

FP11-F

FP11F FLTG PNT PRT B  
CKFPBCO

AH-F635C-MC  
FICHE 1 OF 1

APR 1982  
COPYRIGHT © 79-82  
MADE IN USA



The main body of the document is a large grid of approximately 15 columns and 25 rows of data. Each cell in the grid contains a small, dense table or set of data points. The text is very small and difficult to read, but the overall structure is a comprehensive data matrix. The data appears to be organized into several distinct sections or categories across the rows and columns.



.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

IDENTIFICATION

PRODUCT CODE: AC-F633C-MC  
PRODUCT NAME: CKFPBC0 FP11F FLTG PNT PRT B  
DATE CREATED: OCTOBER, 1981  
MAINTAINER: DIAGNOSTIC ENGINEERING  
AUTHOR: ANTHONY VEZZA, DAN MILLEVILLE

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

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

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

COPYRIGHT (C) 1979, 1982 BY DIGITAL EQUIPMENT CORPORATION



48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93

## HISTORY

-----

NO CHANGES TO THE 11/34 FLOATING POINT DIAGNOSTICS PART 'A' WERE FOUND TO BE NEEDED TO ADAPT IT FOR USE ON THE 11/44.

THE FOLLOWING WAS ADDED TO THE 11/34 FLOATING POINT DIAGNOSTIC TO MAKE THE 'B' VERSION COVER THE 11/44:

1. TEST 22 - PROCESSOR LOOKS TO SEE IF APT IS CONTROLLING THE TEST, AND IF IT IS, CHECKS TO SEE IF THE USER HAS SELECTED THIS TEST BY CHECKING BIT 7 IN THE SWITCH REGISTER. IT HAS ALSO BEEN CHANGED SO THAT IF BIT 7 IS \*ONE\*, THE CODE WILL SELECT THE TEST.

THE FOLLOWING WAS ADDED TO THE 11/34 FLOATING POINT DIAGNOSTIC TO MAKE THE 'C' VERSION COVER THE 11/44:

1. TEST 76 - CHECKS THAT FP PROCESSOR DOESN'T ACCESS D-SPACE UNTIL CONDITIONS WARRANT.
2. TEST 77 - CHECKS THAT SR1 MATCHES WHAT ACTUALLY HAPPENED TO THE REGISTER OF THE INSTRUCTION, AND THAT THE VALUE OF AUTO INCREMENT/DECREMENT WAS PROPER.

ALL THREE PARTS WERE RE-RELEASED WITH NEW SCOPE AND ERROR ROUTINES THAT CHECK BIT 0 OF THE CPU ERROR REGISTER (POWER MONITOR BIT). THE ADDITIONS WERE MADE IN THE SCOPE ROUTINE, EXECUTED AT THE BEGINNING OF EACH TEST. IF THE BIT BECOMES SET, AN ERROR IS CALLED FROM THE SCOPE ROUTINE. THE BIT IS CLEARED, AND THE TEST IS CONTINUED. IF THE BIT BECOMES SET IN THE MIDDLE OF A TEST, AND AN ERROR OCCURS FOR ANY REASON, THE ERROR ROUTINE WILL CALL \*TWO\* ERRORS, THE POWER MONITOR BIT ERROR FIRST, THEN THE ERROR ORIGINALLY CALLED. IN ADDITION, THE \$READ ROUTINE NOW CHECKS FOR A RANDOMLY INPUTED ^Q BEFORE A ^S IS TYPED. THIS BECAME NECESSARY WITH CERTAIN DATA CONNECTIONS OF SOME SYSTEMS.

THE FOLLOWING WAS ADDED TO THE 'D' VERSION:

THE END-OF-PASS MESSAGES ARE NOW PRINTED EVERY 200 PASSES. THESE MESSAGES CAN BE DISABLED BY TYPING ANY KEY. THEY CAN BE REENABLED BY AGAIN TYPING ANY KEY THIS IS HELPFUL FOR OVERNIGHT RUNS ON HARD COPY TERMINALS. IT ALSO MAKES THE DIAGNOSTIC HARDWARE INTENSIVE INSTEAD OF TERMINAL INTENSIVE.



CONTENTS

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  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143

- 1.    ABSTRACT
- 2.    REQUIREMENTS
  - 2.1    EQUIPMENT
  - 2.2    STORAGE
  - 2.3    PRELIMINARY PROGRAMS
- 3.    LOADING PROCEDURE
- 4.    STARTING PROCEDURE
  - 4.1    CONTROL SWITCH SETTINGS
  - 4.2    STARTING ADDRESS
  - 4.3    PROGRAM AND OPERATOR INTERACTION
- 5.    OPERATING PROCEDURE
  - 5.1    OPERATIONAL SWITCH SETTINGS
  - 5.3    OPERATOR ACTION
- 6.    ERRORS
  - 6.1    SUMMARY
  - 6.2    ERROR RECOVERY
- 7.    RESTRICTIONS
  - 7.1    STARTING RESTRICTIONS
  - 7.2    OPERATING RESTRICTIONS
- 8.    MISCELLANEOUS
  - 8.1    EXECUTION TIMES
  - 8.2    STACK POINTER
  - 8.3    PASS COUNT
  - 8.4    T-BIT TRAPPING
  - 8.5    SOFTWARE SWITCH REGISTER
  - 8.6    INTERRUPTS TEST
  - 8.7    ACT, APT AND XXDF COMPATIBILITY
- 9.    PROGRAM DESCRIPTION
  - 9.1    CKFPBCO
- 10.   LISTING
  - 10.1   CKFPBCO



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  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201

1.

ABSTRACT

## THE THREE PROGRAMS:

CKFPADO CKFPBCO CKFPCDO

ARE DESIGN TO DETECT AND REPORT LOGIC FAULTS IN THE PDP 11/44 FP11-F FLOATING POINT PROCESSOR. THE DESIGN IS AN ATTEMPT TO REACH ALL ROM STATES, TAKE ALL BRANCH MICRO TESTS (BUT'S) AND VERIFY ALL THE LOGIC. THEY CONSIST OF 157 (OCT) INDIVIDUAL TESTS SEQUENCED TO DETECT AND ATTEMPT TO IDENTIFY FAULTS WITH A MINIMUM HARDWARE OR SOFTWARE LEVEL. THE TESTS ARE PARTIONED INTO THREE STAND-ALONE PROGRAMS DESCRIBED BELOW.

NOTE THAT ERROR REPORTS IN THESE PROGRAMS ARE BASED UPON THE KNOWLEDGE THAT ALL PREVIOUS TESTS HAVE BEEN RUN AND IN MOST CASE THAT THERE IS ONLY A SINGLE POINT FAULT IN THE FP11-F. IF THE PROGRAMS OR TESTS ARE NOT RUN IN ORDER THEN ERROR MESSAGES MAY NOT BE ACCURATE.

## A. CKFPADO

## CKFPADO TESTS:

LDFPS  
STFPS  
CFCC  
SETF, SETD, SETI AND SETL  
STST  
LDF AND LDD (ALL SOURCE MODES)  
STD (MODE 0 AND 1)  
ADDF, ADDD AND SUBD (MOST CONDITIONS)

## B. CKFPBCO

## CKFPBCO TESTS:

ADDF, ADDC AND SUBD (ALL CONDITIONS NOT TESTED IN CKFPBCO)  
CMPD AND CMPF  
DIVD AND DIVF  
MULD AND MULF  
MODD AND MODF

## C. CKFPCDO

## CKFPCDO TESTS:

STF AND STD (ALL MODES)  
STCFD AND STCDF  
CLRD AND CLRF



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  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258

NEGF AND NEGD  
ABSF AND ABSD  
TSTF AND TSTD  
NEGF, ABSF AND TSTF (ALL SOURCE MODES)  
NEGF, ABSF AND TSTF (ALL SOURCE MODES)  
LDFPS (ALL SOURCE MODES)  
LDCIF AND LDCLF  
LDCID AND LDCLD  
LDEXP  
STFPS (ALL DESTINATION MODES)  
STCFL AND STCFI  
STCDL AND STCDI  
STEXP  
STST

2. REQUIREMENTS

2.1 EQUIPMENT

A PDP 11/44 (WITH OR WITHOUT CONSOLE), LA30 (OR EQUIVALENT) AND AN FP11-F FLOATING POINT PROCESSOR. NOTE THAT A SPECIAL INTERRUPTS TEST MODULE IS BEING DESIGNED FOR USE IN THE MANUFACTURING ENVIRONMENT. WHEN THIS DEVICE IS PRESENT THE PROGRAM CKFPBCO WILL MAKE USE OF IT TO TEST THE FPP INTERRUPT ON BUS REQUEST FUNCTIONS.

2.2 STORAGE

ALL THREE PROGRAM REQUIRE A MEMORY SYSTEM OF AT LEAST 16K TO LOAD AND RUN.

2.3 PRELIMINARY PROGRAMS

THESE THREE DIAGNOSTICS WILL ASSUME THAT THE PDP 11/44 CENTRAL PROCESSOR IS FAULTLESS, THEREFORE WHEN IN DOUBT RUN THE PDP 11/44 PROCESSOR DIAGNOSTICS BEFORE THESE FP11-F DIAGNOSTICS.

3. LOADING PROCEDURE

THE PROGRAMS WILL BE SUPPLIED ON THE 11/44 DIAGNOSTIC MEDIA. REFER TO THE XXDP OPERATING MANUAL FOR FURTHER INFORMATION.

4. STARTING PROCEDURE

4.1 CONTROL SWITCH SETTINGS

SEE SECTION 5.1

4.2 PROGRAM AND OPERATOR ACTION



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  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315

1. LOAD PROGRAM INTO MEMORY
2. LOAD ADDRESS 200
3. SET CONSOLE SWITCHES (IF CONSOLE IS PRESENT)
4. PRESS START  
ON FIRST PASS THE PROGRAM WILL IDENTIFY ITSELF. NOTE THAT IF THERE IS NO PHYSICAL CONSOLE THE PROGRAM WILL REQUEST THE OPERATOR FOR INITIAL VALUE FOR THE SOFTWARE SWITCH REGISTER (SEE SECTION 8.5). IF RUNNING UNDER ACT, APT OR CHAIN THIS DOES NOT APPLY.
5. THE PROGRAM WILL LOOP AND AN END OF PASS AND ERROR SUMMARY WILL BE TYPED AT THE END OF EVERY PASS.

5. OPERATING PROCEDURE  
-----

5.1 OPERATIONAL SWITCH SETTINGS

THE SWITCH SETTING ARE:

	OCTAL	
SW<15>=1...	100000	HALT ON ERROR
SW<14>=1...	40000	LOOP ON CURRENT TEST
SW<13>=1...	20000	INHIBIT ERROR TYPE OUTS
SW<12>=1...	10000	INHIBIT T-BIT TRAPPING
SW<11>=1...	4000	INHIBIT ITERATIONS
SW<10>=1...	2000	RING TTY BELL ON ERROR
SW<9>=1....	1000	LOOP ON ERROR
SW<8>=1....	400	LOOP ON TEST SPECIFIED IN SW<6> THROUGH SW<0>
SW<7>=1....	200	PRINT ERROR SUMMARY EVEN IF SW<13>=1, THIS APPLIES ONLY TO PROGRAM CKFPADO.
SW<7>=1....	200	SELECT CORRECT INTERRUPT TEST IN PROGRAM CKFPBCO. IF APT IS SELECTING THE TEST, THE SWITCH REGISTER IS EXAMINED TO SEE IF THE USER HAS SELECTED THIS TEST BY A <1> IN SW<7>

6. ERRORS  
-----

6.1 SUMMARIES

IN PROGRAM CKFPADO, TESTS 1 AND 11 HAVE A SPECIAL ERROR SUMMARY FEATURE. THESE TWO TEST RUN MANY TEST PATTERNS THROUGH THE LOGIC. AFTER AN ERROR IS ENCOUNTERED, ONLY THE FIRST FIVE ERRORS ARE REPORTED (TYPED ON THE TTY). EVERY ERROR THOUGH IS LOGGED AND AN ERROR SUMMARY IS PRINTED WHEN THE TEST IS COMPLETE. NOTE THAT IF SW<13>=1 THIS



316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372

SUMMARY WILL NOT BE TYPED UNLESS SW<7>=1. IN OTHER WORDS TO GET JUST AN ERROR SUMMARY FROM EITHER OF THESE TWO TESTS 1 AND 11 IN PROGRAM CKFPADO BOTH SWITCHES 13 AND 7 MUST = 1.

## 6.2 ERROR RECOVERY

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

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

## 7. RESTRICTIONS

NONE

## 8. MISCELLANEOUS

### 8.1 EXECUTION TIMES

LESS THAN 10 SECONDS FOR EACH PROGRAM ON ANY PASS.

### 8.2 STACK POINTER

THE STACK POINTER IS INITIALIZED TO 1100 IN EACH OF THE THREE PROGRAMS.

### 8.3 PASS COUNT

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

### 8.4 T-BIT TRAPPING

IF SW<12>=0 EACH PROGRAM WILL RUN WITH TRACE TRAPS ON EVERY OTHER PASS. FIRST PASS WILL NOT ENABLE TRACE TRAPS. NOTE SW<12>=1 DISABLES T-BIT TRAPS.

### 8.5 SOFTWARE SWITCH REGISTER

EACH OF THE THREE PROGRAMS WILL RUN WITH OR WITHOUT A CONSOLE SWITCH REGISTER. IF A PHYSICAL CONSOLE SWITCH REGISTER IS PRESENT ON THE SYSTEM, THEN THESE PROGRAMS WILL GO AHEAD AND USE IT FOR THE SWITCH

373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429

FUNCTIONS DESCRIBED IN 5.1 ABOVE. IF HOWEVER THERE IS NO CONSOLE SWITCH REGISTER ON THE SYSTEM A SOFTWARE SWITCH REGISTER WILL BE USED. THIS SOFTWARE SWITCH REGISTER CAN BE EXAMINED OR MODIFIED AT ANY TIME BY THE USER IF HE TYPES CONTROL G WHILE THE PROGRAM IS RUNNING. THIS CONTROL G WILL CAUSE THE CONTENTS OF THE SOFTWARE SWITCH REGISTER TO BE TYPED ON THE TTY AND ASK THE USER FOR A NEW VALUE. WHEN THE USER TYPES A VALUE AND CARRIAGE RETURN THEN THE PROGRAM WILL RESUME TESTING AT THE SAME POINT AT WHICH IT LEFT OFF WHEN THE USER TYPED CONTROL G. NOTE THAT WHEN NOT RUNNING UNDER ACT, APT OR CHAIN THE USER WILL BE ASKED FOR A SOFTWARE SWITCH REGISTER VALUE AFTER LOADING ADDRESS 200 AND STARTING THE PROGRAM THE FIRST TIME THE PROGRAM IS RUN AFTER LOADING (ONLY IF NO CONSOLE SWITCH REGISTER IS ON THE SYSTEM).

8.6 INTERRUPTS TEST

IN PROGRAM CKFPBCO, THERE IS A SPECIAL TEST FOR CHECKING THE CORRECT FLOWS OF THE FPP. THIS TEST CAN BE RUN ONLY IF A SPECIAL TEST MODULE IS IN THE SYSTEM. THIS MODULE WILL PROBABLY ONLY BE USED IN MANUFACTURING. IF THIS MODULE IS NOT IN THE SYSTEM THIS TEST WILL AUTOMATICALLY BE DESELECTED. IF THIS TEST MODULE IS ON THE SYSTEM AND SW<7>=0 THIS TEST WILL BE RUN. IF SW<7>=1 THIS TEST WILL BE DESELECTED.

8.7 ACT, APT AND XXDP COMPATIBILITY

THESE PROGRAMS ARE FULLY COMPATIBLE WITH:  
APT  
ACT  
XXDP MONITOR AND CHAIN PROGRAMS.

9. PROGRAM DESCRIPTION

TEST 1 ROUND\TRUNK TEST



430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486

THIS IS A TEST OF THE ROUND\TRUNK FLOWS. IN PARTICULAR TWO THINGS ARE TESTED: FIRST A CONDITION IN WHICH ROUNDING RESULTS IN THE NEED FOR RENORMALIZATION, AND SECOND THE PSW CONDITION CODES N AND Z BIT COMBINATIONS

TEST 2 OVER\UNDER TEST  
-----

THIS IS A PARTIAL TEST OF THE OVER\UNDER FLOWS. ONE OVERFLOW AND TWO UNDERFLOW CONDITIONS ARE CHECKED. THE REMAINING UNDERFLOW COND. AND THE REMAINING OVERFLOW COND. WILL BE CHECKED LATER USING THE XXX INSTRUCTION. HERE EACH CONDITION TESTED IS CHECKED BOTH WITH TRAPS ENABLED (FIU=1 OR FIV=1) AND ALSO WITH TRAPS DISABLED (FIU=0 OR FIV=0).

TEST 3 LDCFD AND LDCDF TEST  
-----

THIS IS A TEST OF LDCFD AND LDCDF.

TEST 4 CMPD TEST  
-----

THIS IS A TEST OF THE CMPD INSTRUCTION. NOTE THAT A SUBROUTINE IS USED TO SET UP OPERANDS, EXECUTE THE INSTRUCTION AND CHECK THE RESULTS

TEST 5 DIVD WITH (FSRC=0) AND (BUT FD) TEST  
-----

THIS IS A TEST OF THE DIVD INSTRUCTION WITH A ZERO DIVISOR. THE CONDITION IS CHECKED WITH BOTH TRAP ENABLED AND TRAPS DISABLED.

TEST 6 DIVF TEST  
-----

THIS IS A TEST OF THE DIVF INSTRUCTION. NOTE THAT A SUBROUTINE IS USED TO SET UP THE OPERANDS, EXECUTE THE INSTRUCTION AND CHECK THE RESULTS.

