 000032
 114656 170100
 114660 012700 114776
 114664 011102
 114666 006302
 114670 006302
 114672 012703 114702
 114676 060203
 114700 000113
 114702 170710
 114704 000403
 114706 170610
 114710 000401
 114712 170510

:THE EITHER A TSTD, AN ABSD OR A NEGD INSTRUCTION AND CHECK THE RESULTS. A CALL
 :TO IT IS MADE THUS:

```

      JSR      PC,@#NATSUB
      FLAG:   .WORD   X           ;INSTRUCTION TYPE FLAG.
      ACARG:  .WORD   X,X,X,X    ;OPERAND
      RES:    .WORD   X,X,X,X    ;EXPECTED RESULT
      ERRES:  .WORD   X,X,X,X    ;ERROR RESULT
      FPSB:   .WORD   X           ;FPS BEFORE EXECUTION
      FPSA:   .WORD   X           ;FPS AFTER EXECUTION
      FEC:    .WORD   X           ;EXPECTED FEC
      ERFPS:  .WORD   X           ;ERROR FPS.
      ERR1:   ERROR   X           ;DATA ERROR.
      ERR2:   BR      CONT
      CONT:   ERROR   X           ;FPS ERROR.
                                   ;RETURN ADDRESS
  
```

:THE OPERAND IS SET UP IN NATBF1. THEN
 :THE EITHER THE TSTD, NEGD OR ABSD INSTRUCTION IS EXECUTED.
 :NATSUB USES THE FIRST OPERAND AS A FLAG TO DETERMINE WHICH INSTRUCTION
 :IS TO BE EXECUTED: 0 = NEGD, 1 = ABSD, 2 = TSTD.
 :THE RESULT IS CHECKED AGAINST RES. IF THE RESULT IS CORRECT THEN THE FPS IS
 :COMPARED WITH FPSA. IF THIS TOO IS CORRECT NATSUB RETURNS CONTROL
 :TO THE CALLING ROUTINE AT CONT. IF THE FPS IS BAD NATSUB
 :COMPARE IT TO ERROR FPS. IF THIS MATCHES THEN NATSUB WILL RETURN
 :TO THE ERROR CALL AT ERR2, OTHERWISE NATSUB ITSELF
 :REPORTS THIS FAILURE AND THEN RETURNS TO CONT. IF THE RESULT OF THE
 :INSTRUCTION IS INCORRECT, THE INCORRECT RESULT IS COMPARED WITH THE
 :ANTICIPATED FAILING DATA PATTERN, ERRES. IF THE FAILURE IN
 :THE RESULT WAS ANTICIPATED CORRECTLY TO BE ERRES THEN NATSUB
 :WILL TRANSFER CONTROL TO THE ERROR CALL AT ERR1. OTHERWISE THE
 :RESULT WAS INCORRECT BUT WAS NOT ANTICIPATED AND NATSUB WILL
 :REPORT THE FAILURE AFTER WHICH CONTROL WILL BE PASSED TO CONT.

```

NATSUB: MOV     (SP)+,R1           ;GET A POINTER TO THE ARGUMENTS.
        MOV     R1,R2           ;COPY THE OPERAND.
        ADD     #2,R2
        MOV     #NATBF1,R3
        MOV     #4,R4
1$:     MOV     (R2)+,(R3)+
        SOB     R4,1$
        MOV     32(R1),R0       ;LOAD THE FPS.
        LDFPS  R0
        MOV     #NATBF1,R0     ;SET UP THE OPERAND ADDRESS.
        MOV     (R1),R2       ;GET THE FLAG TO DETERMINE WHICH
        ASL     R2           ;INSTRUCTION TO EXECUTE.
        ASL     R2           ;0 = NEGD, 1 = ABSD, 2 = TSTD
        MOV     #NATINS,R3
        ADD     R2,R3
        JMP     (R3)         ;GO EXECUTE THE INSTRUCTION.
NATINS: NEGD   (R0)
        BR     2$
        ABSD  (R0)
        BR     2$
        TSTD  (R0)
  
```

```
20775
20776 114714 170204      2$:  STFPS  R4          ;GET THE FPS.
20777 114716 170305      STST   R5          ;GET THE FEC.
20778 114720 010100      MOV    R1,R0       ;WAS THE RESULT CORRECT?
20779 114722 062700 000012  ADD   #12,R0
20780 114726 012702 114776  MOV   #NATBF1,R2
20781 114732 012703 000004  MOV   #4,R3
20782 114736 022022      3$:  CMP    (R0)+,(R2)+
20783 114740 001014      BNE   10$         ;BRANCH IF INCORRECT.
20784 114742 077303      SOB   R3,3$
20785 114744 026104 000034  CMP   34(R1),R4   ;WAS THE FPS CORRECT?
20786 114750 001010      BNE   10$         ;BRANCH IF INCORRECT.
20787 114752 005761 000034  TST   34(R1)      ;IF THE EXPECTED FPS WAS NEGATIVE CHECK THE FEC.
20788 114756 100003      BPL   4$
20789 114760 026105 000040  CMP   40(R1),R5   ;WAS THE FEC CORRECT.
20790 114764 001002      BNE   10$         ;BRANCH IF INCORRECT.
20791 114766 000161 000042  4$:  JMP   42(R1)     ;RETURN.
20792
20793 114772      10$:
20794 114772 104000      EMT
20795
20796 114774 177777      .WORD -1
20797 114776 177777 177777 177777 NATBF1: .WORD -1,-1,-1,-1,-1
20798 115004 177777 177777
20799
20800 115010      WWBDONE:
20801 115010 004767 007556  JSR   PC,,RSET    ;GO INITIALIZE THE FPS AND STACK; AND
20802                                     ;SEE IF THE USER HAS EXPRESSED
20803                                     ;THE DESIRE TO CHANGE THE SOFTWARE
20804                                     ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
20805                                     ;THE USER TYPED CONTROL G?).
20806
20807
20808
20809
20810
20811
20812 115014      ;*****
20813                                     ;TEST 556 SOURCE MODES, MODE 1 (FL=0), TEST
20814                                     ;*****
20815      TS556:
20816 115014 012700 115064      MOV   #AACTP1,R0   ;SET UP TEST DATA IN BUFFER.
20817 115020 012710 147517      MOV   #147517,(R0)
20818 115024 012737 115040 037364  MOV   #AAC2,#STMP2
20819 115032 012737 115070 000004  MOV   #AAC10,#ERRVECT ;SET UP FOR TRAPS TO 4.
20820 115040 170110      AAC2:  LDFPS (R0)   ;TEST INSTRUCTION.
20821
20822 115042 170205      STFPS R5          ;GET FPS
20823
20824 115044 020027 115064      CMP   R0,#AACTP1  ;IS R0 CORRECT?
20825 115050 001007      BNE   AAC10       ;BR IF NOT.
20826 115052 022705 147517      CMP   #147517,R5 ;IS FPS CORRECT?
20827 115056 001004      BNE   AAC10       ;BR IF NOT.
20828 115060 000404      BR   AACDONE
20829
20830      ;TEST BUFFER AND DATA:
```

20831 115062 177777
20832 115064 147517
20833 115066 177777
20834 115070
20835 115070 104000
20836
20837 115072
20838 115072 004767 007474
20839
20840
20841
20842
20843
20844
20845
20846
20847
20848 115076
20849
20850
20851 115076 012700 115140
20852 115102 012710 145212
20853 115106 012737 115144 000004
20854
20855 115114 170120
20856
20857 115116 170205
20858
20859 115120 020027 115142
20860 115124 001007
20861 115126 022705 145212
20862 115132 001004
20863 115134 000404
20864
20865
20866
20867 115136 177777
20868 115140 177777
20869 115142 177777
20870

-1
AACTP1: 147517
-1
AAC10:
EMT ;
AACDONE:
JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;TEST 557 SOURCE MODES, MODE 2 (FL=0), TEST

TS557:

MOV #BBCTP1,R0 ;SET UP TEST DATA IN BUFFER.
MOV #145212,(R0)
MOV #BBC10,#ERRVECT ;SET UP FOR TRAPS TO 4.
BBC2: LDFPS (R0)+ ;TEST INSTRUCTION.
STFPS R5 ;GET FPS
CMP R0,#BBCTP1+2 ;IS R0 CORRECT?
BNE BBC10 ;BR IF NOT.
CMP #145212,R5 ;IS THE FPS CORRECT?
BNE BBC10 ;BR IF NOT.
BR BBCDONE

;TEST BUFFER AND DATA:
-1
BBCTP1: .WORD -1
-1

20871
20872 115144
20873 115144 104000
20874 115146
20875 115146 004767 007420
20876
20877
20878
20879
20880
20881
20882
20883
20884
20885 115152
20886
20887
20888 115152 012700 115234
20889 115156 012760 105252 177776
20890 115164 012737 115200 037364
20891 115172 012737 115244 000004
20892 115200 170140
20893 115202 170205
20894 115204 020027 115232
20895 115210 001015
20896 115212 022705 105252
20897 115216 001012
20898 115220 000412
20899
20900 115222 177777 177777 177777
20901 115230 177777
20902 115232 177777
20903 115234 177777 177777 177777
20904 115242 177777
20905 115244
20906 115244 104000
20907 115246
20908 115246 004767 007320
20909
20910
20911
20912
20913
20914
20915
20916 115252
20917 115252 012700 115340
20918 115256 012710 115330
20919 115262 012767 103456 000040
20920 115270 012737 115352 000004
20921 115276 170130
20922 115300 170205
20923 115302 020027 115342
20924 115306 001021
20925 115310 022705 103456
20926 115314 001016

```
BBC10: EMT ;
BBCDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

:*****
:TEST 560 SOURCE MODES, MODE 4 (FL=0), TEST
:*****
TS560:

MOV #DDCTP1+2,R0 ;SET UP THE TEST DATA BUFFER.
MOV #105252,-2(R0)
MOV #DDC2,@#STMP2
MOV #DDC10,@#ERRVEC
DDC2: LDFPS -(R0)
STFPS R5
CMP R0,#DDCTP1
BNE DDC10
CMP #105252,R5
BNE DDC10
BR DDCDONE

-1,-1,-1,-1
DDCTP1: -1
-1,-1,-1,-1

DDC10: EMT ;
DDCDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

:*****
:TEST 561 SOURCE MODES, MODE 3 (FL=0), TEST
:*****
TS561:

MOV #EECTP2,R0
MOV #EECTP1,(R0)
MOV #103456,EECTP1
MOV #EEC10,@#ERRVECT ;SET UP FOR TRAPS TO 4.
EEC2: LDFPS @(R0)+ ;TEST INSTRUCTION.
STFPS R5 ;GET THE FPS.
CMP R0,#EECTP2+2 ;IS R0 CORRECT?
BNE EEC10 ;BR IF NOT.
CMP #103456,R5 ;IS THE FPS CORRECT?
BNE EEC10 ;BR IF NOT.
```

```
20927 115316 000416 BR EEC DONE
20928
20929
20930 ;TEST BUFFER AND DATA:
20931 115320 177777 177777 177777 -1,-1,-1,-1
20932 115326 177777
20933 115330 177777 EECTP1: -1
20934 115332 177777 177777 177777 -1,-1,-1
20935 115340 115330 177777 177777 EECTP2: EECTP1,-1,-1,-1.
20936 115346 177777 000000
20937
20938 115352 EEC10:
20939 115352 104000 EEC DONE: EMT ;
20940 115354 JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND
20941 115354 004767 007212 ;SEE IF THE USER HAS EXPRESSED
20942 ;THE DESIRE TO CHANGE THE SOFTWARE
20943 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
20944 ;THE USER TYPED CONTROL G?).
20945
20946 ;*****
20947 ;TEST 562 SOURCE MODES, MODE 5 (FL=0), TEST
20948 ;*****
20949 TS562:
20950 115360 MOV #FFCTP2+2,R0 ;SET UP THE TEST DATA BUFFER.
20951 115364 012700 115444 MOV #FFCTP1,-2(R0)
20952 115372 012760 115432 177776 MOV #45412,@#FFCTP1
20953 115400 012737 115452 000004 MOV #FFC10,@#ERRVECT ;SET UP FOR TRAPS TO 4.
20954 115406 170150 FFC2: LDFPS @-(R0) ;TEST INSTRUCTION.
20955 115410 170205 STFPS R5 ;GET THE FPS.
20956 115412 020027 115442 CMP R0,#FFCTP2 ;IS R0 CORRECT?
20957 115416 001015 BNE FFC10 ;BR IF NOT.
20958 115420 022705 045412 CMP #45412,R5 ;IS THE FPS CORRECT?
20959 115424 001012 BNE FFC10 ;BR IF NOT.
20960 115426 000412 BR FFC DONE
20961
20962
20963 ;TEST BUFFER AND DATA:
20964 115430 177777 -1
20965 115432 177777 FFCTP1: -1
20966 115434 177777 177777 177777 -1,-1,-1
20967 115442 177777 177777 FFCTP2: FFCTP1,-1,-1,-1
20968 115450 177777
20969
20970 115452 FFC10:
20971 115452 104000 EMT ;
20972 115454 FFC DONE: JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND
20973 115454 004767 007112 ;SEE IF THE USER HAS EXPRESSED
20974 ;THE DESIRE TO CHANGE THE SOFTWARE
20975 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
20976 ;THE USER TYPED CONTROL G?).
20977
20978 ;*****
20979 ;TEST 563 SOURCE MODES, MODE 6 (FL=0), TEST
20980 ;*****
20981 115460 TS563:
20982 115460 012700 110333 MOV #GGCTP1-5201,R0 ;SET UP THE TEST DATA BUFFER.
```

```
20983 115464 012737 046543 115534      MOV      #46543,@#GGCTP1
20984 115472 005001                      CLR      R1
20985 115474 012737 115546 000004      MOV      #GGC10,@#ERRVECT ;SET UP FOR TRAPS TO 4.
20986 115502 170160 005201      GGC2:   LDFPS  5201(R0)          ;TEST INSTRUCTION.
20987 115506 170204                      STFPS   R4          ;GET THE FPS.
20988 115510 005701                      TST     R1          ;WAS PC CORRECT AFTER EXECUTION?
20989 115512 001015                      BNE     GGC10       ;BR IF NOT.
20990 115514 020027 110333      CMP     R0,#GGCTP1-5201 ;IS R0 CORRECT?
20991 115520 001012                      BNE     GGC10       ;BR IF NOT.
20992 115522 022704 046543      CMP     #46543,R4    ;IS THE FPS CORRECT?
20993 115526 001007                      BNE     GGC10       ;BR IF NOT.
20994 115530 000407                      BR      GGCDONE
20995
20996
20997                      ;TEST BUFFER AND DATA:
20998 115532 177777                      -1
20999 115534 177777 177777 177777      GGC1P1: -1,-1,-1,-1
21000 115542 177777                      -1
21001 115544 177777                      GGC10:
21002 115546                      GGC10:   EMT
21003 115546 104000                      GGCDONE:
21004 115550
21005 115550 004767 007016      JSR     PC,,RSET    ;GO INITIALIZE THE FPS AND STACK; AND
21006                      ;SEE IF THE USER HAS EXPRESSED
21007                      ;THE DESIRE TO CHANGE THE SOFTWARE
21008                      ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
21009                      ;THE USER TYPED CONTROL G?).
21010                      ;*****
21011                      ;TEST 564 SOURCE MODES, MODE 7 (FL=0), TEST
21012                      ;*****
21013 115554                      TS564:
21014 115554 012700 110445      MOV      #HHCTP2-5201,R0 ;SET UP THE TEST DATA BUFFER.
21015 115560 012760 115636 005201      MOV      #HHCTP1,5201(R0)
21016 115566 012737 004547 115636      MOV      #4547,@#HHCTP1
21017 115574 005001                      CLR      R1
21018 115576 012737 115656 000004      MOV      #HHC10,@#ERRVECT ;SET UP FOR TRAPS TO 4.
21019 115604 170170 005201      HHC2:   LDFPS  @5201(R0)          ;TEST INSTRUCTION.
21020 115610 170204                      STFPS   R4          ;GET THE FPS.
21021 115612 005701                      TST     R1          ;WAS PC CORRECT AFTER EXECUTION?
21022 115614 001020                      BNE     HHC10       ;BR IF NOT.
21023 115616 020027 110445      CMP     R0,#HHCTP2-5201 ;IS R0 CORRECT?
21024 115622 001015                      BNE     HHC10       ;BR IF NOT.
21025 115624 022704 004547      CMP     #4547,R4    ;IS THE FPS CORRECT?
21026 115630 001012                      BNE     HHC10       ;BR IF NOT.
21027 115632 000412                      BR      HHCDONE
21028
21029
21030                      ;TEST BUFFER AND DATA:
21031 115634 177777                      -1
21032 115636 177777 177777 177777      HHCTP1: .WORD -1,-1,-1,-1
21033 115644 177777                      177777 177777
21034 115646 177777 177777 177777      HHCTP2: .WORD -1,-1,-1,-1
21035 115654 177777
21036 115656                      HHC10:
21037 115656 104000                      HHC10:   EMT
21038 115660                      HHCDONE:
```

21039 115660 004767 006706
21040
21041
21042
21043
21044
21045
21046
21047
21048
21049

JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

:TEST 565 SOURCE MODES, MODE 2 GR7 (FL=1), TEST

TS565:

21050 115664
21051
21052 115664 012737 115722 000004
21053 115672 012700 000300
21054 115676 170100
21055 115700 005001
21056
21057 115702 177027
21058 115704 005201
21059 115706 005201
21060 115710 005201
21061 115712 005201
21062

MOV #IIC20,@#ERRVECT ;SET UP FOR TRAPS TO 4.
MOV #300,R0
LDFPS R0
CLR R1
IIC2: LDCLD (R7)+,ACO ;TEST INSTRUCTION.
5201
5201
5201
5201

21063 115714 020127 000003
21064 115720 001401
21065 115722
21066 115722 104000
21067

CMP R1,#3 ;WAS PC CORRECT AFTER EXECUTION?
BEQ IICDONE
IIC20: EMT ;

21068
21069 115724 004767 006642
21070
21071
21072
21073
21074
21075
21076

IICDONE: JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

:TEST 566 SOURCE MODES, MODE 2 (FL=1), TEST

TS566:

21077
21078
21079 115730
21080
21081 115730 012700 000300
21082 115734 170100
21083 115736 012700 116002
21084 115742 177020
21085
21086 115744 170204
21087 115746 012701 116012
21088 115752 012702 000200
21089 115756 170102
21090 115760 174011
21091 115762 020027 116006
21092 115766 001401
21093 115770 104000
21094

MOV #300,R0
LDFPS R0
MOV #TCCBF0,R0 ;SET UP THE TEST DATA BUFFER.
TCC2: LDCLD (R0)+,ACO ;TEST INSTRUCTION.
STFPS R4 ;GET THE FPS.
MOV #TCCBF1,R1 ;GET THE RESULT.
MOV #200,R2
LDFPS R2
STD ACO,(R1)
CMP R0,#TCCBF0+4 ;IS R0 CORRECT?
BEQ TCC3
EMT ;

21095 115772 022704 000300
21096 115776 001411
21097 116000 104000
21098
21099
21100
21101 116002 001234 067076 054321
21102 116010 012345
21103 116012 177777 177777 177777
21104 116020 177777
21105
21106 116022
21107 116022 004767 006544
21108
21109
21110
21111
21112
21113
21114
21115
21116
21117
21118 116026
21119
21120
21121
21122
21123 116026 004737 116720
21124
21125 116032 000000 000000
21126 116036 000000 000000
21127 116042 177777 177777
21128 116046 000000
21129 116050 000004
21130 116052 177777
21131
21132
21133 116054 004737 116720
21134
21135 116060 000000 177777
21136 116064 000000 000000
21137 116070 004177 177400
21138 116074 000000
21139 116076 000004
21140 116100 177777
21141
21142
21143 116102 004737 116720
21144
21145 116106 000000 000000
21146 116112 000000 000000
21147 116116 177777 177777
21148 116122 000100
21149 116124 000104
21150 116126 000004

TCC3: CMP #300,R4 ;IS THE FPS CORRECT?
BEQ TCCDONE
EMT ;

;TEST BUFFER AND DATA:
TCCBF0: .WORD 01234,67076,54321,012345
TCCBF1: -1,-1,-1,-1

TCCDONE:
JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;TEST 567 LDCIF AND LDCLF TEST

TS567:

;ZERO OPERAND FL=0
KKC1: JSR PC,@#LDCFSUB ;GO EXECUTE INSTRUCTION.
1\$: .WORD 0,0 ;FSRC OPERAND.
2\$: .WORD 0,0 ;EXPECTED RESULT.
3\$: .WORD -1,-1 ;ANTICIPATED ERRONEOUS RESULT.
4\$: 0 ;FPS BEFORE EXECUTION.
4 ;FPS AFTER EXECUTION.
-1 ;ANTICIPATED ERRONEOUS FPS.

;ZERO OPERAND FL=0
KKC2: JSR PC,@#LDCFSUB ;GO EXECUTE THE INSTRUCTION.
1\$: .WORD 0,-1 ;FSRC OPERAND.
2\$: .WORD 0,0 ;EXPECTED RESULT.
3\$: 4177,177400 ;ANTICIPATED ERRONEOUS RESULT.
4\$: 0 ;FPS BEFORE EXECUTION.
4 ;FPS AFTER EXECUTION.
-1 ;ANTICIPATED ERRONEOUS FPS.

;ZERO OPERAND FL=1
KKC3: JSR PC,@#LDCFSUB ;GO EXECUTE THE INSTRUCTION.
1\$: .WORD 0,0 ;FSRC OPERAND.
2\$: .WORD 0,0 ;EXPECTED RESULT.
3\$: .WORD -1,-1 ;ANTICIPATED ERRONEOUS RESULT.
4\$: 100 ;FPS BEFORE EXECUTION.
104 ;FPS AFTER EXECUTION.
4 ;ANTICIPATED ERRONEOUS FPS.

21207
21208 116362 004737 116720
21209 116366 000000 000001
21210 116372 040200 000000
21211 116376 034200 000000
21212 116402 000100
21213 116404 000100
21214 116406 177777
21215
21216 116410 004737 116720
21217 116414 000000 000252
21218 116420 042052 000000
21219 116424 036052 000000
21220 116430 000111
21221 116432 000100
21222 116434 177777
21223
21224 116436 004737 116720
21225 116442 140000 000000
21226 116446 147600 000000
21227 116452 047600 000000
21228 116456 000107
21229 116460 000110
21230 116462 177777
21231
21232 116464 004737 116720
21233 116470 177777 177777
21234 116474 140200 000000
21235 116500 150000 000000
21236 116504 000100
21237 116506 000110
21238 116510 177777
21239
21240 116512 004737 116720
21241 116516 125252 125252
21242 116522 147652 125253
21243 116526 047652 125253
21244 116532 000105
21245 116534 000110
21246 116536 177777
21247
21248 116540 004737 116720
21249 116544 077777 177500
21250 116550 047777 177777
21251 116554 047777 177776
21252 116560 000117
21253 116562 000100
21254 116564 177777
21255
21256 116566 004737 116720
21257 116572 040000 000100
21258 116576 047600 000001
21259 116602 047600 000000
21260 116606 000102
21261 116610 000100
21262 116612 177777

```
;OPERAND=1          FL=1
KKC13: JSR          PC,@#LDCFSUB ;GO EXECUTE THE INSTRUCTION.
1$:      .WORD      0,1          ;FSRC OPERAND.
2$:      .WORD      40200,0      ;EXPECTED RESULT.
3$:      .WORD      34200,0      ;ANTICIPATED ERRONEOUS RESULT.
4$:      100                ;FPS BEFORE EXECUTION.
          100                ;FPS AFTER EXECUTION.
          -1                 ;ANTICIPATED ERRONEOUS FPS.

;OPERAND=          FL=1
KKC14: JSR          PC,@#LDCFSUB ;GO EXECUTE THE INSTRUCTION.
1$:      .WORD      0,252        ;FSRC OPERAND.
2$:      .WORD      42052,0      ;EXPECTED RESULT.
3$:      .WORD      36052,0      ;ANTICIPATED ERRONEOUS RESULT.
4$:      111                ;FPS BEFORE EXECUTION.
          100                ;FPS AFTER EXECUTION.
          -1                 ;ANTICIPATED ERRONEOUS FPS.

;OPERAND=-40000,0   FL=1
KKC15: JSR          PC,@#LDCFSUB ;GO EXECUTE THE INSTRUCTION.
1$:      .WORD      -40000,0     ;FSRC OPERAND.
2$:      .WORD      147600,0     ;EXPECTED RESULT.
3$:      .WORD      47600,0     ;ANTICIPATED ERRONEOUS RESULT.
4$:      107                ;FPS BEFORE EXECUTION.
          110                ;FPS AFTER EXECUTION.
          -1                 ;ANTICIPATED ERRONEOUS FPS.

;OPERAND=-1,-1     FL=1
KKC16: JSR          PC,@#LDCFSUB ;GO EXECUTE THE INSTRUCTION.
1$:      .WORD      -1,-1        ;FSRC OPERAND.
2$:      .WORD      140200,0     ;EXPECTED RESULT.
3$:      .WORD      150000,0     ;ANTICIPATED ERRONEOUS RESULT.
4$:      100                ;FPS BEFORE EXECUTION.
          110                ;FPS AFTER EXECUTION.
          -1                 ;ANTICIPATED ERRONEOUS FPS.

;OPERAND=-PATTERN  FL=1,
KKC17: JSR          PC,@#LDCFSUB ;GO EXECUTE THE INSTRUCTION.
1$:      .WORD      125252,125252 ;FSRC OPERAND.
2$:      .WORD      147652,125253 ;EXPECTED RESULT.
3$:      .WORD      47652,125253 ;ANTICIPATED ERRONEOUS RESULT.
4$:      105                ;FPS BEFORE EXECUTION.
          110                ;FPS AFTER EXECUTION.
          -1                 ;ANTICIPATED ERRONEOUS FPS.

;OPERAND=77777,177500 FL=1,
KKC20: JSR          PC,@#LDCFSUB ;GO EXECUTE THE INSTRUCTION.
1$:      .WORD      77777,177500 ;FSRC OPERAND.
2$:      .WORD      47777,177777 ;EXPECTED RESULT.
3$:      .WORD      47777,177776 ;ANTICIPATED ERRONEOUS RESULT.
4$:      117                ;FPS BEFORE EXECUTION.
          100                ;FPS AFTER EXECUTION.
          -1                 ;ANTICIPATED ERRONEOUS FPS.

;OPERAND=40000,000100 FL=1,
KKC21: JSR          PC,@#LDCFSUB ;GO EXECUTE THE INSTRUCTION.
1$:      .WORD      40000,100    ;FSRC OPERAND.
2$:      .WORD      47600,1      ;EXPECTED RESULT.
3$:      .WORD      47600,0      ;ANTICIPATED ERRONEOUS RESULT.
4$:      102                ;FPS BEFORE EXECUTION.
          100                ;FPS AFTER EXECUTION.
          -1                 ;ANTICIPATED ERRONEOUS FPS.
```

21263
 21264 116614 004737 116720
 21265 116620 040000 000100
 21266 116624 047600 000000
 21267 116630 047600 000001
 21268 116634 000157
 21269 116636 000140
 21270 116640 177777
 21271
 21272 116642 004737 116720
 21273 116646 100000 000000
 21274 116652 144000 000000
 21275 116656 143600 000000
 21276 116662 000007
 21277 116664 000010
 21278 116666 177777
 21279
 21280 116670 004737 116720
 21281 116674 100000 000000
 21282 116700 150000 000000
 21283 116704 147600 000000
 21284 116710 000107
 21285 116712 000110
 21286 116714 177777
 21287 116716 000441
 21288
 21289
 21290
 21291
 21292
 21293
 21294
 21295
 21296
 21297
 21298
 21299
 21300
 21301
 21302
 21303
 21304
 21305
 21306
 21307
 21308
 21309
 21310
 21311
 21312
 21313
 21314
 21315
 21316
 21317
 21318

```

;OPERAND=40000,000100 FL=1, TRUNC MODE
KKC22: JSR PC,@#LDCFSUB ;GO EXECUTE THE INSTRUCTION.
1$: .WORD 40000,100 ;FSRC OPERAND.
2$: .WORD 47600,0 ;EXPECTED RESULT.
3$: .WORD 47600,1 ;ANTICIPATED ERRONEOUS RESULT.
4$: 157 ;FPS BEFORE EXECUTION.
    140 ;FPS AFTER EXECUTION.
    -1 ;ANTICIPATED ERRONEOUS FPS.
;OPERAND=100000,0 (MOST NEG #) FL=0
KKC23: JSR PC,@#LDCFSUB ;GO EXECUTE THE INSTRUCTION.
1$: .WORD 100000,0 ;FSRC OPERAND.
2$: .WORD 144000,0 ;EXPECTED RESULT.
3$: .WORD 143600,0 ;ANTICIPATED ERRONEOUS RESULT.
4$: 7 ;FPS BEFORE EXECUTION.
    10 ;FPS AFTER EXECUTION.
    -1 ;ANTICIPATED ERRONEOUS FPS.
;OPERAND=100000,0 FL=1
KKC24: JSR PC,@#LDCFSUB ;GO EXECUTE THE INSTRUCTION.
1$: .WORD 100000,0 ;FSRC OPERAND.
2$: .WORD 150000,0 ;EXPECTED RESULT.
3$: .WORD 147600,0 ;ANTICIPATED ERRONEOUS RESULT.
4$: 107 ;FPS BEFORE EXECUTION.
    110 ;FPS AFTER EXECUTION.
    -1 ;ANTICIPATED ERRONEOUS FPS.
6$: BR KKCDONE
  
```

;THIS SUBROUTINE, LDCFSUB, IS USED TO SET UP THE OPERANDS, EXECUTE
 ;THE LDCIF OR LDCLF INSTRUCTION AND CHECK THE RESULTS. A CALL
 ;TO IT IS MADE THUS:

```

:
:         JSR     PC,@#LDCFSUB
:         ACARG: .WORD X,X           ;AC OPERAND
:         RES:   .WORD X,X           ;EXPECTED RESULT
:         ERRES: .WORD X,X           ;ERROR RESULT
:         FPSB:  .WORD X             ;FPS BEFORE EXECUTION
:         FPSA:  .WORD X             ;FPS AFTER EXECUTION
:         ERFPS: .WORD X             ;ERROR FPS
:         ERR1:  ERROR X             ;DATA ERROR
:         BR     CONT
:         ERR2:  ERROR X             ;FPS ERROR
:         CONT:  ;RETURN ADDRESS
  
```

;THE OPERANDS ARE SET UP (USING ACO AS THE ACCUMULATOR). THEN
 ;THE LDCIF OR LDCLF INSTRUCTION IS EXECUTED.
 ;THE RESULT IS CHECKED AGAINST RES. IF THE RESULT IS CORRECT THEN THE FPS IS
 ;COMPARED WITH FPSA IF THIS TOO IS CORRECT LDCFSUB RETURNS CONTROL
 ;TO THE CALLING ROUTINE AT CONT. IF THE FPS IS BAD LDCFSUB WILL
 ;COMPARE IT TO ERROR FPS. IF THIS MATCHES THEN LDCFSUB WILL RETURN
 ;TO THE ERROR CALL AT ERR2, OTHERWISE LDCFSUB ITSELF
 ;REPORTS THIS FAILURE AND THEN RETURNS TO CONT. IF THE RESULT OF THE
 ;LDCIF OR LDCLF IS INCORRECT, THE INCORRECT RESULT IS COMPARED WITH THE
 ;ANTICIPATED FAILING DATA PATTERN, ERRES. IF THE FAILURE IN
 ;THE RESULT WAS ANTICIPATED CORRECTLY TO BE ERRES THEN LDCFSUB
 ;WILL TRANSFER CONTROL TO THE ERROR CALL AT ERR1. OTHERWISE THE
 ;RESULT WAS INCORRECT BUT WAS NOT ANTICIPATED AND LDCFSUB
 ;REPORT THE FAILURE AFTER WHICH CONTROL WILL BE PASSED TO CONT.

```
21319
21320 116720 012601
21321 116722 016100 000014
21322 116726 170100
21323 116730 010100
21324
21325 116732 177010
21326
21327 116734 170204
21328 116736 012700 117012
21329 116742 012702 000200
21330 116746 170102
21331 116750 174010
21332
21333 116752 012702 117012
21334 116756 010100
21335 116760 062700 000004
21336 116764 012703 000002
21337 116770 022022
21338 116772 001006
21339 116774 077303
21340
21341 116776 026104 000016
21342 117002 001002
21343 117004 000161 000022
21344 117010
21345 117010 104000
21346
21347
21348 117012 000000 000000 000000
21349 117020 000000
21350
21351 117022
21352 117022 004767 005544
21353
21354
21355
21356
21357
21358
21359
21360
21361
21362 117026
21363
21364 117026 004737 117504
21365 117032 000000 000000
21366 117036 000000 000000 000000
21367 117044 000000
21368 117046 177777 177777 177777
21369 117054 177777
21370 117056 000213
21371 117060 000204
21372 117062 177777
21373
21374 117064 004737 117504

LDCFSUB:  MOV      (SP)+,R1      ;GET A POINTER TO THE ARGUMENTS.
          MOV      14(R1),R0    ;SET THE FPS.
          LDFPS   R0
          MOV      R1,R0

1$:      LDCIF   (R0),ACO      ;TEST INSTRUCTION LDCIF OR LDCLF.

          STFPS   R4          ;GET FPS.
          MOV      #LDCT,R0    ;GET THE RESULT.
          MOV      #200,R2
          LDFPS   R2
          STD     ACO,(R0)

          MOV      #LDCT,R2    ;SEE IF THE RESULT WAS CORRECT.
          MOV      R1,R0
          ADD     #4,R0
          MOV      #2,R3
2$:      CMP     (R0)+,(R2)+    ;BR IF INCORRECT.
          BNE     10$
          SUB     R3,2$

          CMP     16(R1),R4    ;SEE IF THE FPS WAS CORRECT.
          BNE     10$         ;BR IF INCORRECT.
3$:      JMP     22(R1)       ;RETURN.
10$:     EMT                    ;

;DATA BUFFER:
LDCT:    .WORD  0,0,0,0

KKCDONE: JSR     PC,.RSET     ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;*****
;TEST 570      LDCID AND LDCLD TEST
;*****
TS570:
;OPERAND=0      FL=0,   FD=1
LLC1:  JSR     PC,#LDCDSUB   ;GO EXECUTE THE INSTRUCTION.
1$:    .WORD  0,0           ;FSRC OPERAND.
2$:    .WORD  0,0,0,0      ;EXPECTED RESULT.
3$:    .WORD  -1,-1,-1,-1  ;ANTICIPATED ERRONEOUS RESULT.
4$:    213                ;FPS BEFORE EXECUTION.
        204                ;FPS AFTER EXECUTION.
        -1                 ;ANTICIPATED ERRONEOUS FPS.
;OPERAND=0      FL=0,   FD=1
LLC2:  JSR     PC,#LDCDSUB   ;GO EXECUTE THE INSTRUCTION.
```