TEST 7 DIVD TEST  
-----

THIS IS A TEST OF THE DIVD INSTRUCTION. NOTE THAT A SUBROUTINE IS USED TO SET UP THE OPERANDS, EXECUTE THE INSTRUCTION AND CHECK THE RESULTS.

TEST 10 MULF TEST  
-----

THIS IS A TEST OF THE MULF INSTRUCTION. IT MAKES USE OF A SUBROUTINE TO SET UP THE OPERANDS, EXECUTE THE MULF INSTRUCTION AND CHECK THE RESULTS.

487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543

TEST 11            MULD TEST  
-----

THIS IS A TEST OF THE MULD INSTRUCTION. NOTE THAT A SUBROUTINE IS USED TO SET UP THE OPERANDS, EXECUTE THE MULD INSTRUCTION AND CHECK THE RESULTS.

TEST 12            UNDER\OVER FLOW, USING MULF WITH TRAPS DISABLED, TEST  
-----

THIS IS A TEST OF THE OVERFLOW AND UNDERFLOW CONDITIONS USING THE MULF INSTRUCTION WITH TRAPS DISABLED. NOTE THAT A SUBROUTINE IS USED TO SET UP THE OPERANDS, EXECUTE THE MULF INSTRUCTION AND CHECK THE RESULTS.

TEST 13            UNDER\OVER FLOW, USING MULD WITH TRAPS DISABLED, TEST  
-----

THIS IS A TEST OF THE OVERFLOW AND UNDERFLOW CONDITIONS THAT CAN ARRISE USING THE MULD INSTRUCTION WITH TRAPS DISABLED. A SUBROUTINE IS USED TO SET UP THE OPERANDS, EXECUTE THE MULD INSTRUCTION AND CHECK THE RESULTS.

TEST 14            UNDER\OVER FLOW, USING MULF WITH TRAPS ENABLED, TEST  
-----

THIS IS A TEST OF THE UNDERFLOW AND OVERFLOW CONDITIONS THAT CAN OCCUR USING THE MULF INSTRUCTION. A SUBROUTINE IS CALLED TO SET UP THE OPERANDS, EXECUTE THE MULF INSTRUCTION AND CHECK THE RESULTS. HERE THE PARTICULAR INTERRUPT, EITHER OVERFLOW OR UNDERFLOW, IS ENABLED SO A TRAP SHOULD OCCUR.

TEST 15            UNDER\OVER FLOW, USING MULD WITH TRAPS ENABLED, TEST  
-----

THIS IS A TEST OF THE OVER FLOW AND UNDER FLOW CONDITIONS USING THE MULD INSTRUCTION WITH TRAPS ENABLED. A SUBROUTINE IS USED TO SET UP THE OPERANDS, EXECUTE THE MULD INSTRUCTION AND CHECK THE RESULTS.

TEST 16            MODF TEST  
-----

THIS IS A TEST OF THE MODF INSTRUCTION, WHICH MAKES USE OF A SUBROUTINE TO SET UP THE OPERANDS, EXECUTE THE MODF INSTRUCTION AND CHECK THE RESULTS.

TEST 17            MODD TEST  
-----

THIS IS A TEST OF THE MODD INSTRUCTION. IT MAKES USE OF A SUBROUTINE TO SET UP THE ARGUMENTS, EXECUTE



THE INSTRUCTION AND CHECK THE RESULTS.

544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600

TEST 20            UNDER\OVER FLOW, USING MODF WITH TRAPS DISABLED, TEST  
-----

THIS IS A TEST OF THE MODF OVERFLOW AND UNDERFLOW CONDITIONS. IT MAKES USE OF A SUBROUTINE TO SETUP THE OPERANDS, EXECUTE THE MODF INSTRUCTION AND CHECK THE RESULTS. TRAPS ARE DISABLED DURING THIS TEST.

TEST 21            UNDER\OVER FLOW, USING MODD WITH TRAPS DISABLED, TEST  
-----

THIS IS A TEST OF THE MODD INSTRUCTION'S OVER FLOW AND UNDER FLOW CONDITIONS. A SUBROUTINE IS USED TO SET UP THE OPERANDS, EXECUTE THE MODD INSTRUCTION AND CHECK THE RESULTS.

TEST 22            INTERRUPT CORRECT FLOWS TEST  
-----

THIS IS A TEST OF THE 'CORRECT' FLOWS. THIS PART OF THE MICRO CODE HAS AS ITS PURPOSE INSURING THAT INTERRUPT REQUESTS MADE DURING CERTAIN LENGTHY FPP INSTRUCTIONS GET HONORED. THIS IS DONE IN A WAY SUCH THAT IF AN INTERRUPT REQUEST OCCURS DURING ONE OF THESE INSTRUCTIONS THE STATE OF THAT INSTRUCTION'S EXECUTION WILL BE THE SAME AS IF THAT INSTRUCTION HAD NEVER BEEN FETCHED AND ITS EXECUTION NEVER STARTED. THUS THE MICRO CODE WILL RESTORE ALL REGISTERS, BACK UP THE PC AND LEAVE THE FPS AND ACO THROUGH ACS UNMODIFIED.  
THE INSTRUCTIONS FOR WHICH THIS IS NECESSARY ARE:

ADD (OR SUB)  
DIV  
MUL  
MOD

(BOTH DOUBLE AND FLOATING)

ALL ADDRESSING MODES WILL BE TRIED WITH THE ADDD INSTRUCTION. THEN EACH OF THE OTHER INSTRUCTIONS WILL BE TRIED USING MODE 1. NOTE THAT THIS TEST NEEDS A SPECIAL INTERRUPT MODULE, WHICH WILL PROBABLY ONLY BE PRESENT IN DEC'S MANUFACTURING ENVIRONMENT, TO RUN. THIS SPECIAL EQUIPMENT IS DESIGNED TO RAISE AN INTERRUPT REQUEST IN THE PROCESSOR IF A BIT IS SET IN ITS STATUS REGISTER AND ONLY WHEN AN FPP INSTRUCTION IS ENCOUNTERED. THEREFORE THIS TEST WILL BE RUN CONDITIONALLY (DEPENDENT UPON WHETHER OR NOT THE STATUS REGISTER OF THE TEST EQUIPMENT TIMES OUT WHEN REFERENCED). THIS TEST CAN ALSO BE SELECTED BY TURNING SWITCH 7 OF THE SWITCH REGISTER (PHYSICAL OR VIRTUAL) ON. THE TEST ASSUMES THAT THE TEST EQUIPMENT'S STATUS REGISTER IS AT LOCATION 777774 (NOTE THAT ALL REFERENCES TO THIS LOCATION ARE MADE INDIRECT

601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623

THROUGH THIS PROGRAMS LOCATION CORINT, SO THAT IF THE USER HAS MODIFIED THE TEST EQUIPMENT'S STATUS REGISTER TO RESPOND TO A DIFFERENT ADDRESS LOCATION CORINT MUST BE MADE TO CONTAIN THAT STATUS REGISTER'S NEW ADDRESS). THIS PROGRAM ASSUMES THAT THE TRAP VECTOR FOR THE TEST EQUIPMENT IS 110. AGAIN NOTE THAT ALL REFERENCES TO THIS TRAP VECTOR ARE INDIRECT, THROUGH THIS PROGRAM'S LOCATION CORTRP (IF THE TEST EQUIPMENT IS MADE TO TRAP TO A DIFFERENT VECTOR LOCATION CORTRP MUST CONTAIN THE ADDRESS OF THIS VECTOR).

10.

LISTING  
-----g



624  
625  
626  
627  
628  
629  
630  
1645  
1646  
1647  
1648  
1649  
1650

000267  
000002

```

MNUMBER=267
PROGNUM=2
.LIST ME
.NLIST MD
.NLIST MC
.NLIST CND
.NLIST BEX
.MCALL .HEADER, .SWRHI, .EQUAT, .SETUP, .SCATCH, .SACT11, .SCMTAG
.MCALL NEWTST, $$NEWTEST, .SSAVE, .STYPE
.MCALL .STYPDEC, .STRAP, .SPOWER, .SAPTHDR, .SAPTBL
.MCALL .STYPE, .SAPTYPE, .SREAD
.MCALL .EQUIV ;REMOVE FOR PDP-10 ASSEMBLY

```

```

.TITLE CKFPBCO FP11F FLTG PNT PRT B
;*COPYRIGHT (C) 1981
;*DIGITAL EQUIPMENT CORP.
;*MAYNARD, MASS. 01754
;*
;*
;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
;*PACKAGE (MAINDEC-11-DZQAC-C5), JAN, 1981.
;*
$TN=1
$$SWR=160000 ;:HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYP0UT

```

000001  
160000

1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659

000244  
177400  
000200  
000011  
000015

```

FPVECT=244
$$SWR=177400
$$SWRMSK=200
TAB=11
CRLF=15

```

001100  
104000  
000004

```

.SBTTL BASIC DEFINITIONS
;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
STACK= 1100
ERROR=EMT
SCOPE=IOT

```

000011  
000012  
000015  
000200  
177776  
177776  
177774  
177772  
177570  
177570

```

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

```

000000  
000001  
000002  
000003  
000004  
000005  
000006  
000007  
000006

```

;*GENERAL PURPOSE REGISTER DEFINITIONS
R0= %0 ;:GENERAL REGISTER
R1= %1 ;:GENERAL REGISTER
R2= %2 ;:GENERAL REGISTER
R3= %3 ;:GENERAL REGISTER
R4= %4 ;:GENERAL REGISTER
R5= %5 ;:GENERAL REGISTER
R6= %6 ;:GENERAL REGISTER
R7= %7 ;:GENERAL REGISTER
SP= %6 ;:STACK POINTER

```

```
000007      PC=      7      ::PROGRAM COUNTER
              :*PRIORITY LEVEL DEFINITIONS
000000      PR0=      0      ::PRIORITY LEVEL 0
000040      PR1=     40      ::PRIORITY LEVEL 1
000100      PR2=    100      ::PRIORITY LEVEL 2
000140      PR3=    140      ::PRIORITY LEVEL 3
000200      PR4=    200      ::PRIORITY LEVEL 4
000240      PR5=    240      ::PRIORITY LEVEL 5
000300      PR6=    300      ::PRIORITY LEVEL 6
000340      PR7=    340      ::PRIORITY LEVEL 7
              :*"SWITCH REGISTER" SWITCH DEFINITIONS
100000      SW15=  100000
040000      SW14=   40000
020000      SW13=   20000
010000      SW12=   10000
004000      SW11=   4000
002000      SW10=   2000
001000      SW09=   1000
000400      SW08=   400
000200      SW07=   200
000100      SW06=   100
000040      SW05=   40
000020      SW04=   20
000010      SW03=   10
000004      SW02=    4
000002      SW01=    2
000001      SW00=    1
              SW9=SW09
              SW8=SW08
              SW7=SW07
              SW6=SW06
              SW5=SW05
              SW4=SW04
              SW3=SW03
              SW2=SW02
              SW1=SW01
              SW0=SW00
              :*DATA BIT DEFINITIONS (BIT00 TO BIT15)
100000      BIT15=  100000
040000      BIT14=   40000
020000      BIT13=   20000
010000      BIT12=   10000
004000      BIT11=   4000
002000      BIT10=   2000
001000      BIT09=   1000
000400      BIT08=   400
000200      BIT07=   200
000100      BIT06=   100
000040      BIT05=   40
000020      BIT04=   20
000010      BIT03=   10
000004      BIT02=    4
000002      BIT01=    2
000001      BIT00=    1
              BIT9=BIT09
              BIT8=BIT08
              BIT7=BIT07
```



```
000100 BIT6=BIT06
000040 BIT5=BIT05
000020 BIT4=BIT04
000010 BIT3=BIT03
000004 BIT2=BIT02
000002 BIT1=BIT01
000001 BIT0=BIT00

;*BASIC "CPU" TRAP VECTOR ADDRESSES
ERRVEC= 4 ;;TIME OUT AND OTHER ERRORS
RESVEC= 10 ;;RESERVED AND ILLEGAL INSTRUCTIONS
TBITVEC=14 ;;"T" BIT
TRTVEC= 14 ;;TRACE TRAP
BPTVEC= 14 ;;BREAKPOINT TRAP (BPT)
IOTVEC= 20 ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
PWRVEC= 24 ;;POWER FAIL
EMTVEC= 30 ;;EMULATOR TRAP (EMT) **ERROR**
TRAPVEC=34 ;;"TRAP" TRAP
TKVEC= 60 ;;TTY KEYBOARD VECTOR
TPVEC= 64 ;;TTY PRINTER VECTOR
PIRQVEC=240 ;;PROGRAM INTERRUPT REQUEST VECTOR

.SBTTL FPP REGISTER DEFINITIONS
AC0 =%0
AC1 =%1
AC2 =%2
AC3 =%3
AC4 =%4
AC5 =%5
AC6 =%6
AC7 =%7

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

1660
1661 000000
1662 000001
1663 000002
1664 000003
1665 000004
1666 000005
1667 000006
1668 000007
1670
1671 000000

000174 000174
000174 000000
000176 000000

000200 000137 004346
```

1672

001100 001100  
001100 000000  
001102 000  
001103 000  
001104 000000  
001106 000000  
001110 000000  
001112 000000  
001114 000  
001115 001  
001116 000000  
001120 000000  
001122 000000  
001124 000000  
001126 000000  
001130 000000  
001132 000000  
001134 000  
001135 000  
001136 000000  
001140 177570  
001142 177570  
001144 177560  
001146 177562  
001150 177564  
001152 177566  
001154 000  
001155 002  
001156 012  
001157 000  
001160 000000  
  
001162 000024  
001164 000000  
001166 000000  
001170 000000  
001172 000000  
001174 000000  
001176 000000  
001200 000000  
001202 000000  
001204 000000  
001206 000000  
001210 000000  
001212 000000  
001214 000000  
001216 000000  
001220 000000  
001222 000000  
001224 000000  
001226 000000

.SBTTL COMMON TAGS

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

.=1100  
SCMTAG: .WORD 0  
\$TSTNM: .BYTE 0  
\$ERFLG: .BYTE 0  
\$ICNT: .WORD 0  
\$LPADR: .WORD 0  
\$LPERR: .WORD 0  
\$ERTTL: .WORD 0  
\$ITEMB: .BYTE 0  
\$ERMAX: .BYTE 1  
\$ERRPC: .WORD 0  
\$GDADR: .WORD 0  
\$BDADR: .WORD 0  
\$GDDAT: .WORD 0  
\$BDDAT: .WORD 0  
\$AUTOB: .BYTE 0  
\$INTAG: .BYTE 0  
\$SWR: .WORD DSWR  
\$DISPLAY: .WORD DDISP  
\$TKS: 177560  
\$TKB: 177562  
\$TPS: 177564  
\$TPB: 177566  
\$NULL: .BYTE 0  
\$FILLS: .BYTE 2  
\$FILLC: .BYTE 12  
\$TPFLG: .BYTE 0  
\$REGAD: .WORD 0  
  
\$REG0: .REPT \$CM3  
\$REG1: .WORD 0  
\$REG2: .WORD 0  
\$REG3: .WORD 0  
\$REG4: .WORD 0  
\$REG5: .WORD 0  
\$REG6: .WORD 0  
\$REG7: .WORD 0  
\$REG10: .WORD 0  
\$REG11: .WORD 0  
\$REG12: .WORD 0  
\$REG13: .WORD 0  
\$REG14: .WORD 0  
\$REG15: .WORD 0  
\$REG16: .WORD 0  
\$REG17: .WORD 0  
\$REG20: .WORD 0  
\$REG21: .WORD 0  
\$REG22: .WORD 0

:::START OF COMMON TAGS  
:::CONTAINS THE TEST NUMBER  
:::CONTAINS ERROR FLAG  
:::CONTAINS SUBTEST ITERATION COUNT  
:::CONTAINS SCOPE LOOP ADDRESS  
:::CONTAINS SCOPE RETURN FOR ERRORS  
:::CONTAINS TOTAL ERRORS DETECTED  
:::CONTAINS ITEM CONTROL BYTE  
:::CONTAINS MAX. ERRORS PER TEST  
:::CONTAINS PC OF LAST ERROR INSTRUCTION  
:::CONTAINS ADDRESS OF 'GOOD' DATA  
:::CONTAINS ADDRESS OF 'BAD' DATA  
:::CONTAINS 'GOOD' DATA  
:::CONTAINS 'BAD' DATA  
:::RESERVED--NOT TO BE USED  
  
:::AUTOMATIC MODE INDICATOR  
:::INTERRUPT MODE INDICATOR  
  
:::ADDRESS OF SWITCH REGISTER  
:::ADDRESS OF DISPLAY REGISTER  
:::TTY KBD STATUS  
:::TTY KBD BUFFER  
:::TTY PRINTER STATUS REG. ADDRESS  
:::TTY PRINTER BUFFER REG. ADDRESS  
:::CONTAINS NULL CHARACTER FOR FILLS  
:::CONTAINS # OF FILLER CHARACTERS REQUIRED  
:::INSERT FILL CHARS. AFTER A 'LINE FEED'  
:::'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)  
:::CONTAINS THE ADDRESS FROM  
:::WHICH (\$REG0) WAS OBTAINED  
  
:::CONTAINS ((\$REGAD)+0)  
:::CONTAINS ((\$REGAD)+2)  
:::CONTAINS ((\$REGAD)+4)  
:::CONTAINS ((\$REGAD)+6)  
:::CONTAINS ((\$REGAD)+10)  
:::CONTAINS ((\$REGAD)+12)  
:::CONTAINS ((\$REGAD)+14)  
:::CONTAINS ((\$REGAD)+16)  
:::CONTAINS ((\$REGAD)+20)  
:::CONTAINS ((\$REGAD)+22)  
:::CONTAINS ((\$REGAD)+24)  
:::CONTAINS ((\$REGAD)+26)  
:::CONTAINS ((\$REGAD)+30)  
:::CONTAINS ((\$REGAD)+32)  
:::CONTAINS ((\$REGAD)+34)  
:::CONTAINS ((\$REGAD)+36)  
:::CONTAINS ((\$REGAD)+40)  
:::CONTAINS ((\$REGAD)+42)  
:::CONTAINS ((\$REGAD)+44)



```
001230 000000 $REG23: .WORD 0 ;;CONTAINS (($REGAD)+46)
001232 000024 .REPT 24
001234 000000 $TMP0: .WORD 0 ;;USER DEFINED
001236 000000 $TMP1: .WORD 0 ;;USER DEFINED
001240 000000 $TMP2: .WORD 0 ;;USER DEFINED
001242 000000 $TMP3: .WORD 0 ;;USER DEFINED
001244 000000 $TMP4: .WORD 0 ;;USER DEFINED
001246 000000 $TMP5: .WORD 0 ;;USER DEFINED
001250 000000 $TMP6: .WORD 0 ;;USER DEFINED
001252 000000 $TMP7: .WORD 0 ;;USER DEFINED
001254 000000 $TMP10: .WORD 0 ;;USER DEFINED
001256 000000 $TMP11: .WORD 0 ;;USER DEFINED
001260 000000 $TMP12: .WORD 0 ;;USER DEFINED
001262 000000 $TMP13: .WORD 0 ;;USER DEFINED
001264 000000 $TMP14: .WORD 0 ;;USER DEFINED
001266 000000 $TMP15: .WORD 0 ;;USER DEFINED
001270 000000 $TMP16: .WORD 0 ;;USER DEFINED
001272 000000 $TMP17: .WORD 0 ;;USER DEFINED
001274 000000 $TMP20: .WORD 0 ;;USER DEFINED
001276 000000 $TMP21: .WORD 0 ;;USER DEFINED
001300 000000 $TMP22: .WORD 0 ;;USER DEFINED
001302 000000 $TMP23: .WORD 0 ;;USER DEFINED
001304 000000 $TIMES: 0 ;;MAX. NUMBER OF ITERATIONS
001306 207 377 377 $ESCAPE: 0 ;;ESCAPE ON ERROR ADDRESS
001312 077 $BELL: .ASCII <207><377><377> ;;CODE FOR BELL
001313 015 $QUES: .ASCII /?/ ;;QUESTION MARK
001314 012 000 $CRLF: .ASCII <15> ;;CARRIAGE RETURN
$LF: .ASCII <12> ;;LINE FEED
*****
.SBTTL APT MAILBOX-ETABLE
*****
.EVEN
001316 $MAIL: ;;APT MAILBOX
001316 000000 $MSGTY: .WORD AMSGTY ;;MESSAGE TYPE CODE
001320 000000 $FATAL: .WORD AFATAL ;;FATAL ERROR NUMBER
001322 000000 $TESTN: .WORD ATESTN ;;TEST NUMBER
001324 000000 $PASS: .WORD APASS ;;PASS COUNT
001326 000000 $DEVCT: .WORD ADEVCT ;;DEVICE COUNT
001330 000000 $UNIT: .WORD AUNIT ;;I/O UNIT NUMBER
001332 000000 $MSGAD: .WORD AMSGAD ;;MESSAGE ADDRESS
001334 000000 $MSGLG: .WORD AMSGLG ;;MESSAGE LENGTH
001336 $ETABLE: ;;APT ENVIRONMENT TABLE
001336 000 $ENV: .BYTE AENV ;;ENVIRONMENT BYTE
001337 000 $ENVM: .BYTE AENVM ;;ENVIRONMENT MODE BITS
001340 000000 $SWREG: .WORD ASWREG ;;APT SWITCH REGISTER
001342 000000 $USWR: .WORD AUSWR ;;USER SWITCHES
001344 000000 $CPUOP: .WORD ACPUOP ;;CPU TYPE, OPTIONS
BITS 15-11=CPU TYPE
11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
11/70=06,PDQ=07,Q=10
BIT 10=REAL TIME CLOCK
BIT 9=FLOATING POINT PROCESSOR
BIT 8=MEMORY MANAGEMENT
001346 000 $MAMS1: .BYTE AMAMS1 ;;HIGH ADDRESS, M.S. BYTE
001347 000 $MTYP1: .BYTE AMTYP1 ;;MEM. TYPE, BLK#1
MEM. TYPE BYTE -- (HIGH BYTE)
900 NSEC CORE=001
*
```

```

                                300 NSEC BIPOLAR=002
                                500 NSEC MOS=003
001350 000000 $MADR1: .WORD AMADR1 ;;HIGH ADDRESS,BLK#1
                                MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
001352 000 $MAMS2: .BYTE AMAMS2 ;;HIGH ADDRESS,M.S. BYTE
001353 000 $MTYP2: .BYTE AMTYP2 ;;MEM.TYPE,BLK#2
001354 000000 $MADR2: .WORD AMADR2 ;;MEM.LAST ADDRESS,BLK#2
001356 000 $MAMS3: .BYTE AMAMS3 ;;HIGH ADDRESS,M.S.BYTE
001357 000 $MTYP3: .BYTE AMTYP3 ;;MEM.TYPE,BLK#3
001360 000000 $MADR3: .WORD AMADR3 ;;MEM.LAST ADDRESS,BLK#3
001362 000 $MAMS4: .BYTE AMAMS4 ;;HIGH ADDRESS,M.S.BYTE
001363 000 $MTYP4: .BYTE AMTYP4 ;;MEM.TYPE,BLK#4
001364 000000 $MADR4: .WORD AMADR4 ;;MEM.LAST ADDRESS,BLK#4
001366 000000 $VECT1: .WORD AVECT1 ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
001370 000000 $VECT2: .WORD AVECT2 ;;INTERRUPT VECTOR#2BUS PRIORITY#2
001372 000000 $BASE: .WORD ABASE ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
001374 000000 $DEVN: .WORD ADEVN ;;DEVICE MAP
001376 000000 $CDW1: .WORD ACDW1 ;;CONTROLLER DESCRIPTION WORD#1
001400 000000 $CDW2: .WORD ACDW2 ;;CONTROLLER DESCRIPTION WORD#2
001402 000000 $DDW0: .WORD ADDW0 ;;DEVICE DESCRIPTOR WORD#0
001404 000000 $DDW1: .WORD ADDW1 ;;DEVICE DESCRIPTOR WORD#1
001406 000000 $DDW2: .WORD ADDW2 ;;DEVICE DESCRIPTOR WORD#2
001410 000000 $DDW3: .WORD ADDW3 ;;DEVICE DESCRIPTOR WORD#3
001412 000000 $DDW4: .WORD ADDW4 ;;DEVICE DESCRIPTOR WORD#4
001414 000000 $DDW5: .WORD ADDW5 ;;DEVICE DESCRIPTOR WORD#5
001416 000000 $DDW6: .WORD ADDW6 ;;DEVICE DESCRIPTOR WORD#6
001420 000000 $DDW7: .WORD ADDW7 ;;DEVICE DESCRIPTOR WORD#7
001422 000000 $DDW8: .WORD ADDW8 ;;DEVICE DESCRIPTOR WORD#8
001424 000000 $DDW9: .WORD ADDW9 ;;DEVICE DESCRIPTOR WORD#9
001426 000000 $DDW10: .WORD ADDW10 ;;DEVICE DESCRIPTOR WORD#10
001430 000000 $DDW11: .WORD ADDW11 ;;DEVICE DESCRIPTOR WORD#11
001432 000000 $DDW12: .WORD ADDW12 ;;DEVICE DESCRIPTOR WORD#12
001434 000000 $DDW13: .WORD ADDW13 ;;DEVICE DESCRIPTOR WORD#13
001436 000000 $DDW14: .WORD ADDW14 ;;DEVICE DESCRIPTOR WORD#14
001440 000000 $DDW15: .WORD ADDW15 ;;DEVICE DESCRIPTOR WORD#15
001442 SETEND:

```



```
.SBTTL ERROR POINTER TABLE
:*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
:*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
:*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
:*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
:*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
:*      EM      ::POINTS TO THE ERROR MESSAGE
:*      DH      ::POINTS TO THE DATA HEADER
:*      DT      ::POINTS TO THE DATA
:*      DF      ::POINTS TO THE DATA FORMAT
```

	\$ERRTB:	REPT	MNUMBER		
1676	001442	000267			
1678				:ITEM 1	.WORD EM1,DH1,DT1,DF1
	001442	040514	067106	071474	
				:ITEM 2	.WORD EM2,DH2,DT2,DF2
	001452	040546	067176	071532	
				:ITEM 3	.WORD EM3,DH3,DT3,DF3
	001462	040602	067106	071474	
				:ITEM 4	.WORD EM4,DH4,DT4,DF4
	001472	040707	067106	071474	
				:ITEM 5	.WORD EM5,DH5,DT5,DF5
	001502	041014	067106	071474	
				:ITEM 6	.WORD EM6,DH6,DT6,DF6
	001512	041121	067106	071474	
				:ITEM 7	.WORD EM7,DH7,DT7,DF7
	001522	041226	067106	071474	
				:ITEM 10	.WORD EM10,DH10,DT10,DF10
	001532	041333	067106	071474	
				:ITEM 11	.WORD EM11,DH11,DT11,DF11
	001542	041440	067106	071474	
				:ITEM 12	.WORD EM12,DH12,DT12,DF12
	001552	041547	067106	071474	
				:ITEM 13	.WORD EM13,DH13,DT13,DF13
	001562	041656	067106	071474	
				:ITEM 14	.WORD EM14,DH14,DT14,DF14
	001572	041772	067106	071474	
				:ITEM 15	.WORD EM15,DH15,DT15,DF15
	001602	042104	067106	071474	
				:ITEM 16	.WORD EM16,DH16,DT16,DF16
	001612	042216	067106	071474	
				:ITEM 17	.WORD EM17,DH17,DT17,DF17
	001622	042313	067237	071560	
				:ITEM 20	.WORD EM20,DH20,DT20,DF20
	001632	042370	067106	071600	
				:ITEM 21	.WORD EM21,DH21,DT21,DF21
	001642	042422	067312	071622	
				:ITEM 22	.WORD EM22,DH22,DT22,DF22
	001652	042454	067401	071644	
				:ITEM 23	.WORD EM23,DH23,DT23,DF23
	001662	042535	067106	071662	
				:ITEM 24	.WORD EM24,DH24,DT24,DF24
	001672	042612	067106	071662	
				:ITEM 25	.WORD EM25,DH25,DT25,DF25
	001702	042710	067106	071662	
				:ITEM 26	.WORD EM26,DH26,DT26,DF26
	001712	042775	067106	071662	
				:ITEM 27	

001722	042710	067106	071662		.WORD	EM27,DH27,DT27,DF27
001732	043073	067106	071662	:ITEM 30	.WORD	EM30,DH30,DT30,DF30
001742	043145	067106	071662	:ITEM 31	.WORD	EM31,DH31,DT31,DF31
001752	042560	067106	071662	:ITEM 32	.WORD	EM32,DH32,DT32,DF32
001762	043212	067106	071662	:ITEM 33	.WORD	EM33,DH33,DT33,DF33
001772	043235	067106	071662	:ITEM 34	.WORD	EM34,DH34,DT34,DF34
002002	043267	067106	071662	:ITEM 35	.WORD	EM35,DH35,DT35,DF35
002012	043342	067106	071662	:ITEM 36	.WORD	EM36,DH36,DT36,DF36
002022	043410	067106	071662	:ITEM 37	.WORD	EM37,DH37,DT37,DF37
002032	043433	067106	071662	:ITEM 40	.WORD	EM40,DH40,DT40,DF40
002042	043465	067106	071662	:ITEM 41	.WORD	EM41,DH41,DT41,DF41
002052	043540	067106	071662	:ITEM 42	.WORD	EM42,DH42,DT42,DF42
002062	043671	067106	071662	:ITEM 43	.WORD	EM43,DH43,DT43,DF43
002072	044022	067106	071662	:ITEM 44	.WORD	EM44,DH44,DT44,DF44
002102	044070	067106	071662	:ITEM 45	.WORD	EM45,DH45,DT45,DF45
002112	044143	067106	071662	:ITEM 46	.WORD	EM46,DH46,DT46,DF46
002122	044220	067106	071662	:ITEM 47	.WORD	EM47,DH47,DT47,DF47
002132	044354	067106	071662	:ITEM 50	.WORD	EM50,DH50,DT50,DF50
002142	044427	067106	071662	:ITEM 51	.WORD	EM51,DH51,DT51,DF51
002152	044475	067106	071662	:ITEM 52	.WORD	EM52,DH52,DT52,DF52
002162	052325	067106	071734	:ITEM 53	.WORD	EM53,DH53,DT53,DF53
002172	052356	067106	071734	:ITEM 54	.WORD	EM54,DH54,DT54,DF54
002202	052273	067106	071734	:ITEM 55	.WORD	EM55,DH55,DT55,DF55
002212	052406	067106	071734	:ITEM 56	.WORD	EM56,DH56,DT56,DF56
002222	052477	067106	071734	:ITEM 57	.WORD	EM57,DH57,DT57,DF57
002232	052567	067106	071734	:ITEM 60	.WORD	EM60,DH60,DT60,DF60
002242	052671	067106	071734	:ITEM 61	.WORD	EM61,DH61,DT61,DF61
002252	053065	067106	071734	:ITEM 62	.WORD	EM62,DH62,DT62,DF62
002262	053261	067106	071734	:ITEM 63	.WORD	EM63,DH63,DT63,DF63



002272	053372	067106	071734	:ITEM 64	.WORD	EM64,DH64,DT64,DF64
002302	053511	067106	071734	:ITEM 65	.WORD	EM65,DH65,DT65,DF65
002312	053624	067106	071734	:ITEM 66	.WORD	EM66,DH66,DT66,DF66
002322	053673	067106	071734	:ITEM 67	.WORD	EM67,DH67,DT67,DF67
002332	053756	067106	071734	:ITEM 70	.WORD	EM70,DH70,DT70,DF70
002342	054007	067106	071734	:ITEM 71	.WORD	EM71,DH71,DT71,DF71
002352	054037	067106	071734	:ITEM 72	.WORD	EM72,DH72,DT72,DF72
002362	054037	067106	071734	:ITEM 73	.WORD	EM73,DH73,DT73,DF73
002372	054071	067106	071734	:ITEM 74	.WORD	EM74,DH74,DT74,DF74
002402	054200	067106	071734	:ITEM 75	.WORD	EM75,DH75,DT75,DF75
002412	054271	067106	071734	:ITEM 76	.WORD	EM76,DH76,DT76,DF76
002422	054361	067106	071734	:ITEM 77	.WORD	EM77,DH77,DT77,DF77
002432	054471	067106	071734	:ITEM 100	.WORD	EM100,DH100,DT100,DF100
002442	054554	067106	071734	:ITEM 101	.WORD	EM101,DH101,DT101,DF101
002452	054750	067106	071734	:ITEM 102	.WORD	EM102,DH102,DT102,DF102
002462	055144	067106	071734	:ITEM 103	.WORD	EM103,DH103,DT103,DF103
002472	055256	067106	071734	:ITEM 104	.WORD	EM104,DH104,DT104,DF104
002502	055423	067106	071734	:ITEM 105	.WORD	EM105,DH105,DT105,DF105
002512	055532	067106	071734	:ITEM 106	.WORD	EM106,DH106,DT106,DF106
002522	055641	067106	071734	:ITEM 107	.WORD	EM107,DH107,DT107,DF107
002532	055750	067106	071734	:ITEM 110	.WORD	EM110,DH110,DT110,DF110
002542	044565	067106	071662	:ITEM 111	.WORD	EM111,DH111,DT111,DF111
002552	044642	067106	071662	:ITEM 112	.WORD	EM112,DH112,DT112,DF112
002562	044720	067106	071662	:ITEM 113	.WORD	EM113,DH113,DT113,DF113
002572	044776	067106	071662	:ITEM 114	.WORD	EM114,DH114,DT114,DF114
002602	045055	067447	072022	:ITEM 115	.WORD	EM115,DH115,DT115,DF115
002612	045133	067447	072022	:ITEM 116	.WORD	EM116,DH116,DT116,DF116
002622	045212	067447	072022	:ITEM 117	.WORD	EM117,DH117,DT117,DF117
				:ITEM 120		