21375	117070	000000	177777		1\$:	.WORD	0,-1		:FSRC OPERAND.
21376	117074	000000	000000	000000	2\$:	.WORD	0,0,0,0		:EXPECTED RESULT.
21377	117102	000000							
21378	117104	004177	177400	000000	3\$:	.WORD	4177,177400,0,0		:ANTICIPATED ERRONEOUS RESULT.
21379	117112	000000							
21380	117114	000200			4\$:	200			:FPS BEFORE EXECUTION.
21381	117116	000204				204			:FPS AFTER EXECUTION.
21382	117120	177777				-1			:ANTICIPATED ERRONEOUS FPS.
21383									
21384	117122	004737	117504						
21385	117126	000000	000000		LLC3:	JSR	PC, @#LDCDSUB		:GO EXECUTE THE INSTRUCTION.
21386	117132	000000	000000	000000	1\$:	.WORD	0,0		:FSRC OPERAND.
21387	117140	000000			2\$:	.WORD	0,0,0,0		:EXPECTED RESULT.
21388	117142	177777	177777	177777	3\$:	.WORD	-1,-1,-1,-1		:ANTICIPATED ERRONEOUS RESULT.
21389	117150	177777							
21390	117152	000211			4\$:	211			:FPS BEFORE EXECUTION.
21391	117154	000204				204			:FPS AFTER EXECUTION.
21392	117156	177777				-1			:ANTICIPATED ERRONEOUS FPS.
21393									
21394	117160	004737	117504						
21395	117164	040000	000000		LLC4:	JSR	PC, @#LDCDSUB		:GO EXECUTE THE INSTRUCTION.
21396	117170	043600	000000	000000	1\$:	.WORD	4000,0		:FSRC OPERAND.
21397	117176	000000			2\$:	.WORD	43600,0,0,0		:EXPECTED RESULT.
21398	117200	047600	000000	000000	3\$:	.WORD	47600,0,0,0		:ANTICIPATED ERRONEOUS RESULT.
21399	117206	000000							
21400	117210	000217			4\$:	217			:FPS BEFORE EXECUTION.
21401	117212	000200				200			:FPS AFTER EXECUTION.
21402	117214	177777				-1			:ANTICIPATED ERRONEOUS FPS.
21403									
21404	117216	004737	117504						
21405	117222	140000	000000		LLC5:	JSR	PC, @#LDCDSUB		:GO EXECUTE THE INSTRUCTION.
21406	117226	143600	000000	000000	1\$:	.WORD	-4000,0		:FSRC OPERAND.
21407	117234	000000			2\$:	.WORD	143600,0,0,0		:EXPECTED RESULT.
21408	117236	043600	000000	000000	3\$:	.WORD	43600,0,0,0		:ANTICIPATED ERRONEOUS RESULT.
21409	117244	000000							
21410	117246	000200			4\$:	200			:FPS BEFORE EXECUTION.
21411	117250	000210				210			:FPS AFTER EXECUTION.
21412	117252	177777				-1			:ANTICIPATED ERRONEOUS FPS.
21413									
21414	117254	004737	117504						
21415	117260	040000	000000						
21416	117264	047600	000000	000000	LLC6:	JSR	PC, @#LDCDSUB		:GO EXECUTE THE INSTRUCTION.
21417	117272	000000			1\$:	.WORD	4000,0		:FSRC OPERAND.
21418	117274	043600	000000	000000	2\$:	.WORD	47600,0,0,0		:EXPECTED RESULT.
21419	117302	000000			3\$:	.WORD	43600,0,0,0		:ANTICIPATED ERRONEOUS RESULT.
21420	117304	000317							
21421	117306	000300			4\$:	317			:FPS BEFORE EXECUTION.
21422	117310	177777				300			:FPS AFTER EXECUTION.
21423						-1			:ANTICIPATED ERRONEOUS FPS.
21424	117312	004737	117504						
21425	117316	000000	000001						
21426	117322	040200	000000	000000					
21427	117330	000000							
21428	117332	034200	000000	000000	LLC7:	JSR	PC, @#LDCDSUB		:GO EXECUTE THE INSTRUCTION.
21429	117340	000000			1\$:	.WORD	0,1		:FSRC OPERAND.
21430	117342	000300			2\$:	.WORD	40200,0,0,0		:EXPECTED RESULT.

```

21431 117344 000300          300          ;FPS AFTER EXECUTION.
21432 117346 177777          -1          ;ANTICIPATED ERRONEOUS FPS.
21433          ;OPERAND=77777,177777 FL=1      FD=1
21434 117350 004737 117504    LLC10: JSR   PC,#LDCDSUB ;GO EXECUTE THE INSTRUCTION.
21435 117354 077777 177777    1$:  .WORD  77777,177777 ;FSRC OPERAND.
21436 117360 047777 177777 177000 2$:  .WORD  47777,177777,177000,0 ;EXPECTED RESULT.
21437 117366 000000
21439 117370 177777 177777 177777 3$:  .WORD  -1,-1,-1,-1          ;ANTICIPATED ERRONEOUS RESULT.
21439 117376 177777
21440 117400 000317          317          ;FPS BEFORE EXECUTION.
21441 117402 000300          300          ;FPS AFTER EXECUTION.
21442 117404 177777          -1          ;ANTICIPATED ERRONEOUS FPS.
21443          ;OPERAND=-PATTERN          FL=1      FD=1
21444
21445 117406 004767 000072    LLC11: JSR   PC,LDCDSUB ;GO EXECUTE THE INSTRUCTION.
21446 117412 177777 177526    1$:  .WORD  -1,-252        ;FSRC OPERAND.
21447 117416 142052 000000 000000 2$:  .WORD  142052,0,0,0        ;EXPECTED RESULT.
21448 117424 000000
21449 117426 136052 000000 000000 3$:  .WORD  136052,0,0,0        ;ANTICIPATED ERRONEOUS RESULT.
21450 117434 000000
21451 117436 000307          307          ;FPS BEFORE EXECUTION.
21452 117440 000310          310          ;FPS AFTER EXECUTION.
21453 117442 177777          -1          ;ANTICIPATED ERRONEOUS FPS.
21454          ;OPERAND=PATTERN          FL=1      FD=1 FT=1
21455 117444 004767 000034    LLC12: JSR   PC,LDCDSUB ;GO EXECUTE THE INSTRUCTION.
21456 117450 012345 067012    1$:  .WORD  12345,67012      ;FSRC OPERAND.
21457 117454 047247 025560 050000 2$:  .WORD  47247,025560,050000,0 ;EXPECTED RESULT.
21458 117462 000000
21459 117464 177777 177777 177777 3$:  .WORD  -1,-1,-1,-1        ;ANTICIPATED ERRONEOUS RESULT.
21460 117472 177777
21461 117474 000352          352          ;FPS BEFORE EXECUTION.
21462 117476 000340          340          ;FPS AFTER EXECUTION.
21463 117500 177777          -1          ;ANTICIPATED ERRONEOUS FPS.
21464 117502 000435          6$:  BR     LLCDONE
21465
21466          ;THIS SUBROUTINE, LDCDSUB, IS USED TO SET UP THE OPERANDS, EXECUTE
21467          ;THE LDCID OR LDCLD INSTRUCTION AND CHECK THE RESULTS. A CALL
21468          ;TO IT IS MADE THUS:
21469          :
21470          :           JSR   PC,#LDCDSUB
21471          :           ACARG: .WORD  X,X          ;AC OPERAND
21472          :           RES:   .WORD  X,X,X,X      ;EXPECTED RESULT
21473          :           ERRES: .WORD  X,X,X,X      ;ERROR RESULT
21474          :           FPSB:  .WORD  X            ;FPS BEFORE EXECUTION
21475          :           FPSA:  .WORD  X            ;FPS AFTER EXECUTION
21476          :           ERFPS: .WORD  X            ;ERROR FPS.
21477          :           ERR1:  ERROR X            ;DATA ERROR.
21478          :           BR     CONT
21479          :           ERR2:  ERROR X            ;FPS ERROR.
21480          :           CONT:          ;RETURN ADDRESS
21481          :
21482          ;THE OPERANDS ARE SET UP (USING ACO AS THE ACCUMULATOR). THEN
21483          ;THE LDCID OR LDCLD INSTRUCTION IS EXECUTED.
21484          ;THE RESULT IS CHECKED AGAINST RES. IF THE RESULT IS CORRECT THEN THE FPS IS
21485          ;COMPARED WITH FPSA IF THIS TOO IS CORRECT LDCDSUB RETURNS CONTROL
21486          ;TO THE CALLING ROUTINE AT CONT. IF THE FPS IS BAD LDCDSUB

```

```

21487 ;COMPARE IT TO ERROR FPS. IF THIS MATCHES THEN LDCDSUB WILL RETURN
21488 ;TO THE ERROR CALL AT ERR2, OTHERWISE LDCDSUB ITSELF
21489 ;REPORTS THIS FAILURE AND THEN RETURNS TO CONT. IF THE RESULT OF THE
21490 ;LDCID OR LDCLD IS INCORRECT, THE INCORRECT RESULT IS COMPARED WITH THE
21491 ;ANTICIPATED FAILING DATA PATTERN, ERRES. IF THE FAILURE IN
21492 ;THE RESULT WAS ANTICIPATED CORRECTLY TO BE ERRES THEN LDCDSUB
21493 ;WILL TRANSFER CONTROL TO THE ERROR CALL AT ERR1. OTHERWISE THE
21494 ;RESULT WAS INCORRECT BUT WAS NOT ANTICIPATED AND LDCDSUB WILL
21495 ;REPORT THE FAILURE AFTER WHICH CONTROL WILL BE PASSED TO CONT.
21496
21497 117504 012601 LDCDSUB: MOV (SP)+,R1 ;GET A POINTER TO THE ARGUMENTS.
21498 117506 016100 000024 MOV 24(R1),R0 ;SET THE FPS.
21499 117512 170100 LDFPS R0
21500 117514 010100 MOV R1,R0
21501 117516 177010 1$: LDCID (R0),ACO ;TEST INSTRUCTION, LDCID OR LDCLD.
21502
21503 117520 170204 STFPS R4 ;GET FPS.
21504 117522 012700 117012 MOV #LDCT,R0 ;GET THE RESULT.
21505 117526 012702 000200 MOV #200,R2
21506 117532 170102 LDFPS R2
21507 117534 174010 STD ACO,(R0)
21508
21509 ;SEE IF THE RESULT IS CORRECT.
21510 117536 012702 117012 MOV #LDCT,R2
21511 117542 010100 MOV R1,R0
21512 117544 062700 000004 ADD #4,R0
21513 117550 012703 000002 MOV #2,R3
21514 117554 022022 2$: CMP (R0)+,(R2)+
21515 117556 001006 BNE 10$ ;BR IF INCORRECT.
21516 117560 077303 SOB R3,2$
21517
21518 117562 026104 000026 CMP 26(R1),R4 ;IS THE FPS CORRECT?
21519 117566 001002 BNE 10$ ;BR IF INCORRECT.
21520 117570 000161 000032 3$: JMP 32(R1) ;RETURN.
21521 117574 10$:
21522 117574 104000 EMT ;
21523
21524 117576 LLCDONE:
21525 117576 004767 004770 JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
21526 ;SEE IF THE USER HAS EXPRESSED
21527 ;THE DESIRE TO CHANGE THE SOFTWARE
21528 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
21529 ;THE USER TYPED CONTROL G?).
21530
21531
21532 ;*****
21533 ;TEST 571 LDEXP TEST
21534 ;*****
21535 117602 T571:
21536
21537 ;NON-ZERO RES. VALID EXPON=210 (EXCESS 200)=10
21538 117602 004767 001136 MMCI: JSR PC,LXSUB ;GO EXECUTE THE INSTRUCTION.
21539 117606 012345 067012 034567 1$: .WORD 12345,67012,34567,012345 ;ACO OPERAND.
21540 117614 012345
21541 117616 000010 2$: .WORD 10 ;EXPONENT OPERAND.
21542 117620 042145 067012 034567 3$: .WORD 42145,67012,34567,012345 ;EXPECTED RESULT.

```


21543	117626	012345							
21544	117630	002145	067012	034567	4\$:	.WORD	2145,67012,34567,012345		;ANTICIPATED ERRONEOUS RESULT.
21545	117636	012345							
21546	117640	047217			5\$:	47217			;FPS BEFORE EXECUTION.
21547	117642	047200				47200			;FPS AFTER EXECUTION.
21548	117644	147200				147200			;ANTICIPATED ERRONEOUS FPS.
21549	117646	177777				-1			;EXPECTED FEC.
21550									
21551	117650	004737	120744						
21552	117654	123456	070123	045670	1\$:	.WORD	123456,70123,45670,123456		;ACO OPERAND.
21553	117662	123456							
21554	117664	000177			2\$:	.WORD	177		;EXPONENT OPERAND.
21555	117666	177656	070123	045670	3\$:	.WORD	177656,70123,45670,123456		;EXPECTED RESULT.
21556	117674	123456							
21557	117676	137656	070123	045670	4\$:	.WORD	137656,70123,45670,123456		;ANTICIPATED ERRONEOUS RESULT.
21558	117704	123456							
21559	117706	047207			5\$:	47207			;FPS BEFORE EXECUTION.
21560	117710	047210				47210			;FPS AFTER EXECUTION.
21561	117712	147210				147210			;ANTICIPATED ERRONEOUS FPS.
21562	117714	177777				-1			;EXPECTED FEC.
21563									
21564	117716	004737	120744						
21565	117722	073261	057645	043323	1\$:	.WORD	73261,057645,43323,101760		;ACO OPERAND.
21566	117730	101760							
21567	117732	000056			2\$:	.WORD	56		;EXPONENT OPERAND.
21568	117734	053461	057645	043323	3\$:	.WORD	53461,057645,43323,101760		;EXPECTED RESULT.
21569	117742	101760							
21570	117744	177777	177777	177777	4\$:	.WORD	-1,-1,-1,-1		;ANTICIPATED ERRONEOUS RESULT.
21571	117752	177777							
21572	117754	047200			5\$:	47200			;FPS BEFORE EXECUTION.
21573	117756	047200				47200			;FPS AFTER EXECUTION.
21574	117760	147200				147200			;ANTICIPATED ERRONEOUS FPS.
21575	117762	177777				-1			;EXPECTED FEC.
21576									
21577	117764	004737	120744						
21578	117770	012223	024252	062720	1\$:	.WORD	12223,24252,62720,21222		;ACO OPERAND.
21579	117776	021222							
21580	120000	177627			2\$:	.WORD	-151		;EXPONENT OPERAND.
21581	120002	005623	024252	062720	3\$:	.WORD	5623,24252,62720,21222		;EXPECTED RESULT.
21582	120010	021222							
21583	120012	177777	177777	177777	4\$:	.WORD	-1,-1,-1,-1		;ANTICIPATED ERRONEOUS RESULT.
21584	120020	177777							
21585	120022	047200			5\$:	47200			;FPS BEFORE EXECUTION.
21586	120024	047200				47200			;FPS AFTER EXECUTION.
21587	120026	147200				147200			;ANTICIPATED ERRONEOUS FPS.
21588	120030	177777				-1			;EXPECTED FEC.
21589									
21590									
21591	120032	004737	120744						
21592	120036	030131	032334	035363	1\$:	.WORD	30131,32334,35363,73031		;ACO OPERAND.
21593	120044	073031							
21594	120046	177600			2\$:	.WORD	-200		;EXPONENT OPERAND.
21595	120050	000131	032334	035363	3\$:	.WORD	00131,32334,35363,73031		;EXPECTED RESULT.
21596	120056	073031							
21597	120060	000000	000000	000000	4\$:	.WORD	0,0,0,0		;ANTICIPATED ERRONEOUS RESULT.
21598	120066	000000							


```
21655 120326 177777 -1 ;EXPECTED FEC.
21656 ;EXP=-1601 (EXCESS 200)=-2001 (OCT), FIV=1
21657 120330 004737 120744 MMC12: JSR PC,@#LDXSUB ;GO EXECUTE THE INSTRUCTION.
21658 120334 001020 030405 006070 1$: .WORD 01020,30405,06070,00102 ;ACO OPERAND.
21659 120342 000102
21660 120344 175777 2$: .WORD -2001 ;EXPONENT OPERAND.
21661 120346 037620 030405 006070 3$: .WORD 37620,30405,06070,00102 ;EXPECTED RESULT.
21662 120354 000102
21663 120356 000000 000000 000000 4$: .WORD 0,0,0,0 ;ANTICIPATED ERRONEOUS RESULT.
21664 120364 000000
21665 120366 042200 5$: 42200 ;FPS BEFORE EXECUTION.
21666 120370 142200 142200 ;FPS AFTER EXECUTION.
21667 120372 042204 42204 ;ANTICIPATED ERRONEOUS FPS.
21668 120374 000012 12 ;EXPECTED FEC.
21669 ;EXP=1206 (EXCESS 200)=1006 (OCT) FIV=1
21670 120376 004737 120744 MMC13: JSR PC,@#LDXSUB ;GO EXECUTE THE INSTRUCTION.
21671 120402 012131 014151 016171 1$: .WORD 12131,14151,16171,10111 ;ACO OPERAND.
21672 120410 010111
21673 120412 001006 2$: .WORD 1006 ;EXPONENT OPERAND.
21674 120414 041531 014151 016171 3$: .WORD 41531,14151,16171,10111 ;EXPECTED RESULT.
21675 120422 010111
21676 120424 000000 000000 000000 4$: .WORD 0,0,0,0 ;ANTICIPATED ERRONEOUS RESULT.
21677 120432 000000
21678 120434 041200 5$: 41200 ;FPS BEFORE EXECUTION.
21679 120436 141202 141202 ;FPS AFTER EXECUTION.
21680 120440 041204 41204 ;ANTICIPATED ERRONEOUS FPS.
21681 120442 000010 10 ;EXPECTED FEC.
21682 ;EXP=16315 (EXCESS 200)=16115 (OCT) FIV=0
21683 120444 004737 120744 MMC14: JSR PC,@#LDXSUB ;GO EXECUTE THE INSTRUCTION.
21684 120450 027262 025242 023222 1$: .WORD 27262,25242,23222,21202 ;ACO OPERAND.
21685 120456 021202
21686 120460 016115 2$: .WORD 16115 ;EXPONENT OPERAND.
21687 120462 000000 000000 000000 3$: .WORD 0,0,0,0 ;EXPECTED RESULT.
21688 120470 000000
21689 120472 063262 025242 023222 4$: .WORD 63262,25242,23222,21202 ;ANTICIPATED ERRONEOUS RESULT.
21690 120500 021202
21691 120502 046200 5$: 46200 ;FPS BEFORE EXECUTION.
21692 120504 046206 46206 ;FPS AFTER EXECUTION.
21693 120506 146202 146202 ;ANTICIPATED ERRONEOUS FPS.
21694 120510 177777 -1 ;EXPECTED FEC.
21695 ;EXP=11011 (EXCESS 200)=10611 (OCT) FIV=1
21696
21697 120512 004737 120744 MMC15: JSR PC,@#LDXSUB ;GO EXECUTE THE INSTRUCTION.
21698 120516 030313 032333 034353 1$: .WORD 30313,32333,34353,36373 ;ACO OPERAND.
21699 120524 036373
21700 120526 010611 2$: .WORD 10611 ;EXPONENT OPERAND.
21701 120530 002313 032333 034353 3$: .WORD 2313,32333,34353,36373 ;EXPECTED RESULT.
21702 120536 036373
21703 120540 000000 000000 000000 4$: .WORD 0,0,0,0 ;ANTICIPATED ERRONEOUS RESULT.
21704 120546 000000
21705 120550 041200 5$: 41200 ;FPS BEFORE EXECUTION.
21706 120552 141202 141202 ;FPS AFTER EXECUTION.
21707 120554 041204 41204 ;ANTICIPATED ERRONEOUS FPS.
21708 120556 000010 10 ;EXPECTED FEC.
21709 ;EXP=17123 (EXCESS 200)=16723 (OCT) FIV=0
21710
```

```
21711 120560 004737 120744 MMC16: JSR PC,@#LDXSUB :GO EXECUTE THE INSTRUCTION.
21712 120564 040414 042434 044454 1$: .WORD 40414,42434,44454,46474 ;ACO OPERAND.
21713 120572 046474
21714 120574 016723 2$: .WORD 16723 ;EXPONENT OPERAND.
21715 120576 000000 000000 000000 3$: .WORD 0,0,0,0 ;EXPECTED RESULT.
21716 120604 000000
21717 120606 024614 042434 044454 4$: .WORD 24614,42434,44454,46474 ;ANTICIPATED ERRONEOUS RESULT.
21718 120614 046474
21719 120616 046200 5$: 46200 ;FPS BEFORE EXECUTION.
21720 120620 046206 46206 ;FPS AFTER EXECUTION.
21721 120622 146202 146202 ;ANTICIPATED ERRONEOUS FPS.
21722 120624 177777 -1 ;EXPECTED FEC.
21723 :EXP= 254 (OCT)= 454 (EXCESS 200) FIV=1
21724
21725 120626 004737 120744 MMC17: JSR PC,@#LDXSUB :GO EXECUTE THE INSTRUCTION.
21726 120632 050515 052535 054555 1$: .WORD 50515,52535,54555,56575 ;ACO OPERAND.
21727 120640 056575
21728 120642 000254 2$: .WORD 254 ;EXPONENT OPERAND.
21729 120644 013115 052535 054555 3$: .WORD 13115,52535,54555,56575 ;EXPECTED RESULT.
21730 120652 056575
21731 120654 000000 000000 000000 4$: .WORD 0,0,0,0 ;ANTICIPATED ERRONEOUS RESULT.
21732 120662 000000
21733 120664 041200 5$: 41200 ;FPS BEFORE EXECUTION.
21734 120666 141202 141202 ;FPS AFTER EXECUTION.
21735 120670 041204 41204 ;ANTICIPATED ERRONEOUS FPS.
21736 120672 000010 10 ;EXPECTED FEC.
21737 :EXP= 313 (OCT)= 513(EXCESS 200) FIV=0
21738
21739 120674 004737 120744 MMC20: JSR PC,@#LDXSUB :GO EXECUTE THE INSTRUCTION.
21740 120700 060616 062636 064656 1$: .WORD 60616,62636,64656,66676 ;ACO OPERAND.
21741 120706 066676
21742 120710 000313 2$: .WORD 313 ;EXPONENT OPERAND.
21743 120712 000000 000000 000000 3$: .WORD 0,0,0,0 ;EXPECTED RESULT.
21744 120720 000000
21745 120722 022616 062636 064656 4$: .WORD 22616,62636,64656,66676 ;ANTICIPATED ERRONEOUS RESULT.
21746 120730 066676
21747 120732 046200 5$: 46200 ;FPS BEFORE EXECUTION.
21748 120734 046206 46206 ;FPS AFTER EXECUTION.
21749 120736 146202 146202 ;ANTICIPATED ERRONEOUS FPS.
21750 120740 177777 -1 ;EXPECTED FEC.
21751 120742 000457 BR MMCDONE
```

```
: THIS SUBROUTINE, LDXSUB, IS USED TO SET UP THE OPERANDS, EXECUTE
: THE LDEXP INSTRUCTION AND CHECK THE RESULTS. A CALL
: TO IT IS MADE THUS:
```

```
:
: JSR PC,@#LDXSUB
: ACARG: .WORD X,X,X,X ;AC OPERAND
: EXP: .WORD X ;EXPONENT
: RES: .WORD X,X,X,X ;EXPECTED RESULT
: ERRES: .WORD X,X,X,X ;ERROR RESULT
: FPSB: .WORD X ;FPS BEFORE EXECUTION
: FPSA: .WORD X ;FPS AFTER EXECUTION
: ERFPS: .WORD X ;ERROR FPS.
: FEC: .WORD X ;EXPECTED FEC
: ERR1: ERROR X ;DATA ERROR.
```

21767
21768
21769
21770
21771
21772
21773
21774
21775
21776
21777
21778
21779
21780
21781
21782
21783
21784
21785
21786 120744 012601
21787 120746 012700 000200
21788 120752 170100
21789 120754 010100
21790 120756 172410
21791 120760 016100 000032
21792 120764 170100
21793 120766 010100
21794 120770 062700 000010
21795
21796 120774 176410
21797
21798 120776 170204
21799 121000 170305
21800 121002 012700 000200
21801 121006 170100
21802 121010 012700 121072
21803 121014 174010
21804 121016 012702 121072
21805 121022 010103
21806 121024 062703 000012
21807 121030 012700 000004
21808 121034 022223
21809 121036 001014
21810 121040 077003
21811 121042 020461 000034
21812 121046 001010
21813 121050 005761 000034
21814 121054 100003
21815 121056 020561 000040
21816 121062 001002
21817
21818 121064 000161 000042
21819 121070
21820 121070 104000
21821
21822

BR CONT
ERR2: ERROR X ;FPS ERROR.
CONT: ;RETURN ADDRESS
:THE OPERANDS ARE SET UP (USING ACO AS THE ACCUMULATOR). THEN
:THE LDEXP INSTRUCTION IS EXECUTED.
:THE RESULT IS CHECKED AGAINST RES. IF THE RESULT IS CORRECT THEN THE FPS IS
:COMPARED WITH FPSA IF THIS TOO IS CORRECT LDXSUB RETURNS CONTROL
:TO THE CALLING ROUTINE AT CONT. IF THE FPS IS BAD LDXSUB
:COMPARE IT TO ERROR FPS. IF THIS MATCHES THEN LDXSUB WILL RETURN
:TO THE ERROR CALL AT ERR2, OTHERWISE LDXSUB ITSELF
:REPORTS THIS FAILURE AND THEN RETURNS TO CONT. IF THE RESULT OF THE
:LDEXP IS INCORRECT, THE INCORRECT RESULT IS COMPARED WITH THE
:ANTICIPATED FAILING DATA PATTERN, ERRES. IF THE FAILURE IN
:THE RESULT WAS ANTICIPATED CORRECTLY TO BE ERRES THEN LDXSUB
:WILL TRANSFER CONTROL TO THE ERROR CALL AT ERR1. OTHERWISE THE
:RESULT WAS INCORRECT BUT WAS NOT ANTICIPATED AND LDXSUB WILL
:REPORT THE FAILURE AFTER WHICH CONTROL WILL BE PASSED TO CONT.
LDXSUB: MOV (SP)+,R1 ;GET A POINTER TO THE ARGUMENTS.
MOV #200,R0 ;LOAD THE ACO OPERAND.
LDFPS R0
MOV R1,R0
LDD (R0),ACO
MOV 32(R1),R0 ;SET UP THE FPS.
LDFPS R0
MOV R1,R0
ADD #10,R0
1\$: LDEXP (R0),ACO ;TEST INSTRUCTION.
STFPS R4 ;GET THE FPS.
STST R5 ;GET THE FEC.
MOV #200,R0 ;GET THE RESULT.
LDFPS R0
MOV #LDXT,R0
STD ACO,(R0)
MOV #LDXT,R2 ;SEE IF THE RESULT WAS CORRECT.
MOV R1,R3
ADD #12,R3
MOV #4,R0
2\$: CMP (R2)+,(R3)+ ;BRANCH IF NOT CORRECT.
BNE 10\$
SOB R0,2\$
CMP R4,34(R1) ;SEE IF THE FPS WAS CORRECT.
BNE 10\$;BRANCH IF NOT CORRECT.
TST 34(R1)
BPL 3\$
CMP R5,40(R1) ;SEE IF THE FEC WAS CORRECT.
BNE 10\$;BRANCH IF NOT CORRECT.
3\$: JMP 42(R1) ;RETURN.
10\$: EMT ;
:DATA BUFFER:

21823 121072 000000 000000 000000
21824 121100 000000
21825
21826 121102
21827 121102 004767 003464
21828
21829
21830
21831
21832
21833
21834
21835
21836
21837
21838 121106
21839
21840
21841 121106 012700 121176
21842 121112 012701 000006
21843 121116 012720 177777
21844 121122 077103
21845 121124 012700 102345
21846 121130 012737 121212 000004
21847 121136 170100
21848 121140 012700 121202
21849
21850 121144 170210
21851 121146 020027 121202
21852 121152 001017
21853 121154 023727 121202 102345
21854 121162 001013
21855 121164 023727 121204 177777
21856 121172 001007
21857 121174 000407
21858
21859
21860 121176 177777 177777
21861 121202 177777 177777 177777
21862 121210 177777
21863 121212
21864 121212 104000
21865
21866 121214
21867 121214 004767 003352
21868
21869
21870
21871
21872
21873
21874
21875
21876
21877 121220
21878

LDXT: .WORD 0,0,0,0

NNCDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;TEST 572 DESTINATION MODES, MODE 1 (FL=0), TEST

TS572:

MOV #NNCTB0,R0 ;SET UP THE DATA BUFFER.
MOV #6,R1
1\$: MOV #-1,(R0)+
SOB R1,1\$
MOV #102345,R0
MOV #NNC10,@#ERRVECT ;SET UP FOR TRAPS TO 4.
LDFPS R0 ;SET UP FPS.
MOV #NNCTB1,R0
NNC2: STFPS (R0) ;TEST INSTRUCTION.
CMP R0,#NNCTB1 ;IS RC CORRECT?
BNE NNC10 ;BRANCH IF NOT CORRECT.
CMP @#NNCTB1,#102345 ;IS RESULT CORRECT?
BNE NNC10 ;BRANCH IF NOT CORRECT.
CMP @#NNCTB1+2,#-1 ;IS THE RESULT CORRECT?
BNE NNC10 ;BRANCH IF NOT CORRECT.
BR NNCDONE

;TEST DATA BUFFER:
NNCTB0: .WORD -1,-1
NNCTB1: .WORD -1,-1,-1,-1

NNC10:
EMT ;

NNCDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;TEST 573 DESTINATION MODES, MODE 2 (FL=0), TEST

TS573:

```
21879
21880 121220 012700 121310
21881 121224 012701 000006
21882 121230 012720 177777
21883 121234 077103
21884 121236 012700 105412
21885 121242 012737 121324 000004
21886 121250 170100
21887 121252 012700 121314
21888
21889 121256 170220
21890 121260 020027 121316
21891 121264 001017
21892 121266 023727 121314 105412
21893 121274 001013
21894 121276 023727 121316 177777
21895 121304 001007
21896 121306 000407
21897
21898
21899 121310 177777 177777
21900 121314 177777 177777 177777
21901 121322 177777
21902 121324
21903 121324 104000
21904
21905 121326
21906 121326 004767 003240
21907
21908
21909
21910
21911
21912
21913
21914
21915
21916
21917 121332
21918
21919 121332 012700 121422
21920 121336 012701 000006
21921 121342 012720 177777
21922 121346 077103
21923 121350 012700 105555
21924 121354 012737 121436 000004
21925 121362 170100
21926 121364 012700 121430
21927
21928 121370 170240
21929 121372 020027 121426
21930 121376 001017
21931 121400 023727 121426 105555
21932 121406 001013
21933 121410 023727 121430 177777
21934 121416 001007

MOV #OOCB0,R0 ;SET UP THE DATA BUFFER.
MOV #6,R1
1$: MOV #-1,(R0)+
SOB R1,1$
MOV #105412,R0
MOV #OOC10,@#ERRVECT ;SET UP FOR TRAPS TO VECTOR 4.
LDFPS R0 ;SET UP FPS.
MOV #OOCB1,R0

OOC2: STFPS (R0)+ ;TEST INSTRUCTION.
CMP R0,#OOCB1+2 ;IS R0 CORRECT?
BNE OOC10 ;BRANCH IF NOT CORRECT.
CMP @#OOCB1,#105412 ;IS THE RESULT CORRECT?
BNE OOC10 ;BRANCH IF NOT CORRECT.
CMP @#OOCB1+2,#-1 ;IS THE RESULT CORRECT?
BNE OOC10 ;BRANCH IF NOT CORRECT.
BR OOCDONE

;TEST DATA BUFFER:
OOCB0: .WORD -1,-1
OOCB1: .WORD -1,-1,-1,-1

OOC10:
EMT ;

OOCDONE:
JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;*****
;TEST 574 DESTINATION MODES, MODE 4 (FL=0), TEST
;*****
TS574:

MOV #PPCB0,R0 ;SET UP THE DATA BUFFER.
MOV #6,R1
1$: MOV #-1,(R0)+
SOB R1,1$
MOV #105555,R0
MOV #PPC10,@#ERRVECT ;SET UP FOR TRAPS TO VECTOR 4.
LDFPS R0 ;SET UP FPS.
MOV #PPCB1+2,R0

PPC2: STFPS -(R0) ;TEST INSTRUCTION.
CMP R0,#PPCB1 ;IS R0 CORRECT?
BNE PPC10 ;BRANCH IF NOT CORRECT.
CMP @#PPCB1,#105555 ;IS THE RESULT CORRECT?
BNE PPC10 ;BRANCH IF NOT CORRECT.
CMP @#PPCB1+2,#-1 ;IS THE RESULT CORRECT?
BNE PPC10 ;BRANCH IF NOT CORRECT.
```