002632	045350	067447	072022		WORD	EM120,DH120,DT120,DF120
002642	045506	067447	072022	:ITEM 121	WORD	EM121,DH121,DT121,DF121
002652	045643	067447	072022	:ITEM 122	WORD	EM122,DH122,DT122,DF122
002662	046000	067106	071662	:ITEM 123	WORD	EM123,DH123,DT123,DF123
002672	046056	067106	071662	:ITEM 124	WORD	EM124,DH124,DT124,DF124
002702	046135	067106	071662	:ITEM 125	WORD	EM125,DH125,DT125,DF125
002712	046213	067106	071662	:ITEM 126	WORD	EM126,DH126,DT126,DF126
002722	046272	067447	072022	:ITEM 127	WORD	EM127,DH127,DT127,DF127
002732	046350	067447	072022	:ITEM 130	WORD	EM130,DH130,DT130,DF130
002742	046427	067447	072022	:ITEM 131	WORD	EM131,DH131,DT131,DF131
002752	046565	067106	071662	:ITEM 132	WORD	EM132,DH132,DT132,DF132
002762	046723	067447	072022	:ITEM 133	WORD	EM133,DH133,DT133,DF133
002772	047061	067447	072022	:ITEM 134	WORD	EM134,DH134,DT134,DF134
003002	047216	067106	071662	:ITEM 135	WORD	EM135,DH135,DT135,DF135
003012	047353	067447	072022	:ITEM 136	WORD	EM136,DH136,DT136,DF136
003022	047510	067447	072022	:ITEM 137	WORD	EM137,DH137,DT137,DF137
003032	047576	067447	072022	:ITEM 140	WORD	EM140,DH140,DT140,DF140
003042	044565	067447	072022	:ITEM 141	WORD	EM141,DH141,DT141,DF141
003052	044642	067447	072022	:ITEM 142	WORD	EM142,DH142,DT142,DF142
003062	047665	067447	072022	:ITEM 143	WORD	EM143,DH143,DT143,DF143
003072	047743	067447	072022	:ITEM 144	WORD	EM144,DH144,DT144,DF144
003102	050022	067106	071662	:ITEM 145	WORD	EM145,DH145,DT145,DF145
003112	050167	067106	071662	:ITEM 146	WORD	EM146,DH146,DT146,DF146
003122	050334	067106	071662	:ITEM 147	WORD	EM147,DH147,DT147,DF147
003132	050500	067106	071662	:ITEM 150	WORD	EM150,DH150,DT150,DF150
003142	050644	067447	072022	:ITEM 151	WORD	EM151,DH151,DT151,DF151
003152	050732	067447	072022	:ITEM 152	WORD	EM152,DH152,DT152,DF152
003162	046000	067447	072022	:ITEM 153	WORD	EM153,DH153,DT153,DF153
003172	046056	067447	072022	:ITEM 154	WORD	EM154,DH154,DT154,DF154



003202	051021	067447	072022	:ITEM 155	WORD	EM155,DH155,DT155,DF155
003212	051077	067447	072022	:ITEM 156	WORD	EM156,DH156,DT156,DF156
003222	051156	067106	071662	:ITEM 157	WORD	EM157,DH157,DT157,DF157
003232	051323	067447	072022	:ITEM 160	WORD	EM160,DH160,DT160,DF160
003242	051461	067106	071662	:ITEM 161	WORD	EM161,DH161,DT161,DF161
003252	051626	067106	071662	:ITEM 162	WORD	EM162,DH162,DT162,DF162
003262	051772	067447	072022	:ITEM 163	WORD	EM163,DH163,DT163,DF163
003272	052127	067106	071662	:ITEM 164	WORD	EM164,DH164,DT164,DF164
003302	056060	067447	072076	:ITEM 165	WORD	EM165,DH165,DT165,DF165
003312	056144	067447	072076	:ITEM 166	WORD	EM166,DH166,DT166,DF166
003322	056227	067447	072076	:ITEM 167	WORD	EM167,DH167,DT167,DF167
003332	056261	067447	072076	:ITEM 170	WORD	EM170,DH170,DT170,DF170
003342	056347	067447	072076	:ITEM 171	WORD	EM171,DH171,DT171,DF171
003352	056472	067447	072076	:ITEM 172	WORD	EM172,DH172,DT172,DF172
003362	056557	067447	072076	:ITEM 173	WORD	EM173,DH173,DT173,DF173
003372	056725	067447	072076	:ITEM 174	WORD	EM174,DH174,DT174,DF174
003402	057073	067447	072076	:ITEM 175	WORD	EM175,DH175,DT175,DF175
003412	057205	067447	072076	:ITEM 176	WORD	EM176,DH176,DT176,DF176
003422	057271	067546	072164	:ITEM 177	WORD	EM177,DH177,DT177,DF177
003432	057325	067447	072076	:ITEM 200	WORD	EM200,DH200,DT200,DF200
003442	057357	067447	072076	:ITEM 201	WORD	EM201,DH201,DT201,DF201
003452	057446	067447	072076	:ITEM 202	WORD	EM202,DH202,DT202,DF202
003462	057610	067447	072076	:ITEM 203	WORD	EM203,DH203,DT203,DF203
003472	057752	067447	072076	:ITEM 204	WORD	EM204,DH204,DT204,DF204
003502	060040	067447	072076	:ITEM 205	WORD	EM205,DH205,DT205,DF205
003512	060206	067447	072076	:ITEM 206	WORD	EM206,DH206,DT206,DF206
003522	060354	067647	072174	:ITEM 207	WORD	EM207,DH207,DT207,DF207
003532	060416	067737	072246	:ITEM 210	WORD	EM210,DH210,DT210,DF210
				:ITEM 211		

003542	060460	067737	072174	:ITEM 212	WORD	EM211,DH211,DT211,DF211
003552	060624	067647	072270	:ITEM 213	WORD	EM212,DH212,DT212,DF212
003562	061010	067647	072270	:ITEM 214	WORD	EM213,DH213,DT213,DF213
003572	061174	067647	072270	:ITEM 215	WORD	EM214,DH214,DT214,DF214
003602	061360	067647	072270	:ITEM 216	WORD	EM215,DH215,DT215,DF215
003612	061544	067737	072174	:ITEM 217	WORD	EM216,DH216,DT216,DF216
003622	061726	070000	072352	:ITEM 220	WORD	EM217,DH217,DT217,DF217
003632	061770	067607	072174	:ITEM 221	WORD	EM220,DH220,DT220,DF220
003642	062220	067737	072174	:ITEM 222	WORD	EM221,DH221,DT221,DF221
003652	062464	067607	072174	:ITEM 223	WORD	EM222,DH222,DT222,DF222
003662	062715	067737	072174	:ITEM 224	WORD	EM223,DH223,DT223,DF223
003672	063162	067607	072174	:ITEM 225	WORD	EM224,DH224,DT224,DF224
003702	063413	067737	072174	:ITEM 226	WORD	EM225,DH225,DT225,DF225
003712	063660	070053	072270	:ITEM 227	WORD	EM226,DH226,DT226,DF226
003722	064013	070142	072270	:ITEM 230	WORD	EM227,DH227,DT227,DF227
003732	064146	070053	072270	:ITEM 231	WORD	EM230,DH230,DT230,DF230
003742	064302	070142	072246	:ITEM 232	WORD	EM231,DH231,DT231,DF231
003752	060416	070142	072174	:ITEM 233	WORD	EM232,DH232,DT232,DF232
003762	064436	070231	072372	:ITEM 234	WORD	EM233,DH233,DT233,DF233
003772	064475	070272	072426	:ITEM 235	WORD	EM234,DH234,DT234,DF234
004002	064561	070360	072446	:ITEM 236	WORD	EM235,DH235,DT235,DF235
004012	064627	070420	072426	:ITEM 237	WORD	EM236,DH236,DT236,DF236
004022	064662	070272	072426	:ITEM 240	WORD	EM237,DH237,DT237,DF237
004032	064746	070506	072426	:ITEM 241	WORD	EM240,DH240,DT240,DF240
004042	065002	070231	072372	:ITEM 242	WORD	EM241,DH241,DT241,DF241
004052	065105	070506	072426	:ITEM 243	WORD	EM242,DH242,DT242,DF242
004062	065141	070231	072372	:ITEM 244	WORD	EM243,DH243,DT243,DF243
004072	065244	070231	072372	:ITEM 245	WORD	EM244,DH244,DT244,DF244
004102	065303	070231	072372		WORD	EM245,DH245,DT245,DF245



004112	044175	067106	071662	:ITEM 246	WORD	EM246,DH246,DT246,DF246
004122	065365	070576	072460	:ITEM 247	WORD	EM247,DH247,DT247,DF247
004132	065421	070643	072476	:ITEM 250	WORD	EM250,DH250,DT250,DF250
004142	065453	070643	072476	:ITEM 251	WORD	EM251,DH251,DT251,DF251
004152	065506	067176	072510	:ITEM 252	WORD	EM252,DH252,DT252,DF252
004162	065577	067106	072522	:ITEM 253	WORD	EM253,DH253,DT253,DF253
004172	065663	067106	072522	:ITEM 254	WORD	EM254,DH254,DT254,DF254
004202	065750	067106	072522	:ITEM 255	WORD	EM255,DH255,DT255,DF255
004212	066036	067106	072522	:ITEM 256	WORD	EM256,DH256,DT256,DF256
004222	066125	067106	072522	:ITEM 257	WORD	EM257,DH257,DT257,DF257
004232	066213	067106	072522	:ITEM 260	WORD	EM260,DH260,DT260,DF260
004242	066302	067106	072522	:ITEM 261	WORD	EM261,DH261,DT261,DF261
004252	066372	067106	072522	:ITEM 262	WORD	EM262,DH262,DT262,DF262
004262	066463	067106	072522	:ITEM 263	WORD	EM263,DH263,DT263,DF263
004272	066550	067106	072522	:ITEM 264	WORD	EM264,DH264,DT264,DF264
004302	066635	067106	072522	:ITEM 265	WORD	EM265,DH265,DT265,DF265
004312	066722	070703	072510	:ITEM 266	WORD	EM266,DH266,DT266,DF266
004322	067023	067447	072076	:ITEM 267	WORD	EM267,DH267,DT267,DF267

1679  
 1680  
 1681

```

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

1682

```

.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
  
```

000046 004332  
 000052 000046  
 000044 000200  
 000044 000044  
 000044 004332  
 000044 004332

```

004332
004332 000000
004334 001316
004336 000010
004340 000040
004342 000000
004344 000052

1683
1684
1685 004346

004346 012706 001100
004352 005026
004354 022706 001140
004360 001374
004362 012706 001100

004366 012737 034110 000020
004374 012737 000340 000022
004402 012737 034426 000030
004410 012737 000340 000032
004416 012737 036570 000034
004424 012737 000340 000036
004432 012737 036654 000024
004440 012737 000340 000026
004446 013737 033404 033372
004454 005037 001302
004460 005037 001304
004464 112737 000001 001115

004472 012737 034070 000014
004500 012737 000340 000016
004506 012737 000002 034070
004514 012737 004542 000010
004522 005046
004524 012746 004532
004530 000006
004532 012737 000006 034070 64$
004540 000402
004542 062706 000010 65$
004546 012737 000012 000010 66$
004554 005037 034076
004560 012737 004560 001106
004566 012737 004566 001110

004574 013746 000004
004600 012737 004634 000004
004606 012737 177570 001140
004614 012737 177570 001142
004622 022777 177777 174310
004630 001012
    
```

```

*****
: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 40 ;;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)

START:
.SBTTL INITIALIZE THE COMMON TAGS
:;CLEAR THE COMMON TAGS ($CMTAG) AREA
MOV # $CMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
CLR (R6)+ ;;CLEAR MEMORY LOCATION
CMP # $SWR,R6 ;;DONE?
BNE -.6 ;;LOOP BACK IF NO
MOV # $STACK,SP ;;SETUP THE STACK POINTER
:;INITIALIZE A FEW VECTORS
MOV # $SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
MOV #340,@#IOTVEC+2 ;;LEVEL 7
MOV # $ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
MOV #340,@#EMTVEC+2 ;;LEVEL 7
MOV # $TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
MOV #340,@#TRAPVEC+2;LEVEL 7
MOV # $PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
MOV #340,@#PWRVEC+2 ;;LEVEL 7
MOV $ENDCT,$EOPCT ;;SETUP END-OF-PROGRAM COUNTER
CLR $TIMES ;;INITIALIZE NUMBER OF ITERATIONS
CLR $ESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
MOVB #1,$ERMAX ;;ALLOW ONE ERROR PER TEST
:;INITIALIZE THE "T-BIT" TRAP VECTOR. THEN LOAD LOCATION '$RTRN', IN
:;THE "END-OF-PASS" ($EOP) ROUTINE, WITH A 'RTI' OR 'RTT'.
MOV # $RTRN,@#TBITVEC ;;SET 'T' BIT VECTOR TO $RTRN
MOV #340,@#TBITVEC+2 ;;LEVEL 7
MOV #RTI,$RTRN ;;SET $RTRN TO A RTI
MOV #65$,@#RESVEC ;;TRY TO DO A RTT
CLR -(SP) ;;DUMMY PS
MOV #64$,-(SP) ;;AND PC
RTT ;;TRY THE RTT
64$: MOV #RTT,$RTRN ;;RTT IS LEGAL--SET $RTRN TO A RTT
BR 66$
65$: ADD #10,SP ;;RTT ILLEGAL--CLEAN OFF THE STACK
66$: MOV #RESVEC+2,@#RESVEC ;;RESTORE TRAP CATCHER
CLR $TBIT ;;CLEAR 'T' BIT SWITCH
MOV #,$SLPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
MOV #,$SLPERR ;;SETUP THE ERROR LOOP ADDRESS
:;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
:;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
MOV @#ERRVEC,-(SP) ;;SAVE ERROR VECTOR
MOV #67$,@#ERRVEC ;;SET UP ERROR VECTOR
MOV #DSWR,$SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
MOV #DDISP,$DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
CMP #-1,@SWR ;;TRY TO REFERENCE HARDWARE SWR
BNE 69$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
    
```



```

004632 000403
004634 012716 004642
004640 000002
004642 012737 000176 001140
004650 012737 000174 001142
004656 012637 000004
004662 005037 001324
004666 132737 000200 001337
004674 001403
004676 012737 001340 001140
004704
1686
004704 005227 177777
004710 001047
004712 022737 034024 000042
004720 001443
004722 104401 004770
004726 005737 000042
004732 001012
004734 123727 001336 000001
004742 001406
004744 023727 001140 000176
004752 001005
004754 104405
004756 000403
004760 112737 000001 001134
004766
004766 000420
005030
1687 005030 104401 005036
005034 000434
005126
1688 005126 104401 005134
005132 000426
005210
1689 005210 005037 034104
1690 005214
1691
1692
BR 68$
MOV #68$, (SP)
RTI
MOV #SWREG, SWR
MOV #DISPRÉG, DISPLAY
MOV (SP)+, @#ERRVEC
CLR $PASS
BITB #APTSIZE, $ENVM
BEQ 70$
MOV #$$SWREG, SWR
::AND THE HARDWARE SWR IS NOT = -1
::BRANCH IF NO TIMEOUT
::SET UP FOR TRAP RETURN
::POINT TO SOFTWARE SWR
::RESTORE ERROR VECTOR
::CLEAR PASS COUNT
::TEST USER SIZE UNDER APT
::YES, USE NON-APT SWITCH
::NO, USE APT SWITCH REGISTER
70$:
.SBTTL TYPE PROGRAM NAME
::TYPE THE NAME OF THE PROGRAM IF FIRST PASS
INC #-1
BNE 71$
CMP #SENDAD, @#42
BEQ 71$
TYPE ,72$
.SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
TST @#42
BNE 73$
CMPB $ENV, #1
BEQ 73$
CMP SWR, #SWREG
BNE 74$
GTSWR
BR 74$
MOV #1, $AUTOB
73$:
74$:
BR 71$
::72$: .ASCIZ <CRLF>*CKFPBCO FP11F FLTG PNT PRT B*<CRLF>
71$:
TYPE ,76$
BR 75$
::76$: .ASCIZ !EOP MESSAGES WILL PRINT EVERY 2000 PASSES (15 SECONDS)!<CRLF>
75$:
TYPE ,78$
BR 77$
::78$: .ASCIZ !HIT ANY KEY TO DISABLE/ENABLE EOP MESSAGES!<CRLF>
77$:
CLR EPENDS
LOOP:
;CLR EOP ENABLE/DISABLE FLAG ;DPM002

```

1703

```
.SBTTL TEST # 1 - ROUND\TRUNK TEST
:*****
:TEST 1 ROUND\TRUNK TEST
:*
:* THIS IS A TEST OF THE ROUND\TRUNK
:* FLOWS. IN PARTICULAR TWO THINGS ARE TESTED:
:* FIRST A CONDITION IN WHICH ROUNDING
:* RESULTS IN THE NEED FOR RENORMALIZATION, AND
:* SECOND THE PSW CONDITION CODES N AND
:* Z BIT COMBINATIONS
:*
```

```
005214 000004
1704
1705
1706
1707 005216
005216 104413
1708 005220 012704 003200
1709 005224 170104
1710 005226 012737 005246 001236
1711 005234 012700 006776
1712 005240 172410
1713 005242 012700 007006
1714 005246 172010
1715 005250 170205
1716 005252 012700 006766
1717 005256 174010
1718 005260 012701 007016
1719 005264 012702 000004
1720 005270 022021
1721 005272 001415
1722 005274 012700 006766
1723 005300 012701 007026
1724 005304 012702 000004
1725 005310 022021
1726 005312 001402
1727 005314 000137 006006
1728 005320 077205
1729 005322 000137 006054
1730 005326 077220
1731 005330 020405
1732 005332 001402
1733 005334 000137 006122
```

```
TST1: SCOPE
:ROUND AND NORMALIZE TEST
HH1: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
MOV #3200,R4 ;SET FIU, FIV, AND FD
LDFPS R4
MOV #HH2,$TMP2
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
MOV #HHDATO,R0 ;DID FLOW GO
MOV #HHP3,R1 ;FROM STATE 663
MOV #4,R2 ;TO 313 INSTEAD
HH4: CMP (R0)+,(R1)+ ;OF TO 353
BEQ HH5
JMP HHERO
HH5: SOB R2,HH4
JMP HHER1
HH6: SOB R2,HH3
CMP R4,R5 ;FPS CORRECT?
BEQ HH7
JMP HHERO
```

```
1734
1735
1736
1737
1738
1739 005340
005340 104413
1740 005342 012704 043200
1741
1742 005346 170104
1743 005350 012737 005376 001236
1744 005356 012737 006716 000244
1745 005364 012700 007046
```

```
:THIS IS A TEST OF THE ABILITY
:OF NORMALIZE TO PRODUCE A ZERO EXP. AND
:OF THE R\T ALGORITHM TO PORPERLY SET THE FPS
HH7: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
MOV #043200,R4 ;SET FIU,FIV,AND FD
;FID
LDFPS R4
MOV #HH8,$TMP2 ;IN CASE UNDERFLOW
MOV #HHTRAP,FPVECT ;TRAP OCCURS
MOV #HHP5,R0 ;SET ACO OPERAND
```



1746	005370	172410			LDD	(R0),ACO	
1747	005372	012700	007056		MOV	#HHP6,R0	:FSPC
1748	005376	172010		HH8:	ADDD	(R0),ACO	:TEST INSTRUCTION
1749	005400	170205			STFPS	R5	:GET FPS
1750	005402	012700	006766		MOV	#HHDATO,R0	:GET THE RESULT
1751	005406	174010			STD	ACO,(R0)	
1752	005410	012701	007036		MOV	#HHP4,R1	:IS IT CORRECT
1753	005414	012702	000004		MOV	#4,R2	
1754	005420	022021		HH9:	CMP	(R0)+,(R1)+	
1755	005422	001402			BEQ	HH10	
1756	005424	000137	006170		JMP	HHER2	
1757	005430	077205		HH10:	SOB	R2,HH9	
1758	005432	052704	100004		BIS	#100004,R4	:FPS CORRECT?
1759	005436	020405			CMP	R4,R5	
1760	005440	001402			BEQ	HH11	
1761	005442	000137	006236		JMP	HHER3	
1762							
1763							
1764	005446						
	005446	104413		HH11:	LPERR		:SET UP THE LOOP ON ERROR ADDRESS. :SET FIV, FIV, AND FD
1765							
1766	005450	012704	043200		MOV	#043200,R4	
1767	005454	170104			LDFPS	R4	
1768	005456	012737	005476	001236	MOV	#HH12,\$TMP2	
1769	005464	012700	007076		MOV	#HHP8,R0	:SET ACO OPERAND
1770	005470	172410			LDD	(R0),ACO	
1771	005472	012700	007106		MOV	#HHP9,R0	:FSPC
1772	005476	172010		HH12:	ADDD	(R0),ACO	:TEST INSTRUCTION
1773	005500	170205			STFPS	R5	:GET FPS
1774	005502	012700	006766		MOV	#HHDATO,R0	:GET THE RESULT
1775	005506	174010			STD	ACO,(R0)	
1776	005510	012701	007066		MOV	#HHP7,R1	:IS IT CORRECT
1777	005514	012702	000004		MOV	#4,R2	
1778	005520	022021		HH13:	CMP	(R0)+,(R1)+	
1779	005522	001415			BEQ	HH16	
1780	005524	012700	006766		MOV	#HHDATO,R0	
1781	005530	012701	007036		MOV	#HHP4,R1	
1782	005534	012702	000004		MOV	#4,R2	
1783	005540	022021		HH14:	CMP	(R0)+,(R1)+	
1784	005542	001402			BEQ	HH15	
1785	005544	000137	006304		JMP	HHER4	
1786	005550	077205		HH15:	SOB	R2,HH14	
1787	005552	000137	006352		JMP	HHER5	
1788	005556	077220		HH16:	SOB	R2,HH13	
1789	005560	052704	100014		BIS	#100014,R4	:FPS CORRECT?
1790	005564	020405			CMP	R4,R5	
1791	005566	001402			BEQ	HH17	
1792	005570	000137	006420		JMP	HHER6	
1793							
1794	005574						
	005574	104413		HH17:	LPERR		:SET UP THE LOOP ON ERROR ADDRESS. :SET FIV, FIV, AND FD
1795	005576	012704	000200		MOV	#00200,R4	
1796	005602	170104			LDFPS	R4	
1797	005604	012737	005632	001236	MOV	#HH18,\$TMP2	
1798	005612	012737	037450	000244	MOV	#FPSPUR,FPVECT	
1799	005620	012700	007076		MOV	#HHP8,R0	:SET ACO OPERAND
1800	005624	172410			LDD	(R0),ACO	

:THIS IS A TEST OF THE R\T ALGORITHM'S  
:ABILITY TO SET BOTH N AND Z ON A - 0 RESULT.

:TEST THAT CC ARE CLEARED BY R\T

1801	005626	012700	007076	
1802	005632	172010		
1803	005634	170205		
1804	005636	012700	006766	
1805	005642	174010		
1806	005644	012701	007116	
1807	005650	012702	000004	
1808	005654	022021		
1809	005656	001402		
1810	005660	000137	006466	
1811	005664	077205		
1812	005666	052704	000000	
1813	005672	020405		
1814	005674	001402		
1815	005676	000137	006534	
1816				
1817	005702			
	005702	104413		
1818	005704	012704	003200	
1819	005710	170104		
1820	005712	012737	005732	001236
1821	005720	012700	007046	
1822	005724	172410		
1823	005726	012700	007046	
1824	005732	172010		
1825	005734	170205		
1826	005736	012700	006766	
1827	005742	174010		
1828	005744	012701	007126	
1829	005750	012702	000004	
1830	005754	022021		
1831	005756	001402		
1832	005760	000137	006602	
1833	005764	077205		
1834	005766	052704	000010	
1835	005772	020405		
1836	005774	001402		
1837	005776	000137	006650	
1838	006002	000137	007136	
1839	006006			
	006006	010537	001252	
	006012	010437	001254	
	006016	012737	007006	001240
	006024	012737	006776	001242
	006032	012737	006766	001244
	006040	012737	007016	001246
	006046	104207		
	006050	000137	007136	
1840	006054			
	006054	010537	001252	
	006060	010437	001254	
	006064	012737	007006	001240
	006072	012737	006776	001242
	006100	012737	006766	001244
	006106	012737	007016	001246
	006114	104211		
	006116	000137	007136	

```

HH18:  MOV #HHP8,R0           ;FSPC
        ADD (R0),ACO         ;TEST INSTRUCTION
        STFPS R5             ;GET FPS
        MOV #HHDATA,R0      ;GET THE RESULT
        STD ACO,(R0)
        MOV #HHP10,R1       ;IS IT CORRECT
        MOV #4,R2
HH19:  CMP (R0)+,(R1)+
        BEQ HH20
        JMP HHER7
HH20:  SOB R2,HH19
        BIS #00000,R4       ;FPS CORRECT?
        CMP R4,R5
        BEQ HH21
        JMP HHER8
;TEST THAT N IS SET BY R\T
HH21:  LPERR
        MOV #3200,R4
        LDFPS R4
        MOV #HH22,$TMP2
        MOV #HHP5,R0
        LDD (R0),ACO
        MOV #HHP5,R0
        ADDD (R0),ACO
        STFPS R5
        MOV #HHDATA,R0
        STD ACO,(R0)
        MOV #HHP11,R1
        MOV #4,R2
HH22:  CMP (R0)+,(R1)+
        BEQ HH24
        JMP HHER9
HH23:  SOB R2,HH23
        BIS #10,R4
        CMP R4,R5
        BEQ HH25
        JMP HHER10
HH24:  SOB R2,HH23
        BIS #10,R4
        CMP R4,R5
        BEQ HH25
        JMP HHER10
HH25:  JMP HHDONE
HHER0: MOV R5,$TMP10
        MOV R4,$TMP11
        MOV #HHP1,$TMP3
        MOV #HHP0,$TMP4
        MOV #HHDATA,$TMP5
        MOV #HHP2,$TMP6
1$:   ERROR +207
        JMP HHDONE
HHER1: MOV R5,$TMP10
        MOV R4,$TMP11
        MOV #HHP1,$TMP3
        MOV #HHP0,$TMP4
        MOV #HHDATA,$TMP5
        MOV #HHP2,$TMP6
1$:   ERROR +211
        JMP HHDONE
;SET UP THE LOOP ON ERROR ADDRESS.
;SET FIV, FIV, AND FD
;SET ACO OPERAND
;FSPC
;TEST INSTRUCTION
;GET FPS
;GET THE RESULT
;IS IT CORRECT
;FPS CORRECT?

```



1841	006122				HHER00:	MOV	R5,\$TMP10
	006122	010537	001252			MOV	R4,\$TMP11
	006126	010437	001254			MOV	#HHP1,\$TMP3
	006132	012737	007006	001240		MOV	#HHP0,\$TMP4
	006140	012737	006776	001242		MOV	#HHDAT0,\$TMP5
	006146	012737	006766	001244		MOV	#HHP2,\$TMP6
	006154	012737	007016	001246		1\$:	ERROR
	006162	104210					+210
	006164	000137	007136			JMP	HHDONE
1842	006170				HHER2:	MOV	R5,\$TMP10
	006170	010537	001252			MOV	R4,\$TMP11
	006174	010437	001254			MOV	#HHP6,\$TMP3
	006200	012737	007056	001240		MOV	#HHP5,\$TMP4
	006206	012737	007046	001242		MOV	#HHDAT0,\$TMP5
	006214	012737	006766	001244		MOV	#HHP4,\$TMP6
	006222	012737	007036	001246		1\$:	ERROR
	006230	104207					+207
	006232	000137	007136			JMP	HHDONE
1843	006236				HHER3:	MOV	R5,\$TMP10
	006236	010537	001252			MOV	R4,\$TMP11
	006242	010437	001254			MOV	#HHP6,\$TMP3
	006246	012737	007056	001240		MOV	#HHP5,\$TMP4
	006254	012737	007046	001242		MOV	#HHDAT0,\$TMP5
	006262	012737	006766	001244		MOV	#HHP4,\$TMP6
	006270	012737	007036	001246		1\$:	ERROR
	006276	104214					+214
	006300	000137	007136			JMP	HHDONE
1844	006304				HHER4:	MOV	R5,\$TMP10
	006304	010537	001252			MOV	R4,\$TMP11
	006310	010437	001254			MOV	#HHP9,\$TMP3
	006314	012737	007106	001240		MOV	#HHP8,\$TMP4
	006322	012737	007076	001242		MOV	#HHDAT0,\$TMP5
	006330	012737	006766	001244		MOV	#HHP7,\$TMP6
	006336	012737	007066	001246		1\$:	ERROR
	006344	104207					+207
	006346	000137	007136			JMP	HHDONE
1845	006352				HHER5:	MOV	R5,\$TMP10
	006352	010537	001252			MOV	R4,\$TMP11
	006356	010437	001254			MOV	#HHP9,\$TMP3
	006362	012737	007106	001240		MOV	#HHP8,\$TMP4
	006370	012737	007076	001242		MOV	#HHDAT0,\$TMP5
	006376	012737	006766	001244		MOV	#HHP7,\$TMP6
	006404	012737	007066	001246		1\$:	ERROR
	006412	104216					+216
	006414	000137	007136			JMP	HHDONE
1846	006420				HHER6:	MOV	R5,\$TMP10
	006420	010537	001252			MOV	R4,\$TMP11
	006424	010437	001254			MOV	#HHP9,\$TMP3
	006430	012737	007106	001240		MOV	#HHP8,\$TMP4
	006436	012737	007076	001242		MOV	#HHDAT0,\$TMP5
	006444	012737	006766	001244		MOV	#HHP7,\$TMP6
	006452	012737	007066	001246		1\$:	ERROR
	006460	104215					+215
	006462	000137	007136			JMP	HHDONE
1847	006466				HHER7:	MOV	R5,\$TMP10
	006466	010537	001252			MOV	R4,\$TMP11
	006472	010437	001254				

	006476	012737	007076	001240		MOV	#HHP8,\$TMP3	
	006504	012737	007076	001242		MOV	#HHP8,\$TMP4	
	006512	012737	006766	001244		MOV	#HHDATO,\$TMP5	
	006520	012737	007116	001246		MOV	#HHP10,\$TMP6	
	006526	104207			1\$:	ERROR	+207	
1848	006530	000137	007136			JMP	HHDONE	
	006534				HHER8:			
	006534	010537	001252			MOV	R5,\$TMP10	
	006540	010437	001254			MOV	R4,\$TMP11	
	006544	012737	007076	001240		MOV	#HHP8,\$TMP3	
	006552	012737	007076	001242		MOV	#HHP8,\$TMP4	
	006560	012737	006766	001244		MOV	#HHDATO,\$TMP5	
	006566	012737	007116	001246		MOV	#HHP10,\$TMP6	
	006574	104212			1\$:	ERROR	+212	
1849	006576	000137	007136			JMP	HHDONE	
	006602				HHER9:			
	006602	010537	001252			MOV	R5,\$TMP10	
	006606	010437	001254			MOV	R4,\$TMP11	
	006612	012737	007046	001240		MOV	#HHP5,\$TMP3	
	006620	012737	007046	001242		MOV	#HHP5,\$TMP4	
	006626	012737	006766	001244		MOV	#HHDATO,\$TMP5	
	006634	012737	007126	001246		MOV	#HHP11,\$TMP6	
	006642	104207			1\$:	ERROR	+207	
1850	006644	000137	007136			JMP	HHDONE	
	006650				HHER10:			
	006650	010537	001252			MOV	R5,\$TMP10	
	006654	010437	001254			MOV	R4,\$TMP11	
	006660	012737	007046	001240		MOV	#HHP5,\$TMP3	
	006666	012737	007046	001242		MOV	#HHP5,\$TMP4	
	006674	012737	006766	001244		MOV	#HHDATO,\$TMP5	
	006702	012737	007126	001246		MOV	#HHP11,\$TMP6	
	006710	104213			1\$:	ERROR	+213	
	006712	000137	007136			JMP	HHDONE	
1851	006716	013703	001236		HHTRAP:	MOV	\$TMP2,R3	:WAS THE TRAP TO 264
1852	006722	062703	000002			ADD	#2,R3	:ON THE INSTRUCTION
1853	006726	020316				CMP	R3,(SP)	:BEING TESTED?
1854	006730	001402				BEQ	1\$	
1855	006732	000137	037450			JMP	FPSPUR	
1856	006736	011637	001236		1\$:	MOV	(SP),\$TMP2	:FAILURE OF FPS INTERRUPT
1857								:DISABLE BIT (FID=1)
1858	006742	022626				CMP	(SP)+,(SP)+	:TO INHIBIT TRAP.
1859	006744	170201				STFPS	R1	
1860	006746	010137	001240			MOV	R1,\$TMP3	
1861	006752	170301				STST	R1	
1862	006754	010137	001242			MOV	R1,\$TMP4	
1863	006760	104217			2\$:	ERROR	+217	
1864	006762	000137	007136			JMP	HHDONE	
1865	006766	000000			HHDATO:	0		
1866	006770	000000				0		
1867	006772	000000				0		
1868	006774	000000				0		
1869	006776	000452			HHP0:	452		
1870	007000	125252				125252		
1871	007002	125252				125252		
1872	007004	125253				125253		
1873	007006	000252			HHP1:	252		
1874	007010	125252				125252		



1875 007012 125252  
1876 007014 125252  
1877 007016 000600  
1878 007020 000000  
1879 007022 000000  
1880 007024 000000  
1881 007026 000400  
1882 007030 000000  
1883 007032 000000  
1884 007034 000000  
1885 007036 000000  
1886 007040 000000  
1887 007042 000000  
1888 007044 000000  
1889 007046 100200  
1890 007050 000000  
1891 007052 000000  
1892 007054 000000  
1893 007056 000300  
1894 007060 000000  
1895 007062 000000  
1896 007064 000000  
1897 007066 100000  
1898 007070 000000  
1899 007072 000000  
1900 007074 000000  
1901 007076 000200  
1902 007100 000000  
1903 007102 000000  
1904 007104 000000  
1905 007106 100300  
1906 007110 000000  
1907 007112 000000  
1908 007114 000000  
1909 007116 000400  
1910 007120 000000  
1911 007122 000000  
1912 007124 000000  
1913 007126 100400  
1914 007130 000000  
1915 007132 000000  
1916 007134 000000  
1917 007136 104412  
007136 104412

HHP2: 125252  
125252  
600  
0  
0  
0  
HHP3: 400  
0  
0  
0  
0  
HHP4: 0  
0  
0  
0  
0  
HHP5: 100200  
0  
0  
0  
0  
HHP6: 300  
0  
0  
0  
0  
HHP7: 100000  
0  
0  
0  
0  
HHP8: 200  
0  
0  
0  
0  
HHP9: 100300  
0  
0  
0  
0  
HHP10: 400  
0  
0  
0  
0  
HHP11: 100400  
0  
0  
0  
0  
HHDONE: RSETUP

:HHP0 + HHP1 WITH  
:PROPER NORMALIZATION

:HHP0 + HHP1 WITH  
:BAD NORMALIZATION

:HHP7 = HHP8 + HHP9  
: = HHP5 + HHP6

:HHP10 = HHP8 + HHP8

:HHP11 = HHP5 + HHP5

: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?).

1918  
1932

1933

```
.SBTTL TEST # 2 - OVER\UNDER TEST
:*****
:*TEST 2 OVER\UNDER TEST
:*
:*THIS IS A PARTIAL TEST OF THE OVER\UNDER
:*FLOWS. ONE OVERFLOW AND TWO UNDERFLOW
:*CONDITIONS ARE CHECKED. THE REMAINING
:*UNDERFLOW COND. AND THE REMAINING OVERFLOW
:*COND. WILL BE CHECKED LATER USING THE
:*XXX INSTRUCTION. HERE EACH CONDITION TESTED
:*IS CHECKED BOTH WITH TRAPS ENABLED
:*(FIU=1 OR FIV=1) AND ALSO WITH TRAPS
:*DISABLED (FIU=0 OR FIV=0).
:*
:*****
```