21935 121420 000407
21936
21937
21938 121422 177777 177777
21939 121426 177777 177777 177777
21940 121434 177777
21941 121436
21942 121436 104000
21943 121440
21944 121440 004767 003126
21945
21946
21947
21948
21949
21950
21951
21952
21953
21954
21955 121444
21956
21957 121444 012700 121540
21958 121450 012701 000010
21959 121454 012720 177777
21960 121460 077103
21961 121462 012700 106653
21962 121466 012737 121560 000004
21963 121474 170100
21964 121476 012700 121554
21965 121502 012710 121544
21966
21967 121506 170230
21968 121510 020027 121556
21969 121514 001021
21970 121516 023727 121544 106653
21971 121524 001015
21972 121526 023727 121554 121544
21973 121534 001011
21974 121536 000411
21975
21976
21977 121540 177777 177777
21978 121544 177777 177777 177777
21979 121552 177777
21980 121554 177777 177777
21981 121560
21982 121560 104000
21983 121562
21984 121562 004767 003004
21985
21986
21987
21988
21989
21990

```
BR PPCDONE  
:TEST DATA BUFFER:  
PPCTB0: .WORD -1,-1  
PPCTB1: .WORD -1,-1,-1,-1  
PPC10:  
EMT ;  
PPCDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND  
;SEE IF THE USER HAS EXPRESSED  
;THE DESIRE TO CHANGE THE SOFTWARE  
;VIRTUAL CONSOLE SWITCH REGISTER (HAS  
;THE USER TYPED CONTROL G?).  
:*****  
:TEST 575 DESTINATION MODES, MODE 3 (FL=0), TEST  
:*****  
TS575:  
MOV #QQCTB0,R0 ;SET UP THE DATA BUFFER.  
MOV #10,R1  
1$: MOV #-1,(R0)+  
SOB R1,1$  
MOV #106653,R0  
MOV #QQC10,@#ERRVECT ;SET UP FOR TRAPS TO VECTOR 4.  
LDFPS R0 ;SET UP FPS.  
MOV #QQCTB2,R0  
MOV #QQCTB1,(R0)  
QQC2: STFPS @(R0)+ ;TEST INSTRUCTION.  
CMP R0,#QQCTB2+2 ;IS R0 CORRECT?  
BNE QQC10 ;BRANCH IF NOT CORRECT.  
CMP @#QQCTB1,#106653 ;IS THE RESULT CORRECT?  
BNE QQC10 ;BRANCH IF NOT CORRECT.  
CMP @#QQCTB2,#QQCTB1 ;IS THE RESULT CORRECT?  
BNE QQC10 ;BRANCH IF NOT CORRECT.  
BR QQCDONE  
:TEST DATA BUFFER:  
QQCTB0: .WORD -1,-1  
QQCTB1: .WORD -1,-1,-1,-1  
QQCTB2: .WORD -1,-1  
QQC10:  
EMT ;  
QQCDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND  
;SEE IF THE USER HAS EXPRESSED  
;THE DESIRE TO CHANGE THE SOFTWARE  
;VIRTUAL CONSOLE SWITCH REGISTER (HAS  
;THE USER TYPED CONTROL G?).
```


21991
21992
21993
21994
21995 121566
21996
21997
21998 121566 012700 121664
21999 121572 012701 000006
22000 121576 012720 177777
22001 121602 077103
22002 121604 012700 004301
22003 121610 012737 121704 000004
22004 121616 170100
22005 121620 012700 121702
22006 121624 012760 121670 177776
22007
22008 121632 170250
22009 121634 020027 121700
22010 121640 001021
22011 121642 023727 121670 004301
22012 121650 001015
22013 121652 023727 121700 121670
22014 121660 001011
22015 121662 000411
22016
22017
22018 121664 177777 177777
22019 121670 177777 177777 177777
22020 121676 177777
22021 121700 177777 177777
22022 121704
22023 121704 104000
22024 121706
22025 121706 004767 002660
22026
22027
22028
22029
22030
22031
22032
22033
22034
22035 121712
22036
22037
22038 121712 012700 122014
22039 121716 012701 000006
22040 121722 012720 177777
22041 121726 077103
22042 121730 012700 102514
22043 121734 012737 122030 000004
22044 121742 170100
22045 121744 005001
22046 121746 012700 114617

```
*****  
:TEST 576 DESTINATION MODES, MODE 5 (FL=0), TEST  
*****  
TS576:  
  
MOV #RRCTB0,R0 ;SET UP THE DATA BUFFER.  
MOV #6,R1  
1$: MOV #-1,(R0)+  
SOB R1,1$  
MOV #004301,R0  
MOV #RRC10,@#ERRVECT ;SET UP FOR TRAPS TO VECTOR 4.  
LDFPS R0 ;SET UP FPS.  
MOV #RRCTB2+2,R0  
MOV #RRCTB1,-2(R0)  
  
RRC2: STFPS @-(R0) ;TEST INSTRUCTION.  
CMP R0,#RRCTB2 ;IS R0 CORRECT?  
BNE RRC10 ;BRANCH IF NOT CORRECT.  
CMP @#RRCTB1,#004301 ;IS THE RESULT CORRECT?  
BNE RRC10 ;BRANCH IF NOT CORRECT.  
CMP @#RRCTB2,#RRCTB1 ;IS THE RESULT CORRECT?  
BNE RRC10 ;BRANCH IF NOT CORRECT.  
BR RRCDONE  
  
;TEST DATA BUFFER:  
RRC2: RRC10: RRC10: RRC10: RRC10:  
RRC10: .WORD -1,-1  
RRC10: .WORD -1,-1,-1,-1  
RRC10: .WORD -1,-1  
RRC10: EMT ;  
RRC10: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND  
;SEE IF THE USER HAS EXPRESSED  
;THE DESIRE TO CHANGE THE SOFTWARE  
;VIRTUAL CONSOLE SWITCH REGISTER (HAS  
;THE USER TYPED CONTROL G?).  
  
*****  
:TEST 577 DESTINATION MODES, MODE 6 (FL=0), TEST  
*****  
TS577:  
  
MOV #SSCTB0,R0 ;SET UP THE DATA BUFFER.  
MOV #6,R1  
1$: MOV #-1,(R0)+  
SOB R1,1$  
MOV #102514,R0  
MOV #SSC10,@#ERRVECT ;SET UP FOR TRAPS TO VECTOR 4.  
LDFPS R0 ;SET UP FPS.  
CLR R1  
MOV #SSCTB1-5201,R0
```

22047
22048 121752 170260 005201
22049 121756 020127 000000
22050 121762 001022
22051 121764 020027 114617
22052 121770 001017
22053 121772 023727 122020 102514
22054 122000 001013
22055 122002 023727 122022 177777
22056 122010 001007
22057 122012 000407
22058
22059
22060 122014 177777 177777
22061 122020 177777 177777 177777
22062 122026 177777
22063 122030
22064 122030 104000
22065 122032
22066 122032 004767 002534
22067
22068
22069
22070

SSC2: STFPS 5201(R0) ;TEST INSTRUCTION.
CMP R1,#0 ;WAS PC CORRECT AFTER EXECUTION?
BNE SSC10 ;BRANCH IF NOT CORRECT.
CMP R0,#SSCTB1-5201 ;IS R0 CORRECT?
BNE SSC10 ;BRANCH IF NOT CORRECT.
CMP @SSCTB1,#102514 ;IS THE RESULT CORRECT?
BNE SSC10 ;BRANCH IF NOT CORRECT.
CMP @SSCTB1+2,#-1 ;IS THE RESULT CORRECT?
BNE SSC10 ;BRANCH IF NOT CORRECT.
BR SSCDONE

:TEST DATA BUFFER:
SSCTB0: .WORD -1,-1
SSCTB1: .WORD -1,-1,-1,-1

SSC10:
EMT ;
SSCDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

22071
22072
22073
22074
22075
22076 122036
22077
22078 122036 012700 122146
22079 122042 012701 000010
22080 122046 012720 177777
22081 122052 077103
22082 122054 012700 103747
22083 122060 012737 122166 000004
22084 122066 170100
22085 122070 005001
22086 122072 012700 114761
22087 122076 012760 122152 005201
22088
22089 122104 170270 005201
22090 122110 022701 000000
22091 122114 001024
22092 122116 020027 114761
22093 122122 001021
22094 122124 023727 122152 103747
22095 122132 001015
22096 122134 023727 122154 177777
22097 122142 001011
22098 122144 000411
22099
22100
22101 122146 177777 177777
22102 122152 177777 177777 177777
22103 122160 177777
22104 122162 177777 177777
22105 122166
22106 122166 104000
22107 122170
22108 122170 004767 002376
22109
22110
22111
22112
22113
22114
22115
22116
22117 122174
22118 122174 012700 000300
22119 122200 170100
22120 122202 012700 122226
22121 122206 172410
22122 122210 012700 122240
22123
22124 122214 175420
22125
22126 122216 020027 122244

```
*****  
:TEST 600 DESTINATION MODES, MODE 7 (FL=0), TEST  
*****  
T5600:  
      MOV #TTCTB0,R0 ;SET UP THE DATA BUFFER.  
      MOV #10,R1  
1$:   MOV #-1,(R0)+  
      SOB R1,1$  
      MOV #103747,R0  
      MOV #TTC10,@#ERRVECT ;SET UP FOR TRAPS TO VECTOR 4.  
      LDFPS R0 ;SET UP FPS.  
      CLR R1  
      MOV #TTCTB2-5201,R0  
      MOV #TTCTB1,5201(R0)  
  
TTC2: STFPS @5201(R0) ;TEST INSTRUCTION.  
      CMP #0,R1 ;WAS PC CORRECT AFTER EXECUTION?  
      BNE TTC10 ;BRANCH IF NOT CORRECT.  
      CMP R0,#TTCTB2-5201 ;IS R0 CORRECT?  
      BNE TTC10 ;BRANCH IF NOT CORRECT.  
      CMP @#TTCTB1,#103747 ;IS THE RESULT CORRECT?  
      BNE TTC10 ;BRANCH IF NOT CORRECT.  
      CMP @#TTCTB1+2,#-1 ;IS THE RESULT CORRECT?  
      BNE TTC10 ;BRANCH IF NOT CORRECT.  
      BR TTCDONE  
  
:TEST DATA BUFFER:  
TTCTB0: .WORD -1,-1  
TTCTB1: .WORD -1,-1,-1,-1  
  
TTCTB2: .WORD -1,-1  
TTC10:  
      EMT ;  
TTCDONE:  
      JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND  
;SEE IF THE USER HAS EXPRESSED  
;THE DESIRE TO CHANGE THE SOFTWARE  
;VIRTUAL CONSOLE SWITCH REGISTER (HAS  
;THE USER TYPED CONTROL G?).  
  
*****  
:TEST 601 DESTINATION MODES, MODE 2 (FL=1), TEST  
*****  
T5601:  
      MOV #300,R0 ;SET UP FPS.  
      LDFPS R0  
      MOV #UUCTP1,R0 ;SET UP THE ACO OPERAND.  
      LDD (R0),ACO  
      MOV #UUCBFO,R0  
  
UUC2: STCDL ACO,(R0)+ ;TEST INSTRUCTION.  
      CMP R0,#UUCBFO+4 ;IS R0 CORRECT?
```

```
22127 122222 001411          BEQ      UUCDONE
22128 122224 104000          EMT
22129          ;TEST DATA BUFFER:
22130 122226 000000 000000 000000 UUCTP1: .WORD 0,0,0,0
22131 122234 000000
22132 122236 177777          -1
22133 122240 177777 177777 177777 UUCBFO: .WORD -1,-1,-1
22134
22135 122246          UUCDONE:
22136 122246 004767 002320          JSR      PC,.RSET          ;GO INITIALIZE THE FPS AND STACK; AND
22137          ;SEE IF THE USER HAS EXPRESSED
22138          ;THE DESIRE TO CHANGE THE SOFTWARE
22139          ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
22140          ;THE USER TYPED CONTROL G?).
22141
22142          ;*****
22143          ;TEST 602          DESTINATION MODES, MODE 4 (FL=1), TEST
22144          ;*****
22145 122252          TS602:
22146
22147 122252 012700 000300          MOV      #300,R0          ;SET UP FPS.
22148 122256 170100          LDFPS   R0
22149 122260 012700 122304          MOV      #VVC1P1,R0      ;SET UP THE ACO OPERAND.
22150 122264 172410          LDD     (R0),ACO
22151 122266 012700 122322          MOV      #VVCBFO+4,R0
22152
22153 122272 175440          VVC2:   STCDL   ACO,-(R0)          ;TEST INSTRUCTION.
22154
22155 122274 020027 122316          CMP     R0,#VVCBFO      ;IS R0 CORRECT?
22156 122300 001411          BEQ     VVCDONE
22157 122302 104000          EMT
22158          ;TEST DATA BUFFER:
22159 122304 000000 000000 000000 VVC1P1: .WORD 0,0,0,0
22160 122312 000000
22161 122314 177777          -1
22162 122316 177777 177777 177777 VVCBFO: .WORD -1,-1,-1
22163
22164 122324          VVCDONE:
22165 122324 004767 002242          JSR     PC,.RSET          ;GO INITIALIZE THE FPS AND STACK; AND
22166          ;SEE IF THE USER HAS EXPRESSED
22167          ;THE DESIRE TO CHANGE THE SOFTWARE
22168          ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
22169          ;THE USER TYPED CONTROL G?).
22170
22171          ;*****
22172          ;TEST 603          STCDI AND STCDL TEST
22173          ;*****
22174 122330          TS603:
22175
22176          ;FIRST TEST STC WITH EXP=100 (EXCESS 200)
22177 122330 004737 123266          WWC1:   JSR     PC,#STCSUB      ;GO EXECUTE THE INSTRUCTION.
22178 122334 020000 000000 000000 1$:      .WORD 2000,0,0,0      ;ACO OPERAND.
22179 122342 000000
22180 122344 000000 000000          2$:      .WORD 0,0          ;EXPECTED RESULT.
22181 122350 177777 177777          3$:      .WORD -1,-1        ;ERROR RES.
22182 122354 040300          4$:      40300          ;FPS BEFORE EXECUTION.
```

```

22183 122356 040304          40304          ;FPS AFTER EXECUTION.
22184 122360 140304          140304          ;ANTICIPATED ERRONEOUS FPS.
22185 122362 177777          -1              ;REPORT RESULT INCORRECT.
22186          ;EXP=0 (OCT)   FL=1   FIC=0
22187 122364 004737 123266    WWC2: JSR      PC,@#STCSUB ;GO EXECUTE THE INSTRUCTION.
22188 122370 040000 000000 000000 1$: .WORD    40000,0,0,0 ;AC      ;ACO OPERAND.
22189 122376 000000
22190 122400 000000 000000
22191 122404 177777 177777    2$: .WORD    0,0          ;EXPECTED RESULT.
22192 122410 040313          40313          ;ANTICIPATED ERRONEOUS RESULT.
22193 122412 040304          40304          ;FPS BEFORE EXECUTION.
22194 122414 140304          140304          ;FPS AFTER EXECUTION.
22195 122416 177777          -1              ;ANTICIPATED ERRONEOUS FPS.
22196          ;EXP=37 (OCT)  FL=1   FIC=1
22197 122420 004737 123266    WWC3: JSR      PC,@#STCSUB ;GO EXECUTE THE INSTRUCTION.
22198 122424 047667 075757 157737 1$: .WORD    47667,75757,157737,167773 ;ACO OPERAND.
22199 122432 167773
22200 122434 055675 173757    2$: .WORD    55675,173757 ;EXPECTED RESULT.
22201 122440 122102 004021    3$: .WORD    122102,004021 ;ANTICIPATED ERRONEOUS RESULT.
22202 122444 040717          40717          ;FPS BEFORE EXECUTION.
22203 122446 040700          40700          ;FPS AFTER EXECUTION.
22204 122450 140705          140705          ;ANTICIPATED ERRONEOUS FPS.
22205 122452 177777          -1              ;EXPECTED FEC.
22206          ;EXP=40 (OCT)  FL=1   FIC=1
22207 122454 004737 123266    WWC4: JSR      PC,@#STCSUB ;GO EXECUTE THE INSTRUCTION.
22208 122460 050000 000000 000000 1$: .WORD    50000,0,0,0 ;ACO OPERAND.
22209 122466 000000
22210 122470 000000 000000
22211 122474 177777 177777    2$: .WORD    0,0          ;EXPECTED RESULT.
22212 122500 040700          40700          ;ANTICIPATED ERRONEOUS RESULT.
22213 122502 140705          140705          ;FPS BEFORE EXECUTION.
22214 122504 040705          40705          ;FPS AFTER EXECUTION.
22215 122506 000006          6              ;ANTICIPATED ERRONEOUS FPS.
22216          ;EXPECTED FEC.
22217          ;EXP=40 (OCT)  FL=1   FIC=0
22218 122510 004737 123266    WWC5: JSR      PC,@#STCSUB ;GO EXECUTE THE INSTRUCTION.
22219 122514 050000 000000 000000 1$: .WORD    50000,0,0,0 ;ACO OPERAND.
22220 122522 000000
22221 122524 000000 000000
22222 122530 177777 177777    2$: .WORD    0,0          ;EXPECTED RESULT.
22223 122534 040312          40312          ;ANTICIPATED ERRONEOUS RESULT.
22224 122536 040305          40305          ;FPS BEFORE EXECUTION.
22225 122540 140305          140305          ;FPS AFTER EXECUTION.
22226 122542 177777          -1              ;ANTICIPATED ERRONEOUS FPS.
22227          ;EXPECTED FEC.
22228 122544 004737 123266    WWC6: JSR      PC,@#STCSUB ;GO EXECUTE THE INSTRUCTION.
22229 122550 046000 000001 000000 1$: .WORD    46000,1,0,0 ;ACO OPERAND.
22230 122556 000000
22231 122560 000200 000001
22232 122564 177777 177777    2$: .WORD    200,1        ;EXPECTED RESULT.
22233 122570 040700          40700          ;ANTICIPATED ERRONEOUS RESULT.
22234 122572 040700          40700          ;FPS BEFORE EXECUTION.
22235 122574 177777          -1              ;FPS AFTER EXECUTION.
22236 122576 177777          -1              ;ANTICIPATED ERRONEOUS FPS.
22237          ;EXPECTED FEC.
22238 122600 004737 123266    WWC7: JSR      PC,@#STCSUB ;GO EXECUTE THE INSTRUCTION.
    
```

22239	122604	045600	000001	000000	1\$:	.WORD	45600,1,0,0	:ACO OPERAND.
22240	122612	000000						
22241	122614	000100	000000		2\$:	.WORD	100,0	:EXPECTED RESULT.
22242	122620	177777	177777		3\$:	.WORD	-1,-1	:ANTICIPATED ERRONEOUS RESULT.
22243	122624	040707			4\$:	40707		:FPS BEFORE EXECUTION.
22244	122626	040700				40700		:FPS AFTER EXECUTION.
22245	122630	177777				-1		:ANTICIPATED ERRONEOUS FPS.
22246	122632	177777				-1		:EXPECTED FEC.
22247					:EXP=17 (OCT)	FL=0 FIC=1		
22248	122634	004737	123266		WVC10: JSR	PC, @#STCSUB		:GO EXECUTE THE INSTRUCTION.
22249	122640	043600	000000	000000	1\$:	.WORD	43600,0,0,0	:ACO OPERAND.
22250	122646	000000						
22251	122650	040000	177777		2\$:	.WORD	40000,-1	:EXPECTED RESULT.
22252	122654	000000	177777		3\$:	.WORD	0,-1	:ANTICIPATED ERRONEOUS RESULT.
22253	122660	040600			4\$:	40600		:FPS BEFORE EXECUTION.
22254	122662	040600				40600		:FPS AFTER EXECUTION.
22255	122664	140604				140604		:ANTICIPATED ERRONEOUS FPS.
22256	122666	177777				-1		:EXPECTED FEC.
22257								
22258					:EXP=20 (OCT)	FL=0 FIC=1		
22259	122670	004737	123266		WVC11: JSR	PC, @#STCSUB		:GO EXECUTE THE INSTRUCTION.
22260	122674	044000	000000	000000	1\$:	.WORD	44000,0,0,0	:ACO OPERAND.
22261	122702	000000						
22262	122704	000000	177777		2\$:	.WORD	0,-1	:EXPECTED RESULT.
22263	122710	177777	177777		3\$:	.WORD	-1,-1	:ANTICIPATED ERRONEOUS RESULT.
22264	122714	040600			4\$:	40600		:FPS BEFORE EXECUTION.
22265	122716	140605				140605		:FPS AFTER EXECUTION.
22266	122720	040600				40600		:ANTICIPATED ERRONEOUS FPS.
22267	122722	000006				6		:EXPECTED FEC.
22268					:EXP=10 (OCT),	AC NEGATIVE, FL=0, FIC=1		
22269	122724	004737	123266		WVC12: JSR	PC, @#STCSUB		:GO EXECUTE THE INSTRUCTION.
22270	122730	142000	000000	000000	1\$:	.WORD	142000,0,0,0	:ACO OPERAND.
22271	122736	000000						
22272	122740	177600	177777		2\$:	.WORD	177600,-1	:EXPECTED RESULT.
22273	122744	000200	000000		3\$:	.WORD	200,0	:ANTICIPATED ERRONEOUS RESULT.
22274	122750	040600			4\$:	40600		:FPS BEFORE EXECUTION.
22275	122752	040610				40610		:FPS AFTER EXECUTION.
22276	122754	040600				40600		:ANTICIPATED ERRONEOUS FPS.
22277	122756	177777				-1		:EXPECTED FEC.
22278					:EXP=37 (OCT),	FL=1, FIC=1, AC NEG.		
22279	122760	004737	123266		WVC13: JSR	PC, @#STCSUB		:GO EXECUTE THE INSTRUCTION.
22280	122764	147600	000000	000000	1\$:	.WORD	147600,0,0,0	:ACO OPERAND.
22281	122772	000000						
22282	122774	140000	000000		2\$:	.WORD	140000,0	:EXPECTED RESULT.
22283	123000	137777	000000		3\$:	.WORD	137777,0	:ANTICIPATED ERRONEOUS RESULT.
22284	123004	040700			4\$:	40700		:FPS BEFORE EXECUTION.
22285	123006	040710				40710		:FPS AFTER EXECUTION.
22286	123010	177777				-1		:ANTICIPATED ERRONEOUS FPS.
22287	123012	177777				-1		:EXPECTED FEC.
22288					:EXP=37 (OCT),	FL=1, FIC=1, AC NEG		
22289	123014	004737	123266		WVC14: JSR	PC, @#STCSUB		:GO EXECUTE THE INSTRUCTION.
22290	123020	147600	000000	001000	1\$:	.WORD	147600,0,1000,0	:ACO OPERAND.
22291	123026	000000						
22292	123030	137777	177777		2\$:	.WORD	137777,177777	:EXPECTED RESULT.
22293	123034	140000	177777		3\$:	.WORD	140000,177777	:ANTICIPATED ERRONEOUS RESULT.
22294	123040	040707			4\$:	40707		:FPS BEFORE EXECUTION.

```
22295 123042 040710 40710 ;FPS AFTER EXECUTION.
22296 123044 177777 -1 ;ANTICIPATED ERRONEOUS FPS.
22297 123046 177777 -1 ;EXPECTED FEC.
22298 ;EXP=41 (OCT), AC NEG, FL=1, FIC=1
22299 123050 004737 123266 WWC15: JSR PC,#STCSUB ;GO EXECUTE THE INSTRUCTION.
22300 123054 150200 000000 000000 1$: .WORD 150200,0,0,0 ;ACO OPERAND.
22301 123062 000000
22302 123064 000000 000000 2$: .WORD 0,0 ;EXPECTED RESULT.
22303 123070 177777 177777 3$: .WORD -1,-1 ;ANTICIPATED ERRONEOUS RESULT.
22304 123074 040700 4$: 40700 ;FPS BEFORE EXECUTION.
22305 123076 140705 140705 ;FPS AFTER EXECUTION.
22306 123100 177777 -1 ;ANTICIPATED ERRONEOUS FPS.
22307 123102 000006 6 ;EXPECTED FEC.
22308 ;EXP=40 (OCT), AC NEG, FL=1, FIC=1
22309 123104 004737 123266 WWC16: JSR PC,#STCSUB ;GO EXECUTE THE INSTRUCTION.
22310 123110 150000 000001 000000 1$: .WORD 150000,1,0,0 ;ACO OPERAND.
22311 123116 000000
22312 123120 000000 000000 2$: .WORD 0,0 ;EXPECTED RESULT.
22313 123124 100000 177600 3$: .WORD 100000,-200 ;ANTICIPATED ERRONEOUS RESULT.
22314 123130 040700 4$: 40700 ;FPS BEFORE EXECUTION.
22315 123132 140705 140705 ;FPS AFTER EXECUTION.
22316 123134 040700 40700 ;ANTICIPATED ERRONEOUS FPS.
22317 123136 000006 6 ;EXPECTED FEC.
22318 ;EXP=40, AC NEGATIVE, FL=1, FIC=1
22319 123140 004737 123266 WWC17: JSR PC,#STCSUB ;GO EXECUTE THE INSTRUCTION.
22320 123144 150001 000000 000000 1$: .WORD 150001,0,0,0 ;ACO OPERAND.
22321 123152 000000
22322 123154 000000 000000 2$: .WORD 0,0 ;EXPECTED RESULT.
22323 123160 077400 000000 3$: .WORD 77400,0 ;ANTICIPATED ERRONEOUS RESULT.
22324 123164 040700 4$: 40700 ;FPS BEFORE EXECUTION.
22325 123166 140705 140705 ;FPS AFTER EXECUTION.
22326 123170 177777 -1 ;ANTICIPATED ERRONEOUS FPS.
22327 123172 000006 6 ;EXPECTED FEC.
22328 ;EXP 40 (OCT), AC MOST NEG LONG INT, FL=1
22329 ;FIC=1
22330 123174 004737 123266 WWC20: JSR PC,#STCSUB ;GO EXECUTE THE INSTRUCTION.
22331 123200 150000 000000 000000 1$: .WORD 150000,0,0,0 ;ACO OPERAND.
22332 123206 000000
22333 123210 100000 000000 2$: .WORD 100000,0 ;EXPECTED RESULT.
22334 123214 000000 000000 3$: .WORD 0,0 ;ANTICIPATED ERRONEOUS RESULT.
22335 123220 040700 4$: 40700 ;FPS BEFORE EXECUTION.
22336 123222 040710 40710 ;FPS AFTER EXECUTION.
22337 123224 140705 140705 ;ANTICIPATED ERRONEOUS FPS.
22338 123226 177777 -1 ;EXPECTED FEC.
22339 ;EXP=20, AC = MOST NEG INTEGER, FL=0, FIC=1
22340
22341 123230 004737 123266 WWC21: JSR PC,#STCSUB ;GO EXECUTE THE INSTRUCTION.
22342 123234 144000 000001 000000 1$: .WORD 144000,1,0,0 ;ACO OPERAND.
22343 123242 000000
22344 123244 100000 177777 2$: .WORD 100000,-1 ;EXPECTED RESULT.
22345 123250 100000 177400 3$: .WORD 100000,177400 ;ANTICIPATED ERRONEOUS RESULT.
22346 123254 040600 4$: 40600 ;FPS BEFORE EXECUTION.
22347 123256 040610 40610 ;FPS AFTER EXECUTION.
22348 123260 140605 140605 ;ANTICIPATED ERRONEOUS FPS.
22349 123262 177777 -1 ;EXPECTED FEC.
22350 123264 000457 6$: BR WWCDONE
```

22351
 22352
 22353
 22354
 22355
 22356
 22357
 22358
 22359
 22360
 22361
 22362
 22363
 22364
 22365
 22366
 22367
 22368
 22369
 22370
 22371
 22372
 22373
 22374
 22375
 22376
 22377
 22378
 22379
 22380
 22381
 22382
 22383
 22384 123266 012601
 22385 123270 012700 000200
 22386 123274 170100
 22387 123276 010100
 22388 123300 172410
 22389 123302 012702 123414
 22390 123306 012700 000004
 22391 123312 012722 177777
 22392 123316 077003
 22393 123320 016100 000020
 22394 123324 170100
 22395 123326 012700 123414
 22396 123332 175410
 22397
 22398 123334 170204
 22399 123336 170305
 22400 123340 010102
 22401 123342 062702 000010
 22402 123346 012700 123414
 22403 123352 012703 000002
 22404 123356 022022
 22405 123360 001014
 22406 123362 077303

: THIS SUBROUTINE, STCSUB, IS USED TO SET UP THE OPERANDS, EXECUTE
 : THE STCDI OR STCDL INSTRUCTION AND CHECK THE RESULTS. A CALL
 : TO IT IS MADE THUS:

```

JSR      PC,@#STCSUB
ACARG:   .WORD  X,X,X,X      :AC OPERAND
RES:     .WORD  X,X          :EXPECTED RESULT
ERRES:   .WORD  X,X          :ERROR RESULT
FPSB:    .WORD  X            :FPS BEFORE EXECUTION
FPSA:    .WORD  X            :FPS AFTER EXECUTION
ERFPS:   .WORD  X            :ERROR FPS.
FEC:     .WORD  X            :EXPECTED FEC
ERR1:    ERROR  X            :DATA ERROR.
          BR      CONT
ERR2:    ERROR  X            :FPS ERROR.
CONT:    CONT
  
```

: THE OPERANDS ARE SET UP (USING ACO AS THE ACCUMULATOR). THEN
 : THE STCDI OR STCDL INSTRUCTION IS EXECUTED.
 : THE RESULT IS CHECKED AGAINST RES. IF THE RESULT IS CORRECT THEN THE FPS IS
 : COMPARED WITH FPSA IF THIS TOO IS CORRECT STCSUB RETURNS CONTROL
 : TO THE CALLING ROUTINE AT CONT. IF THE FPS IS BAD STCSUB
 : COMPARE IT TO ERROR FPS. IF THIS MATCHES THEN STCSUB WILL RETURN
 : TO THE ERROR CALL AT ERR2, OTHERWISE STCSUB ITSELF
 : REPORTS THIS FAILURE AND THEN RETURNS TO CONT. IF THE RESULT OF THE
 : STCDI OR STCDL IS INCORRECT, THE INCORRECT RESULT IS COMPARED WITH THE
 : ANTICIPATED FAILING DATA PATTERN, ERRES. IF THE FAILURE IN
 : THE RESULT WAS ANTICIPATED CORRECTLY TO BE ERRES THEN STCSUB
 : WILL TRANSFER CONTROL TO THE ERROR CALL AT ERR1. OTHERWISE THE
 : RESULT WAS INCORRECT BUT WAS NOT ANTICIPATED AND STCSUB WILL
 : REPORT THE FAILURE AFTER WHICH CONTROL WILL BE PASSED TO CONT.

```

STCSUB:  MOV      (SP)+,R1      :GET A POINTER TO THE ARGUMENTS.
          MOV      #200,R0      :SET UP THE ACO OPERAND.
          LDFPS   R0
          MOV      R1,R0
          LDD      (R0),ACO
          MOV      #STCIBF,R2   :INITIALIZE THE OUT PUT BUFFER.
          MOV      #4,R0
1$:      MOV      #-1,(R2)+
          SOB     R0,1$
          MOV      20(R1),R0    :SET THE FPS.
          LDFPS   R0
          MOV      #STCIBF,R0
2$:      STCDL   ACO,(R0)      :TEST INSTRUCTION.

          STFPS   R4           :GET THE FPS.
          STST    R5           :GET THE FEC.
          MOV     R1,R2
          ADD     #10,R2
          MOV     #STCIBF,R0   :SEE IF THE RESULT IS CORRECT.
          MOV     #2,R3
3$:      CMP     (R0)+,(R2)+
          BNE    10$
          SOB    R3,3$
  