```
1934 007140 000004
1934
1935
1936 007142
1937 007142 104413
1938 007144 012704 000200
1939 007150 170104
1940 007152 012737 007200 001236
1941 007160 012737 010232 000244
1942 007166 012700 012004
1943 007172 172410
1944 007174 012700 012004
1945 007200 172010
1946 007202 170205
1947 007204 012700 011734
1948 007210 174010
1949 007212 012701 012014
1950 007216 012702 000004
1951 007222 022021
1952 007224 001402
1953 007226 000137 010330
1954 007232 077205
1955 007234 052704 000006
1956 007240 020405
1957 007242 001402
1958 007244 000137 010376
1959
1960 007250
1961 007250 104413
1962 007252 012704 001200
1963 007256 170104
1964 007260 012737 007306 001236
1965 007266 012737 007324 000244
1966 007274 012700 012004
1967 007300 172410
1968 007302 012700 012004
1969 007306 172010
1970 007310 170000
1971 007312 012700 011734
1972 007316 174010
1973 007320 000137 010444
```

```
TST2: SCOPE
:TEST OVERFLOW CONDITION WITH TRAP DISABLER FIV=0
GG1:
LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
MOV #200,R4 ;CLEAR FIU, FIV, AND SET FD
LDFPS R4
MOV #GG2,$TMP2
MOV #GGER0,FPVECT
MOV #GGP5,R0 ;SET ACO OPERAND
LDD (R0),ACO
MOV #GGP5,R0 ;FSRC
GG2: ADDD (R0),ACO ;TEST INSTRUCTION
STFPS R5 ;GET FPS
MOV #GGDAT0,R0 ;GET THE RESULT
STD ACO,(R0)
MOV #GGP6,R1 ;IS IT CORRECT
MOV #4,R2
GG3: CMP (R0)+,(R1)+
BEQ GG4
JMP GGER1
GG4: SOB R2,GG3
BIS #6,R4 ;FPS CORRECT?
CMP R4,R5
BEQ GG5
JMP GGER2
;TEST OVERFLOW WITH TRAPS ENABLED
;FIV = 1
GG5:
LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
MOV #1200,R4 ;CLEAR FIU, SET FIV, AND FD
LDFPS R4
MOV #GG6,$TMP2
MOV #GG7,FPVECT
MOV #GGP5,R0 ;SET ACO OPERAND
LDD (R0),ACO
MOV #GGP5,R0 ;FSRC
GG6: ADDD (R0),ACO ;TEST INSTRUCTION
CFCC
MOV #GGDAT0,R0
STD ACO,(R0)
JMP GGER3
```



1973	007324	013703	001236		GG7:	MOV	\$TMP2,R3	
1974	007330	062703	000002			ADD	#2,R3	
1975	007334	020316				CMP	R3,(SP)	
1976	007336	001402				BEQ	1\$	
1977	007340	000137	037450			JMP	FPSPUR	
1978	007344	011637	001236		1\$:	MOV	(SP),\$TMP2	
1979	007350	022626				CMP	(SP)+,(SP)+	
1980	007352	170205				STFPS	R5	
1981	007354	012700	011734			MOV	#GGDATO,R0	;GET THE RESULT
1982	007360	174010				STD	ACO,(R0)	
1983	007362	012701	012014			MOV	#GGP6,R1	;IS IT CORRECT
1984	007366	012702	000004			MOV	#4,R2	
1985	007372	022021			GG8:	CMP	(R0)+,(R1)+	
1986	007374	001402				BEQ	GG9	
1987	007376	000137	010512			JMP	GGER4	
1988	007402	077205			GG9:	SOB	R2,GG8	
1989	007404	052704	100006			BIS	#100006,R4	
1990	007410	020405				CMP	R4,R5	;FPS CORRECT?
1991	007412	001402				BEQ	1\$	
1992	007414	000137	010562			JMP	GGER6	
1993	007420	012704	000010		1\$:	MOV	#10,R4	
1994					:CHECK	FEC		
1995	007424	170305				STST	R5	
1996	007426	020405				CMP	R4,R5	
1997	007430	001402				BEQ	GG10	
1998	007432	000137	010514			JMP	GGER5	
1999					:CHECK	UNDER FLOW CONDITION WITH		
2000					:TRAPS	DISABLED (FIU = 0)		
2001	007436				GG10:	LPERR		;SET UP THE LOOP ON ERROR ADDRESS.
	007436	104413				MOV	#0200,R4	;SET FIU, FIV, AND FD
2002	007440	012704	000200			LDFPS	R4	
2003	007444	170104				MOV	#GG11,\$TMP2	
2004	007446	012737	007474	001236		MOV	#GGER7,FPVECT	
2005	007454	012737	010630	000244		MOV	#GGP2,R0	;SET ACO OPERAND
2006	007462	012700	011754			LDD	(R0),ACO	;FSRC
2007	007466	172410				MOV	#GGP3,R0	
2008	007470	012700	011764		GG11:	ADDD	(R0),ACO	;TEST INSTRUCTION
2009	007474	172010				STFPS	R5	;GET FPS
2010	007476	170205				MOV	#GGDATO,R0	;GET THE RESULT
2011	007500	012700	011734			STD	ACO,(R0)	
2012	007504	174010				MOV	#GGP6,R1	;IS IT CORRECT
2013	007506	012701	012014			MOV	#4,R2	
2014	007512	012702	000004		GG12:	CMP	(R0)+,(R1)+	
2015	007516	022021				BEQ	GG13	
2016	007520	001402				JMP	GGER8	
2017	007522	000137	010726		GG13:	SOB	R2,GG12	
2018	007526	077205				BIS	#4,R4	;FPS CORRECT?
2019	007530	052704	000004			CMP	R4,R5	
2020	007534	020405				BEQ	GG14	
2021	007536	001402				JMP	GGER9	
2022	007540	000137	010774		:CHECK	UNDERFLOW CONDITION WITH		
2023					:TRAP	ENABLED (FIU = 1)		
2024					GG14:	LPERR		;SET UP THE LOOP ON ERROR ADDRESS.
2025	007544					MOV	#2200,R4	;SET FIU, FIV, AND FD
	007544	104413				LDFPS	R4	
2026	007546	012704	002200					
2027	007552	170104						

```

2028 007554 012737 007602 001236      MOV      #GG15,$TMP2
2029 007562 012737 007620 000244      MOV      #GG16,FPVECT
2030 007570 012700 011754      MOV      #GGP2,R0      ;SET ACO OPERAND
2031 007574 172410      LDD      (R0),ACO      ;FSPC
2032 007576 012700 011764      MOV      #GGP3,R0
2033 007602 172010      GG15:    ADDD     (R0),ACO      ;TEST INSTRUCTION
2034 007604 170000      CFCC
2035 007606 012700 011734      MOV      #GGDATO,R0
2036 007612 174010      STD      ACO,(R0)
2037 007614 000137 011042      JMP      GGER10
2038 007620 013703 001236      GG16:    MOV      $TMP2,R3
2039 007624 062703 000002      ADD      #2,R3
2040 007630 021603      CMP      (SP),R3
2041 007632 001402      BEQ      1$
2042 007634 000137 037450      JMP      FPSPUR
2043 007640 011637 001236      1$:     MOV      (SP),$TMP2
2044 007644 022626      CMP      (SP)+,(SP)+
2045 007646 170205      STFPS   R5      ;GET FPS
2046 007650 012700 011734      MOV      #GGDATO,R0      ;GET THE RESULT
2047 007654 174010      STD      ACO,(R0)
2048 007656 012701 012024      MOV      #GGP7,R1      ;IS IT CORRECT
2049 007662 012702 000004      MOV      #4,R2
2050 007666 022021      GG17:    CMP      (R0)+,(R1)+
2051 007670 001402      BEQ      GG18
2052 007672 000137 011110      JMP      GGER11
2053 007676 077205      GG18:    SOB      R2,GG17
2054 007700 052704 100000      BIS      #100000,R4
2055 007704 020405      CMP      R4,R5      ;FPS CORRECT?
2056 007706 001402      BEQ      2$
2057 007710 000137 011156      JMP      GGER12
2058 007714      2$:
2059 007714 012704 000012      1$:     MOV      #12,R4
2060      ;CHECK FEC
2061 007720 170305      STST    R5
2062 007722 020405      CMP      R4,R5
2063 007724 001402      BEQ      GG19
2064 007726 000177 001272      JMP      @GGER13
2065      ;CHECK UNDERFLOW CONDITION WITH TRAPS
2066      ;DISABLED (FIU = 0)
2067      GG19:
2068 007732 104413      LPERR   ;SET UP THE LOOP ON ERROR ADDRESS.
2069 007734 012704 000200      MOV      #0200,R4      ;SET FIU, FIV, AND FD
2070 007740 170104      LDFPS  R4
2071 007742 012737 007770 001236      MOV      #GG20,$TMP2
2072 007750 012737 011272 000244      MOV      #GGER14,FPVECT
2073 007756 012700 011754      MOV      #GGP2,R0      ;SET ACO OPERAND
2074 007762 172410      LDD      (R0),ACO
2075 007770 012700 012034      MOV      #GGP8,R0      ;FSPC
2076 007772 170205      GG20:    ADDD     (R0),ACO      ;TEST INSTRUCTION
2077 007774 012700 011734      STFPS   R5      ;GET FPS
2078 010000 174010      MOV      #GGDATO,R0      ;GET THE RESULT
2079 010002 012701 012014      STD      ACO,(R0)
2080 010006 012702 000004      MOV      #GGP6,R1      ;IS IT CORRECT
2081 010012 022021      GG21:    MOV      #4,R2
2082 010014 001402      CMP      (R0)+,(R1)+
2083 010016 000137 011370      BEQ      GG22
      JMP      GGER15
    
```



2084 010022 077205  
2085 010024 052704 000004  
2086 010030 020405  
2087 010032 001402  
2088 010034 000137 011436

GG22: SOB R2,GG21  
BIS #4,R4  
CMP R4,R5  
BEQ GG23  
JMP GGER16

;FPS CORRECT?

2090

;CHECK UNDERFLOW CONDITION WITH TRAP