```



```
22407 123364 016102 000022      MOV      22(R1),R2
22408 123370 020204      CMP      R2,R4          ;SEE IF THE FPS IS CORRECT.
22409 123372 001007      BNE     10$            ;BRANCH IF INCORRECT.
22410 123374 005702      TST     R2
22411 123376 100003      BPL     4$
22412 123400 026105 000026      CMP      26(R1),R5     ;SEE IF THE FEC IS CORRECT.
22413 123404 001002      BNE     10$            ;BRANCH IF INCORRECT.
22414
22415 123406 000161 000030      4$:     JMP      30(R1)   ;RETURN.
22416 123412      10$:
22417 123412 104000      EMT
22418
22419      ;DATA BUFFER:
22420 123414 177777 177777 177777  STCIBF: .WORD  -1,-1,-1,-1
22421 123422 177777
22422
22423 123424      WWC DONE:
22424 123424 004767 001142      JSR      PC,.RSET      ;GO INITIALIZE THE FPS AND STACK; AND
22425      ;SEE IF THE USER HAS EXPRESSED
22426      ;THE DESIRE TO CHANGE THE SOFTWARE
22427      ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
22428      ;THE USER TYPED CONTROL G?).
22429
22430
22431
22432      ;*****
22433      ;TEST 604      STCFL AND STCFI TEST
22434      ;*****
22435      TS604:
22436
22437      ;EXPONENT=37, FL=1
22438 123430 004737 123266      JSR      PC,@#STCSUB   ;GO EXECUTE THE INSTRUCTION.
22439 123434 047777 177777 177777  1$:     .WORD  47777,-1,-1,-1 ;ACO OPERAND.
22440 123442 177777
22441 123444 077777 177600      2$:     .WORD  77777,177600   ;EXPECTED RESULT.
22442 123450 077777 177777      3$:     .WORD  77777,177777   ;ANTICIPATED ERRONEOUS RESULT.
22443 123454 040100      4$:     40100              ;FPS BEFORE EXECUTION.
22444 123456 040100      -1      ;FPS AFTER EXECUTION.
22445 123460 177777      -1      ;ANTICIPATED ERRONEOUS FPS.
22446 123462 177777      -1      ;EXPECTED FEC.
22447 123464
22448 123464 004767 001102      XXC DONE: JSR      PC,.RSET      ;GO INITIALIZE THE FPS AND STACK; AND
22449      ;SEE IF THE USER HAS EXPRESSED
22450      ;THE DESIRE TO CHANGE THE SOFTWARE
22451      ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
22452      ;THE USER TYPED CONTROL G?).
22453
22454
22455
22456      ;*****
22457      ;TEST 605      STEXP TEST
22458      ;*****
22459      TS605:
22460
22461 123470 004737 123676      ; EXP = 100 (EXCESS 200)
22462 123474 020000 000000 000000  YYC1: JSR      PC,@#STXSUB
22462 123474 020000 000000 000000  1$:     .WORD  20000,0,0,0   ;AC
```

```
22463 123502 000000
22464 123504 177700      2$: -100                :EXP RES
22465 123506 052525      3$: 52525                :ERROR EXP.
22466 123510 040000      4$: 40000                :FPSB
22467 123512 040010                40010                :FPSA
22468 123514 040000                40000                :ERROR FPS
22469
22470 123516 004737 123676      ; EXP = 200 (EXCESS 200)
22471 123522 040000 000000 000000 YYC2: JSR PC,#STXSUB ;GO EXECUTE THE INSTRUCTION.
22472 123530 000000      1$: .WORD 40000,0,0,0 ;ACO OPERAND.
22473 123532 000000      2$: 0                    ;EXPECTED EXPONENT RESULT.
22474 123534 052525                3$: 52525                ;ANTICIPATED ERRONEOUS RESULT.
22475 123536 040000      4$: 40000                ;FPS BEFORE EXECUTION.
22476 123540 040004                40004                ;FPS AFTER EXECUTION.
22477 123542 040000                40000                ;ANTICIPATED ERRONEOUS FPS.
22478
22479 123544 004737 123676      ; EXP = 201 (EXCESS 200)
22480 123550 040200 000000 000000 YYC3: JSR PC,#STXSUB ;GO EXECUTE THE INSTRUCTION.
22481 123556 000000      1$: .WORD 40200,0,0,0 ;ACO OPERAND.
22482 123560 000001      2$: 1                    ;EXPECTED EXPONENT RESULT.
22483 123562 052525                3$: 52525                ;ANTICIPATED ERRONEOUS RESULT.
22484 123564 040000      4$: 40000                ;FPS BEFORE EXECUTION.
22485 123566 040000                40000                ;FPS AFTER EXECUTION.
22486 123570 040004                40004                ;ANTICIPATED ERRONEOUS FPS.
22487
22488
22489 123572 004737 123676      ; EXP = 375 (EXCESS 200)
22490 123576 077200 000000 000000 YYC4: JSR PC,#STXSUB ;GO EXECUTE THE INSTRUCTION.
22491 123604 000000      1$: .WORD 77200,0,0,0 ;ACO OPERAND.
22492 123606 000175      2$: 175                  ;EXPECTED EXPONENT RESULT.
22493 123610 052525                3$: 52525                ;ANTICIPATED ERRONEOUS RESULT.
22494 123612 040000      4$: 40000                ;FPS BEFORE EXECUTION.
22495 123614 040000                40000                ;FPS AFTER EXECUTION.
22496 123616 040010                40010                ;ANTICIPATED ERRONEOUS FPS.
22497
22498
22499 123620 004737 123676      ; EXP = 1 (EXCESS 200)
22500 123624 000200 000000 000000 YYC5: JSR PC,#STXSUB ;GO EXECUTE THE INSTRUCTION.
22501 123632 000000      1$: .WORD 200,0,0,0 ;ACO OPERAND.
22502 123634 177601      2$: -177                 ;EXPECTED EXPONENT RESULT.
22503 123636 052525                3$: 52525                ;ANTICIPATED ERRONEOUS RESULT.
22504 123640 040000      4$: 40000                ;FPS BEFORE EXECUTION.
22505 123642 040010                40010                ;FPS AFTER EXECUTION.
22506 123644 040000                40000                ;ANTICIPATED ERRONEOUS FPS.
22507
22508
22509 123646 004737 123676      ; EXP = 156 (EXCESS 200)
22510 123652 033400 000000 000000 YYC6: JSR PC,#STXSUB ;GO EXECUTE THE INSTRUCTION.
22511 123660 000000      1$: .WORD 33400,0,0,0 ;ACO OPERAND.
22512 123662 177756      2$: -22                  ;EXPECTED EXPONENT RESULT.
22513 123664 052525                3$: 52525                ;ANTICIPATED ERRONEOUS RESULT.
22514 123666 047707      4$: 47707                ;FPS BEFORE EXECUTION.
22515 123670 047710                47710                ;FPS AFTER EXECUTION.
22516 123672 177777                -1                    ;ANTICIPATED ERRONEOUS FPS.
22517 123674 000452      BR YYCDONE
22518
```

22519
 22520
 22521
 22522
 22523
 22524
 22525
 22526
 22527
 22528
 22529
 22530
 22531
 22532
 22533
 22534
 22535
 22536
 22537
 22538
 22539
 22540
 22541
 22542
 22543
 22544
 22545
 22546
 22547
 22548
 22549
 22550
 22551
 22552
 22553
 22554
 22555
 22556
 22557
 22558
 22559
 22560
 22561
 22562
 22563
 22564
 22565
 22566
 22567
 22568
 22569
 22570
 22571
 22572
 22573
 22574

:THIS SUBROUTINE, STXSUB, IS USED TO SET UP THE OPERANDS, EXECUTE
 :THE STEXP INSTRUCTION AND CHECK THE RESULTS. A CALL
 :TO IT IS MADE THUS:

```

:
:      JSR      PC,@#STXSUB
:      ACARG:  .WORD  X,X,X,X      ;AC OPERAND
:      RES:    .WORD  X            ;EXPECTED RESULT
:      ERRES:  .WORD  X            ;ERROR RESULT
:      FPSB:   .WORD  X            ;FPS BEFORE EXECUTION
:      FPSA:   .WORD  X            ;FPS AFTER EXECUTION
:      ERFPS:  .WORD  X            ;ERROR FPS.
:      ERR1:   ERROR X            ;DATA ERROR.
:      BR      CONT
:      ERR2:   ERROR X            ;FPS ERROR.
:      CONT:   CONT              ;RETURN ADDRESS

```

:THE OPERANDS ARE SET UP (USING ACO AS THE ACCUMULATOR). THEN
 :THE STEXP INSTRUCTION IS EXECUTED.
 :THE RESULT IS CHECKED AGAINST RES. IF THE RESULT IS CORRECT THEN THE FPS IS
 :COMPARED WITH FPSA IF THIS TOO IS CORRECT STXSUB RETURNS CONTROL
 :TO THE CALLING ROUTINE AT CONT. IF THE FPS IS BAD STXSUB
 :COMPARE IT TO ERROR FPS. IF THIS MATCHES THEN STXSUB WILL RETURN
 :TO THE ERROR CALL AT ERR2, OTHERWISE STXSUB ITSELF
 :REPORTS THIS FAILURE AND THEN RETURNS TO CONT. IF THE RESULT OF THE
 :STEXP IS INCORRECT, THE INCORRECT RESULT IS COMPARED WITH THE
 :ANTICIPATED FAILING DATA PATTERN, ERRES. IF THE FAILURE IN
 :THE RESULT WAS ANTICIPATED CORRECTLY TO BE ERRES THEN STXSUB
 :WILL TRANSFER CONTROL TO THE ERROR CALL AT ERR1. OTHERWISE THE
 :RESULT WAS INCORRECT BUT WAS NOT ANTICIPATED AND STXSUB WILL
 :REPORT THE FAILURE AFTER WHICH CONTROL WILL BE PASSED TO CONT.

```

STXSUB: MOV      (SP)+,R1      ;GET A POINTER TO THE ARGUMENTS.
        MOV      R1,R2
        MOV      #123456,@#STXBF
        MOV      #76543,@#STXBF+2
        MOV      #200,R0
        LDFPS   R0
        MOV      R1,R0      ;SET UP THE ACO OPERAND.
        LDD     (R0),ACO
        MOV      16(R1),R0  ;SET THE FPS.
        LDFPS   R0
        MOV      #STXBF,R0
1$:     STEXP   ACO,(R0)    ;TEST INSTRUCTION.
        STFPS   R4         ;GET FPS.
        CMP     10(R1),@#STXBF ;WAS RESULT CORRECT?
        BEQ     5$
        EMT
        CMP     R4,16(R1)   ;SEE IF THE FPS IS CORRECT.
        BEQ     10$
        EMT
;SEE IF MORE THAN ONE WORD WAS WRITTEN IN THE OUTPUT BUFFER.
10$:   CMP     #76543,@#STXBF+2
        BEQ     4$
        EMT
4$:     JMP     22(R1)

```

```

123676 012601
123700 010102
123702 012737 123456 124010
123710 012737 076543 124012
123716 012700 000200
123722 170100
123724 010100
123726 172410
123730 016100 000016
123734 170100
123736 012700 124010
123742 175010
123744 170204
123746 026137 000010 124010
123754 001401
123756 104000
123760 J20461 000016
123764 001401
123766 104000
123770 022737 076543 124012
123776 001401
124000 104000
124002 000161 000022

```

```
22575 124006 177777
22576 124010 177777 177777 177777 STXBF: .WORD -1,-1,-1,-1,-1
22577 124016 177777 177777
22578
22579 124022
22580 124022 004767 000544 YYCDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
22581 ;SEE IF THE USER HAS EXPRESSED
22582 ;THE DESIRE TO CHANGE THE SOFTWARE
22583 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
22584 ;THE USER TYPED CONTROL G?).
22585
22586 ;*****
22587 ;TEST 606 STST TEST
22588 ;*****
22589 124026 TS606:
22590
22591 124026 012700 040000 MOV #40000,R0 ;SET FPS. FID=1.
22592 124032 170100 LDFPS R0
22593
22594 124034 170003 ZZC2: .WORD 170003 ;ILLEGAL FPP
22595 ;OP CODE
22596 124036 012700 124116 MOV #ZZCBF,R0 ;SET UP THE OUTPUT BUFFER.
22597 124042 012710 177777 MOV #-1,(R0)
22598 124046 012760 177777 000002 MOV #-1,2(R0)
22599 124054 170310 ZZC3: STST (R0) ;GET FEC AND
22600 ;FEA
22601 124056 170204 STFPS R4 ;GET FPS.
22602 124060 012700 124116 MOV #ZZCBF,R0
22603 124064 022710 000002 CMP #2,(R0) ;SEE IF FEC IS CORRECT.
22604 124070 001010 BNE ZZC10 ;BRANCH IF INCORRECT.
22605 124072 022760 124034 000002 CMP #ZZC2,2(R0) ;SEE IF FEA, ADDRESS, IS CORRECT.
22606 124100 001004 BNE ZZC10 ;BRANCH IF INCORRECT.
22607 124102 022704 140000 CMP #140000,R4 ;SEE IF FPS IS CORRECT.
22608 124106 001001 BNE ZZC10 ;BRANCH IF INCORRECT.
22609 124110 000407 BR ZZCDONE
22610 124112 ZZC10:
22611 124112 104000 EMT ;
22612
22613 ;DATA BUFFER:
22614 124114 177777 -1
22615 124116 177777 177777 177777 ZZCBF: .WORD -1,-1,-1,-1
22616 124124 177777
22617 124126 177777 -1
22618
22619 124130 ZZCDONE:
22620 124130 004767 000436 JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
22621 ;SEE IF THE USER HAS EXPRESSED
22622 ;THE DESIRE TO CHANGE THE SOFTWARE
22623 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
22624 ;THE USER TYPED CONTROL G?).
22625
22626
22627 ;*****
22628 ;TEST 607 SPECIAL CASE TEST
22629 ;*****
22630 124134 TS607:
```

22631 124134 012746 144724
22632 124140 012746 040600
22633 124144 005046
22634 124146 012746 040600
22635 124152 172466 000004
22636 124156 173026
22637 124160 174037 124210
22638 124164 022737 036711 124210
22639 124172 001401
22640 124174 104000
22641 124176 022737 152000 124212 1\$:
22642 124204 001403
22643 124206 104000
22644
22645 124210 000000
22646 124212 000000
22647
22648 124214 012706 001000
22649 124220 004767 000346
22650
22651
22652
22653
22654
22655
22656
22657
22658
22659
22660 124224
22661 124224 004567 124160
22662 124230 000520
22663
22664 124232 005001
22665 124234 005000
22666 124236 013767 000064 113114
22667 124244 013767 000066 113110
22668 124252 012737 124324 000064
22669 124260 005037 000066
22670 124264 005067 053506
22671 124270 005067 053272
22672 124274 105767 053264 1\$:
22673 124300 100375
22674 124302 005067 053260
22675 124306 052767 000100 053250
22676 124314 005200 2\$:
22677 124316 001376
22678 124320 000005
22679 124322 104000
22680 124324 166700 000110 3\$:
22681 124330 010067 000106
22682 124334 012737 124412 000064
22683 124342 005100 4\$:
22684 124344 005067 053214
22685 124350 005067 053212
22686 124354 105767 053204 5\$:

AAD1: MOV #144724, -(SP) ;PUT FRACTION ON STACK
MOV #40600, -(SP) ;PUT EXPONENT ON STACK
CLR -(SP) ;PUT SUBTRAHEND FRACTION ON STACK
MOV #40600, -(SP) ;PUT SUBTRAHEND EXPONENT ON STACK
LDF 4(SP), ACO ;LOAD FP ACCUMULATORS
SUBF (SP)+, ACO ;DO SUBTRACTION
STF ACO, @#AADBFB ;GET AND STORE ANSWER
CMP #36711, @#AADBFB ;IS EXPONENT CORRECT
BEQ 1\$
EMT ;BAD EXPONENT FROM SUBTRACTION
1\$: CMP #152000, @#AADBFB+2 ;IS FRACTION CORRECT
BEQ AADDONE
EMT ;FRACTION INCORRECT
AADBFB: .WORD 0
.WORD 0
AADDONE: MOV #STBOT, SP ;RESTORE STACK POINTER
JSR PC, RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

:TEST 610 INTERRUPTABILITY TEST

TS610:
BBD1: JSR R5,CHKAPT
BR FPEXIT ;SKIP TEST IF ON APT AND NOT FIRST PASS
CLR R1 ;INITIALIZE A COUPLE OF COUNTERS
CLR R0
MOV @#64, STMP0 ;SAVE INTERRUPT VECTOR
MOV @#66, STMP1 ;SAVE INTERRUPT PRIORITY
MOV #3\$, @#64 ;SET UP INTERRUPT PRIORITY FOR THIS TEST
CLR @#66 ;AND PRIORITY
CLR PS ;PUT PROCESSOR PRIORITY AT 0
CLR TPB ;SEND A NULL CHARACTER
1\$: TSTB TTCSR ;WAIT FOR DONE TO SET
BPL 1\$
CLR TPB ;SEND A SECOND CHARACTER
BIS #BIT6, TTCSR ;SET INTERRUPT ENABLE
2\$: INC R0 ;INCREMENT COUNTER TO GET BASE TIME
BNE 2\$;CONTINUE LOOPING UNLESS COUNTER GOES TO 0
RESET ;IF NO INTERRUPT YET KILL IT
EMT ;NO INTERRUPT OCCURRED IN ALLOTTED TIME
3\$: SUB Y, R0 ;SUBTRACT TIME FOR FP INSTRUCTION
MOV R0, Z ;SAVE FIRST TIME
MOV #7\$, @#64 ;SET UP FOR NEXT INTERRUPT
4\$: COM R0 ;MAKE PRE LOOP COUNTER NEGATIVE
CLR TTCSR ;MAKE SURE NO INTERRUPT YET
CLR TPB ;SEND A CHARACTER
5\$: TSTB TTCSR ;WAIT FOR READY BIT TO SET

```

22687 124360 100375          BPL      5$
22688 124362 005067 053200  CLR      TPB          ;SEND SECOND CHARACTER
22689 124366 052767 000100 053170  BIS     #BIT6, TTCSR ;SET INTERRUPT ENABLE
22690 124374 005200          INC      R0          ;DO PRE LOOP
22691 124376 001376          BNE     6$,          ;
22692 124400 171227 040400  MULF   #2, AC2      ;DO FLOATING POINT INSTRUCTION
22693 124404 000240          NOP                    ;JUST IN CASE INTERRUPT TAKES TOO LONG
22694 124406 000005          RESET                ;IF NO INTERRUPT CLEAR THE WORLD
22695 124410 104000          EMT                    ;INTERRUPT NOT BACK IN ALLOTTED TIME
22696 124412 005201          INC      R1          ;INCREMENT TIMES THROUGH COUNTER
22697 124414 020127 000015  CMP    R1, #15      ;HAVE WE PASSED HERE 15 TIMES BEFORE
22698 124420 001411          BEQ     BBDDONE      ;IF YES I MAY NEVER PASS HERE AGAIN
22699 124422 062767 000002 000012  ADD    #2, Z          ;IF NO ADD A LITTLE TIME TO PRELOOP
22700 124430 016700 000006  MOV    Z, R0         ;PUT NEW COUNT IN COUNTER
22701 124434 000742          BR     4$,          ;DO IT ALL AGAIN
22702
22703 124436 000000          X:      .WORD      0
22704 124440 000026          Y:      .WORD      26
22705 124442 000000          Z:      .WORD      0
22706
22707 124444 042767 000100 053112 BBDDONE: BIC   #100, TTCSR ;CLEAR INTERRUPT ENABLE BEFORE EXITING TEST
22708 124452 016737 112702 000064  MOV    $TMP0, @#64 ;RESTORE PRINTER VECTOR
22709 124460 016737 112676 000066  MOV    $TMP1, @#66 ;RESTORE PRINTER PRIORITY
22710 124466 004767 000100          JSR    PC, .RSET   ;GO INITIALIZE THE FPS AND STACK; AND
22711          ;SEE IF THE USER HAS EXPRESSED
22712          ;THE DESIRE TO CHANGE THE SOFTWARE
22713          ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
22714          ;THE USER TYPED CONTROL G?).
22715 124472 000167 000160          FPEXIT: JMP    SLU1ST ;GET OVER SUBROUTINES TO NEXT TEST
22716
22717
22718 124476 012737 000003 001002 ERROR4: MOV   #3, @#SFATAL ;SET UP FATAL ERROR NUMBER
22719 124504 012767 000001 054266  MOV   #1, $MSGTY ;SET FATAL ERROR FLAG
22720 124512 032737 000001 001020  BIT   #1, @#SENV ;UNDER APT
22721 124520 001004          BNE    FPHLT ;YES
22722 124522 012700 124534  MOV   #FPMSG, R0
22723 124526 004767 006576  JSR   PC, TYPE
22724 124532 000777          FPHLT: BR     . ;STAY HERE FOREVER
22725
22726 124534 040506 046111 042105 FPMSG: .ASCIZ /FAILED DURING THE FPP TESTS/<12><15>
22727 124542 042040 051125 047111
22728 124550 020107 044124 020105
22729 124556 050106 020120 042524
22730 124564 052123 005123 000015
22731          .EVEN
22732
22733          .SBTTL FLAG RESET ROUTINE
22734          ;*****
22735          ;*THIS ROUTINE WILL BE CALLED AT THE END OF EACH FLOATING POINT TEST
22736          ;*TO RESET THE STACK, CLEAR THE FPS AND REINITIALIZE TRAP VECTORS
22737
22738 124572 012737 124476 000244 .RSET: MOV   #ERROR4, @#FPVCT
22739 124600 012737 021336 000004  MOV   #T04, @#ERRVCT
22740 124606 012737 021340 000010  MOV   #T010, @#10
22741 124614 011600          MOV   (SP), R0
22742 124616 012706 001000          MOV   #STBOT, SP
  
```

22743 124622 005004
 22744 124624 170104
 22745 124626 000110

CLR R4
 LDFPS R4
 JMP (R0)

;THESE ARE SOME EQUATES USED IN THE PROGRAM

22748 000001
 22749 000002
 22750 000004
 22751 000010
 22752 000020
 22753 000040
 22754 000100
 22755 000200
 22756 000400
 22757 001000
 22758 002000
 22759 004000
 22760 010000
 22761 020000
 22762 040000
 22763 100000

BIT0=000001
 BIT1=000002
 BIT2=000004
 BIT3=000010
 BIT4=000020
 BIT5=000040
 BIT6=000100
 BIT7=000200
 BIT8=000400
 BIT9=001000
 BIT10=002000
 BIT11=004000
 BIT12=010000
 BIT13=020000
 BIT14=040000
 BIT15=100000

22764
 22765 124630 177560
 22766 124632 177562
 22767 124634 177564
 22768 124636 177566
 22769 124640 000060
 22770 124642 000062
 22771 124644 000064
 22772 124646 000066

RCSR: 177560
 RBUF: 177562
 TCSR: 177564
 TBUF: 177566
 RVECT: 60
 RPSW: 62
 TVECT: 64
 TPSW: 66

;ADDRESS OF RECEIVER COMMAND/STATUS REGISTER
 ;ADDRESS OF RECEIVER BUFFER
 ;ADDRESS OF TRANSMITTER COMMAND/STATUS REGISTER
 ;ADDRESS OF TRANSMITTER BUFFER
 ;RECEIVER INTERRUPT VECTOR
 ;TRANSMITTER INTERRUPT VECTOR

22773
 22774
 22775 124650 177546
 22776 124652 000100
 22777 124654 000102

;REAL TIME CLOCK REGISTER AND VECTOR ADDRESSES
 LKS: .WORD 177546
 RTCVT: .WORD 100
 RTCPSW: .WORD 102

22778
 22779 124656 000244
 22780 124660 032777 000004 074516
 22781 124666 001402
 22782 124670 000167 001106
 22783 124674 012737 000004 001004
 22784 124702 012737 125720 000030

SLU1ST: CLZ
 BIT #4,@SWR
 BEQ 1\$
 JMP KWSTRT
 1\$: MOV #4,@\$TESTN
 MOV #ERRORS,@#30

;PUT TEST NUMBER IN MAILBOX
 ;SET UP FOR CORRECT ERROR CALL

22785
 22786
 22787
 22788
 22789

 ;TEST 611 TEST ABILITY TO REFERENCE TCSR

 TS611:

22790 124710
 22791 124710 013703 000004
 22792 124714 012737 124730 000004
 22793 124722 005777 177706
 22794 124726 000401
 22795 124730
 22796 124730 104000
 22797 124732 010337 000004
 22798

MOV @#4,R3
 MOV #1\$,@#4
 TST @TCSR
 BR 4\$
 1\$: EMT
 4\$: MOV R3,@#4

;SAVE TIMEOUT VECTOR
 ;SET UP TIMEOUT VECTOR
 ;REFERENCE THE XMIT COMMAND/STATUS REG.
 ;RESTORE TIMEOUT VECTOR

```
22799
22800
22801 :*****
22802 :TEST 612 TEST ABILITY TO REFERENCE TBUF
22803 :*****
22804 TS612:
22805 124736 013703 000004 000004 MOV @#4,R3 ;SAVE TIMEOUT VECTOR
22806 124742 012737 124756 MOV #1$,@#4 ;SET UP TIMEOUT VECTOR
22807 124750 005777 177662 TST @TBUF ;REFERENCE THE XMIT BUFFER
22808 124754 000401 BR 4$
22809 124756 1$:
22810 124756 104000 EMT ;
22811 124760 010337 000004 4$: MOV R3,@#4 ;RESTORE TIMEOUT VECTOR
22812
22813
22814 :*****
22815 :TEST 613 TEST THAT TCSR BIT7(DONE) CLEARS WHEN XBUF IS LOADED
22816 :*****
22817 TS613:
22818 124764 032737 000001 001020 BIT #1,@#SENV ;ARE WE RUNNING UNDER APT
22819 124772 001405 BEQ 70$ ;IF NO THEN SERIES OF TESTS
22820 124774 005737 001006 TST @#$PASS ;IS THIS FIRST PASS
22821 125000 001402 BEQ 70$ ;IF YES THEN DO SERIES OF TESTS
22822 125002 000167 000774 JMP KWSTRT ;IF NO THEN BYPASS SERIES OF TESTS
22823 125006 005077 177624 70$: CLR @TBUF ;LOAD XBUF
22824 125012 105777 177616 TSTB @TCSR ;CHECK DONE
22825 125016 100006 BPL 3$ ;BR IF CLEAR
22826 ;FILL SECOND BUFFER BECUASE REFRESH COULD CAUSE
22827 ;FIRST TEST TO FAIL
22828 125020 005077 177612 CLR @TBUF ;FILL DOUBLE BUFFER
22829 125024 105777 177604 TSTB @TCSR ;CHECK DONE
22830 125030 100001 BPL 3$
22831 125032 104000 EMT ;
22832 125034 005000 3$: CLR R0 ;CLEAR TIMER
22833 125036 105777 177572 4$: TSTB @TCSR ;CHECK FOR XMIT DONE
22834 125042 100403 BMI 5$ ;IF DONE SETS, BR TO END OF TEST
22835 125044 005200 INC R0 ;INCREMENT TIMER
22836 125046 001373 BNE 4$
22837 125050 104000 EMT ;
22838 5$:
22839
22840
22841 :*****
22842 :TEST 614 TEST THAT TCSR "DONE" SETS WITH RESET
22843 :*****
22844 TS614:
22845 125052 005077 177560 CLR @TBUF ;LOAD TRANSMIT BUFFER
22846 125056 105777 177552 1$: TSTB @TCSR ;WAIT FOR DONE
22847 125062 100375 BPL 1$
22848 125064 005077 177546 CLR @TBUF ;LOAD SECOND BUFFER
22849 125070 000240 NOP
22850 125072 000005 RESET ;SET DONE WITH RESET
22851 125074 105777 177534 TSTB @TCSR ;CHECK FOR DONE SET
22852 125100 100401 BMI TS615
22853 125102 104000 EMT ;
22854
```


22855
22856
22857
22858
22859
22860 125104
22861 125104 013703 000004
22862 125110 012737 125124 000004
22863 125116 005777 177506
22864 125122 000401
22865 125124
22866 125124 104000
22867 125126 010337 000004
22868
22869
22870
22871
22872
22873 125132
22874 125132 013703 000004
22875 125136 012737 125152 000004
22876 125144 005777 177462
22877 125150 000401
22878 125152
22879 125152 104000
22880 125154 010337 000004
22881
22882
22883
22884
22885
22886
22887
22888 125160
22889 125160 017703 177454
22890 125164 012777 125206 177446
22891 125172 106427 000340
22892 125176 032777 000100 177424
22893 125204 001401
22894 125206
22895 125206 104000
22896 125210 052777 000100 177412
22897 125216 032777 000100 177404
22898 125224 001001
22899 125226 104000
22900 125230 042777 000100 177372
22901 125236 032777 000100 177364
22902 125244 001401
22903 125246 104000
22904 125250
22905 125250 052777 000100 177352
22906 125256 000005
22907 125260 032777 000100 177342
22908 125266 001401
22909 125270 104000
22910 125272 010377 177342

```
*****  
:TEST 615 TEST ABILITY TO ACCESS RCSR  
*****  
TS615:  
MOV @#4,R3 ;SAVE TIMEOUT VECTOR  
MOV #1$,@#4 ;SET UP TIMEOUT VECTOR  
TST @RCSR ;ACCESS RCSR  
BR 2$  
1$: EMT ;  
2$: MOV R3,@#4 ;RESTORE TIMEOUT VECTOR  
*****  
:TEST 616 TEST ABILITY TO ACCESS RBUF  
*****  
TS616:  
MOV @#4,R3 ;SAVE TIMEOUT VECTOR  
MOV #1$,@#4 ;SET UP TIMEOUT VECTOR  
TST @RBUF ;ACCESS RBUF  
BR 2$  
1$: EMT ;  
2$: MOV R3,@#4 ;RESTORE TIMEOUT VECTOR  
*****  
:TEST 617 TEST THAT BIT6 OF RCSR CAN BE SET & RESET  
*****  
TS617:  
MOV @RVECT,R3 ;SAVE RECEIVE VECTOR  
MOV #1$,@RVECT ;SET UP INTERRUPT VECTOR FOR ERROR REPORT  
MTPS #340 ;SET PSW TO PRIORITY 7  
BIT #BIT6,@RCSR ;TEST BIT6 OF RCSR  
BEQ 2$  
1$: EMT ;  
2$: BIS #BIT6,@RCSR ;SET BIT6 OF RCSR  
BIT #BIT6,@RCSR ;TEST BIT6 OF RCSR  
BNE 3$  
3$: EMT ;  
BIC #BIT6,@RCSR ;CLEAR BIT6 OF RCSR  
BIT #BIT6,@RCSR ;TEST BIT6 OF RCSR  
BEQ 4$  
4$: EMT ;  
BIS #BIT6,@RCSR ;SET BIT6 OF RCSR  
RESET ;CLEAR BIT6 OF RCSR WITH RESET  
BIT #BIT6,@RCSR ;TEST BIT6 OF RCSR  
BEQ 5$  
5$: EMT ;  
MOV R3,@RVECT ;RESTORE RECEIVE VECTOR
```

22911
22912
22913
22914
22915
22916
22917 125276
22918 125276 042777 000100 177330
22919 125304 017703 177334
22920 125310 012777 125332 177326
22921 125316 105777 177312
22922 125322 100375
22923 125324 106427 000140
22924 125330 000401
22925 125332
22926 125332 104000
22927 125334 012777 125354 177302
22928 125342 052777 000100 177264
22929 125350 000240
22930
22931 125352 104000
22932
22933 125354 042777 000100 177252
22934 125362 022626
22935 125364 010377 177254
22936
22937
22938
22939
22940
22941 125370
22942 125370 042777 000100 177236
22943 125376 106427 000340
22944 125402 017703 177236
22945 125406 012777 125434 177230
22946 125414 105777 177214
22947 125420 100375
22948 125422 052777 000100 177204
22949 125430 000240
22950 125432 000401
22951 125434
22952 125434 104000
22953 125436 042777 000100 177170
22954 125444 012777 125462 177172
22955 125452 106427 000140
22956 125456 000240
22957 125460 000401
22958 125462
22959 125462 104000
22960 125464 010377 177154
22961
22962
22963
22964
22965
22966 125470

```
*****  
:TEST 620 TEST THAT XMIT INTERRUPTS ONLY WHEN ENABLED  
*****  
TS620:  
BIC #BIT6,@TCSR ;CLEAR TRANSMIT INTERRUPT ENABLE  
MOV @TVECT,R3 ;SAVE XMIT VECTOR  
MOV #2$,@TVECT ;POINT XMIT VECTOR TO ERROR REPORT  
1$: TSTB @TCSR ;WAIT FOR DONE  
BPL 1$  
MTPS #140 ;SET PSW TO PRIORITY 3  
BR 3$  
2$: EMT ;  
3$: MOV #4$,@TVECT ;SET XMIT VECTOR TO END OF TEST  
BIS #BIT6,@TCSR ;ENABLE INTERRUPTS  
NOP  
EMT ;XMIT DID NOT INTERRUPT  
4$: BIC #BIT6,@TCSR ;DISABLE INTERRUPTS  
CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT  
MOV R3,@TVECT ;RESTORE XMIT VECTOR  
*****  
:TEST 621 TEST THAT XMIT INTERRUPTS DO NOT OCCUR WHEN DISABLED  
*****  
TS621:  
BIC #BIT6,@TCSR ;DISABLE INTERRUPTS  
MTPS #340 ;SET PSW TO PRIORITY 7  
MOV @TVECT,R3 ;SAVE XMIT VECTOR  
MOV #2$,@TVECT ;POINT XMIT VECTOR TO ERROR REPORT  
1$: TSTB @TCSR ;WAIT FOR DONE  
BPL 1$  
BIS #BIT6,@TCSR ;ENABLE INTERRUPT  
NOP  
BR 3$  
2$: EMT ;  
3$: BIC #BIT6,@TCSR ;CLEAR INTERRUPT ENABLE  
MOV #4$,@TVECT ;POINT XMIT VECTOR TO ERROR REPORT  
MTPS #140 ;SET PSW TO PRIORITY 3  
NOP  
BR 5$  
4$: EMT ;  
5$: MOV R3,@TVECT ;RESTORE XMIT VECTOR  
*****  
:TEST 622 TEST TRANSMITTER FOR DOUBLE INTERRUPTS  
*****  
TS622:
```

```

22967 125470 042777 000100 177136 BIC #BIT6,@TCSR ;CLEAR INTERRUPT ENABLE
22968 125476 017703 177142 MOV @TVECT,R3 ;SAVE XMIT VECTOR
22969 125502 017704 177140 MOV @TPSW,R4 ;SAVE XMIT PSW VECTOR
22970 125506 012777 125546 177130 MOV #2,@TVECT ;SET UP XMIT VECTOR
22971 125514 012777 000340 177124 MOV #340,@TPSW ;SET PIO 7 AFTER INTERRUPT
22972 125522 106427 000140 MTPS #140 ;SET PSW TO PRIORITY 3
22973 125526 105777 177102 1$: TSTB @TCSR ;WAIT FOR DONE
22974 125532 100375 BPL 1$
22975 125534 052777 000100 177072 BIS #BIT6,@TCSR ;ENABLE INTERRUPTS
22976 125542 000240 NOP
22977
22978 125544 104000 EMT ;
22979 ;XMIT INTERRUPT DID NOT OCCUR
22980 125546 022626 2$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
22981 125550 012777 125574 177066 MOV #4,@TVECT ;POINT XMIT VECTOR TO ERROR
22982 125556 106427 000140 MTPS #140 ;SET PSW TO PRIORITY 3
22983 125562 000240 NOP ;GIVE TIME FOR ANY INTERRUPTS
22984 125564 042777 000100 177042 BIC #BIT6,@TCSR ;DISABLE INTERRUPTS
22985 125572 000401 BR 5$
22986 125574 4$: EMT ;
22987 125574 104000 5$: MOV R3,@TVECT ;RESTORE XMIT VECTOR
22988 125576 010377 177042 MOV R4,@TPSW ;RESTORE XMIT PSW VECTOR
22989 125602 010477 177040
22990
22991 ;*****
22992 ;TEST 623 TEST THAT XMIT INTERRUPT CLEARS WITH LOADING TBUF
22993 ;*****
22994 TS623:
22995 125606 042777 000100 177020 BIC #BIT6,@TCSR ;DISABLE INTERRUPTS
22996 125614 106427 000340 MTPS #340 ;SET PSW TO PRIORITY 7
22997 125620 017703 177020 MOV @TVECT,R3 ;SAVE XMIT VECTOR
22998 125624 012777 125674 177012 MOV #2,@TVECT ;POINT XMIT VECTOR TO ERROR
22999 125632 052777 000100 176774 BIS #BIT6,@TCSR ;ENABLE INTERRUPTS
23000 125640 005077 176772 CLR @TBUF ;LOAD TBUF
23001 125644 105777 176764 1$: TSTB @TCSR ;WAIT FOR DONE (INTERRUPT)
23002 125650 100375 BPL 1$
23003 125652 005077 176760 CLR @TBUF ;FILL SECOND BUFFER TO RESET INT.
23004 125656 106427 000140 MTPS #140 ;SET PSW TO PRIORITY 3
23005 125662 000240 NOP ;GIVE TIME FOR ANY INTERRUPTS
23006 125664 042777 000100 176742 BIC #BIT6,@TCSR ;DISABLE INTERRUPTS
23007 125672 000401 BR 2$
23008 125674 2$: EMT ;
23009 125674 104000 3$: MOV R3,@TVECT ;RESTORE XMIT VECTOR
23010 125676 010377 176742 CLR R0 ;INITIALIZE LOOP COUNTER
23011 125702 005000 4$: INC R0 ;INCREMENT LOOP COUNTER
23012 125704 005200 BNE 4$ ;UNTIL COUNTER = 0
23013 125706 001376 TST @RBUF ;CLEAR RECEIVER BUFFER
23014 125710 005777 176716 JMP KWSTRT ;GET TO NEXT TEST
23015 125714 000167 000062
23016
23017 125720 012737 000004 001002 ERROR5: MOV #4,@$FATAL ;SET UP FATAL ERROR NUMBER
23018 125726 012767 000001 053044 MOV #1,$MSGTY ;SET FATAL ERROR FLAG
23019 125734 032737 000001 001020 BIT #1,@$ENV ;UNDER APR ?
23020 125742 001004 SLIHLT
23021 125744 012700 125756 MOV #SL1MSG,R0
23022 125750 004767 005354 JSR PC,TYPE
  
```

23023 125754 000777
23024
23025 125756 040506 046111 042105
23026 125764 051440 052514 020061
23027 125772 042524 052123 006412
23028 126000 000
23029 126002
23030
23031
23032 126002 000244
23033 126004 032777 000010 073372
23034 126012 001402
23035 126014 000167 000730
23036 126020 012737 000005 001004
23037 126026 012737 126640 000030
23038
23039 126034
23040
23041
23042
23043 126034
23044 126034 013703 000004
23045 126040 012737 126054 000004
23046 126046 005777 176576
23047 126052 000401
23048 126054
23049 126054 104000
23050 126056 010337 000004
23051
23052
23053
23054
23055 126062
23056 126062 017703 176564
23057 126066 012777 126110 176556
23058 126074 106427 000340
23059 126100 032777 000100 176542
23060 126106 001401
23061 126110
23062 126110 104000
23063 126112 052777 000100 176530
23064 126120 032777 000100 176522
23065 126126 001001
23066 126130 104000
23067 126132 042777 000100 176510
23068 126140 032777 000100 176502
23069 126146 001401
23070 126150 104000
23071 126152 032737 000001 001020
23072 126160 001403
23073 126162 005737 001006
23074 126166 001011
23075 126170
23076 126170 052777 000100 176452
23077 126176 000005
23078 126200 032777 000100 176442

SL1HLT: BR .
SL1MSG: .ASCIZ /FAILED SLU1 TEST/<12><15>
.EVEN
KWSTRT: CLZ
BIT #10,@SWR
BEQ 1\$
JMP SLU2ST
1\$: MOV #5,@\$TESTN ;PUT TEST NUMBER IN MAILBOX
MOV #ERROR6,@#30 ;SET UP ERROR CALL
LKSTST:
:*****
:TEST 624 TEST ABILITY TO ACCESS LKS
:*****
TS624:
MOV @#4,R3 ;SAVE TIMEOUT VECTOR
MOV #1\$,@#4 ;SET UP TIMEOUT VECTOR
TST @LKS ;ACCESS LKS
BR 2\$
1\$: EMT ;
2\$: MOV R3,@#4 ;RESTORE TIMEOUT VECTOR
:*****
:TEST 625 TEST THAT BIT6 OF LKS CAN BE SET & RESET
:*****
TS625:
MOV @RTCVT,R3 ;SAVE LINE CLOCK VECTOR
MOV #1\$,@RTCVT ;SET UP INTERRUPT VECTOR FOR ERROR REPORT
MTPS #340 ;SET PSW TO PRIORITY 7
BIT #BIT6,@LKS ;TEST BIT6 OF LKS
BEQ 2\$
1\$: EMT ;
2\$: BIS #BIT6,@LKS ;SET BIT6 OF LKS
BIT #BIT6,@LKS ;TEST BIT6 OF LKS
BNE 3\$
3\$: EMT ;
BIC #BIT6,@LKS ;CLEAR BIT6 OF LKS
BIT #BIT6,@LKS ;TEST BIT6 OF LK
BEQ 4\$
4\$: EMT ;
BIT #1,@\$ENV ;ARE WE RUNNING UNDER APT
BEQ 70\$;IF NO THEN DO TEST
TST @\$SPASS ;IS THIS FIRST PASS
BNE 5\$;IF NO SKIP TO TEST END
70\$: BIS #BIT6,@LKS ;SET BIT6 OF LKS
RESET ;CLEAR BIT6 OF LKS WITH RESET
BIT #BIT6,@LKS ;TEST BIT6 OF LKS

```
23079 126206 001401
23080 126210 104000
23081 126212 010377 176434
23082
23083
23084
23085
23086
23087 126216
23088
23089 126216 106427 000340
23090 126222 017703 176424
23091 126226 017704 176422
23092 126232 012777 126274 176412
23093 126240 012777 000340 176406
23094 126246 052777 000100 176374
23095
23096 126254 012701 024000
23097 126260 077101
23098 126262 012777 126276 176362
23099 126270 106427 000240
23100 126274 104000
23101 126276 012777 126314 176346
23102 126304 012701 024000
23103 126310 077101
23104 126312 000401
23105 126314 104000
23106 126316 005077 176326
23107 126322 012701 024000
23108 126326 077101
23109 126330 106427 000240
23110 126334 012701 024000
23111 126340 077101
23112 126342 012777 126366 176302
23113 126350 052777 000100 176272
23114 126356 012701 024000
23115 126362 077101
23116 126364 104000
23117
23118 126366 022626
23119 126370 042777 000100 176252
23120 126376 010377 176250
23121 126402 010477 176246
23122
23123
23124
23125
23126
23127
23128
23129
23130 126406
23131 126406 032737 000001 001020
23132 126414 001403
23133 126416 005737 001006
23134 126422 001045
```

```
BEQ 5$
EMT
5$: MOV R3,@RTCVT ;RESTORE LINE CLOCK VECTOR
```

```
*****
:TEST 626 TEST THAT THE REAL TIME CLOCK INTERRUPTS PROPERLY
*****
TS626:
```

```
MTPS #340 ;SET PSW TO PRIORITY 7
MOV @RTCVT,R3 ;SAVE LINE CLOCK VECTOR
MOV @RTCP SW,R4 ;SAVE LINE CLOCK PSW VECTOR
MOV #ESR51,@RTCVT ;SET RTC INTERRUPT VECTOR TO ERROR REPORT
MOV #340,@RTCP SW ;KEEP PRIORITY AT 7
BIS #BIT6,@LKS ;SET INTERRUPT ENABLE

MOV #24000,R1 ;SET UP A WAIT LOOP
ESR36: SOB R1,ESR36 ;WAIT 30 MILLISEC
MOV #ESR37,@RTCVT ;ALTER VECTOR
MTPS #240 ;PRIORITY NOW TO FIVE

ESR51: EMT
ESR37: MOV #ESR38,@RTCVT
MOV #24000,R1
ESR39: SOB R1,ESR39 ;WAIT 30 MORE MILLISEC
BR ESR38+2

ESR38: EMT ;AN ERROR IF WE ARE HERE
CLR @LKS ;CLR INTERRUPT ENABLE
MOV #24000,R1
ESR52: SOB R1,ESR52 ;WAIT LOOP
MTPS #240 ;ALTER PRIORITY TO FIVE
MOV #24000,R1
ESR53: SOB R1,ESR53 ;WAIT AGAIN
MOV #ESR55,@RTCVT
BIS #BIT6,@LKS ;ENABLE LTC INTERRUPTS
MOV #24000,R1
ESR54: SOB R1,ESR54
EMT ;SHOULD HAVE INTERRUPTED

ESR55: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
BIC #BIT6,@LKS ;DISABLE INTERRUPTS
MOV R3,@RTCVT ;RESTORE LINE CLOCK VECTOR
MOV R4,@RTCP SW ;RESTORE LINE CLOCK PSW VECTOR
```

```
*****
:TEST 627 TEST RTC FOR DOUBLE INTERRUPTS
*****
TS627:
```

```
BIT #1,@$ENV ;ARE WE RUNNING UNDER APT
BEQ 70$ ;IF NO THEN DO TEST
TST @$PASS ;IS THIS FIRST PASS
BNE TS630 ;IF NO THEN SHIP TO NEXT TEST
```

```
23135 126424
23136 126424 017703 176222
23137 126430 017704 176220
23138 126434 012777 126476 176210
23139 126442 012777 000340 176204
23140 126450 106427 000240
23141
23142 126454 005000
23143 126456 052777 000100 176164
23144 126464 005200
23145 126466 005700
23146 126470 100375
23147 126472 000240
23148
23149 126474 104000
23150
23151 126476 022626
23152 126500 012777 126516 176144
23153 126506 106427 000240
23154 126512 000240
23155 126514 000401
23156 126516
23157 126516 104000
23158 126520 042777 000100 176122
23159 126526 010377 176120
23160 126532 010477 176116
23161
23162
23163
23164
23165
23166 126536
23167 126536 032737 000001 001020
23168 126544 001403
23169 126546 005737 001006
23170 126552 001113
23171 126554
23172 126554 106427 000340
23173 126560 017703 176066
23174 126564 012777 126630 176060
23175 126572 005000
23176
23177 126574 052777 000100 176046
23178 126602 005200
23179 126604 005700
23180 126606 100375
23181 126610 000005
23182 126612 106427 000240
23183 126616 000240
23184 126620 042777 000100 176022
23185 126626 000401
23186 126630
23187 126630 104000
23188 126632 010377 176014
23189 126636 000444
23190

70$:
MOV @RTCVT,R3 ;SAVE LINE CLOCK VECTOR
MOV @RTCPSW,R4 ;SAVE LINE CLOCK PSW VECTOR
MOV #2$,@RTCVT ;SET UP RTC INTERRUPT VECTOR
MOV #340,@RTCPSW ;DISALLOW INTERRUPTS AFTER THE INTERRUPT
MTPS #240 ;SET PSW TO PRIORITY 5

1$:
CLR R0
BIS #BIT6,@LKS ;ENABLE CLOCK INTERRUPTS
INC R0
TST R0
BPL 1$
NOP ;GIVE TIME FOR ANY INTERRUPT

EMT ;RTC INTERRUPT DID NOT OCCUR

2$:
CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
MOV #3$,@RTCVT ;POINT RTC VECTOR TO ERROR REPORT
MTPS #240 ;SET PSW TO PRIORITY 5
NOP ;GIVE SOME TIME FOR AN INTERRUPT
BR 4$

3$:
EMT

4$:
BIC #BIT6,@LKS ;DISABLE CLOCK INTERRUPTS
MOV R3,@RTCVT ;RESTORE LINE CLGCK VECTOR
MOV R4,@RTCPSW ;RESTORE LINE CLOCK PSW VECTOR

:*****
:TEST 630 TEST THAT RTC INTERRUPT CLEARS WITH RESET
:*****
TS630:
BIT #1,@#SENV ;ARE WE RUNNING UNDER APT
BEQ 70$ ;IF NO THEN DO TEST
TST @#SPASS ;IS THIS FIRST PASS
BNE TS631 ;IF NO THEN SHIP TO NEXT TEST

70$:
MTPS #340 ;SET PSW TO PRIORITY 7
MOV @RTCVT,R3 ;SAVE LINE CLOCK VECTOR
MOV #2$,@RTCVT ;POINT RTC VECTOR TO ERROR REPORT
CLR R0

1$:
BIS #BIT6,@LKS ;ENABLE CLOCK INTERRUPTS
INC R0
TST R0
BPL 1$
RESET ;CLEAR PENDING INTERRUPT WITH RESET
MTPS #240 ;SET PSW TO PRIORITY 5
NOP ;GIVE TIME FOR ANY INTERRUPT
BIC #BIT6,@LKS ;DISALLOW INTERRUPTS
BR 3$

2$:
EMT

3$:
MOV R3,@RTCVT ;RESTORE LINE CLOCK VECTOR
BR SLU2ST
```

```
23191
23192
23193 126640 012737 000005 001002 ERROR6: MOV #5,@#SFATAL ;SET UP FATAL ERROR NUMBER
23194 126646 012767 000001 052124 MOV #1,@#MSGTY ;SET FATAL ERROR FLAG
23195 126654 032737 000001 001020 BIT #1,@#SENV ;UNDER APT /
23196 126662 001004 BNE LTCHLT
23197 126664 012700 126676 MOV #LTCMSG,R0
23198 126670 004767 004434 JSR PC,TYPE
23199 126674 000777 LTCHLT: BR
23200 126676 040506 046111 051125 LTCMSG: .ASCIZ /FAILURE DURING LTC TEST/<12><15>
23201 126704 020105 052504 044522
23202 126712 043516 046040 041524
23203 126720 052040 051505 005124
23204 126726 000015
23205 .EVEN
23206
23207 ;SERIAL LINE UNIT REGISTER AND VECTOR ADDRESSES FOR SLU2
23208
23209 126730 176500 RCSR2: 176500 ;ADDRESS OF RECEIVER COMMAND/STATUS REGISTER
23210 126732 176502 RBUF2: 176502 ;ADDRESS OF RECEIVER BUFFER
23211 126734 176504 TCSR2: 176504 ;ADDRESS OF TRANSMITTER COMMAND/STATUS REGISTER
23212 126736 176506 TBUF2: 176506 ;ADDRESS OF TRANSMITTER BUFFER
23213 126740 000300 RVECT2: 300 ;RECEIVER INTERRUPT VECTOR
23214 126742 000302 RPSW2: 302
23215 126744 000304 TVECT2: 304 ;TRANSMITTER INTERRUPT VECTOR
23216 126746 000306 TPSW2: 306
23217
23218 126750 000244 SLU2ST: CLZ
23219 126752 032777 000020 072424 BIT #20,@#SWR
23220 126760 001402 BEQ 1$
23221 126762 000167 002622 JMP UNIQUE
23222 126766 012737 000006 001004 1$: MOV #6,@#STESTN ;PUT TEST NUMBER IN MAILBOX
23223 126774 012737 131504 000030 MOV #ERROR7,@#30 ;SET UP FOR CORRECT ERROR CALL
23224
23225
23226 ;*****
23227 ;TEST 631 TEST ABILITY TO REFERENCE TCSR2
23228 ;*****
23229 TS631:
23230 127002 013703 000004 000004 MOV @#4,R3 ;SAVE TIMEOUT VECTOR
23231 127006 012737 127022 000004 MOV #1$,@#4 ;SET UP TIMEOUT VECTOR
23232 127014 005777 177714 TST @TCSR2 ;REFERENCE THE XMIT COMMAND/STATUS REG.
23233 127020 000401 BR 4$
23234 127022 1$:
23235 127022 104000 EMT ;
23236 127024 010337 000004 4$: MOV R3,@#4 ;RESTORE TIMEOUT VECTOR
23237
23238
23239
23240 ;*****
23241 ;TEST 632 TEST ABILITY TO REFERENCE TBUF2
23242 ;*****
23243 TS632:
23244 127030 013703 000004 000004 MOV @#4,R3 ;SAVE TIMEOUT VECTOR
23245 127034 012737 127050 000004 MOV #1$,@#4 ;SET UP TIMEOUT VECTOR
23246 127042 005777 177670 TST @TBUF2 ;REFERENCE THE XMIT BUFFER
```

23247 127046 000401
23248 127050
23249 127050 104000
23250 127052 010337 000004
23251
23252
23253
23254
23255
23256 127056
23257 127056 032737 000001 001020
23258 127064 001403
23259 127066 005737 001006
23260 127072 001022
23261 127074
23262 127074 005077 177636
23263 127100 105777 177630
23264 127104 100006
23265
23266
23267 127106 005077 177624
23268 127112 105777 177616
23269 127116 100001
23270 127120 104000
23271 127122 005000
23272 127124 105777 177604
23273 127130 100403
23274 127132 005200
23275 127134 001373
23276 127136 104000
23277 127140
23278
23279
23280
23281
23282
23283 127140
23284 127140 032737 000001 001020
23285 127146 001403
23286 127150 005737 001006
23287 127154 001015
23288 127156
23289 127156 005077 177554
23290 127162 105777 177546
23291 127166 100375
23292 127170 005077 177542
23293 127174 000240
23294 127176 000005
23295 127200 105777 177530
23296 127204 100401
23297 127206 104000
23298
23299
23300
23301
23302

```
BR 4$
1$:
EMT
4$: MOV R3,@#4 ;RESTORE TIMEOUT VECTOR

:*****
:TEST 633 TEST THAT TCSR2 BIT7(DONE) CLEARS WHEN XBUF IS LOADED
:*****
TS633:
BIT #1,@#SENV ;ARE WE RUNNING UNDER APT
BEQ 70$ ;IF NO THEN DO TEST
TST @#SPASS ;IS THIS FIRST PASS
BNE TS634 ;IF NO THEN SHIP TO NEXT TEST
70$:
CLR @TBUF2 ;LOAD XBUF
TSTB @TCSR2 ;CHECK DONE
BPL 3$ ;BR IF CLEAR
;FILL SECOND BUFFER BECUASE REFRESH COULD CAUSE
;FIRST TEST TO FAIL
;FILL DOUBLE BUFFER
CLR @TBUF2 ;CHECK DONE
TSTB @TCSR2
BPL 3$
3$: CLR R0 ;CLEAR TIMER
4$: TSTB @TCSR2 ;CHECK FOR XMIT DONE
BMI 5$ ;IF DONE SETS, BR TO END OF TEST
INC R0 ;INCREMENT TIMER
BNE 4$
5$:

:*****
:TEST 634 TEST THAT TCSR2 'DONE' SETS WITH RESET
:*****
TS634:
BIT #1,@#SENV ;ARE WE RUNNING UNDER APT
BEQ 70$ ;IF NO THEN DO TEST
TST @#SPASS ;IS THIS FIRST PASS
BNE TS635 ;IF NO THEN SHIP TO NEXT TEST
70$:
CLR @TBUF2 ;LOAD TRANSMIT BUFFER
1$: TSTB @TCSR2 ;WAIT FOR DONE
BPL 1$
CLR @TBUF2 ;LOAD SECOND BUFFER
NOP
RESET ;SET DONE WITH RESET
TSTB @TCSR2 ;CHECK FOR DONE SET
BMI TS635
EMT ;

:*****
:TEST 635 TEST ABILITY TO ACCESS RCSR2
```


23303
23304 127210
23305 127210 013703 000004
23306 127214 012737 127230 000004
23307 127222 005777 177502
23308 127226 000401
23309 127230
23310 127230 104000
23311 127232 010337 00G004
23312
23313
23314
23315
23316
23317 127236
23318 127236 013703 000004
23319 127242 012737 127256 000004
23320 127250 005777 177456
23321 127254 000401
23322 127256
23323 127256 104000
23324 127260 010337 000004
23325
23326
23327
23328
23329
23330
23331
23332
23333
23334 127264
23335 127264 032777 000001 177442
23336 127272 001401
23337 127274 104000
23338 127276 052777 000001 177430
23339 127304 032777 000001 177422
23340 127312 001001
23341 127314 104000
23342 127316 042777 000001 177410
23343 127324 032777 000001 177402
23344 127332 001401
23345 127334 104000
23346 127336
23347 127336 032737 000001 001020
23348 127344 001403
23349 127346 005737 001006
23350 127352 001011
23351 127354
23352 127354 052777 000001 177352
23353 127362 000005
23354 127364 032777 000001 177342
23355 127372 001401
23356 127374 104000
23357
23358

```
*****  
TS635:  MOV    @#4,R3          ;SAVE TIMEOUT VECTOR  
        MOV    #1$,@#4      ;SET UP TIMEOUT VECTOR  
        TST    @RCSR2       ;ACCESS RCSR  
        BR     2$  
1$:     EMT  
2$:     MOV    R3,@#4       ;RESTORE TIMEOUT VECTOR  
*****  
:TEST 636 TEST ABILITY TO ACCESS RBUF2  
*****  
TS636:  MOV    @#4,R3          ;SAVE TIMEOUT VECTOR  
        MOV    #1$,@#4      ;SET UP TIMEOUT VECTOR  
        TST    @RBUF2       ;ACCESS RBUF  
        BR     2$  
1$:     EMT  
2$:     MOV    R3,@#4       ;RESTORE TIMEOUT VECTOR  
*****  
:TEST 637 TEST THAT BIT0(BREAK BIT) CAN BE SET & CLEARED & RESET  
*****  
TS637:  BIT    #BIT0,@TCSR2  ;CHECK BIT0 OF TCSR CLEAR  
        BEQ   3$  
3$:     EMT  
        BIS   #BIT0,@TCSR2  ;SET BIT0 IN TCSR  
        BIT   #BIT0,@TCSR2  ;TEST BIT0 OF TCSR  
        BNE   4$  
4$:     EMT  
        BIC   #BIT0,@TCSR2  ;CLEAR BIT0 OF TCSR  
        BIT   #BIT0,@TCSR2  ;TEST BIT0 OF TCSR  
        BEQ   7$  
7$:     EMT  
        BIT   #1,@#SENV     ;ARE WE RUNNING UNDER APT  
        BEQ   70$          ;IF NO THEN DO TEST  
        TST   @#SPASS      ;IS THIS FIRST PASS  
        BNE   TS640        ;IF NO THEN SHIP TO NEXT TEST  
70$:    BIS   #BIT0,@TCSR2  ;SET BIT0 IN TCSR  
        RESET ;CLEAR BIT0 WITH RESET  
        BIT   #BIT0,@TCSR2  ;TEST BIT0 CLEAR  
        BEQ   TS640  
        EMT  
;
```

```
23359 :*****  
23360 :TEST 640 TEST THAT BIT6(XMIT INT EN) CAN BE SET & RESET  
23361 :*****  
23362 TS640:  
23363 127376 017703 177342 MOV @TVECT2,R3 ;SAVE XMIT VECTOR  
23364 127402 012777 127424 177334 MOV #1$,@TVECT2 ;SET UP INTERRUPT VECTOR FOR ERROR REPORT  
23365 127410 106427 000340 MTPS #340 ;SET PSW TO PRIORITY 7  
23366 127414 032777 000100 177312 BIT #BIT6,@TCSR2 ;TEST BIT6 OF TCSR  
23367 127422 001401 BEQ 2$  
23368 127424 1$:  
23369 127424 104000 EMT ;  
23370 127426 052777 000100 177300 2$: BIS #BIT6,@TCSR2 ;SET BIT6 OF TCSR  
23371 127434 032777 000100 177272 BIT #BIT6,@TCSR2 ;TEST BIT6 OF TCSR  
23372 127442 001001 BNE 3$  
23373 127444 104000 EMT ;  
23374 127446 042777 000100 177260 3$: BIC #BIT6,@TCSR2 ;CLEAR BIT6 OF TCSR  
23375 127454 032777 000100 177252 BIT #BIT6,@TCSR2 ;TEST BIT6 OF TCSR  
23376 127462 001401 BEQ 4$  
23377 127464 104000 EMT ;  
23378 127466 032737 000001 001020 4$: BIT #1,@SENV ;ARE WE RUNNING UNDER APT  
23379 127474 001403 BEQ 70$ ;IF NO THEN DO TEST  
23380 127476 005737 001006 TST @SPASS ;IF THIS FIRST PASS  
23381 127502 001011 BNE 5$ ;IF NO THEN SKIP TO END OF TEST  
23382 127504 70$:  
23383 127504 052777 000100 177222 BIS #BIT6,@TCSR2 ;SET BIT6 OF TCSR  
23384 127512 000005 RESET ;CLEAR BIT6 WITH RESET  
23385 127514 032777 000100 177212 BIT #BIT6,@TCSR2 ;TEST BIT6 OF TCSR  
23386 127522 001401 BEQ 5$  
23387 127524 104000 EMT ;  
23388 127526 010377 177212 5$: MOV R3,@TVECT2 ;RESTORE XMIT VECTOR  
23389  
23390  
23391
```

```
23392 :*****  
23393 :TEST 641 TEST THAT BIT6 OF RCSR2 CAN BE SET & RESET  
23394 :*****  
23395 TS641:  
23396 127532 017703 177202 MOV @RVECT2,R3 ;SAVE RECEIVE VECTOR  
23397 127536 012777 127560 177174 MOV #1$,@RVECT2 ;SET UP INTERRUPT VECTOR FOR ERROR REPORT  
23398 127544 106427 000340 MTPS #340 ;SET PSW TO PRIORITY 7  
23399 127550 032777 000100 177152 BIT #BIT6,@RCSR2 ;TEST BIT6 OF RCSR  
23400 127556 001401 BEQ 2$  
23401 127560 1$:  
23402 127562 104000 EMT ;  
23403 127570 052777 000100 177140 2$: BIS #BIT6,@RCSR2 ;SET BIT6 OF RCSR  
23404 127576 032777 000100 177132 BIT #BIT6,@RCSR2 ;TEST BIT6 OF RCSR  
23405 127576 001001 BNE 3$  
23406 127600 104000 EMT ;  
23407 127602 042777 000100 177120 3$: BIC #BIT6,@RCSR2 ;CLEAR BIT6 OF RCSR  
23408 127610 032777 000100 177112 BIT #BIT6,@RCSR2 ;TEST BIT6 OF RCSR  
23409 127616 001401 BEQ 4$  
23410 127620 104000 EMT ;  
23411 127622 032737 000001 001020 4$: BIT #1,@SENV ;ARE WE RUNNING UNDER APT  
23412 127630 001403 BEQ 70$ ;IF NO THEN DO TEST  
23413 127632 005737 001006 TST @SPASS ;IS THIS FIRST PASS  
23414 127636 001011 BNE 5$ ;IF NO THEN SKIP TO END OF TEST  
70$:  
5$:
```

23415 127640 052777 000100 177062
23416 127646 000005
23417 127650 032777 000100 177052
23418 127656 001401
23419 127660 104000
23420 127662 010377 177052
23421
23422
23423
23424
23425
23426
23427 127666
23428 127666 042777 000100 177040
23429 127674 017703 177044
23430 127700 012777 127722 177036
23431 127706 105777 177022
23432 127712 100375
23433 127714 106427 000140
23434 127720 000401
23435 127722
23436 127722 104000
23437 127724 012777 127744 177012
23438 127732 052777 000100 176774
23439 127740 000240
23440
23441 127742 104000
23442
23443 127744 042777 000100 176762
23444 127752 022626
23445 127754 010377 176764
23446
23447
23448
23449
23450
23451 127760
23452 127760 042777 000100 176746
23453 127766 106427 000340
23454 127772 017703 176746
23455 127776 012777 130024 176740
23456 130004 105777 176724
23457 130010 100375
23458 130012 052777 000100 176714
23459 130020 000240
23460 130022 000401
23461 130024
23462 130024 104000
23463 130026 042777 000100 176700
23464 130034 012777 130052 176702
23465 130042 106427 000140
23466 130046 000240
23467 130050 000401
23468 130052
23469 130052 104000
23470 130054 010377 176664

```
BIS #BIT6,@RCSR2 ;SET BIT6 OF RCSR
RESET ;CLEAR BIT6OF RCSR2 WITH RESET
BIT #BIT6,@RCSR2 ;TEST BIT6 OF RCSR
BCQ 5$
EMT
5$: MOV R3,@RVECT2 ;RESTORE RECEIVE VECTOR

:*****
:TEST 642 TEST THAT XMIT INTERRUPTS ONLY WHEN ENABLED
:*****
TS642:
BIC #BIT6,@TCSR2 ;CLEAR TRANSMIT INTERRUPT ENABLE
MOV @TVECT2,R3 ;SAVE XMIT VECTOR
MOV #2$,@TVECT2 ;POINT XMIT VECTOR TO ERROR REPORT
1$: TSTB @TCSR2 ;WAIT FOR DONE
BPL 1$
MTPS #140 ;SET PSW TO PRIORITY 3
BR 3$
2$: EMT
3$: MOV #4$,@TVECT2 ;SET XMIT VECTOR TO END OF TEST
BIS #BIT6,@TCSR2 ;ENABLE INTERRUPTS
NOP
EMT ;XMIT DID NOT INTERRUPT
4$: BIC #BIT6,@TCSR2 ;DISABLE INTERRUPTS
CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
MOV R3,@TVECT2 ;RESTORE XMIT VECTOR

:*****
:TEST 643 TEST THAT XMIT INTERRUPTS DO NOT OCCUR WHEN DISABLED
:*****
TS643:
BIC #BIT6,@TCSR2 ;DISABLE INTERRUPTS
MTPS #340 ;SET PSW TO PRIORITY 7
MOV @TVECT2,R3 ;SAVE XMIT VECTOR
MOV #2$,@TVECT2 ;POINT XMIT VECTOR TO ERROR REPORT
1$: TSTB @TCSR2 ;WAIT FOR DONE
BPL 1$
BIS #BIT6,@TCSR2 ;ENABLE INTERRUPT
NOP
BR 3$
2$: EMT
3$: BIC #BIT6,@TCSR2 ;CLEAR INTERRUPT ENABLE
MOV #4$,@TVECT2 ;POINT XMIT VECTOR TO ERROR REPORT
MTPS #140 ;SET PSW TO PRIORITY 3
NOP
BR 5$
4$: EMT
5$: MOV R3,@TVECT2 ;RESTORE XMIT VECTOR
```