```

2092                                     :ENABLED (FIU = 1)
2093 010040                               GG23: LPERR
      010040 104413                       MOV #2200,R4           ;SET UP THE LOOP ON ERROR ADDRESS.
2094 010042 012704 002200                 LDFPS R4              ;SET FIU, FIV, AND FD
2095 010046 170104                         MOV #GG24,$TMP2
2096 010050 012737 010076 001236         MOV #GG25,FPVECT
      010114 000244
2097 010056 012737 010114
2098 010064 012700 011754                 MOV #GGP2,R0          ;SET ACO OPERAND
2099 010070 172410                         LDD (R0),ACO
2100 010072 012700 012034                 MOV #GGP8,R0          ;FSRC
2101 010076 172010                         GG24: ADDD (R0),ACO    ;TEST INSTRUCTION
2102 010100 170000                         CFCC
2103 010102 012700 011734                 MOV #GGDATO,R0
2104 010106 174010                         STD ACO,(R0)
2105 010110 000137 011504                 JMP GGER17
2106 010114 013700 001236                 GG25: MOV $TMP2,R0
2107 010120 062700 000002                 ADD #2,R0
2108 010124 020016                         CMP R0,(SP)
2109 010126 001402                         BEQ 1$
2110 010130 000137 037450                 JMP FPSPUR
2111 010134 011637 001236                 1$: MOV (SP),$TMP2
2112 010140 022626                         CMP (SP)+,(SP)+
2113 010142 170205                         STFPS R5              ;GET FPS
2114 010144 012700 011734                 MOV #GGDATO,R0        ;GET THE RESULT
2115 010150 174010                         STD ACO,(R0)
2116 010152 012701 012044                 MOV #GGP9,R1          ;IS IT CORRECT
2117 010156 012702 000004                 MOV #4,R2
2118 010162 022021                         GG26: CMP (R0)+,(R1)+
2119 010164 001402                         BEQ GG27
2120 010166 000137 011552                 JMP GGER18
2121 010172 077205                         GG27: SOB R2,GG26
2122 010174 052704 100004                 BIS #100004,R4
2123 010200 020405                         CMP R4,R5              ;FPS CORRECT?
2124 010202 001402                         BEQ 1$
2125 010204 000137 011666                 JMP GGER20
2126 010210 012704 000012                 1$: MOV #12,R4
2127                                     :CHECK FEC
2128 010214 170305                         STST R5
2129 010216 020405                         CMP R4,R5
2130 010220 001402                         BEQ GG28
2131 010222 000137 011620                 JMP GGER19
2132
2133 010226 000137 012054                 GG28: JMP GGDONE
2134
2135 010232 013701 001236                 GGER0: MOV $TMP2,R1
2136 010236 062701 000002                 ADD #2,R1
2137 010242 020116                         CMP R1,(SP)
2138 010244 001402                         BEQ 10$
2139 010246 000137 037450                 5$: JMP FPSPUR
2140 010252                                     10$:
2141 010252 170301                         STST R1
2142 010254 020127 000010                 CMP R1,#10
2143 010260 001372                         BNE 5$
2144 010262 022626                         CMP (SP)+,(SP)+
2145 010264 012700 011734                 MOV #GGDATO,R0
2146 010270 174010                         STD ACO,(R0)
2147 010272 012737 012004 001240         MOV #GGP5,$TMP3

```

	010300	012737	012004	001242		MOV	#GGP5,\$TMP4
	010306	012737	011734	001244		MOV	#GGDATO,\$TMP5
	010314	012737	012014	001246		MOV	#GGP6,\$TMP6
	010322	104220			1\$:	ERROR	+220
	010324	000137	012054			JMP	GGDONE
2148							
2149	010330				GGER1:		
	010330	010537	001252			MOV	R5,\$TMP10
	010334	010437	001254			MOV	R4,\$TMP11
	010340	012737	012004	001240		MOV	#GGP5,\$TMP3
	010346	012737	012004	001242		MOV	#GGP5,\$TMP4
	010354	012737	011734	001244		MOV	#GGDATO,\$TMP5
	010362	012737	012014	001246		MOV	#GGP6,\$TMP6
	010370	104207			1\$:	ERROR	+207
	010372	000137	012054			JMP	GGDONE
2150							
2151	010376				GGER2:		
	010376	010537	001252			MOV	R5,\$TMP10
	010402	010437	001254			MOV	R4,\$TMP11
	010406	012737	012004	001240		MOV	#GGP5,\$TMP3
	010414	012737	012004	001242		MOV	#GGP5,\$TMP4
	010422	012737	011734	001244		MOV	#GGDATO,\$TMP5
	010430	012737	012014	001246		MOV	#GGP6,\$TMP6
	010436	104232			1\$:	ERROR	+232
	010440	000137	012054			JMP	GGDONE
2152							
2153	010444				GGER3:		
	010444	010537	001252			MOV	R5,\$TMP10
	010450	010437	001254			MOV	R4,\$TMP11
	010454	012737	012004	001240		MOV	#GGP5,\$TMP3
	010462	012737	012004	001242		MOV	#GGP5,\$TMP4
	010470	012737	011734	001244		MOV	#GGDATO,\$TMP5
	010476	012737	012014	001246		MOV	#GGP6,\$TMP6
	010504	104221			1\$:	ERROR	+221
	010506	000137	012054			JMP	GGDONE
2154							
2155	010512	000706			GGER4:	BR	GGER1
2156							
2157	010514				GGER5:		
	010514	010537	001252			MOV	R5,\$TMP10
	010520	010437	001254			MOV	R4,\$TMP11
	010524	012737	012004	001240		MOV	#GGP5,\$TMP3
	010532	012737	012004	001242		MOV	#GGP5,\$TMP4
	010540	012737	011734	001244		MOV	#GGDATO,\$TMP5
	010546	012737	012014	001246		MOV	#GGP6,\$TMP6
	010554	104226			1\$:	ERROR	+226
	010556	000137	012054			JMP	GGDONE
2158							
2159	010562				GGER6:		
	010562	010537	001252			MOV	R5,\$TMP10
	010566	010437	001254			MOV	R4,\$TMP11
	010572	012737	012004	001240		MOV	#GGP5,\$TMP3
	010600	012737	012004	001242		MOV	#GGP5,\$TMP4
	010606	012737	011734	001244		MOV	#GGDATO,\$TMP5
	010614	012737	012014	001246		MOV	#GGP6,\$TMP6
	010622	104227			1\$:	ERROR	+227
	010624	000137	012054			JMP	GGDONE

2160						
2161	010630	013701	001236		GGER7:	MOV \$TMP2,R1
2162	010634	062701	000002			ADD #2,R1
2163	010640	020116				CMP R1,(SP)
2164	010642	001402				BEQ 10\$
2165	010644	000137	037450		5\$:	JMP FPSPUR
2166	010650				10\$:	
2167	010650	170301				STST R1
2168	010652	020127	000012			CMP R1,#12
2169	010656	001372				BNE 5\$
2170	010660	022626				CMP (SP)+,(SP)+
2171	010662	012700	011734			MOV #GGDATO,R0
2172	010666	174010				STD ACO,(R0)
2173	010670	012737	011764	001240		MOV #GGP3,\$TMP3
	010676	012737	011754	001242		MOV #GGP2,\$TMP4
	010704	012737	011734	001244		MOV #GGDATO,\$TMP5
	010712	012737	012014	001246		MOV #GGP6,\$TMP6
	010720	104224			1\$:	ERROR +224
	010722	000137	012054			JMP GGDONE
2174						
2175	010726				GGER8:	MOV R5,\$TMP10
	010726	010537	001252			MOV R4,\$TMP11
	010732	010437	001254			MOV #GGP3,\$TMP3
	010736	012737	011764	001240		MOV #GGP2,\$TMP4
	010744	012737	011754	001242		MOV #GGDATO,\$TMP5
	010752	012737	011734	001244		MOV #GGP6,\$TMP6
	010760	012737	012014	001246		MOV #GGP6,\$TMP6
	010766	104207			1\$:	ERROR +207
	010770	000137	012054			JMP GGDONE
2176						
2177	010774				GGER9:	MOV R5,\$TMP10
	010774	010537	001252			MOV R4,\$TMP11
	011000	010437	001254			MOV #GGP3,\$TMP3
	011004	012737	011764	001240		MOV #GGP2,\$TMP4
	011012	012737	011754	001242		MOV #GGDATO,\$TMP5
	011020	012737	011734	001244		MOV #GGP6,\$TMP6
	011026	012737	012014	001246		MOV #GGP6,\$TMP6
	011034	104232			1\$:	ERROR +232
	011036	000137	012054			JMP GGDONE
2178						
2179	011042				GGER10:	MOV R5,\$TMP10
	011042	010537	001252			MOV R4,\$TMP11
	011046	010437	001254			MOV #GGP3,\$TMP3
	011052	012737	011764	001240		MOV #GGP2,\$TMP4
	011060	012737	011754	001242		MOV #GGDATO,\$TMP5
	011066	012737	011734	001244		MOV #GGP7,\$TMP6
	011074	012737	012024	001246		MOV #GGP7,\$TMP6
	011102	104225			1\$:	ERROR +225
	011104	000137	012054			JMP GGDONE
2180						
2181	011110				GGER11:	MOV R5,\$TMP10
	011110	010537	001252			MOV R4,\$TMP11
	011114	010437	001254			MOV #GGP3,\$TMP3
	011120	012737	011764	001240		MOV #GGP2,\$TMP4
	011126	012737	011754	001242		MOV #GGDATO,\$TMP5
	011134	012737	011734	001244		MOV #GGP7,\$TMP6
	011142	012737	012024	001246		MOV #GGP7,\$TMP6



	011150	104207		1\$:	ERROR	+207
	011152	000137	012054		JMP	GGDONE
2182	011156			GGER12:		
2183	011156	010537	001252		MOV	R5,\$TMP10
	011162	010437	001254		MOV	R4,\$TMP11
	011166	012737	011764	001240	MOV	#GGP3,\$TMP3
	011174	012737	011754	001242	MOV	#GGP2,\$TMP4
	011202	012737	011734	001244	MOV	#GGDAT0,\$TMP5
	011210	012737	012024	001246	MOV	#GGP7,\$TMP6
	011216	104231		1\$:	ERROR	+231
	011220	000137	012054		JMP	GGDONE
2184	011224			GGER13:		
2185	011224	010537	001252		MOV	R5,\$TMP10
	011230	010437	001254		MOV	R4,\$TMP11
	011234	012737	011764	001240	MOV	#GGP3,\$TMP3
	011242	012737	011754	001242	MOV	#GGP2,\$TMP4
	011250	012737	011734	001244	MOV	#GGDAT0,\$TMP5
	011256	012737	012024	001246	MOV	#GGP7,\$TMP6
	011264	104230		1\$:	ERROR	+230
	011266	000137	012054		JMP	GGDONE
2186	011272	013701	001236	GGER14:	MOV	\$TMP2,R1
2187	011276	062701	000002		ADD	#2,R1
2188	011302	020116			CMP	R1,(SP)
2189	011304	001402			BEQ	10\$
2190	011306	000137	037450	5\$:	JMP	FPSPUR
2191	011312			10\$:		
2192	011312	170301			STST	R1
2193	011314	020127	000012		CMP	R1,#12
2194	011320	001372			BNE	5\$
2195	011322	022626			CMP	(SP)+,(SP)+
2196	011324	012700	011734		MOV	#GGDAT0,R0
2197	011330	174010			STD	AC0,(R0)
2198	011332	012737	011744	001240	MOV	#GGP1,\$TMP3
2199	011340	012737	011764	001242	MOV	#GGP3,\$TMP4
2200	011346	012737	011734	001244	MOV	#GGDAT0,\$TMP5
	011354	012737	012014	001246	MOV	#GGP6,\$TMP6
	011362	104222		1\$:	ERROR	+222
	011364	000137	012054		JMP	GGDONE
2201	011370			GGER15:		
2202	011370	010537	001252		MOV	R5,\$TMP10
	011374	010437	001254		MOV	R4,\$TMP11
	011400	012737	011754	001240	MOV	#GGP2,\$TMP3
	011406	012737	012034	001242	MOV	#GGP8,\$TMP4
	011414	012737	011734	001244	MOV	#GGDAT0,\$TMP5
	011422	012737	012014	001246	MOV	#GGP6,\$TMP6
	011430	104207		1\$:	ERROR	+207
	011432	000137	012054		JMP	GGDONE
2203	011436			GGER16:		
2204	011436	010537	001252		MOV	R5,\$TMP10
	011442	010437	001254		MOV	R4,\$TMP11
	011446	012737	011754	001240	MOV	#GGP2,\$TMP3

	011454	012737	012034	001242	MOV	#GGP8,\$TMP4
	011462	012737	011734	001244	MOV	#GGDATO,\$TMP5
	011470	012737	012014	001246	MOV	#GGP6,\$TMP6
	011476	104232			1\$: ERROR	+232
2205	011500	000137	012054		JMP	GGDONE
2206	011504				GGER17:	
	011504	010537	001252		MOV	R5,\$TMP10
	011510	010437	001254		MOV	R4,\$TMP11
	011514	012737	011754	001240	MOV	#GGP2,\$TMP3
	011522	012737	012034	001242	MOV	#GGP8,\$TMP4
	011530	012737	011734	001244	MOV	#GGDATO,\$TMP5
	011536	012737	012044	001246	MOV	#GGP9,\$TMP6
	011544	104223			1\$: ERROR	+223
	011546	000137	012054		JMP	GGDONE
2207	011552				GGER18:	
2208	011552	010537	001252		MOV	R5,\$TMP10
	011556	010437	001254		MOV	R4,\$TMP11
	011562	012737	011754	001240	MOV	#GGP2,\$TMP3
	011570	012737	012034	001242	MOV	#GGP8,\$TMP4
	011576	012737	011734	001244	MOV	#GGDATO,\$TMP5
	011604	012737	012044	001246	MOV	#GGP9,\$TMP6
	011612	104207			1\$: ERROR	+207
	011614	000137	012054		JMP	GGDONE
2209	011620				GGER19:	
2210	011620	010537	001252		MOV	R5,\$TMP10
	011624	010437	001254		MOV	R4,\$TMP11
	011630	012737	011754	001240	MOV	#GGP2,\$TMP3
	011636	012737	012034	001242	MOV	#GGP8,\$TMP4
	011644	012737	011734	001244	MOV	#GGDATO,\$TMP5
	011652	012737	012044	001246	MOV	#GGP9,\$TMP6
	011660	104230			1\$: ERROR	+230
	011662	000137	012054		JMP	GGDONE
2211	011666				GGER20:	
2212	011666	010537	001252		MOV	R5,\$TMP10
	011672	010437	001254		MOV	R4,\$TMP11
	011676	012737	011754	001240	MOV	#GGP2,\$TMP3
	011704	012737	012034	001242	MOV	#GGP8,\$TMP4
	011712	012737	011734	001244	MOV	#GGDATO,\$TMP5
	011720	012737	012044	001246	MOV	#GGP9,\$TMP6
	011726	104231			1\$: ERROR	+231
	011730	000137	012054		JMP	GGDONE
2213	011734	000000			GGDATO:	0
2214	011736	000000				0
2215	011740	000000				0
2216	011742	000000				0
2217	011744	000300			GGP1:	300
2218	011746	000000				0
2219	011750	000000				0
2220	011752	000000				0
2221	011754	100200			GGP2:	100200
2222	011756	000000				0

2225 011760 000000  
2226 011762 000000  
2227 011764 000200  
2228 011766 000000  
2229 011770 000000  
2230 011772 000001  
2231 011774 010200  
2232 011776 000000  
2233 012000 000000  
2234 012002 000000  
2235 012004 077600  
2236 012006 000000  
2237 012010 000000  
2238 012012 000000  
2239 012014 000000  
2240 012016 000000  
2241 012020 000000  
2242 012022 000000  
2243  
2244 012024 062400  
2245 012026 000000  
2246 012030 000000  
2247 012032 000000  
2248 012034 000340  
2249 012036 000000  
2250 012040 000000  
2251 012042 000000  
2252 012044 000100  
2253 012046 000000  
2254 012050 000000  
2255 012052 000000  
2256 012054 104412

GGP3: 200  
GGP4: 10200  
GGP5: 77600  
GGP6: 0  
GGP7: 62400  
GGP8: 340  
GGP9: 100  
GGDONE: RSETUP

:OVER FLOW = GGP5 + GGP5

:OVERFLOW RESULT  
:UNDERFLOW RESULT  
:GGP6 = GGP4 + GGP5  
: = GGP3 + GGP2 (FIU = 0)  
: = GGP3 + GGP1  
:GGP7 = GGP3 + GGP2 (FIU = 1)

: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?).

2257  
2263



2264

```
.SBTTL TEST # 3 - LDCFD AND LDCDF TEST
*****
*TEST 3          LDCFD AND LDCDF TEST
*
*THIS IS A TEST OF LDCFD AND LDCDF.
*
```

```

012056 000004
2265 012056 000004
2266 012060 104413
012060 104413
2267 012062 012704 000200
2268 012066 170104
2269 012070 012700 013474
2270 012074 172410
2271 012076 012700 013504
2272 012102 012737 012110 001236
2273 012110 177420
2274 012112 020027 013510
2275 012116 001402
2276 012120 000137 013052
2277 012124
2278 012124 170205
2279 012126 012700 013464
2280 012132 174010
2281 012134 012701 013554
2282 012140 012702 000004
2283 012144 022120
2284 012146 001415
2285 012150 012701 013504
2286 012154 012700 013464
2287 012160 012702 000004
2288 012164 022120
2289 012166 001402
2290 012170 000137 013112
2291 012174 077205
2292 012176 000137 013142
2293 012202 077220
2294 012204 012704 000200
2295 012210 020405
2296 012212 001402
2297 012214 000137 013210
2298
2299 012220
012220 104413
2300 012222 012704 000200
2301 012226 170104
2302
2303 012230 012700 013474
2304 012234 172410
2305
2306 012236 012700 013504
2307 012242 012737 012252 001236
2308
2309 012250 170001
2310
2311 012252 177420
```

```

TST3: SCOPE
;TEST FOR CORRECT AUTO INCREMENT CONSTANT.
HX1: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
      MOV #200,R4
      LDFPS R4
      MOV #HXP1,R0
      LDD (R0),ACO
      MOV #HXP2,R0
      MOV #HX2,$TMP2
      LDCFD (R0)+,ACO
      CMP R0,#HXP2+4 ;IS R0 CORRECT
      BEQ HX3
      JMP HXER1
HX2:
HX3: STFPS R5 ;GET FPS
      MOV #HXDATO,R0
      STD ACO,(R0) ;GET ACO
      MOV #HXP7,R1 ;SEE IF RESULT IS
      MOV #4,R2 ;CORRECT
      CMP (R1)+,(R0)+
      BEQ HX7
      MOV #HXP2,R1 ;DID FD GET
      MOV #HXDATO,R0 ;COMPLIMENTED?
      MOV #4,R2
      CMP (R1)+,(R0)+
      BEQ HX6
      JMP HXER2
HX4: SOB R2,HX5
      JMP HXER3
HX5: SOB R2,HX4
      MOV #200,R4 ;FPS CORRECT?
      CMP R4,R5
      BEQ HX8
      JMP HXER8
HX6:
HX7:
HX8: ;NOW
      TEST LDCDF
      LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
      MOV #200,R4
      LDFPS R4
      MOV #HXP1,R0
      LDD (R0),ACO
      MOV #HXP2,R0
      MOV #HX9,$TMP2
      SETF
HX9: LDCFD (R0)+,ACO ;TEST INSTRUCTION
```

2312										
2313	012254	020027	013514			CMP	R0,#HXP2+10		:WAS A GOOD	
2314	012260	001402				BEQ	HX10		:CONSTANT USED	
2315	012262	000137	013072			JMP	HXER5		:TO INCREMENT R0?	
2316										
2317	012266					HX10:	STFPS	R5		
2318	012266	170205					MOV	#HXDATO,R0		
2319	012270	012700	013464				SETD			
2320	012274	170011					STD	AC0,(R0)		:GET RESULT
2321	012276	174010					MOV	#HXP8,R1		
2322	012300	012701	013564				MOV	#4,R2		
2323	012304	012702	000004			HX11:	CMP	(R1)+,(R0)+		:IS IT CORRECT?
2324	012310	022120					BEQ	HX14		
2325	012312	001415								
2326										
2327	012314	012701	013554				MOV	#HXP7,R1		
2328	012320	012700	013464				MOV	#HXDATO,R0		
2329	012324	012702	000004				MOV	#4,R2		
2330	012330	022110				HX12:	CMP	(R1)+,(R0)		:DID FD FAIL TO GET
2331	012332	001402					BEQ	HX13		:COMPLIMENTED?
2332	012334	000137	013226				JMP	HXER6		
2333	012340	077205				HX13:	SOB	R2,HX12		
2334	012342	000137	013256				JMP	HXER7		
2335										
2336	012346	077220				HX14:	SOB	R2,HX11		
2337										
2338	012350	012704	000000				MOV	#0,R4		:FPS CORRECT?
2339	012354	020405					CMP	R4,R5		
2340	012356	001402					BEQ	HX15		
2341	012360	000137	013210				JMP	HXER8		
2342										
2343										
2344										
2345	012364									
2346	012364	104413				HX15:	LPERR			:SET UP THE LOOP ON ERROR ADDRESS.
2347	012366	012704	000200				MOV	#200,R4		
2348	012372	170104					LDFPS	R4		:SET FD
2349	012374	012737	012412	001236			MOV	#HX16,\$TMP2		
2350	012402	012737	013306	000004			MOV	#HXER9,ERRVECT		
2351	012410	005001					CLR	R1		
2352	012412	177427	043243			HX16:	LDCFD	#5201,AC0		
2353	012416	005201				HX165:	INC	R1		
2354	012420	005201					INC	R1		
2355	012422	005201					INC	R1		
2356	012424	012737	037502	000004			MOV	#CPSPUR,ERRVECT		
2357	012432	020127	000003				CMP	R1,#3		:SEE IF PC WAS
2358	012436	001402					BEQ	HX17		:CORRECT
2359	012440	000137	013342				JMP	HXER10		
2360										
2361	012444					HX17:	LPERR			:SET UP THE LOOP ON ERROR ADDRESS.
2362	012444	104413								
2363	012446	012704	000200				MOV	#200,R4		
2364	012452	170104					LDFPS	R4		
2365	012454	012737	012502	001236			MOV	#HX18,\$TMP2		
2366	012462	012700	013544				MOV	#HXP6,R0		

```

2367 012466 172410          LDD      (R0),ACO
2368 012470 012737 037502 000004    MOV      #CPSPUR,ERRVECT
2369 012476 012700 013504          MOV      #HXP2,R0
2370 012502 177410          HX18:   LDCFD   (R0),ACO
2371
2372 012504 012700 013464          MOV      #HXDATO,R0
2373 012510 174010          STD      ACO,(R0)          ;GET RESULT.
2374 012512 012701 013554          MOV      #HXP7,R1
2375 012516 012702 000004          MOV      #4,R2
2376 012522 022021          HX19:   CMP      (R0)+,(R1)+  ;IS RESULT CORRECT?
2377 012524 001402          BEQ      HX20
2378 012526 000137 013112          JMP      HXER2
2379 012532 077205          HX20:   SOB      R2,HX19
2380
2381          ;TEST LDCFD WITH NEGATIVE OPERAND
2382          HX21:
2383 012534 104413          LPERR                    ;SET UP THE LOOP ON ERROR ADDRESS.
2384 012536 012704 000200          MOV      #200,R4
2385 012542 170104          LDFPS   R4
2386 012544 012737 012564 001236    MOV      #HX22,$TMP2
2387 012552 012700 013544          MOV      #HXP6,R0
2388 012560 012700 013524          LDD      (R0),ACO
2389 012564 177410          HX22:   MOV      #HXP4,R0
2390          LDCFD   (R0),ACO
2391 012566 012700 013464          MOV      #HXDATO,R0
2392 012572 174010          STD      ACO,(R0)          ;GET RESULT
2393
2394 012574 012701 013534          MOV      #HXP5,R1
2395 012600 012702 000004          MOV      #4,R2
2396 012604 022120          HX23:   CMP      (R1)+,(R0)+
2397 012606 001415          BEQ      HX26
2398
2399 012610 012701 013554          MOV      #HXP7,R1
2400 012614 012700 013464          MOV      #HXDATO,R0
2401 012620 012702 000004          MOV      #4,R2
2402 012624 022120          HX24:   CMP      (R1)+,(R0)+  ;WAS SIGN INCORRECT
2403 012626 001402          BEQ      HX25
2404 012630 000137 013374          JMP      HXER11
2405 012634 077205          HX25:   SOB      R2,HX24
2406 012636 000137 013414          JMP      HXER12
2407
2408 012642 077220          HX26:   SOB      R2,HX23
2409
2410          ;TEST LDCFD 0
2411          HX27:
2412 012644 104413          LPERR                    ;SET UP THE LOOP ON ERROR ADDRESS.
2413 012646 012704 000200          MOV      #200,R4
2414 012652 170104          LDFPS   R4
2415
2416 012654 012700 013474          MOV      #HXP1,R0
2417 012660 172410          LDD      (R0),ACO
2418 012662 172010          ADDD    (R0),ACO
2419
2420 012664 012737 012676 001236    MOV      #HX28,$TMP2
2421 012672 012700 013474          MOV      #HXP1,R0
  
```



TEST # 3 - LDCFD AND LDCDF TEST