23471
23472
23473
23474
23475
23476 130060
23477 130060 042777 000100 176646
23478 130066 017703 176652
23479 130072 017704 176650
23480 130076 012777 130136 176640
23481 130104 012777 000340 176634
23482 130112 106427 000140
23483 130116 105777 176612 1\$:
23484 130122 100375
23485 130124 052777 000100 176602
23486 130132 000240
23487
23488 130134 104000
23489
23490 130136 022626
23491 130140 012777 130164 176576 2\$:
23492 130146 106427 000140
23493 130152 000240
23494 130154 042777 000100 176552
23495 130162 000401
23496 130164
23497 130164 104000
23498 130166 010377 176552 5\$:
23499 130172 010477 176550
23500
23501
23502
23503
23504 130176
23505 130176 032737 000001 001020
23506 130204 001403
23507 130206 005737 001006
23508 130212 001043
23509 130214
23510 130214 042777 000100 176512
23511 130222 106427 000340
23512 130226 017703 176512
23513 130232 012777 130302 176504
23514 130240 052777 000100 176466
23515 130246 005077 176464
23516 130252 105777 176456 1\$:
23517 130256 100375
23518 130260 005077 176452
23519 130264 106427 000140
23520 130270 000240
23521 130272 042777 000100 176434
23522 130300 000401
23523 130302
23524 130302 104000
23525 130304 010377 176434 3\$:
23526 130310 005000

```
*****  
:TEST 644 TEST TRANSMITTER FOR DOUBLE INTERRUPTS  
*****  
TS644:  
BIC #BIT6,@TCSR2 ;CLEAR INTERRUPT ENABLE  
MOV @TVECT2,R3 ;SAVE XMIT VECTOR  
MOV @TPSW2,R4 ;SAVE XMIT PSW VECTOR  
MOV #2$,@TVECT2 ;SET UP XMIT VECTOR  
MOV #340,@TPSW2 ;SET PIO 7 AFTER INTERRUPT  
MTPS #140 ;SET PSW TO PRIORITY 3  
TSTB @TCSR2 ;WAIT FOR DONE  
BPL 1$  
BIS #BIT6,@TCSR2 ;ENABLE INTERRUPTS  
NOP  
EMT ;  
;XMIT INTERRUPT DID NOT OCCUR  
2$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT  
MOV #4$,@TVECT2 ;POINT XMIT VECTOR TO ERROR  
MTPS #140 ;SET PSW TO PRIORITY 3  
NOP ;GIVE TIME FOR ANY INTERRUPTS  
BIC #BIT6,@TCSR2 ;DISABLE INTERRUPTS  
BR 5$  
4$: EMT ;  
5$: MOV R3,@TVECT2 ;RESTORE XMIT VECTOR  
MOV R4,@TPSW2 ;RESTORE XMIT PSW VECTOR  
*****  
:TEST 645 TEST THAT XMIT INTERRUPT CLEARS WITH LOADING TBUF2  
*****  
TS645:  
BIT #1,@$ENV ;ARE WE RUNNING UNDER APT  
BEQ 70$ ;IF NO THEN DO TEST  
TST @$PASS ;IS THIS FIRST PASS  
BNE TS646 ;IF NO THEN SHIP TO NEXT TEST  
70$: BIC #BIT6,@TCSR2 ;DISABLE INTERRUPTS  
MTPS #340 ;SET PSW TO PRIORITY 7  
MOV @TVECT2,R3 ;SAVE XMIT VECTOR  
MOV #2$,@TVECT2 ;POINT XMIT VECTOR TO ERROR  
BIS #BIT6,@TCSR2 ;ENABLE INTERRUPTS  
CLR @TBUF2 ;LOAD TBUF  
TSTB @TCSR2 ;WAIT FOR DONE (INTERRUPT)  
BPL 1$  
CLR @TBUF2 ;FILL SECOND BUFFER TO RESET INT.  
MTPS #140 ;SET PSW TO PRIORITY 3  
NOP ;GIVE TIME FOR ANY INTERRUPTS  
BIC #BIT6,@TCSR2 ;DISABLE INTERRUPTS  
BR 3$  
2$: EMT ;  
3$: MOV R3,@TVECT2 ;RESTORE XMIT VECTOR  
CLR R0 ;INIT LOOP COUNTER
```

23527 130312 005200
23528 130314 001376
23529 130316 005777 176410
23530
23531
23532
23533
23534
23535 130322
23536 130322 005000
23537 130324 005077 176406
23538 130330 105777 176374
23539 130334 100403
23540 130336 005200
23541 130340 001373
23542 130342 104000
23543
23544 130344 032737 000001 001020
23545 130352 001403
23546 130354 005737 001006
23547 130360 001005
23548 130362
23549 130362 000005
23550 130364 105777 176340
23551 130370 001401
23552 130372 104000
23553 130374 005000
23554 130376 005200
23555 130400 001376
23556 130402 005777 176324
23557
23558
23559
23560
23561
23562 130406
23563 130406 005077 176324
23564 130412 105777 176312
23565 130416 100375
23566 130420 017700 176306
23567 130424 105777 176300
23568 130430 001401
23569 130432 104000
23570
23571
23572
23573
23574
23575
23576 130434
23577 130434 042777 000100 176272
23578 130442 042777 000100 176260
23579 130450 017703 176264
23580 130454 012777 130502 176256
23581 130462 106427 000140
23582 130466 005077 176244

```
4$: INC R0 ;INCREMENT COUNTER
    BNE 4$ ;UNTIL COUNTER = 0
    TST @RBUF2 ;CLEAR RECEIVER BUFFER

:*****
:TEST 646 TEST THAT RCVR DONE (?) SET & CLEAR PROPERLY
:*****
TS646:
    CLR R0 ;CLEAR A TIMER
    CLR @TBUF2 ;LOAD TRANSMIT BUFFER
    TSTB @RCSR2 ;CHECK FOR RECEIVER DONE
    BMI 6$ ;BR, IF DONE
    INC R0 ;INCREMENT TIMER, IF NOT DONE
    BNE WDONE2
    EMT ;RECEIVER DONE NEVER SET

WDONE2:
6$: BIT #1,@SENV ;ARE WE RUNNING UNDER APT
    BEQ 70$ ;IF NO THEN DO TEST
    TST @SPASS ;IS THIS FIRST PASS
    BNE 2$ ;IF NO THEN SKIP TO END OF TEST

70$: RESET ;CLEAR DONE WITH RESET
    TSTB @RCSR2 ;CHECK FOR DONE CLEAR
    BEQ 2$
    EMT ;RESET DID NOT CLEAR RCVR DONE

2$: CLR R0 ;INIT LOOP COUNTER
3$: INC R0 ;INCREMENT COUNTER
    BNE 3$ ;UNTIL COUNTER = 0
    TST @RBUF2 ;CLEAR RECEIVER BUFFER

:*****
:TEST 647 TEST THAT READING RBUF2 CLEARS RECEIVER DONE
:*****
TS647:
1$: CLR @TBUF2 ;LOAD TRANSMITTER
    TSTB @RCSR2 ;WAIT FOR RECEIVER DONE
    BPL 1$
    MOV @RBUF2,R0 ;READ RECEIVE BUFFER
    TSTB @RCSR2 ;CHECK FOR RECEIVE DONE CLEAR
    BEQ TS650
    EMT ;READING RBUF2 DID NOT CLEAR RCVR DONE

:*****
:TEST 650 TEST THAT RCVR INTERRUPTS ONLY WHEN ENABLED
:*****
TS650:
    BIC #BIT6,@TCSR2 ;DISABLE TRANSMIT INTERRUPTS
    BIC #BIT6,@RCSR2 ;DISABLE RECEIVER INTERRUPTS
    MOV @RVECT2,R3 ;SAVE RECEIVE VECTOR
    MOV #2$,@RVECT2 ;POINT RCY VECTOR TO ERROR REPORT
    MTPS #140 ;SET PSW TO PRIORITY 3
    CLR @TBUF2 ;SEND A CHARACTER
```

```
23583 130472 105777 176232 1$: TSTB @RCSR2 ;WAIT FOR RECEIVER DONE
23584 130476 100375 BPL 1$
23585 130500 000401 BR 3$
23586 130502 2$:
23587 130502 104000 EMT ;
23588 130504 012777 130524 176226 3$: MOV #4$,@RVECT2 ;POINT RCV VECTOR TO END OF TEST
23589 130512 052777 000100 176210 BIS #BIT6,@RCSR2 ;ENABLE RCV INTERRUPTS
23590 130520 000240 NOP ;GIVE ANY INTERRUPTS TIME
23591 130522 104000 EMT ;
23592 130524 042777 000100 176176 4$: BIC #BIT6,@RCSR2 ;DISABLE INTERRUPTS
23593 130532 022626 CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
23594 130534 005777 176172 TST @RBUF2 ;CLEAR CHARACTER FROM RECEIVER BUFFER
23595 130540 010377 176174 MOV R3,@RVECT2 ;RESTORE RECEIVE VECTOR
23596
23597
23598
23599 :*****
23600 :TEST 651 TEST THAT RCVR INTERRUPTS DO NOT OCCUR WHEN DISABLED
23601 :*****
23602 130544 TS651:
23603 130544 106427 000340 MTPS #340 ;SET PSW TO PRIORITY 7
23604 130550 017703 176164 MOV @RVECT2,R3 ;SAVE RECEIVE VECTOR
23605 130554 012777 130606 176156 MOV #2$,@RVECT2 ;POINT RCV VECTOR TO ERROR REPORT
23606 130562 005077 176150 CLR @TBUF2 ;SEND A CHARACTER
23607 130566 105777 176136 1$: TSTB @RCSR2 ;WAIT FOR RECEIVER DONE
23608 130572 100375 BPL 1$
23609 130574 052777 000100 176126 BIS #BIT6,@RCSR2 ;ENABLE INTERRUPTS
23610 130602 000240 NOP ;GIVE TIME FOR INTERRUPT
23611 130604 000401 BR 3$
23612 130606 2$:
23613 130610 104000 EMT ;RCVR INTERRUPTS AT PRIORITY 7
23614 130616 042777 000100 176112 3$: BIC #BIT6,@RCSR2 ;CLEAR INTERRUPT ENABLE
23615 130624 012777 130634 176114 MOV #4$,@RVECT2 ;POINT RCV VECTOR TO ERROR REPORT
23616 130630 106427 000140 MTPS #140 ;SET PSW TO PRIORITY 3
23617 130632 000240 NOP ;GIVE TIME FOR ANY INTERRUPT
23618 130634 000401 BR 5$
23619 130636 104000 EMT ;RCVR INTERRUPT REQUEST PASSED WITH BIT6 CLEAR
23620 130638 005777 176070 5$: TST @RBUF2 ;CLEAR CHARACTER FROM RECEIVER BUFFER
23621 130642 010377 176072 MOV R3,@RVECT2 ;RESTORE RECEIVE VECTOR
23622
23623
23624 :*****
23625 :TEST 652 TEST RECEIVER FOR DOUBLE INTERRUPTS
23626 :*****
23627 130646 TS652:
23628 130646 017703 176066 MOV @RVECT2,R3 ;SAVE RECEIVE VECTOR
23629 130652 017704 176064 MOV @RPSW2,R4 ;SAVE RECEIVE PSW VECTOR
23630 130656 012777 130722 176054 MOV #2$,@RVECT2 ;POINT RCV VECTOR TO CONTINUE TEST
23631 130664 012777 000340 176050 MOV #340,@RPSW2 ;SET PRIORITY TO 7 AFTER INTERRUPT
23632 130672 106427 000140 MTPS #140 ;SET PSW TO PRIORITY 3
23633 130676 005077 176034 CLR @TBUF2 ;SEND A CHARACTER
23634 130702 105777 176022 1$: TSTB @RCSR2 ;WAIT FOR RCVR DONE
23635 130706 100375 BPL 1$
23636 130710 052777 000100 176012 BIS #BIT6,@RCSR2 ;ENABLE RCV INTERRUPTS
23637 130716 000240 NOP ;GIVE SOME TIME
23638 130720 104000 EMT ;
```

23639 130722 022626
23640 130724 012777 130760 176006
23641 130732 106427 000140
23642 130736 000240
23643 130740 042777 000100 175762
23644 130746 010377 175766
23645 130752 010477 175764
23646 130756 000401
23647 130760
23648 130760 104000
23649 130762 005777 175744
23650 130766 010377 175746

2\$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
MOV #3\$,@RVECT2 ;POINT RCV VECTOR TO ERROR REPORT
MTPS #140 ;SET PSW TO PRIORITY 3
NOP ;GIVE SOME TIME
BIC #BIT6,@RCSR2 ;CLEAR INTERRUPT ENABLE
MOV R3,@RVECT2 ;RESTORE RECEIVE VECTOR
MOV R4,@RPSW2 ;RESTORE RECEIVE PSW VECTOR
BR 4\$
3\$: EMT ;
4\$: TST @RBUF2 ;CLEAR CHARACTER FROM RECEIVER BUFFER
MOV R3,@RVECT2 ;RESTORE RECEIVE VECTOR

23651
23652
23653
23654
23655

:TEST 653 TEST THAT RCVR INTERRUPT CLEARS BY READING RBUF2

TS653:

23656 130772
23657 130772 106427 000340
23658 130776 017703 175736
23659 131002 012777 131052 175730
23660 131010 052777 000100 175712
23661 131016 005077 175714
23662 131022 105777 175702
23663 131026 100375
23664 131030 005777 175676
23665 131034 106427 000140
23666 131040 000240
23667 131042 042777 000100 175660
23668 131050 000401
23669 131052
23670 131052 104000
23671 131054 010377 175660

MTPS #340 ;SET PSW TO PRIORITY 7
MOV @RVECT2,R3 ;SAVE RECEIVE VECTOR
MOV #2\$,@RVECT2 ;POINT RCV VECTOR TO ERROR REPORT
BIS #BIT6,@RCSR2 ;SET RCVR INTERRUPT ENABLE
CLR @RBUF2 ;SEND A CHARACTER
1\$: TSTB @RCSR2 ;WAIT FOR DONE (INTERRUPT)
BPL 1\$
TST @RBUF2 ;READ RBUF TO CLEAR PENDING INTERRUPT
MTPS #140 ;SET PSW TO PRIORITY 3
NOP ;ALLOW TIME FOR ANY ERRONEOUS INTERRUPT
BIC #BIT6,@RCSR2 ;NO INTERRUPT-CLEAR INT. ENABLE
BR 3\$
2\$: EMT ;
3\$: MOV R3,@RVECT2 ;RESTORE RECEIVE VECTOR

23672
23673
23674
23675
23676
23677

:TEST 654 TEST THAT RESET CLEARS RECEIVE INTERRUPT

TS654:

23678 131060
23679 131060 032737 000001 001020
23680 131066 001403
23681 131070 005737 001006
23682 131074 001036
23683 131076
23684 131076 106427 000340
23685 131102 017703 175632
23686 131106 012777 131164 175624
23687 131114 052777 000100 175606
23688 131122 012777 000377 175606
23689 131130 105777 175574
23690 131134 100375
23691 131136 000005
23692 131140 052777 000100 175562
23693 131146 106427 000140
23694 131152 000240

BIT #1,@\$ENV ;ARE WE RUNNING UNDER APT
BEQ 70\$;IF NO THEN DO TEST
TST @\$SPASS ;IS THIS FIRST PASS
BNE TS655 ;IF NO THEN SHIP TO NEXT TEST
70\$: MTPS #340 ;SET PSW TO PRIORITY 7
MOV @RVECT2,R3 ;SAVE RECEIVE VECTOR
MOV #2\$,@RVECT2 ;POINT RCV VECTOR TO ERROR REPORT
BIS #BIT6,@RCSR2 ;SET RCV INTERRUPT ENABLE
MOV #377,@RBUF2 ;SEND AN ALL 1'S CHARACTER
1\$: TSTB @RCSR2 ;WAIT FOR RCV DONE
BPL 1\$
RESET ;CLEAR RCV INTERRUPT & RBUF2
BIS #BIT6,@RCSR2 ;SET RECEIVER INTERRUPT
MTPS #140 ;SET PSW TO PRIORITY 3
NOP ;ALLOW TIME FOR AN ERRONEOUS INTERRUPT

23695 131154 042777 000100 175546
 23696 131162 000401
 23697 131164
 23698 131164 104000
 23699 131166 010377 175546
 23700
 23701
 23702
 23703
 23704
 23705 131172
 23706 131172 012700 000003
 23707 131176 005077 175534
 23708 131202 105777 175526
 23709 131206 100375
 23710 131210 005300
 23711 131212 001371
 23712 131214 032777 040000 175510
 23713 131222 001001
 23714 131224 104000
 23715 131226 032777 100000 175476
 23716 131234 001001
 23717 131236 104000
 23718 131240 005000
 23719 131242 005200
 23720 131244 001376
 23721 131246 005777 175460
 23722
 23723
 23724
 23725
 23726
 23727 131252
 23728 131252 032737 000001 001020
 23729 131260 001403
 23730 131262 005737 001006
 23731 131266 001027
 23732 131270
 23733 131270 012777 177777 175440
 23734 131276 105777 175426
 23735 131302 100375
 23736 131304 005777 175422
 23737 131310 052777 000001 175416
 23738 131316 005000
 23739 131320 105777 175404
 23740 131324 100403
 23741 131326 005200
 23742 131330 001373
 23743 131332 104000
 23744
 23745 131334 105777 175372
 23746 131340 001401
 23747 131342 104000
 23748
 23749 131344 000005
 23750

```

BIC #BIT6,@RCSR2 ;NO INTERRUPT-CLEAR INT. ENABLE
BR 3$
2$:
EMT ;
3$: MOV R3,@RVECT2 ;RESTORE RECEIVE VECTOR
  
```

```

:*****
:TEST 655 TEST THAT THE 'OR' ERROR (BIT14) & 'ERROR' (BIT15) CAN BE SET
:*****
TS655:
  
```

```

MOV #3,R0 ;SET CHARACTER COUNT TO SEND 3 CHAR.
1$: CLR @RBUF2 ;LOAD TRANSMIT BUFFER
2$: TSTB @TCSR2 ;WAIT FOR TRANSMIT DONE
BPL 2$
RO ;DECREMENT CHARACTER COUNT
DEC RO ;BR IF ALL CHARACTERS NOT TRANSMITTED
BNE 1$ ;TEST FOR 'OR' ERROR FLAG
BIT #BIT14,@RBUF2 ;
BNE 3$
EMT ;
3$: BIT #BIT15,@RBUF2 ;TEST 'ERROR' FLAG
BNE 4$
EMT ;
4$: CLR RO ;CLEAR LOOP COUNTER
5$: INC RO ;INCREMENT LOOP COUNTER
BNE 5$ ; UNTIL COUNTER = 0
TST @RBUF2 ;CLEAR CHARACTER FROM RECEIVER BUFFER
  
```

```

:*****
:TEST 656 TEST THAT BREAK TRANSMITS ALL ZEROES
:*****
TS656:
  
```

```

BIT #1,@SENV ;ARE WE RUNNING UNDER APT
BEQ 70$ ;IF NO THEN DO TEST
TST @SPASS ;IS THIS FIRST PASS
BNE TS657 ;IF NO THEN SHIP TO NEXT TEST
70$:
MOV #-1,@RBUF2 ;TRANSMIT ALL ONES TO RCVR
1$: TSTB @RCSR2 ;WAIT FOR RCVR DONE
BPL 1$
TST @RBUF2 ;CLEAR DONE (LEAVING ALL ONES IN RBUF)
BIS #BIT0,@TCSR2 ;TRANSMIT BREAK
CLR RO ;CLEAR A TIMER
2$: TSTB @RCSR2 ;WAIT FOR RCVR DONE
BMI CONT42 ;BR IF DONE
INC RO ;IF NOT, INCREMENT TIMER
BNE 2$
EMT ;BREAK DID NOT TRANSMIT ANYTHING
CONT42: TSTB @RBUF2 ;CHECK RECEIVE BUFFER FOR ZERO
BEQ 3$
EMT ;BREAK DID NOT TRANSMIT ALL ZEROES
3$: RESET ;CLEAR ERRORS
  
```



```

23751
23752
23753
23754
23755 131346
23756 131346 052777 000001 175360
23757 131354 005077 175356
23758 131360 105777 175344
23759 131364 100375
23760 131366 042777 000001 175340
23761 131374 032777 020000 175330
23762 131402 001001
23763 131404 104000
23764 131406 032777 100000 175316
23765 131414 001001
23766 131416 104000
23767 131420 005777 175306
23768
23769
23770
23771
23772 131424
23773 131424 005001
23774 131426 105201
23775 131430 010177 175302
23776 131434 005000
23777 131436 105777 175266
23778 131442 100403
23779 131444 005200
23780 131446 001373
23781 131450 104000
23782 131452 017702 175254
23783 131456 020102
23784 131460 001401
23785 131462 104000
23786 131464 105701
23787 131466 001357
23788 131470 000167 000114
23789
23790 131474 000000
23791 131476 000000
23792 131500 000000
23793 131502 000000
23794
23795
23796 131504 012737 000006 001002
23797 131512 012767 000001 047260
23798 131520 032737 000001 001020
23799 131526 001004
23800 131530 012700 131542
23801 131534 004767 001570
23802 131540 000777
23803
23804 131542 040506 046111 051125
23805 131550 020105 052504 044522
23806 131556 043516 051440 052514
  
```

```

:*****
:TEST 657 TEST THAT 'FR' ERROR CAN BE SET DURING BREAK
:*****
TS657:
  
```

```

BIS #BIT0,@TCSR2 ;SEND BREAK
CLR @TBUF2 ;TRANSMIT A CHARACTER TO TIME BREAK
1$: TSTB @RCSR2 ;WAIT FOR RCVR DONE
BPL 1$
BIC #BIT0,@TCSR2 ;CLEAR BREAK BITS
BIT #BIT13,@RBUF2 ;CHECK FOR FRAMING ERROR FLAG
2$: BNE 2$ ;BREAK DID NOT SET FRAMING ERROR
BIT #BIT15,@RBUF2 ;TEST 'ERROR' FLAG
BNE 3$
EMT ;'ERROR' FLAG DID NOT SET WITH 'DR' FLAG
3$: TST @RBUF2 ;CLEAP RECEIVER BUFFER
  
```

```

:*****
:TEST 660 TEST DATA PATHS USING WRAP CABLE
:*****
TS660:
  
```

```

CLR R1 ;CLEAR REGISTER FOR TEST DATA
1$: INCB R1 ;INCREMENT THE TEST DATA
MOV R1,@TBUF2 ;XMIT A CHARACTER
CLR R0 ;CLEAR A TIMER
2$: TSTB @RCSR2 ;WAIT FOR RECEIVER DONE
BMI 3$ ;BR IF DONE
INC R0 ;INCREMENT TIMER IF NOT
BNE 2$
3$: MOV @RBUF2,R2 ;GET RECEIVED CHARACTER
CMP R1,R2 ;COMPARE DATA
BEQ 4$
EMT ;
4$: TSTB R1 ;TEST XMIT DATA FOR ZERO
BNE 1$ ;BR, IF NOT FINISHED
JMP UNIQUE ;FINISHED TESTING DEVICES SEPARATELY
;GO TEST THEM ALL TOGETHER
  
```

```

$BDADR: 0
$BD DAT: 0
$GDADR: 0
$GD DAT: 0
  
```

```

ERROR7: MOV #6,@$FATAL ;SET UP FATAL ERROR NUMBER
MOV #1,$MSGTY ;SET FATAL ERROR FLAG
BIT #1,@$SENV ;UNDER APT ?
BNE SL2HLT ;YES
MOV #SL2MSG,R0
JSR PC,TYPE
SL2HLT: BR .
  
```

```

SL2MSG: .ASCIZ /FAILURE DURING SLU 2 TEST/<12><15>
  
```

23807 131564 031040 052040 051505
 23808 131572 005124 000015
 23809
 23810
 23811
 23812 131576 177560
 23813 131600 177564
 23814 131602 176500
 23815 131604 176504
 23816 131606 177564
 23817
 23818
 23819 131610 032777 000034 067566
 23820 131616 001402
 23821 131620 000167 001320
 23822 131624 012737 000007 001004
 23823 131632 012737 132450 000030
 23824
 23825
 23826
 23827
 23828 131640
 23829 131640 032737 000001 001020
 23830 131646 001403
 23831 131650 005737 001006
 23832 131654 001044
 23833 131656
 23834 131656 012767 000340 046112
 23835
 23836 131664 012700 131576
 23837 131670 012703 131576
 23838
 23839 131674 012701 000005
 23840 131700 005033
 23841 131702 077102
 23842 131704 012770 000100 000000
 23843 131712 012701 131576
 23844 131716 012702 000005
 23845 131722 032731 000100
 23846 131726 001006
 23847 131730 077204
 23848 131732 005030
 23849
 23850 131734 020027 131606
 23851 131740 001407
 23852 131742 000752
 23853 131744 021041
 23854 131746 001401
 23855 131750 104000
 23856
 23857 131752 062701 000002
 23858 131756 000764
 23859 131760 005000
 23860 131762 005200
 23861 131764 001376
 23862

.EVEN

DADTBL: .WORD 177560
 .WORD 177564
 .WORD 176500
 .WORD 176504
 TBLEND: .WORD 177564

UNIQUE: BIT #34,@SWR
 BEQ 1\$
 JMP ENDPAS
 1\$: MOV #7,@\$TESTN ;UPDATE TEST NUMBER FOR APT
 MOV #ERROR8,@#30 ;SET UP FOR CORRECT ERROR CALL

 :TEST 661 UNIQUE INTERNAL ADDRESS TEST

TS661:
 BIT #1,@\$SENV ;ARE WE RUNNING UNDER APT
 BEQ 70\$;IF NO THEN DO TEST
 TST @\$SPASS ;IS THIS FIRST PASS
 BNE TS662 ;IF NO THEN SHIP TO NEXT TEST
 70\$: MOV #340,PS ;WE WILL BE PLAYING WITH BIT6
 ;SO LOCK OUT EXTRANEIOUS INTERRUPTS
 MOV #DADTBL,R0 ;GET LOCATION OF FIRST REGISTER ADDRESS
 1\$: MOV #DADTBL,R3 ;MAKE R3 POINT TO LOCATION OF FIRST
 ;REGISTER ADDRESS
 MOV #5,R1 ;SET LOOP COUNTER TO CLEAR ALL REG.
 2\$: CLR @R3+ ;CLEAR A REGISTER
 SOB R1,2\$;LOOP UNTIL ALL REGISTERS CLEARED
 MOV #BIT6,@R0 ;SET TEST BIT IN DEVICE REGISTERS
 MOV #DADTBL,R1 ;GET LOCATION OF FIRST REGISTER ADDRESS
 MOV #5,R2 ;SET UP TEST LOOP COUNTER
 3\$: BIT #BIT6,@R1+ ;IS TEST BIT SET IN THIS REGISTER
 BNE 5\$;IF YES GO SEE IF THERE IS AN ERROR
 4\$: SOB R2,3\$;LOOP UNTIL ALL REGISTER CHECKED
 CLR @R0+ ;CLEAR REGISTER JUST TESTED AND POINT
 ;TO NEXT ONE
 CMP R0,#TBLEND ;ARE WE DONE TESTING
 BEQ 7\$;IF YES GO TO NEXT TEST
 BR 1\$;CONTINUE TESTING
 5\$: CMP (R0),-(R1) ;DID WE COMPARE THE REGISTER TO ITSELF?
 BEQ 6\$
 EMT ;WRITE TO 1 INTERNAL ADDRESS MODIFIED
 ;ANOTHER SO ADDRESS NOT UNIQUE
 6\$: ADD #2,R1 ;RESTORE POINTER
 BR 4\$;GET BACK IN TEST LOOP
 7\$: CLR R0 ;INITIALIZE LOOP COUNTER
 8\$: INC R0 ;INCREMENT COUNTER
 BNE 8\$; UNTIL LOOP COUNTER = 0

```

23863
23864
23865
23866
23867
23868 131766
23869 131766 032737 000001 001020
23870 131774 001405
23871 131776 005737 001006
23872 132002 001402
23873 132004 000167 001134
23874 132010 000005
23875 132012 012767 000340 045756
23876 132020 017767 174720 105332
23877 132026 017767 174706 105326
23878 132034 017767 172604 105322
23879 132042 017767 172572 105316
23880 132050 017767 172576 105312
23881 132056 005067 000360
23882 132062 005067 000356
23883 132066 005067 000354
23884 132072 012777 132210 174644
23885 132100 012777 000340 174640
23886 132106 012777 132244 174624
23887 132114 012777 000340 174620
23888 132122 012777 132200 172522
23889 132130 012777 000340 172516
23890 132136 052777 000100 174570
23891 132144 052777 000100 174556
23892 132152 012703 132544
23893 132156 052777 000100 172464
23894 132164 012701 177777
23895 132170 005067 045602
23896 132174 000001
23897 132176 000776
23898
23899 132200 005267 000242
23900 132204 000167 000056
23901
23902 132210 005267 000226
23903 132214 005201
23904 132216 010177 174514
23905 132222 026727 000214 000400
23906 132230 002403
23907 132232 042777 000100 174474
23908 132240 000167 000022
23909
23910 132244 005267 000174
23911 132250 005777 174456
23912 132254 100002
23913 132256 000005
23914
23915 132260 104000
23916 132262 117723 174444
23917
23918 132266 026727 000154 000074
  
```

```

:*****
:TEST 662 TEST ALL INTERNAL OPTIONS SIMULTANEOUSLY
:*****
TS662:
      BIT #1, @#SENV ;ARE WE RUNNING UNDER APT
      BEQ 70$ ;IF NO DO TEST
      TST @#$PASS ;IS THIS FIRST PASS
      BEQ 70$ ;IF YES DO TEST
      JMP ENDPAS ;IF NO THEN SKIP THIS TEST
70$: RESET ;CLEAR EVERY BODY
      MOV #340,PS ;SET PROCESSOR PRIORITY TO 7
      MOV @TVECT2,$TMP0
      MOV @RVECT2,$TMP1
      MOV @TVECT,$TMP2
      MOV @RVECT,$TMP3
      MOV @RTCVT,$TMP4
      CLR XMTCT2
      CLR RECCT2
      CLR TICKS
      MOV #XMIT2, @TVECT2 ;SET UP SLU2 TRANSMIT VECTOR
      MOV #340, @TPSW2 ;AND PSW
      MOV #REC2, @RVECT2 ;SET UP SLU2 RECEIVER VECTOR
      MOV #340, @RPSW2 ;AND PSW
      MOV #TICKER, @RTCVT ;SET UP RTC VECTOR
      MOV #340, @RTCPSW ;AND PSW
      BIS #BIT6, @TCSR2 ;ENABLE SLU2 XMIT INTERRUPT
      BIS #BIT6, @RCSR2 ;ENABLE SLU2 RECEIVER INTERRUPT
      MOV #BUF2, R3 ;SET UP RECEIVER BUFFER
      BIS #BIT6, @LKS ;ENABLE RTC INTERRUPTS
3$: MOV #-1, R1 ;INITIALIZE DATA FOR SLU2
      CLR PS ;DROP PROCESSOR PRIORITY TO 0
WAITIO: WAIT ;WAIT FOR INTERRUPT
      BR WAITIO
TICKER: INC TICKS ;UPDATE COUNT
      JMP IOHAND ;GO TO INTERRUPT HANDLER
XMIT2: INC XMTCT2 ;UPDATE XMIT INTERRUPT COUNT
      INC R1 ;UPDATE XMIT DATA
      MOV R1, @TBUF2 ;SEND NEXT CHARACTER
      CMP XMTCT2, #400 ;IF 256 CHARACTERS HAVE NOT
      BLT 1$ ;BEEN TRANSFERRED CONTINUE
      BIC #BIT6, @TCSR2 ;ELSE NO MORE XMIT INTERRUPTS
1$: JMP IOHAND ;GO TO INTERRUPT HANDLER
REC2: INC RECCT2 ;UPDATE RECEIVER INTERRUPT COUNT
      TST @RBUF2 ;BIT 15 SETS IF ANY ERRORS OCCURRED
      BPL 3$ ;IF BIT IS CLEAR NO ERRORS
      RESET ;CLEAR THE WORLD - STOP ALL
      ;INTERRUPTS
3$: EMT ;RECEIVER STATUS ERROR
      MOV @RBUF2, (R3)+ ;GET DATA AND STORE IT
IOHAND: CMP TICKS, #74 ;HAS 1 SEC ELAPSED
  
```

```
23919 132274 001401 BEQ 1$ ;IF YES STOP TEST
23920 132276 000002 RTI ;RETURN FROM INTERRUPT TO AWAIT NEXT
23921 132300 042777 000100 172326 1$: BIC #BIT6, @TCSR ;IF YES STOP TRANSMISSIONS
23922 132306 042777 000100 174420 BIC #BIT6, @TCSR2 ;
23923 132314 042777 000100 172326 BIC #BIT6, @LKS ;TURN OFF LINE CLOCK
23924
23925 132322 106427 000000 WAITER: MTPS #0 ;LOWER PRIORITY TO ALLOW TIME FOR RECEIVER TO FINISH
23926 132326 012705 140000 MOV #-40000,R5 ;SET UP LOOP COUNTER
23927 132332 062705 000001 1$: ADD #1, R5 ;DO LOOP UNTIL R5 = 0
23928 132336 001375 BNE 1$ ;
23929 132340 000005 RESET ;STOP EVERYONE SHOULD BE DONE
23930
23931
23932 132342 026767 000074 000074 CHECK2: CMP XMTCT2, RECCT2 ;#OF XMIT INTERRUPTS = REC INTERRUPTS
23933 132350 001401 BEQ 1$ ;
23934 132352 104000 EMT ;INTERRUPT COMPARISON ERROR
23935 132354 012703 132544 1$: MOV #BUF2, R3 ;INITIALIZE TO FIRST RECEIVED DATA
23936 132360 005001 CLR R1 ;INITIALIZE TO FIRST XMIT DATA
23937 132362 016704 000054 MOV XMTCT2, R4 ;GET # OF BYTES TRANSFERRED
23938 132366 122301 2$: CMPB (R3)+, R1 ;IS RECEIVED DATA = EXPECTED DATA
23939 132370 001401 BEQ 3$ ;
23940 132372 104000 EMT ;SLU2 DATA COMPARISON ERROR
23941 132374 005201 3$: INC R1 ;UPDATE TO NEXT GOOD DATA
23942 132376 077405 SOB R4, 2$ ;LOOP UNTIL ALL DATA CHECKED
23943 132400 016777 104754 174336 FINIE: MOV $TMP0, @TVECT2 ;RESTORE VECTORS
23944 132406 016777 104750 174324 MOV $TMP1, @RVECT2
23945 132414 016777 104744 172222 MOV $TMP2, @TVECT
23946 132422 016777 104740 172210 MOV $TMP3, @RVECT
23947 132430 016777 104734 172214 MOV $TMP4, @RTCVT
23948 132436 000167 000502 JMP ENDPAS ;FINISHED TESTING GO TO END OF PASS
23949
23950 132442 000000 XMTCT2: .WORD 0
23951 132444 000000 RECCT2: .WORD 0
23952 132446 000000 TICKS: .WORD 0
23953
23954 132450 012737 000007 001002 ERROR8: MOV #7, @#$FATAL ;SET UP FATAL ERROR NUMBER
23955 132456 012767 000001 046314 MOV #1, $MSGTY ;SET FATAL ERROR FLAG
23956 132464 032737 000001 001020 BIT #1, @#$ENV ;UNDER APT ?
23957 132472 001004 BNE COMHLT ;YES
23958 132474 012700 132506 MOV #COMMSG, R0
23959 132500 004767 000624 JSR PC, TYPE
23960 132504 000777 COMHLT: BR .
23961
23962 132506 040506 046111 051125 COMMSG: .ASCIZ /FAILURE DURING COMMON TEST/<12><15>
23963 132514 020105 052504 044522
23964 132522 043516 041440 046517
23965 132530 047515 020116 042524
23966 132536 052123 006412 000
23967 132544 .EVEN
23968
23969 132544 000200 BUF2: .BLKW 200
23970
23971
23972
23973
23974 133144 005327 ENDPAS: DEC (PC)+ ;DECREMENT TEST LOOP COUNTER
```

23975 133146 000001
23976 133150 003043
23977 133152 005267 045630
23978 133156 042767 100000 045622
23979 133164 016767 045634 177754
23980 133172 012700 133403
23981 133176 004767 000126
23982 133202 016700 044634
23983 133206 001405
23984 133210 000005
23985 133212 004710
23986 133214 000240
23987 133216 000240
23988 133220 000240
23989 133222 013737 000004 037360
23990 133230 012737 133246 000004
23991 133236 012737 000001 164000
23992 133244 000402
23993 133246 062706 000004
23994 133252 013737 037360 000004
23995 133260 000137
23996 133262 001226

```
$EOPCT: .WORD 1
        BGT $DOAGN ;IF COUNTER NOT 0 DO TEST AGAIN
        INC $PASS ;INCREMENT PASS COUNTER
        BIC #100000,$PASS ;DON'T LET IT BE NEGATIVE
        MOV $USWR,$EOPCT ;RESET TEST LOOP COUNTER
        MOV #ENDMSG,RO ;LET RO POINT TO ENDPASS MESSAGE
        JSR PC,TYPE ;GO TYPE END PASS MESSAGE
        MOV 42,RO ;GET MONITOR ADDRESS
        BEQ DOAGIN ;IF = 0 NO MONITOR SO DON'T STOP
        RESET ;IF MONITOR CLEAR THE WORLD
$ENDAD: JSR PC,(RO) ;GO TO MONITOR
        NOP ;THESE THREE LOCATIONS RESERVED
        NOP
        NOP
DOAGIN: MOV @#4,@#$TMP0
        MOV #1$,@#4
        MOV #1,@#164000
        BR 2$
1$: ADD #4,SP
2$: MOV @#$TMP0,@#4
$DOAGN: JMP @($PC)+ ;RETURN TO TEST AT LOCATION RESTRT
        .WORD RESTRT
```

23997
23998
23999

```
:*****
:*COMMON SUBROUTINES THAT ARE NEEDED BY THE PROGRAM
:*
:*****
```

24000
24001
24002
24003
24004
24005 133264 012737 133274 000024
24006 133272 000000
24007
24008 133274 012737 133264 000024
24009 133302 012706 001000
24010 133306 005737 000172
24011 133312 001004
24012 133314 012700 133364
24013 133320 004767 000004
24014 133324 000167 045676
24015
24016
24017 133330 132767 000040 045463
24018 133336 001011
24019 133340 105737 177564
24020 133344 100375
24021 133346 112037 177566
24022 133352 001372
24023 133354 105737 177564
24024 133360 100375
24025 133362 000207

```
PWRDN: MOV #PWRUF,@#24 ;SET UP POWER FAIL VECTOR FOR POWER UP
        HALT
PWRUP: MOV #PWRDN,@#24 ;SET UP POWER FAIL VECTOR FOR POWER DOWN
        MOV #STBOT,SP ;SET UP STACK
        TST @#MTFLAG ;ARE WE ON MULTI-OPTION TESTER
        BNE 1$ ;IF YES SKIP TYPE OUT
        MOV #PWRMSG,RO ;POINT RO TO POWER FAIL MESSAGE
        JSR PC,TYPE ;GO TYPE IT
1$: JMP RESTRT ;GO RESTART TEST
TYPE: BITB #40,$ENVM ;TYPE OUTS DISABLED
        BNE 3$ ;IF YES GET TO EXIT
1$: TSTB @#TTCSR ;TEST FOR PRINTER READY BIT
        BPL 1$ ;IF NOT READY WAIT FOR IT
        MOVB (RO)+,@#TPB ;WHEN READY PRINT A CHARACTER
        BNE 1$ ;IF LAST CHARACTER NOT NULL CONTINUE TYPING
2$: TSTB @#TTCSR ;WAIT FOR PRINTER TO FINISH
        BPL 2$
3$: RTS PC
```

24026
24027
24028
24029
24030

```
:*****
:*MESSAGES
:*
:*****
```

CJKDJBO 11/23-B CPU CLUSTER DIAG.
CJKDJB.P11 26-MAY-82 11:14

DNMAC X24.07-563 26-MAY-82^{D 4} 11:18 PAGE 455
T662 TEST ALL INTERNAL OPTIONS SIMULTANEOUSLY

SEQ 0454

24031					
24032	133364	047520	042527	020122	PWRMSG: .ASCIIZ /POWER FAILED/<12><15>
24033	133372	040506	046111	042105	
24034	133400	006412	000		
24035	133403	105	042116	047440	ENDMSG: .ASCIIZ /END OF PASS CJKDJBO/<12><15>
24036	133410	020106	040520	051523	
24037	133416	041440	045512	045104	
24038	133424	030102	006412	000	
24039	133431	012	051415	040524	STRMSG: .ASCIIZ <12><15>/START TESTING/<12><15>
24040	133436	052122	052040	051505	
24041	133444	044524	043516	006412	
24042	133452	000			
24043					
24044					
24045	000001				.END

A	020450	AC1	=%0000C1	AMSGLG=	000000	BBBP1	074466	BDONE	055022
AAADON	074232	AC2	=%000002	AMSGTY=	000000	BBBP2	074476	BDVHLT	054472
AAA1	073432	AC3	=%000003	AMTYP1=	000000	BBB0	074236	BDVMSG	054474
AAA10	073762	AC4	=%000004	AMTYP2=	000000	BBB1	074264	BDVTST	053266
AAA11	074012	AC5	=%000005	AMTYP3=	000000	BBB2	074316	BDVTS2	054030
AAA12	074042	AC6	=%000006	AMTYP4=	000000	BBB3	074336	BELL	= 000240
AAA13	074072	AC7	=%000007	ANSR	053754	BBB4	074370	BIC1	013656
AAA2	073462	ADC1	014414	APASS	= 000000	BBB5	074416	BIC2	013660
AAA3	073512	ADC2	014416	APRIOR=	000000	BBB6	074424	BIC3	013676
AAA4	073542	ADC3	014436	APTENV=	000001	BBB7	074304	BIS1	013722
AAA5	073572	ADC4	014440	AROUN	001562	BBBDON	115146	BIS2	013724
AAA6	073622	ADC5	014456	AROUND	025044	BBCTP1	115140	BIS3	013744
AAA7	073652	ADDW0	= 000000	ASLB1	015774	BBCTP10	115144	BITCHK	020124
AAA8	073702	ADDW1	= 000000	ASLB2	015762	BBCTP2	115114	BITCLR	020052
AAA9	073732	ADDW10	= 000000	ASLB3	015764	BBDATO	067274	BITCON	020200
AABBF0	110230	ADDW11	= 000000	ASLB4	016006	BBDDON	124444	BITSET	020070
AABDON	110250	ADDW12	= 000000	ASL1	015366	BBDONE	067424	BITST1	013612
AABTP1	110240	ADDW13	= 000000	ASL2	015370	BBD1	124224	BITST2	013614
AAB2	110226	ADDW14	= 000000	ASL3	015406	BBPATO	067304	BITST3	013632
AACDON	115072	ADDW15	= 000000	ASL4	015410	BBPAT1	067314	BIT0	= 000001
AAC10	115064	ADDW2	= 000000	ASL5	015424	BBPAT2	067324	BIT1	= 000002
AAC2	115040	ADDW3	= 000000	ASL6	015426	BBPAT3	067334	BIT10	= 002000
AADATO	065714	ADDW4	= 000000	ASL7	015452	BBPAT4	067344	BIT11	= 004000
AADBF	124210	ADDW5	= 000000	ASRB1	016032	BBPAT5	067354	BIT12	= 010000
AADDON	124214	ADDW6	= 000000	ASRB2	016044	BBPAT6	067364	BIT13	= 020000
AADONE	066014	ADDW7	= 000000	ASRB3	016046	BBP10	067404	BIT14	= 040000
AAD1	124134	ADDW8	= 000000	ASRB4	016054	BBP11	067414	BIT15	= 100000
AAPATO	065724	ADDW9	= 000000	ASRB5	016056	BBP7	067334	BIT2	= 000004
AAPAT1	065734	ADD1	014276	ASRB6	016076	BB10	066752	BIT3	= 000010
AAPAT2	065744	ADD2	014300	ASRB7	016100	BB13	066760	BIT4	= 000020
AAPAT3	065754	ADD3	014314	ASR1	015476	BB14	066770	BIT5	= 000040
AAPAT4	065764	ADD4	014316	ASR2	015500	BB15	067010	BIT6	= 000100
AAPAT5	065774	ADD5	014334	ASR3	015522	BB16	067034	BIT7	= 000200
AAPAT6	066004	ADD6	014336	ASR4	015524	BB17	067042	BIT8	= 000400
AA11	065604	ADD7	014350	ASR5	015540	BB2	066650	BIT9	= 001000
AA12	065606	ADD8	014352	ASR6	015542	BB20	067052	BRCT	001606
AA13	065624	ADD9	014372	ASR7	015572	BB21	067072	BRH	001572
AA2	065534	ADEVCT=	000000	ASTART	025534	BB22	067116	BRTAB	021242
AA20	065632	ADEVN =	000000	ASWREG=	000000	BB25	067124	BR1	021442
AA22	065656	ADONE	054766	ATESTN=	000000	BB26	067134	BR10	021604
AA23	065660	AENV	= 000000	ATRAP	023460	BB27	067154	BR11	021664
AA24	065700	AENVN =	000000	AUNIT =	000000	BB3	066654	BR12	021724
AA27	065706	AERR1	054756	AUSWR =	000000	BB30	067176	BR13	021764
AA3	065536	AFATAL=	000000	AUTO1	023552	BB31	067204	BR14	022024
AA4	065554	AMADR1=	000000	AVECT1=	000000	BB32	067214	BR15	022076
AA7	065562	AMADR2=	000000	AVECT2=	000000	BB33	067234	BR16	022110
ABASE =	000000	AMADR3=	000000	A1	054726	BB34	067256	BR17	022122
ACDW1 =	000000	AMADR4=	000000	A11	054726	BB35	067264	BR2	021462
ACDW2 =	000000	AMAMS1=	000000	A12	054742	BB4	06672	BR20	022134
ACPUOP=	000000	AMAMS2=	000000	A2	054760	BB5	066700	BR21	022146
ACTSUM	053762	AMAMS3=	000000	BA	051536	BB6	066710	BR22	022160
ACO	=%000000	AMAMS4=	000000	BBBDON	074506	BB7	066730	BR23	022172
		AMSGAD=	000000	BBBER1	074302	BCF	053750	BR3	021476

BR4	021514	CCX20	066262	CPUHLT	002164	DDD7	075544	DIVDT	075722
BR5	021532	CCX21	066306	CPUMSG	002166	DDONE	055420	DIVFSU	075122
BR6	021550	CCX24	066314	CSRMSG	051451	DDPO	070204	DIVFT	075230
BR7	021566	CCX25	066324	CSR1	051532	DDP1	070214	GLOOP	054072
BR70	023300	CCX26	066344	CSR2	051534	DDP2	070224	DI11W	023620
BR71	024014	CCX27	066366	CTRAP	023476	DDP3	070234	DNMB0A	007472
BTCON	020236	CCX3	066062	C15	055032	DDP4	070244	DNMB0B	007474
BTERR	020230	CCX30	066374	C2	055046	DDP5	070254	DNMB2A	007626
BTRAP	023526	CCX31	066404	C25	055054	DDP6	070264	DNMB2B	007630
BUF2	132544	CCX32	066424	C3	055072	DDP7	070274	DNMB2C	007636
B1	054776	CCX33	066446	C35	055100	DDP8	070304	DNMB2D	007652
B2	055000	CCX36	066454	C4	055116	DDP9	070314	DNMB2E	007654
B3	055020	CCX6	066070	C45	055122	DD10	067556	DNMB2F	007664
CARRY1	003266	CCX7	066100	C5	055140	DD11	067564	DNMB3A	007730
CC =	177776	CCX8	066120	C55	055144	DD12	067600	DNMB3B	007732
CCBDON	110310	CCX9	066142	C6	055160	DD13	067620	DNMB3C	007742
CCB10	110306	CC1	001566	C65	055166	DD14	067642	DNMB3D	007760
CCB2	110262	CC2	020332	C7	055204	DD15	067650	DNMB3E	007762
CCCDON	075240	CC3	020316	C75	055212	DD16	067664	DNMB4A	010102
CCC1	074512	CDONE	055252	C8	055230	DD17	067704	DNMB4B	010104
CCC10	074776	CHECK2	132342	C85	055234	DD18	067726	DNMB4C	010114
CCC11	075022	CHKAPT	050410	DADTBL	131576	DD2	067450	DNMB4D	010124
CCC12	075046	CHKSUM	053764	DAERR	020176	DD21	067734	DNMB4E	010126
CCC13	075072	CHKWRD	054412	DALTB1	044126	DD22	067744	DNMB4F	010134
CCC2	074536	CIS	025332	DALTB2	044162	DD23	067764	DNMB03A	007072
CCC3	074562	CISADR	025036	DALTB3	044412	DD24	070006	DNMB03B	007074
CCC4	074606	CISTRP	025022	DALTB4	044446	DD27	070014	DNMB03C	007104
CCC5	074632	CLRCD	020314	DATA	051542	DD3	067472	DNM1	007004
CCC6	074656	CLR1	014152	DBE	024644	DD30	070024	DNM1A	007524
CCC7	074702	CMPREG	050226	DBE1	024652	DD31	070044	DNM1B	007526
CCC8	074726	CMPSUB	074126	DBE2	024650	DD32	070066	DNM12	007012
CCC9	074752	CMP TMP	074222	DBE3	024664	DD35	070074	DNM2A	007556
CCDATO	066464	CMP1	014574	DBE4	024676	DD36	070110	DNM2B	007560
CCERR	020422	CMP2	014576	DBE5	024706	DD37	070130	DNM2C	007566
CCPO	066474	CMP3	014620	DDBBFO	110474	DD38	070152	DNM2D	007570
CCP1	066504	CMP4	014622	DDBDON	110504	DD41	070160	DNM3	007022
CCP10	066574	CMP5	014646	DDBTP1	110454	DD6	067500	DNM4	007036
CCP11	066604	CMP6	014650	DDBTP2	110464	DD7	067514	DNM4A	010026
CCP12	066614	CMP7	014670	DDB2	110410	DD8	067534	DNM4B	010030
CCP2	066514	CNTR	050534	DDB5	110452	DEC1	014052	DNM4C	010036
CCP3	066524	COMHLT	132504	DDCDON	115246	DEC2	014054	DNM5A	010176
CCP4	066534	COMMSG	132506	DDCTP1	115232	DEC3	014070	DNM5B	010200
CCP5	066544	COMPLE	054406	DDC10	115244	DEC4	014072	DNM5C	010210
CCP6	066554	COM1	014712	DDC2	115200	DEC5	014106	DNM6A	010252
CCP7	066564	CONCIS	025076	DDDATO	070174	DEC6	014110	DNM6B	010254
CCXDON	066624	CONT	001610	DDDDON	075732	DEC7	014132	DNM6C	010264
CCX12	066150	CONTIN	001220	DDDONE	070324	DERR1	055354	DNM7A	010330
CCX13	066160	CONT42	131334	DD1	075244	DERR2	055322	DNM7B	010332
CCX14	066200	CON1	020340	DD2	075304	DEVADR	051530	DNM7C	010342
CCX15	066224	CON2	020424	DD3	075344	DEVECT	051524	DOAGIN	133222
CCX18	066232	CORH	023520	DD4	075404	DEV1	051526	DOPB2A	007312
CCX19	066242	COUNT	025412	DD5	075444	DISPRE	000174	DOPB2B	007344
CCX2	066040	COUNTR	053752	DD6	075504	DIVDSU	075610	DOP0A	006640

DOPOB	006656	EEP3	070542	ESR29	052310	FDAT00	055764	FXDAT4	056016
DOPOC	006670	EEP4	070552	ESR3	050576	FDAT01	055766	FXDAT5	056020
DOPOD	006712	EERRO	055464	ESR30	052346	FDAT02	055770	FXDAT6	056022
DOPO3A	006752	EE12	070464	ESR31	052344	FDAT03	055772	FXDAT7	056024
DOPO3B	006754	EE2	070350	ESR32	052566	FDAT04	055774	F10	055654
DOP1	007212	EE3	070372	ESR33	052404	FDAT05	055776	F11	055656
DOP2	007262	EE6	070400	ESR34	051374	FDAT06	056000	F12	055672
DOP4	011324	EE7	070414	ESR35	052460	FDAT07	056002	F13	055716
DOP5	011372	EE8	070434	ESR36	126260	FDONE	056026	F135	055702
DPAT3	062146	EE9	070456	F3R37	126276	FERR20	055740	F14	055724
DRLP	054410	EISEND	037340	ESR38	126314	FFBBF1	110756	F2	055514
DTRAP1	023516	EMTA =	104377	ESR39	126310	FFBDON	110770	F22	055726
DUMMY =	000000	ENDMSG	133403	ESR4	051062	FFBTP1	110746	F3	055534
D1	055276	ENDPAS	133144	ESR40	053370	FFB10	110766	F4	055536
D2	055310	ENT176	031222	ESR41	053412	FFB2	110706	F5	055552
D3	055312	ENT51	027146	ESR42	053434	FFCDON	115454	F6	055620
D4	055316	ER	001604	ESR43	053454	FFCTP1	115432	F7	055634
D5	055324	ERRORA	051332	ESR44	053474	FFCTP2	115442	GAND0	060414
D6	055340	ERRORB	053160	ESR45	053516	FFC10	115452	GAND1	060416
D7	055350	ERRORC	054436	ESR46	053536	FFC2	115406	GAND2	060420
D8	055404	ERRORD	054532	ESR47	053556	FFDATO	070726	GAND3	060422
D9	055416	ERORE	051520	ESR48	053604	FFDONE	071006	GCMP	060344
EDONE	055466	ERROR1	002130	ESR49	053616	FFFON	077016	GDATA0	060434
EEBBF0	110620	ERROR2	002466	ESR5	051064	FFF1	076470	GDATA1	060436
EEBBF1	110630	ERROR3	050434	ESR50	053646	FFF2	076530	GDATA2	060440
EEBDON	110642	ERROR4	124476	ESR51	126274	FFF3	076570	GDATA3	060442
EEBTP1	110610	ERROR5	125720	ESR52	126326	FFF4	076630	GDONE	060444
EEB10	110640	ERROR6	126640	ESR53	126340	FFP0	070736	GFLAG1	060370
EEB2	110550	ERROR7	131504	ESR54	126362	FFP1	070746	GFLAG2	060372
EECDON	115354	ERROR8	132450	ESR55	126366	FFP2	070756	GGBBF0	111104
EECTP1	115330	ERRVEC =	000004	ESR57	051122	FFP3	070766	GGBBF1	111114
EECTP2	115340	ESR0	050714	ESR6	051176	FFP4	070776	GGBDON	111126
EECT10	115352	ESR1	050656	ESR7	051200	FF10	070716	GGBTP1	111074
EECT2	115276	ESR10	053342	ESR8	051254	FF2	070606	GGB10	111124
EEDATO	070500	ESR11	051640	ESR9	051234	FF3	070630	GGB2	111034
EEDONE	070562	ESR12	052674	ESR98	052034	FF4	070636	GGCDON	115550
EEEDON	076464	ESR13	052706	ESR99	052006	FF5	070646	GGCTP1	115534
EEE1	075736	ESR14	052710	EXADHT	051404	FF6	070666	GGC10	115546
EEE10	076222	ESR15	053046	EXATST	050540	FF7	070710	GGC2	115502
EEE11	076246	ESR16	053124	EXPSUM	053760	FJNIE	132400	GGDATO	072444
EEE12	076272	ESR17	053026	EXTMSG	051406	FINISH	025376	GGDONE	072564
EEE13	076316	ESR18	053102	E1	055432	FIRST =	X000005	GGERO	071712
EEE2	075762	ESR19	051264	E3	055440	FOVER	022720	GGER14	072350
EEE3	076006	ESR2	050726	E4	055442	FPEXIT	124472	GGGDON	101602
EEE4	076032	ESR20	052606	F =	000063	FPHLT	124532	GGG1	100652
EEE5	076056	ESR21	052114	FDAT10	055742	FPMSG	124534	GGG10	101202
EEE6	076102	ESR22	052150	FDAT11	055744	FPP	025366	GGG11	101232
EEE7	076126	ESR23	052152	FDAT12	055746	FPSTRT	054642	GGG12	101262
EEE8	076152	ESR24	052214	FDAT13	055750	FPVECT =	000244	GGG13	101312
EEE9	076176	ESR25	052546	FDAT14	055752	FXDAT0	056006	GGG14	101342
EEPO	070510	ESR26	052534	FDAT15	055754	FXDAT1	056010	GGG2	100702
EEP1	070522	ESR27	052250	FDAT16	055756	FXDAT2	056012	GGG3	100732
EEP2	070532	ESR28	052252	FDAT17	055760	FXDAT3	056014	GGG4	100762

GGG5	101012	GPAT12	060410	HCLR1	061130	HH12	071176	HX4	072642
GGG6	101042	GPAT13	060412	HCMP	061064	HH13	071220	HX7	072650
GGG7	101072	GRESET	060324	HCMP1	061104	HH16	071225	HX8	072664
GGG8	101122	GSETUP	060246	HCMP2	061112	HH17	071242	HX9	072706
GGG9	101152	GS1	060276	HDAT1	061262	HH18	071262	H1	060450
GGP1	072454	G1	056662	HDAT2	061272	HH19	071304	H10	060712
GGP2	072464	G10	057114	HDAT3	061302	HH2	071032	H11	060744
GGP3	072474	G11	057116	HDAT4	061312	HH20	071312	H12	060776
GGP4	072504	G12	057162	HDAT5	061322	HH21	071326	H2	060470
GGP5	072514	G13	057214	HDONE	061332	HH22	071346	H3	060500
GGP6	072524	G14	057216	HERE =	000000	HH23	071370	H4	060552
GGP7	072534	G15	057262	HFLAG	061140	HH24	071376	H5	060574
GGP8	072544	G16	057314	HHBBF0	111252	HH3	071054	H6	060626
GGP9	072554	G17	057316	HHBBF1	111262	HH6	071062	H7	060660
GG10	072006	G2	056714	HHBDON	111274	HH7	071072	IDATIO	056320
GG11	072034	G20	057362	HHBTP1	111232	HH8	071112	IDAT11	056322
GG12	072056	G21	057414	HHB10	111272	HH9	071134	IDAT12	056324
GG13	072064	G22	057416	HHB2	111172	HICORE	023412	IDAT13	056326
GG14	072100	G23	057462	HHCDON	115660	HLT =	000000	IDAT00	056310
GG15	072126	G24	057514	HHCTP1	115636	HSTD	061006	IDAT01	056312
GG16	072132	G25	057516	HHCTP2	115646	HXDATO	073316	IDAT02	056314
GG17	072166	G26	057562	HHC10	115656	HXDONE	073426	IDAT03	056316
GG18	072174	G27	057614	HHC2	115604	HXER9	073022	IDONE	056330
GG19	072224	G3	056716	HHDATO	071412	HXP1	073326	IIBBF0	111420
GG2	071614	G30	057616	HHDONE	071562	HXP2	073336	IIBBF1	111430
GG20	072252	G31	057662	HHHDON	103016	HXP3	073346	IIBDON	111442
GG21	072274	G32	057714	HH1	101606	HXP4	073356	IIBTP1	111400
GG22	072302	G33	057716	HH10	102356	HXP5	073366	IIB10	111440
GG23	072316	G34	057762	HH11	102426	HXP6	073376	IIB2	111340
GG24	072344	G35	060014	HH12	102476	HXP7	073406	IICDON	115724
GG25	072352	G36	060016	HH13	102546	HXP8	073416	IIC2	115702
GG26	072406	G37	060062	HH2	101656	HX10	072720	IIC20	115722
GG27	072414	G4	056762	HH3	101726	HX11	072742	IIDON	077320
GG3	071636	G40	060114	HH4	101776	HX14	072750	III1	077022
GG4	071644	G41	060116	HH5	102046	HX15	072764	III2	077052
GG5	071660	G42	060162	HH6	102116	HX16	073002	III3	077102
GG6	071706	G43	060214	HH7	102166	HX165	073006	III4	077132
GG7	071714	G44	060216	HH8	102236	HX17	073024	IJMP	013074
GG8	071750	G5	057014	HH9	102306	HX18	073044	IJMP4	012712
GG9	071756	G6	057016	HHP0	071422	HX19	073064	IJMP5	013044
GIN1	025110	G7	057062	HHP1	071432	HX2	072610	ILLA =	004700
GIN2	025152	HADR	061136	HHP10	071542	HX20	073072	ILLB =	000100
GIN3	025200	HA1R	061212	HHP11	071552	HX22	073114	INC1	013766
GORO	060424	HA1W	061142	HHP2	071442	HX23	073134	INC2	013770
GOR1	060426	HA2R	061222	HHP3	071452	HX26	073142	INC3	014012
GOR2	060430	HA2W	061152	HHP4	071462	HX28	073166	INC4	014014
GOR3	060432	HA3R	061232	HHP5	071472	HX29	073210	INC5	014030
GPAT00	060374	HA3W	061162	HHP6	071502	HX3	072622	INST	025400
GPAT01	060376	HA4R	061242	HHP7	071512	HX30	073216	IOHAND	132266
GPAT02	060400	HA4W	061172	HHP8	071522	HX31	073232	IPAT10	056270
GPAT03	060402	HA5R	061252	HHP9	071532	HX32	073252	IPAT11	056272
GPAT10	060404	HA5W	061202	HH10	071142	HX33	073274	IPAT12	056274
GPAT11	060406	HCLR	061120	HH11	071156	HX34	073302	IPAT13	056276

IPAT20	056300	JMP3	012656	KDAT00	062130	KKK5	100016	LLC5	117216
IPAT21	056302	JMP3A	012666	KDAT01	062132	KPATO	062140	LLC6	117254
IPAT22	056304	JMP3B	012700	KDAT02	062134	KPAT1	062142	LLC7	117312
IPAT23	056306	JMP4	012736	KDAT03	062136	KPAT2	062144	LLLDON	100646
ITRAP5=	000004	JMP4A	012746	KDONE	062150	KSP	=X000006	LLL1	100240
I1	056032	JMP4B	012760	KERSTK=	001000	KWSTR	126002	LLL2	100304
I105	056132	JMP5	013020	KIPAR0=	172340	KX2	062026	LLL3	100350
I12	056134	JMP5A	013032	KIPAR1=	172342	KX3	062030	LLL4	100414
I13	056150	JMP6	012772	KIPAR2=	172344	KX4	062032	LPAT10	062302
I14	056176	JMP6A	013004	KIPAR3=	172346	KX5	062066	LPAT11	062304
I15	056200	JMP7	013046	KIPAR4=	172350	KX6	062074	LPAT12	062306
I16	056202	JMP7A	013060	KIPAR5=	172352	KX7	062106	LPAT13	062310
I17	056216	JSRCK	013460	KIPAR6=	172354	K1	021360	LPAT20	062312
I2	056050	JSRCKA	013454	KIPAR7=	172356	K10	021376	LPAT21	062314
I20	056256	JSRCK1	013476	KIPDR0=	172300	K11	021400	LPAT22	062316
I23	056264	JSRSEQ	013456	KIPDR1=	172302	K12	021402	LPAT23	062320
I3	056074	JSR0	013114	KIPDR2=	172304	K2	021362	LSREG =	177524
I4	056076	JSR1	013120	KIPDR3=	172306	K3	021364	LTCHLT	126674
I5	056100	JSR1A	013142	KIPDR4=	172310	K4	021366	LTCMSG	126676
I6	056124	JSR2	013216	KIPDR5=	172312	K5	021370	L1	062154
JBUFO	061744	JSR2A	013246	KIPDR6=	172314	K6	021372	L2	062214
JBUF1	061746	JSR2B	013250	KIPDR7=	172316	K7	021374	L3	062216
JBUF2	061750	JSR3	013144	KKBBF0	111722	LAST	=X000001	L4	062220
JBUF3	061752	JSR3A	013204	KKBBF1	111732	LATCNT	051544	L5	062270
JDAT10	061754	JSR3B	013206	KKBDON	111744	LDAT10	062324	L6	062276
JDAT11	061756	JSR4	013272	KKBTP1	111702	LDAT11	062326	MBDM2A	010464
JDAT12	061760	JSR4A	013310	KKB10	111742	LDAT12	062330	MBDM2B	010466
JDAT13	061762	JSR4B	013312	KKB2	111640	LDAT13	062332	MBDM2C	010476
JDAT00	061764	JSR5	013362	KKCDON	117022	LDAT00	062336	MBDM2D	010510
JDAT0	061774	JSR5A	013406	KKC1	116026	LDAT01	062340	MBDM2E	010512
JDAT01	061766	JSR5B	013410	KKC10	116260	LDAT02	062342	MBDM2F	010522
JDAT02	061770	JSR6	013324	KKC11	116306	LDAT03	062344	MBDM4A	010732
JDAT03	061772	JSR6A	013350	KKC12	116334	LDCDSU	117504	MBDM4B	010742
JDAT1	061776	JSR6AD	013452	KKC13	116362	LDCFSU	116720	MBDM4C	010754
JDAT2	062000	JSR6B	013352	KKC14	116410	LDCT	117012	MBDM4D	010756
JDAT3	062002	JSR7	013422	KKC15	116436	LDONE	062346	MBDM4E	010764
JDONE	062004	JSR7A	013440	KKC16	116464	LDXSUB	120744	MDAT00	062474
JJBBF0	111562	JSR7B	013442	KKC17	116512	LDXT	121072	MDAT01	062476
JJBON	111574	J1	061644	KKC2	116054	LKS	124650	MDAT02	062500
JJBTP1	111550	J2	061662	KKC20	116540	LKSTST	126034	MDAT03	062502
JJB10	111572	J3	061664	KKC21	116566	LLBBF0	112050	MDM1A	010374
JJB2	111506	J4	061666	KKC22	116614	LLBDON	112062	MDM1B	010376
JJJDON	077702	J5	061722	KKC23	116642	LLBTP1	112040	MDM2A	010424
JJJ1	077324	J6	061730	KKC24	116670	LLB10	112060	MDM2B	010426
JJJ2	077370	J7	061742	KKC3	116102	LLB2	112004	MDM2C	010434
JJJ3	077434	KBUFO	062120	KKC4	116130	LLCDON	117576	MDM2D	010436
JJJ4	077500	KBUF1	062122	KKC5	116156	LLC1	117026	MDM3A	010560
JMPCK	013076	KBUF2	062124	KKC6	116204	LLC10	117350	MDM3B	010562
JMPERR	020262	KBUF3	062126	KKC7	116232	LLC11	117406	MDM3C	010572
JMPSEQ	013110	KDAT10	062110	KKKDON	100234	LLC12	117444	MDM3D	010604
JMPT	020252	KDAT11	062112	KKK1	077706	LLC2	117064	MDM3E	010624
JMP2	012714	KDAT12	062114	KKK3	077736	LLC3	117122	MDM4A	010666
JMP2A	012724	KDAT13	062116	KKK4	077766	LLC4	117160	MDM4B	010670

MDM4C	010676	MMC14	120444	MPAT23	062472	NEG11	004726	NPAT12	062644
MDM5A	011016	MMC15	120512	MRK1	016354	NEG12	004736	NPAT13	062646
MDM5B	011020	MMC16	120560	MRK2	016376	NEG13	004752	NPAT20	062626
MDM5C	011030	MMC17	120626	MRK3	016400	NEG14	004754	NPAT21	062630
MDM5D	011040	MMC2	117650	MRK4	016422	NEG2	014504	NPAT22	062632
MDM5E	011060	MMC20	120674	MRK5	016426	NEG20	005004	NPAT23	062634
MDM6A	011112	MMC3	117716	MRK6	016442	NEG21	005006	N1	062510
MDM6B	011114	MMC4	117764	MTFLAG	000172	NEG22	005026	N12	062560
MDM6C	011124	MMC5	120032	MTP10	020036	NEG3	014526	N13	062574
MDM6D	011136	MMC6	120100	MTPS1	016466	NEG30	005236	N14	062602
MDM6E	011160	MMC7	120146	MTPS1A	016506	NEG31	005240	N2	062534
MDM7A	011216	MMMDON	103456	MTPS2	016536	NEG32	005246	N3	062536
MDM7B	011220	MMM1	103022	MTPS3	016602	NEG33	005264	N4	062540
MDM7C	011230	MMM2	103056	MTPS4	016644	NEG34	005272	ODATI0	063024
MDM7D	011242	MMM3	103112	MTPS5	016700	NEG4	014530	ODATI1	063026
MDM7E	011260	MMM4	103146	MTPS6	016744	NEG40	005526	ODATI2	063030
MDONE	062504	MMM5	103202	MTPS7	017010	NEG41	005530	ODATI3	063032
MEM	050532	MMUHLT	050470	MULDSU	076674	NEG42	005536	ODAT00	062762
MERR3	062412	MMUMSG	050472	MULDT	077006	NEG5	014550	ODAT01	062764
MFPI0	017750	MMUTST	037372	MULFSU	076346	NEG50	005574	ODAT02	062766
MFPI0A	017770	MMVEC =	000250	MULFT	076454	NEG51	005576	ODAT03	062770
MFPS1	017040	MODDDO	104142	MVLCNT	051546	NEG52	005604	ODONE	063034
MFPS2A	017112	MODDD1	104152	M1	062352	NEG60	005644	OERR0	062732
MFPS2B	017114	MODDDOV	103746	M15	062372	NEG61	005646	OEBDON	112474
MFPS2C	017126	MODDSU	102622	M2	062376	NEG70	005700	OEBTP1	112452
MFPS3A	017166	MODDT0	102776	M3	062400	NEG71	005702	OEBTP2	112462
MFPS3B	017170	MODDT1	103006	M4	062402	NERR0	062600	OEB1C	112472
MFPS3C	017202	MODFDO	103436	M5	062436	NEXT	050756	OEB2	112406
MFPS4A	017242	MODFD1	103446	M6	062442	NBBF0	112332	OCDON	121326
MFPS4B	017244	MODFOV	103242	M7	062450	NBDON	112344	OCTB0	121310
MFPS4C	017256	MODFSU	101376	M8	062414	NBTP1	112312	OCTB1	121314
MFPS5A	017316	MODFT0	101552	N =	000307	NBTP2	112322	OOC10	121324
MFPS5B	017320	MODFT1	101562	NATBF1	114776	NB10	112342	OOC2	121256
MFPS5C	017332	MODP1	101572	NATINS	114702	NB2	112240	OODON	105154
MFPS6A	017374	MOR0	020456	NATSUB	114626	NBCDON	121214	OOT	105110
MFPS6B	017376	MOR1	020500	NBR	001576	NBCTB0	121176	OOC2	105106
MFPS6C	017410	MOR2	020534	NDATIO	062650	NBCTB1	121202	OOC3	105122
MFPS7A	017452	MOR3	020602	NDATI1	062652	NB10	121212	OOC4	105146
MFPS7B	017454	MOR4	020614	NDATI2	062654	NB2	121144	OPAT10	063014
MFPS7C	017466	MOR5	020626	NDATI3	062656	NBDON	104162	OPAT11	063016
MFPT =	000007	MOR6	020672	NDAT00	062606	NN1	103462	OPAT12	063020
MMBBF0	112176	MOR7	020704	NDAT01	062610	NN2	103536	OPAT13	063022
MMBBF1	112206	MOP8	020716	NDAT02	062612	NN3	103612	OPAT20	063000
MMBDON	112220	MOV1	013546	NDAT03	062614	NN4	103666	OPAT21	063002
MMBTP1	112156	MOV2	013550	NDONE	062660	NOCMP	054362	OPAT22	063004
MMB10	112216	MOV3	013566	NEGAT	026546	NODL	023606	OPAT23	063006
MMB2	112122	MPAT10	062454	NEG00	004650	NODL2	024234	OPAT24	063010
MMCDON	121102	MPAT11	062456	NEG01	004652	NOP =	000240	OVDNTT	077672
MMC1	117602	MPAT12	062460	NEG02	004660	NOR	023406	OVDTT	100636
MMC10	120214	MPAT13	062462	NEG03	004674	NJSUB	023502	OVFNTT	077310
MMC11	120262	MPAT20	062464	NEG04	004676	NOTEQ	054372	OVFTT	100224
MMC12	120330	MPAT21	062466	NEG1	014502	NPAT10	062640	OVUNDN	077550
MMC13	120376	MPAT22	062470	NEG10	004724	NPAT11	062642	OVUNT	100464

OVUNFN	077166	PSW	= 177776	Q2	063176	RETR3	024600	ROT3E	012134
OVUNFT	100052	PSWORD	025414	Q22HLT	053214	RET1	025222	ROT4	012172
O1	062664	PUSRM	= 030000	Q22MSG	053216	RET2	025234	ROT5	012236
O12	062734	PWRDN	133264	Q22TST	051552	RET3	025244	ROT6	012270
O13	062750	PWRMSG	133364	Q22TS1	052462	RET4	025242	ROT7	012326
O14	062756	PWRUP	133274	Q22TS2	052570	RFLAG	053756	RPSW	124642
O2	062710	P1	063040	Q22TS3	052734	RITSH	030444	RPSW2	126742
O3	062712	P2	063054	Q3	= 063200	ROL1	015156	RRBDON	113126
O4	062714	P3	= 063056	Q4	063202	ROL2	015160	RRBTP1	113102
PCNO1	020732	P4	063060	Q5	063224	ROL3	015176	RRBTP2	113112
PCN1	024510	P5	063102	RBUF	124632	ROL4	015200	RRBTP3	113122
PCN2	020750	QDAT10	063300	RBUF2	126732	ROL5	015214	RRB10	113124
PCN3	021000	QDAT11	063302	RCSR	124630	ROL6	015216	RRB2	113036
PCN4	021024	QDAT12	063304	RCSR2	126730	ROL7	015240	RRCDON	121706
PCN5	021046	QDAT13	063306	REC2	132444	RORB1	015616	RRCB0	121664
PCR	= 177520	QDAT00	063270	REC2	132244	RORB10	015700	RRCB1	121670
PDAT10	063136	QDAT01	063272	REGR1	025662	RORB11	015702	RRCB2	121700
PDAT11	063140	QDAT02	063274	REGR2	026010	RORB12	015714	RRC1U	121704
PDAT12	063142	QDAT03	063276	REGR23	030104	RORB13	015726	RRC2	121632
PDAT13	063144	QDONE	063310	REGR3	026136	RORB14	015730	RRRDON	105776
PDAT00	063146	QPAT10	063250	REGR4	026262	RORB2	015630	RRREXP	105766
PDAT01	063150	QPAT11	063252	REGR5	026412	RORB3	015632	RRRTP1	105746
PDAT02	063152	QPAT12	063254	REG01	027724	RORB4	015642	RRRTP2	105756
PDAT03	063154	QPAT13	063256	REG1	003502	RORB5	015644	RRR10	105744
PDONE	063156	QPAT20	063260	REG1A	003530	RORB7	015654	RRR2	105676
PDRTB1	044144	QPAT21	063262	REG1E	003514	ROR1	015264	RRR3	105700
PDRTB2	044176	QPAT22	063264	REG2	003562	ROR2	015266	RRR4	105724
PDRTB3	044430	QPAT23	063266	REG2B	003610	ROR3	015304	RTCPSW	124654
PDRTB4	044462	QQBDON	112766	REG2E	003574	ROR4	015306	RTCVT	124652
PNTR	054434	QQBTP1	112734	REG3	003642	ROR5	015324	RTI1	023220
PPAT10	063126	QQBTP2	112754	REG3A	003670	ROR6	015326	RTI2	023224
PPAT11	063130	QQB10	112764	REG3E	003654	ROR7	015342	RTRAP	= 000010
PPAT12	063132	QQB2	112670	REG4	003722	ROTLP1	053574	RTRAP1	= 000034
PPAT13	063134	QQCDON	121562	REG4A	003750	ROTLP2	053636	RTRAP2	= 000020
PPBDON	112624	QQCTB0	121540	REG4E	003734	ROTX	012240	RTRAP3	= 000030
PPBTP1	112602	QQCTB1	121544	REG45	030260	ROTXAD	012330	RTRAP4	= 000014
PPBTP2	112612	QQCTB2	121554	REG5	004002	ROTA	011544	RTRAP5	= 000004
PPB10	112622	QQC10	121560	REG5A	004030	ROTB	011546	RTS1	013520
PPB2	112536	QQC2	121506	REG5E	004014	ROTC	011570	RTT1	023110
PPCDON	121440	QQQBF0	105456	REG6	004062	ROT1A	011620	RTT2	023116
PPCTB0	121422	QQQBF1	105472	REG6A	004110	ROT1B	011622	RTT3	023150
PPCTB1	121426	QQQDON	105624	REG6E	004074	ROT1C	011646	RTT4	023160
PPC10	121436	QQQTP1	105506	RESET2	024304	ROT1D	011650	RTT5	023122
PPC2	121370	QQQ10	105516	RESET3	024302	ROT1E	011700	RTT6	023172
PPPBF0	105274	QQQ2	105356	REST	017536	ROT2A	011734	RVECT	124640
PPPBF1	105310	QQQ20	105520	RESTR	001226	ROT2B	011736	RVECT2	126740
PPPDON	105336	QQQ22	105532	RET	025212	ROT2C	011766	RWREG	= 177522
PPPTP1	105324	QQQ23	105564	RETA	050432	ROT2D	011770	ROTRAP	023432
PPP10	105334	QQQ24	105576	RETAT	022764	ROT2E	012024	R1ERR	003546
PPP2	105174	QQQ25	105622	RETB	023014	ROT3A	012054	R2ERR	003626
PPP3	105224	QQQ3	105406	RETC	023050	ROT3B	012056	R3ERR	003706
PPP4	105236	QQC4	105420	RETR1	024516	ROT3C	012104	R4ERR	003766
PS	= 177776	Q1	063162	RETR2	024554	ROT3D	012106	RSERR	004046

R6	=X000006	SL2HLT	131540	SOP0B	004252	STP4	024460	TBUF	124636
R6ERR	004126	SL2MSG	131542	SOP0C	004270	STP4E	024462	TBUF2	126736
R7	=X000007	SNMB0A	006016	SOP0D	004314	STRMSG	133431	TCCBF0	116002
SADR	061614	SNMB1A	006066	SOP1A	004346	STXBF	124010	TCCBF1	116012
SBC1	015050	SNMB1B	006070	SOP1B	004360	STXSUB	123676	TCCDON	116022
SBC2	015052	SNMB1C	006112	SOP2B	004516	SUB0	006620	TCC2	115742
SBC3	015070	SNMB2A	006172	SOP3A	005054	SUB0A	006622	TCC3	115772
SBC4	015072	SNMB2B	006174	SOP3B	005070	SUB1	014736	TCSR	124634
SBC5	015110	SNMB2C	006202	SOP4A	005324	SUB2	014740	TCSR2	126734
SBC6	015112	SNMB2D	006222	SOP4B	005344	SUB3	014762	TDATIO	056624
SBC7	015132	SNMB2E	006224	SOP5A	005376	SUB4	014764	TDATI1	056626
SBO	012344	SNMB3A	006324	SOP5B	005412	SUB5	015002	TDATI2	056630
SB2	012424	SNMB3B	006326	SOP6A	005432	SUB6	015004	TDATI3	056632
SB4	012506	SNMB3C	006344	SOP6B	005446	SUB7	015024	TDAT00	056614
SB5	012550	SNMB3D	006346	SOP7A	005470	SWB1	014234	TDAT01	056616
SB5A	012542	SNM0A	005774	SOP7B	005504	SWB2	014236	TDAT02	056620
SB5X	012552	SNM1A	006042	SPAT10	061620	SWB3	014254	TDAT03	056622
SB5XAD	012554	SNM2A	006136	SPAT11	061622	SWMSG	054570	TDEC1	022222
SB6	012604	SNM2B	006140	SPAT12	061624	SWR	021404	TDEC2	022236
SB6X	012606	SNM3A	006262	SPAT13	061626	SWREG	000176	TDEC3	022266
SB7	012636	SNM3B	006264	SRO	= 177572	SXT0	016136	TDEC4	022320
SB7X	012640	SNM4A	006400	SR1	= 177574	SXT1	016140	TDEC6	022334
SB7XAD	012642	SNM4B	006402	SR2	= 177576	SXT2	016170	TDEC7	022336
SCOPE	= 000240	SNM5A	006436	SR3	= 172516	SX10	061466	TDEC77	023702
SC3	020402	SNM5B	006440	SSBDON	113270	SX11	061512	TDEC8	023700
SC4	020416	SNM6A	006476	SSBTP1	113244	SX12	061520	TDONE	056634
SDAT00	061630	SNM6B	006500	SSBTP2	113254	SX2	061370	TEMP1	025416
SDAT01	061632	SNM7A	006534	SSBTP3	113264	SX3	061372	TEMP2	025420
SDAT02	061634	SNM7B	006536	SSB10	113266	SX4	061376	TEMP3	025422
SDAT03	061636	SOB1	016270	SSB2	113200	SX5	061422	TEMP4	025424
SDONE	061640	SOB2	016276	SSCDON	122032	SX6	061430	TEMP5	025426
SERRO	061524	SOB3	016300	SSCTB0	122014	SX7	061434	TEMP6	025430
SERR1	061432	SOB4	016320	SSCTB1	122020	SX8	061460	TEMSAV	025042
SERR10	061542	SOPA	005730	SSC10	122030	SX9	061462	TESTN1	021406
SERR4	061574	SOPB	005750	SSC2	121752	S1	025432	TEST1	014172
SETBR	001456	SOPB0A	004324	SSSA1	106140	S10	025454	TEST2	014174
SETCC	001476	SOPB0B	004334	SSSBF0	106130	S11	025456	TEST3	014212
SETCD	020400	SOPB1A	004376	SSSDON	106162	S2	025434	THRPR1	025464
SETREG	050134	SOPB1B	004414	SSSTP1	106150	S3	025436	TICKER	132200
SETUP	001440	SOPB1C	004434	SSS10	106160	S4	025440	TICKS	132446
SET2BR	001546	SOPB1D	004454	SSS2	106050	S5	025442	TKB	= 177562
SFPTBL	053666	SOPB2A	004540	START	001200	S6	025444	TMP	050526
SHL	003314	SOPB2B	004560	STATUS=	177776	S7	025446	TMP1	050530
SHLE	003330	SOPB2C	004604	STBOT	001000	S8	025450	TOFF	050046
SHR	003374	SOPB2D	004630	STCDF5	107730	S9	025452	TON	050102
SHRE	003410	SOPB3A	005114	STCDT	110060	TAB	=X000003	TONT1	023302
SIMG0A	051550	SOPB3B	005140	STCFDS	107276	TABLE	025256	TO10	021340
SIXDLR	054312	SOPB3C	005162	STCF1	107426	TABLES	054432	TO114	021352
SKTST2	024322	SOPB3D	005204	STCIBF	123414	TBIT	= 000020	TO14	021342
SLU1ST	124656	SOPX	005722	STCSUB	123266	TBITPS	037356	TO20	021344
SLU2ST	126750	SOPXAD	005752	STP	022722	TBLEND	131606	TO24	021354
SL1HLT	125754	SOPZA	004476	STP3	023356	TEL1	011336	TO250	021356
SL1MSG	125756	SOP0A	004240	STP3D	023360	TBL2	011404	TO30	021346

TO34	021350	TST206	031750	TST273	036304	TS121	007046	TS20	003332
TO4	021336	TST207	032026	TST274	036352	TS122	007106	TS200	012460
TPAT10	056574	TST210	032104	TST275	036430	TS123	007126	TS201	012514
TPAT11	056576	TST211	032162	TST276	036504	TS124	007146	TS202	012556
TPAT12	056600	TST212	032236	TST277	036560	TS125	007166	TS203	012610
TPAT13	056602	TST213	032312	TST300	036634	TS126	007222	TS204	012644
TPAT20	056604	TST214	032366	TST301	036706	TS127	007244	TS205	013112
TPAT21	056606	TST215	032444	TST302	036760	TS13	003160	TS206	013500
TPAT22	056610	TST216	032522	TST303	037032	TS130	007272	TS207	013530
TPAT23	056612	TST217	032576	TST304	037106	TS131	007322	TS21	003360
TPB =	177566	TST220	032652	TST305	037162	TS132	007354	TS210	013570
TPS =	177564	TST221	032722	TST306	037234	TS133	007400	TS211	013634
TPSW	124646	TST222	032772	TST37	026574	TS134	007424	TS212	013700
TPSW2	126746	TST223	033042	TST40	026630	TS135	007450	TS213	013746
TRACE	023276	TST224	033116	TST41	026644	TS136	007502	TS214	014032
TRAPA =	000077	TST225	033166	TST42	026662	TS137	007534	TS215	014134
TRAPB	023554	TST226	033236	TST43	026716	TS14	003200	TS216	014154
TRAPC =	104777	TST227	033312	TST44	026750	TS140	007600	TS217	014214
TRAPPC	037346	TST230	033362	TST45	027002	TS141	007674	TS22	003412
TRAPPS	037350	TST231	033432	TST46	027034	TS142	007774	TS220	014256
TRAP10	025044	TST232	033502	TST47	027072	TS143	010050	TS221	014374
TRCSR =	177560	TST233	033556	TST50	027122	TS144	010144	TS222	014460
TRC1	023346	TST234	033626	TST51	027174	TS145	010222	TS223	014552
TRPADR	023430	TST235	033676	TST52	027234	TS146	010276	TS224	014672
TRT =	000003	TST236	033746	TST53	027274	TS147	010354	TS225	014714
TRO	024012	TST237	034022	TST54	027334	TS15	003220	TS226	015026
TR2	024016	TST240	034072	TST55	027372	TS150	010404	TS227	015134
TR3	024124	TST241	034142	TST56	027430	TS151	010446	TS23	003424
TR4	024130	TST242	034232	TST57	027466	TS152	010534	TS230	015242
TR5	024126	TST243	034302	TST60	027526	TS153	010644	TS231	015344
TSTSPC	037306	TST244	034352	TST61	027566	TS154	010706	TS232	015454
TST160	030500	TST245	034422	TST62	027624	TS155	010774	TS233	015574
TST161	030526	TST246	034470	TS1	001440	TS156	011070	TS234	015742
TST162	030544	TST247	034536	TS10	002734	TS157	011172	TS235	016010
TST163	030562	TST250	034604	TS100	005776	TS16	003246	TS236	016110
TST164	030624	TST251	034654	TS101	006020	TS160	011272	TS237	016172
TST165	030662	TST252	034724	TS102	006044	TS161	011340	TS24	003440
TST166	030706	TST253	034772	TS103	006114	TS162	011406	TS240	016260
TST167	030736	TST254	035040	TS104	006150	TS163	011454	TS241	016322
TST170	030774	TST255	035114	TS105	006234	TS164	011522	TS242	016444
TST171	031020	TST256	035170	TS106	006274	TS165	011572	TS243	016510
TST172	031046	TST257	035250	TS107	006356	TS166	011702	TS244	016546
TST173	031070	TST260	035324	TS11	003122	TS167	012026	TS245	016612
TST174	031126	TST261	035400	TS110	006410	TS17	003300	TS246	016652
TST175	031162	TST262	035460	TS111	006450	TS170	012136	TS247	016710
TST176	031234	TST263	035534	TS112	006506	TS171	012174	TS25	003454
TST177	031302	TST264	035610	TS113	006544	TS172	012242	TS250	016754
TST200	031350	TST265	035664	TS114	006562	TS173	012272	TS251	017020
TST201	031422	TST266	035744	TS115	006600	TS174	012332	TS252	017066
TST202	031470	TST267	036020	TS116	006630	TS175	012354	TS253	017136
TST203	031536	TST270	036074	TS117	006724	TS176	012400	TS254	017212
TST204	031610	TST271	036154	TS12	003140	TS177	012434	TS255	017266
TST205	031656	TST272	036230	TS120	006770	TS2	001662	TS256	017342

TS257	017420	TS335	023626	TS413	046066	TS472	072570	TS550	113132
TS26	003470	TS336	023726	TS414	046562	TS473	073432	TS551	113274
TS260	017476	TS337	024036	TS415	047314	TS474	074236	TS552	113424
TS261	017536	TS34	003710	TS416	047714	TS475	074512	TS553	113560
TS262	017576	TS340	024170	TS417	047776	TS476	075244	TS554	113722
TS263	017710	TS341	024234	TS42	004130	TS477	075736	TS555	114074
TS264	017770	TS342	024322	TS420	050540	TS5	002342	TS556	115014
TS265	020052	TS343	024476	TS421	050756	TS50	004316	TS557	115076
TS266	020204	TS344	024530	TS422	051104	TS500	076470	TS56	004562
TS267	020242	TS345	024560	TS423	051552	TS501	077022	TS560	115152
TS27	003516	TS346	024610	TS424	052050	TS502	077324	TS561	115252
TS270	020264	TS347	024726	TS425	052462	TS503	077706	TS562	115360
TS271	020456	TS35	003736	TS426	052570	TS504	100240	TS563	115460
TS272	020512	TS350	024764	TS427	052734	TS505	100652	TS564	115554
TS273	020546	TS351	037456	TS43	004152	TS506	101606	TS565	115664
TS274	020636	TS352	037512	TS430	053266	TS507	103022	TS566	115730
TS275	020726	TS353	037554	TS431	053666	TS51	004336	TS567	116026
TS276	020744	TS354	037642	TS432	054700	TS510	103462	TS57	004632
TS277	020766	TS355	037700	TS433	054772	TS511	104166	TS570	117026
TS3	002004	TS356	037724	TS434	055026	TS512	105064	TS571	117602
TS30	003550	TS357	037744	TS435	055256	TS513	105160	TS572	121106
TS300	021016	TS36	003770	TS436	055424	TS514	105342	TS573	121220
TS301	021042	TS360	037764	TS437	055472	TS515	105630	TS574	121332
TS302	021064	TS361	040004	TS44	004172	TS516	106002	TS575	121444
TS303	021102	TS362	040024	TS440	056032	TS517	106166	TS576	121566
TS304	021132	TS363	040142	TS441	056334	TS52	004362	TS577	121712
TS305	021174	TS364	040176	TS442	056640	TS520	106332	TS6	002470
TS306	021424	TS365	040260	TS443	060450	TS521	106476	TS60	004704
TS307	021624	TS366	040350	TS444	061336	TS522	106634	TS600	122036
TS31	003576	TS367	040446	TS445	051644	TS523	107010	TS601	122174
TS310	022064	TS37	004016	TS446	062010	TS524	107442	TS602	122252
TS311	022204	TS370	040552	TS447	062154	TS525	110074	TS603	122330
TS312	022222	TS371	040656	TS45	004212	TS526	110134	TS604	123430
TS313	022246	TS372	041022	TS450	062352	TS527	110254	TS605	123470
TS314	022336	TS373	041206	TS451	062510	TS53	004416	TS606	124026
TS315	022372	TS374	041662	TS452	062664	TS530	110314	TS607	124134
TS316	022426	TS375	042212	TS453	063040	TS531	110354	TS61	004764
TS317	022462	TS376	042560	TS454	063162	TS532	110510	TS610	124224
TS32	003630	TS377	042734	TS455	063314	TS533	110646	TS611	124710
TS320	022516	TS4	002220	TS456	064102	TS534	110774	TS612	124736
TS321	022552	TS40	004050	TS457	064456	TS535	111132	TS613	124764
TS322	022616	TS400	043122	TS46	004232	TS536	111300	TS614	125052
TS323	022652	TS401	043244	TS460	065052	TS537	111446	TS615	125104
TS324	022734	TS402	043522	TS461	065302	TS54	004456	TS616	125132
TS325	022764	TS403	043712	TS462	065514	TS540	111600	TS617	125160
TS326	023024	TS404	044212	TS463	066020	TS541	111750	TS62	005036
TS327	023062	TS405	044476	TS464	066630	TS542	112066	TS620	125276
TS33	003656	TS406	044652	TS465	067430	TS543	112224	TS621	125370
TS330	023116	TS407	045226	TS466	070330	TS544	112350	TS622	125470
TS331	023172	TS41	004076	TS467	070566	TS545	112500	TS623	125606
TS332	023232	TS410	045364	TS47	004260	TS546	112630	TS624	126034
TS333	023316	TS411	045430	TS470	071012	TS547	112772	TS625	126062
TS334	023366	TS412	045562	TS471	071566	TS55	004520	TS626	126216

TS627	126406	TTC2	122104	UPAT12	064032	U2	063422	WPAT02	064436
TS63	005072	TTTA1	106310	UPAT13	064034	U3	063440	WPAT03	064440
TS630	126536	TTTA2	106312	UPAT20	064036	U4	063502	WVBDON	115010
TS631	127002	TTTBFO	106276	UPAT21	064040	U5	063520	WVBI	114074
TS632	127030	TTTDON	106326	UPAT22	064042	U6	063562	WVBI0	114506
TS633	127056	TTTTP1	106314	UPAT23	064044	U7	063600	WVBI1	114554
TS634	127140	TTT10	106324	UPAT30	064046	VDEC	022364	WVBI2	114142
TS635	127210	TTT2	106242	UPAT31	064050	VDEC10	022540	WVBI3	114210
TS636	127236	TTT3	106266	UPAT32	064052	VDEC11	022574	WVBI4	114256
TS637	127264	TVECT	124644	UPAT33	064054	VDEC12	022600	WVBI5	114324
TS64	005142	TVECT2	126744	UPAT40	064056	VDEC13	022640	WVBI6	114372
TS640	127376	TYPE	133330	UPAT41	064060	VDEC14	022644	WVBI7	114440
TS641	127532	T105	056434	UPAT42	064062	VDEC2	022360	WVCDON	123424
TS642	127666	T12	056436	UPAT43	064064	VDEC3	022420	WVCI	122330
TS643	127760	T13	056452	UROM1	064074	VDEC4	022414	WVCI0	122634
TS644	130060	T14	056500	UROM2	064076	VDEC5	022454	WVCI1	122670
TS645	130176	T15	056502	UROM3	064100	VDEC6	022450	WVCI2	122724
TS646	130322	T16	056504	USESTK=	000600	VDEC7	022510	WVCI3	122760
TS647	130406	T17	056522	USP	=X000006	VDEC8	022504	WVCI4	123014
TS65	005206	T2	056352	USP1	017552	VDEC9	022544	WVCI5	123050
TS650	130434	T20	056562	USP1A	017570	VECT1	051522	WVCI6	123104
TS651	130544	T23	056570	USP2	017626	VRTPCR	053746	WVCI7	123140
TS652	130646	T3	056376	USP3	017644	VVBDON	110350	WVCI2	122364
TS653	130772	T4	056400	USP4	017672	VVB10	110346	WVCI20	123174
TS654	131060	T5	056402	USRM	= 140000	VVB2	110322	WVCI21	123230
TS655	131172	T6	056426	UTMP1	064070	VVCBFO	122316	WVCI3	122420
TS656	131252	UDONE	064102	UTMP2	064072	VVCDON	122324	WVCI4	122454
TS657	131346	UERR0	063436	UUBDON	113554	VVCTP1	122304	WVCI5	122510
TS66	005310	UERR3	063770	UUBTP1	113470	VVC2	122272	WVCI6	122544
TS660	131424	UFLAG	064066	UUBTP2	113542	VVVBFO	106606	WVCI7	122600
TS661	131640	UIPAR0=	177640	UUB10	113552	VVVDON	106630	WVWBFO	106752
TS662	131766	UIPAR1=	177642	UUB2	113466	VVVTP1	106616	WVWBFO1	106772
TS67	005346	UIPAR2=	177644	UUCBFO	122240	VVV10	106626	WVWDON	107004
TS7	002612	UIPAR3=	177646	UUCDON	122246	VVV2	106550	WVWTP1	106762
TS70	005414	UIPAR4=	177650	UUCTP1	122226	WAITER	132322	WVW10	107002
TS71	005450	UIPAR5=	177652	UUC2	122214	WAIT10	132174	WVW2	106714
TS72	005506	UIPAR6=	177654	UUUA1	106454	WASR6	037344	W10	064244
TS73	005544	UIPAR7=	177656	UUUA2	106456	WASSR0	037352	W11	064256
TS74	005614	UIPDR0=	177600	UUUBFO	106442	WASSR2	037354	W12	064276
TS75	005654	UIPDR1=	177602	UUUDON	106472	WATE	024440	W13	064322
TS76	005720	UIPDR2=	177604	UUUTP1	106460	WATE1	024372	W14	064330
TS77	005754	UIPDR3=	177606	UUU10	106470	WATE2	024406	W15	064342
TTBDON	113420	UIPDR4=	177610	UUU2	106406	WATE3	024434	W16	064366
TTBTTP1	113376	UIPDR5=	177612	UUU3	106432	WBIT	= 000100	W17	064412
TTBTTP2	113406	UIPDR6=	177614	U0	063330	WC	051540	W2	064122
TTB10	113416	UIPDR7=	177616	U1	063352	WDAPO0	064442	W20	064420
TTB2	113332	UNIQUE	131610	U10	063642	WDAT01	064444	W3	064146
TTCDON	122170	UPAT00	064016	U11	063644	WDAT02	064446	W4	064154
TTCR =	177564	UPAT01	064020	U12	063662	WDAT03	064450	W5	064166
TTCTB0	122146	UPAT02	064022	U13	063700	WDONE	064452	W6	064212
TTCTB1	122152	UPAT03	064024	U14	063732	WDONE2	130330	W7	064236
TTCTB2	122162	UPAT10	064026	U15	063760	WPAT00	064432	X	124436
TTC10	122166	UPAT11	064030	U16	063762	WPAT01	064434	XAPT11	065030

XBUF	104750	XT4	104306	YDAT01	065240	Y3	065144	\$DEVCT	001010
XDAT00	065006	XT4A	104336	YDAT02	065242	Y4	065164	\$DOAGN	133260
XDAT01	065010	XT4B	104350	YDAT03	065244	Y5	065224	\$ENDAD	133212
XDAT02	065012	XT5	104366	YDONE	065276	Z	124442	\$ENV	001020
XDAT03	065014	XT5A	104406	YFLAG	065226	ZDAT00	065450	\$ENVM	001021
XDONE	065046	XT5B	104420	YNTAB	021300	ZDAT01	065452	\$EOPCT	133146
XMI12	132210	XT6	104436	YPAT00	065246	ZDAT02	065454	\$ETABL	001020
XMTCT2	132442	XT6A	104462	YPAT01	065250	ZDAT03	065456	\$ETEND	001030
XOR1	016226	XT6B	104474	YPAT02	065252	ZDONE	065510	\$FATAL	001002
XOR2	016230	XT7	104512	YPAT03	065254	ZFLAG	065442	\$GDADR	131500
XOR3	016256	XT8	104540	YPAT10	065256	ZPAT00	065460	\$GDDAT	131502
XPATO	105050	XT9	104566	YPA111	065260	ZPAT01	065462	\$HIBTS	001030
XPATO	104760	XXBDON	113716	YPAT12	065262	ZPAT02	065464	\$MAIL	001000
XPAT00	065016	XXBTP1	113674	YPAT13	065264	ZPAT03	065466	\$MBADR	001032
XPAT01	065020	XXBTP2	113704	YPAT20	065266	ZPAT10	065470	\$MSGAD	001014
XPAT02	065022	XXB10	113714	YPAT21	065270	ZPAT11	065472	\$MSGLG	001016
XPAT03	065024	XXB2	113622	YPAT22	065272	ZPAT12	065474	\$MSGTY	001030
XPAT1	104770	XXCDON	123464	YPAT23	065274	ZPAT13	065476	\$PASS	001006
XPAT10	065026	XXXDON	107436	YTMP1	065230	ZPAT20	065500	\$PASTM	001036
XPAT12	065032	XXX1	107010	YTMP2	065232	ZPAT21	065502	\$SETUP=	000020
XPAT13	065034	XXX2	107054	YTMP3	065234	ZPAT22	065504	\$STUP =	177777
XPAT2	105000	XXX3	107120	YYBDON	114070	ZPAT23	065506	\$SVPC =	000400
XPAT20	065036	XXX4	107164	YYBTP1	114044	ZTMP1	065444	\$SWR =	000000
XPAT21	065040	XXX5	107230	YYBTP2	114054	ZTMP2	065446	\$SWREG	001022
XPAT22	065042	X10	064616	YYBTP3	114064	ZZCBF	124116	\$TESTN	001004
XPAT23	065044	X11	064632	YYB10	114066	ZZCDON	124130	\$TMP0	037360
XPAT3	105010	X12	064652	YYB2	113772	ZZC10	124112	\$TMP1	037362
XPAT4	105020	X13	064676	YYCDON	124022	ZZC2	124034	\$TMP2	037364
XPAT5	105030	X14	064704	YYC1	123470	ZZC3	124054	\$TMP3	037366
XPAT6	105040	X15	064720	YYC2	123516	ZZZDON	110130	\$TMP4	037370
XTDONE	105060	X16	064740	YYC3	123544	ZZZ10	110126	\$TN =	000663
XT1	104166	X17	064764	YYC4	123572	ZZZ2	110102	\$TPB	025460
XT1A	104202	X2	064476	YYC5	123620	Z1	065322	\$TPS	025462
XT10	104612	X20	064772	YYC6	123646	Z2	065342	\$TSTM	001034
XT11	104640	X3	064522	YYYDON	110070	Z3	065366	\$UNIT	001012
XT12	104666	X4	064530	YYY1	107442	Z4	065374	\$UNITM	001040
XT13	104722	X5	064544	YYY2	107506	Z5	065406	\$USWR	001024
XT2	104204	X6	064564	YYY3	107552	Z6	065440	\$X =	131766
XT2A	104224	X7	064610	YYY4	107616	\$APTHD	001030	=	133453
XT2B	104242	Y	124440	YYY5	107662	\$BDADR	131474	.RSET	124572
XT3	104254	YBR	001602	Y1	065100	\$BDDAT	131476	.\$X =	001030
XT3A	104276	YDAT00	065236	Y2	065120	\$CPUOP	001026		

. ABS. 133453 000 CON RW REL LCL I

ERRORS DETECTED: 0

CJKDJB,CJKDJB/SOL/NL:TOC=SYSMAC.SML,CJKDJB.P11
 RUN-TIME: 65 77 3 SECONDS
 RUN-TIME RATIO: 273/145=1.8
 CORE USED: 48K (95 PAGES)