```

2422 012676 177410          HX28:  LDCFD  (R0),ACO
2423
2424 012700 170205          STFPS  R5
2425
2426 012702 012700 013464          MOV    #HXDATO,R0
2427 012706 174010          STD    ACO,(R0)          ;GET RESULT
2428
2429 012710 012701 013474          MOV    #HXP1,R1
2430 012714 012702 000004          MOV    #4,R2
2431 012720 022120          HX29:  CMP    (R1)+,(R0)+    ;IS IT 0?
2432 012722 001402          BEQ    HX30
2433 012724 000137 013444          JMP    HXER13
2434 012730 077205          HX30:  SOB    R2,HX29
2435
2436 012732 012704 000204          MOV    #204,R4          ;FPS CORRECT
2437 012736 020405          CMP    R4,R5
2438 012740 001402          BEQ    HX31
2439 012742 000137 013172          JMP    HXER4
2440
2441          ;TEST  LDCFD  0
2442
2443          HX31:
2443 012746 104413          LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
2444 012750 012704 000200          MOV    #200,R4
2445 012754 170104          LDFPS  R4
2446
2447 012756 012700 013544          MOV    #HXP6,R0
2448 012762 172410          LDD    (R0),ACO
2449
2450 012764 012737 012776 001236          MOV    #HX32,$TMP2
2451 012772 012700 013474          MOV    #HXP1,R0
2452 012776 177410          HX32:  LDCFD  (R0),ACO
2453
2454 013000 170205          STFPS  R5
2455
2456 013002 012700 013464          MOV    #HXDATO,R0
2457 013006 174010          STD    ACO,(R0)          ;GET RESULT
2458
2459 013010 012701 013474          MOV    #HXP1,R1
2460 013014 012702 000004          MOV    #4,R2
2461 013020 022120          HX33:  CMP    (R1)+,(R0)+    ;IS IT ZERO?
2462 013022 001402          BEQ    HX34
2463 013024 000137 013444          JMP    HXER13
2464 013030 077205          HX34:  SOB    R2,HX33
2465
2466 013032 012704 000204          MOV    #204,R4          ;FPS CORRECT?
2467 013036 020405          CMP    R4,R5
2468 013040 001402          BEQ    HX35
2469 013042 000137 013172          JMP    HXER4
2470 013046 000137 013574          HX35:  JMP    HXDONE
2471
2472          ;RO INCORRECT
2473
2474 013052 012737 013510 001242          HXER1:  MOV    #HXP2+4,$TMP4
2475 013060 010037 001240          MOV    R0,$TMP3
2476 013064 104234          1$:    ERROR +234
2477 013066 000137 013574          JMP    HXDONE
    
```

```

2478
2479 013072 012737 013514 001242 HXER5: MOV #HXP2+10,$TMP4
2480 013100 010037 001240          MOV R0,$TMP3
2481 013104 104237          1$: ERROR +237
2482 013106 000137 013574          JMP HXDONE
2483          ;REPORT BAD DATA
2484 013112 012737 013504 001244 HXER2: MOV #HXP2,$TMP5
2485 013120 012737 013554 001250          MOV #HXP7,$TMP7
2486 013126 012737 013464 001246 HXER22: MOV #HXDAT0,$TMP6
2487 013134 104233          1$: ERROR +233
2488 013136 000137 013574          JMP HXDONE
2489          ;
2490 013142 012737 013504 001244 HXER3: MOV #HXP2,$TMP5
2491 013150 012737 013554 001250          MOV #HXP7,$TMP7
2492 013156 012737 013464 001246 HXER33: MOV #HXDAT0,$TMP6
2493 013164 104241          1$: ERROR +241
2494 013166 000137 013574          JMP HXDONE
2495
2496 013172 010537 001240          HXER4: MOV R5,$TMP3
2497 013176 010437 001242          MOV R4,$TMP4
2498 013202 104240          1$: ERROR +240
2499 013204 000137 013574          JMP HXDONE
2500
2501 013210 010537 001240          HXER8: MOV R5,$TMP3
2502 013214 010437 001242          MOV R4,$TMP4
2503 013220 104242          1$: ERROR +242
2504 013222 000137 013574          JMP HXDONE
2505 013226 012737 013504 001244 HXER6: MOV #HXP2,$TMP5
2506 013234 012737 013564 001250          MOV #HXP8,$TMP7
2507 013242 012737 013464 001246 HXER66: MOV #HXDAT0,$TMP6
2508 013250 104244          1$: ERROR +244
2509 013252 000137 013574          JMP HXDONE
2510
2511 013256 012737 013504 001244 HXER7: MOV #HXP2,$TMP5
2512 013264 012737 013564 001250          MOV #HXP8,$TMP7
2513 013272 012737 013464 001246          MOV #HXDAT0,$TMP6
2514 013300 104243          1$: ERROR +243
2515 013302 000137 013574          JMP HXDONE
2516
2517 013306 032716 000001          HXER9: BIT #1,(SP)          ;SEE IF IT
2518 013312 001005          BNE 1$          ;AN ODD ADDRESS
2519 013314 022716 012416          CMP #HX165,(SP)
2520 013320 001402          BEQ 1$
2521 013322 000137 037502          JMP CPSPUR
2522
2523 013326 011637 001236          1$: MOV (SP),$TMP2
2524 013332 022626          CMP (SP)+,(SP)+
2525 013334 104235          2$: ERROR +235
2526 013336 000137 013574          JMP HXDONE
2527
2528 013342 162701 000003          HXER10: SUB #3,R1
2529 013346 006301          ASL R1
2530 013350 012702 012416          MOV #HX165,R2
2531 013354 010237 001242          MOV R2,$TMP4
2532 013360 160102          SUB R1,R2
2533 013362 010237 001240          MOV R2,$TMP3
2534 013366 104236          1$: ERROR +236
    
```

```

2535 013370 000137 013574          JMP      HXDONE
2536
2537 013374 012737 013524 001244 HXER11: MOV    #HXP4,$TMP5
2538 013402 012737 013534 001250      MOV    #HXP5,$TMP7
2539 013410 000137 013126          JMP      HXER22
2540 013414 012737 013524 001244 HXER12: MOV    #HXP4,$TMP5
2541 013422 012737 013534 001250      MOV    #HXP5,$TMP7
2542 013430 012737 013464 001246      MOV    #HXDAT0,$TMP6
2543 013436 104245          1$:     ERROR  +245
2544 013440 000137 013574          JMP      HXDONE
2545
2546 013444 012737 013474 001244 HXER13: MOV    #HXP1,$TMP5
2547 013452 012737 013474 001250      MOV    #HXP1,$TMP7
2548 013460 000137 013126          JMP      HXER22
2549
2550 013464 000000          HXDAT0: 0
2551 013466 000000              0
2552 013470 000000              0
2553 013472 000000              0
2554
2555 013474 000000          HXP1:   0
2556 013476 000000              0
2557 013500 000000              0
2558 013502 000000              0
2559
2560 013504 000577          HXP2:   577
2561 013506 177776              177776
2562 013510 177777              177777
2563 013512 177776              177776
2564 013514 005201          HXP3:   5201
2565 013516 000000              0
2566 013520 000000              0
2567 013522 000000              0
2568 013524 100577          HXP4:  100577
2569 013526 177776              177776
2570 013530 177777              177777
2571 013532 177776              177776
2572 013534 100577          HXP5:  100577
2573 013536 177776              177776
2574 013540 000000              0
2575 013542 000000              0
2576 013544 000252          HXP6:  252
2577 013546 125252              125252
2578 013550 125252              125252
2579 013552 125252              125252
2580
2581 013554 000577          HXP7:   577
2582 013556 177776              177776
2583 013560 000000              0
2584 013562 000000              0
2585 013564 000577          HXP8:   577
2586 013566 177777              177777
2587 013570 000000              0
2588 013572 000000              0
2589
2590 013574          HXDONE:
      013574 104412      RSETUP
    
```

;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?).

2591  
2592  
2593  
2601

2602

```
.SBTTL TEST # 4 - CMPD TEST  
:*****  
:*TEST 4      CMPD TEST  
:*  
:*THIS IS A TEST OF THE CMPD INSTRUCTION. NOTE THAT A SUBROUTINE  
:*IS USED TO SET UP OPERANDS, EXECUTE THE INSTRUCTION AND CHECK THE  
:*RESULTS  
:*  
:*****
```

```
013576 000004  
2603  
2604  
2605 013600  
013600 104413  
2606 013602 004737 014412  
2607 013606 000000 000000 000000 1$:  
2608 013616 000000 000000 000000 2$:  
2609 013626 000200 3$:  
2610 013630 000204  
2611 013632 000200  
2612 013634 104001 4$:
```

```
TST4:  SCOPE  
:TEST THE CMPD INSTRUCTION WITH (FSRC=AC=0)  
AAA1:  
LPERR ;SET UP THE LOOP ON ERROR ADDRESS.  
JSR PC,CMPD SUB  
.WORD 0,0,0,0 ;AC0  
.WORD 0,0,0,0 ;FSRC  
200 ;FPS BEFORE EXECUTION  
204 ;FPS AFTER EXECUTION  
200 ;ERROR FPS  
ERROR +1 ;FPS ERROR
```

```
2613  
2614  
2615  
2616 013636  
013636 104413  
2617 013640 004737 014412  
2618 013644 000000 000000 000000 1$:  
2619 013654 025252 2$:  
2620 013656 052525  
2621 013660 125252  
2622 013662 052525  
2623 013664 000200 3$:  
2624 013666 000200  
2625 013670 000210  
2626 013672 104003 4$:
```

```
:TEST CMPD WITH (AC=0) AND FSRC POSITIVE.  
AAA2:  
LPERR ;SET UP THE LOOP ON ERROR ADDRESS.  
JSR PC,CMPD SUB  
.WORD 0,0,0,0 ;AC  
25252 ;FSRC  
52525  
125252  
52525  
200 ;FPS BEFORE EXECUTION  
200 ;FPS AFTER EXECUTION  
210 ;ERROR FPS  
ERROR +3 ;FPS ERROR
```

```
2627  
2628  
2629 013674  
013674 104413  
2630 013676 004737 014412  
2631 013702 000000 000000 000000 1$:  
2632 013712 125252 2$:  
2633 013714 125252  
2634 013716 052525  
2635 013720 125252  
2636 013722 000200 3$:  
2637 013724 000210  
2638 013726 000200  
2639 013730 104004 4$:
```

```
:TEST CMPD WITH (AC=0) AND FSRC NEGATIVE  
AAA3:  
LPERR ;SET UP THE LOOP ON ERROR ADDRESS.  
JSR PC,CMPD SUB  
.WORD 0,0,0,0 ;AC  
125252 ;FSRC  
125252  
52525  
125252  
200 ;FPS BEFORE EXECUTION  
210 ;FPS AFTER EXECUTION  
200 ;ERROR FPS  
ERROR +4 ;FPS ERROR.
```

```
2640  
2641  
2642 013732  
013732 104413  
2643 013734 004737 014412  
2644 013740 025252  
2645 013742 052525
```

```
:TEST CMPD WITH (FSRC=0) AND AC POSITIVE.  
AAA4:  
LPERR ;SET UP THE LOOP ON ERROR ADDRESS.  
JSR PC,CMPD SUB  
25252 ;AC  
52525 ;FSRC
```

TEST # 4 - CMPD TEST

```

2646 013744 125252 125252
2647 013746 052525 52525
2648 013750 000000 000000 000000 2$: .WORD 0,0,0,0 ;FSRC
2649 013760 000200 3$: 200 ;FPS BEFORE EXECUTION
2650 013762 000210 210 ;FPS AFTER EXECUTION
2651 013764 000200 200 ;ERROR FPS
2652 013766 104005 4$: ERROR +5 ;FPS ERROR
2653
2654
2655 ;TEST CMPD WITH (FSRC=0) AND AC NEGATIVE
2656 013770 AAA5: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
013770 104413 JSR PC,CMPSUB
2657 013772 004737 014412 1$: 125252 ;AC
013776 125252
2658 013776 125252
2659 014000 125252
2660 014002 052525
2661 014004 125252
2662 014006 000000 000000 2$: .WORD 0,0,0,0 ;FSRC
2663 014016 000200 3$: 200 ;FPS BEFORE EXECUTION
2664 014020 000200 200 ;FPS AFTER EXECUTION
2665 014022 000210 210 ;ERROR FPS
2666 014024 104006 4$: ERROR +6 ;FPS ERROR
2667
2668 ;TEST CMPD WITH AC POSITIVE AND FSRC NEGATIVE
2669 014026 AAA6: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
014026 104413 JSR PC,CMPSUB
2670 014030 004737 014412 1$: 52525 ;AC
014034 052525
2671 014034 052525
2672 014036 125252
2673 014040 052525
2674 014042 125252
2675 014044 125252 2$: 125252 ;:FSRC
2676 014046 052525
2677 014050 125252
2678 014052 052525
2679 014054 000200 3$: 200 ;FPS BEFORE EXECUTION
2680 014056 000210 210 ;FPS AFTER EXECUTION
2681 014060 000200 200 ;ERROR FPS
2682 014062 104007 4$: ERROR +7 ;FPS ERROR
2683
2684
2685 ;TEST CMPD WITH AC NEGATIVE AND FSRC POSITIVE
2686 014064 AAA7: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
014064 104413 JSR PC,CMPSUB
2687 014066 004737 014412 1$: 125252 ;AC
014072 125252
2688 014072 125252
2689 014074 052525
2690 014076 125252
2691 014100 052525
2692 014102 052525 2$: 52525 ;:FSRC
2693 014104 125252
2694 014106 052525
2695 014110 125252
2696 014112 000200 3$: 200 ;FPS BEFORE EXECUTION
2697 014114 000200 200 ;FPS AFTER EXECUTION
2698 014116 000210 210 ;ERROR FPS
2699 014120 104010 4$: ERROR +10 ;FPS ERROR.

```



2700

2701

2702

2703 014122

014122 104413

2704 014124 004737 014412

2705 014130 012345

2706 014132 067654

2707 014134 032101

2708 014136 023456

2709 014140 023456

2710 014142 076543

2711 014144 021012

2712 014146 034567

2713 014150 000200

2714 014152 000200

2715 014154 000210

2716 014156 104011

2717

2718

2719

2720 014160

014160 104413

2721 014162 004737 014412

2722 014166 045676

2723 014170 054321

2724 014172 012345

2725 014174 067654

2726 014176 034567

2727 014200 065432

2728 014202 101234

2729 014204 056765

2730 014206 000200

2731 014210 000210

2732 014212 000200

2733 014214 104012

2734

2735

2736 014216

014216 104413

2737 014220 004737 014412

2738 014224 012345

2739 014226 067012

2740 014230 034567

2741 014232 012345

2742 014234 012345

2743 014236 067012

2744 014240 034567

2745 014242 012345

2746 014244 000200

;TEST CMPD WITH AC POSITIVE AND FSRC POSITIVE  
;AND EAC LESS THAN EFSRC.

AAA8:

LPERR

;SET UP THE LOOP ON ERROR ADDRESS.

JSR PC,CMPSUB

1\$:

12345

;AC

67654

32101

23456

2\$:

23456

;FSRC

76543

21012

34567

3\$:

200

;FPS BEFORE EXECUTION

200

;FPS AFTER EXECUTION

210

;ERROR FPS

4\$:

ERROR +11

;FPS ERROR

;TEST CMPD WITH AC POSITIVE, FSRC POSITIVE AND EAC GREATER THAN EFSRC  
AAA9:

LPERR

;SET UP THE LOOP ON ERROR ADDRESS.

JSR PC,CMPSUB

1\$:

45676

;AC

54321

12345

67654

2\$:

34567

;FSRC

65432

101234

56765

3\$:

200

;FPS BEFORE EXECUTION

210

;FPS AFTER EXECUTION

200

;ERROR FPS

4\$:

ERROR +12

;TEST CMPD WITH AC POSITIVE, FSRC POSITIVE AND AC EQUAL TO FSRC  
AAA10:

LPERR

;SET UP THE LOOP ON ERROR ADDRESS.

JSR PC,CMPSUB

1\$:

12345

;AC

67012

34567

012345

2\$:

12345

;FSRC

67012

34567

012345

3\$:

200

;FPS BEFORE EXECUTION

```

2748 014246 000204          204          :FPS AFTER EXECUTION
2749 014250 000200          200          :ERROR FPS
2750 014252 104013          4$: ERROR +13      :FPS ERROR
2751
2752          :TEST CMPD WITH AC POSITIVE, FSRC POSITIVE, EAC EQUAL TO EFSRC,
2753          :AND FSRC GREATER THAN AC.
2754 014254          AAA11:
      014254 104413          LPERR          :SET UP THE LOOP ON ERROR ADDRESS.
2755 014256 004737 014412          JSR          PC,CMP SUB
2756 014262 012345          1$: 12345          :AC
      014264 067012          67012
2757 014264 067012          67012
2758 014266 034567          34567
2759 014270 012345          012345
2760 014272 012345          2$: 12345          :FSRC
2761 014274 070123          70123
2762 014276 045670          45670
2763 014300 123456          123456
2764 014302 000200          3$: 200          :FPS BEFORE EXECUTION
2765 014304 000200          200          :FPS AFTER EXECUTION
2766 014306 000210          210          :ERROR FPS
2767 014310 104014          4$: ERROR +14      :FPS ERROR
2768
2769          :TEST CMPD WITH AC POSITIVE, FSRC POSITIVE, EAC EQUAL TO EFSRC,
2770          :AND AC GREATER THAN FSRC.
2771 014312          AAA12:
      014312 104413          LPERR          :SET UP THE LOOP ON ERROR ADDRESS.
2772 014314 004737 014412          JSR          PC,CMP SUB
2773 014320 054321          1$: 54321          :AC
      014322 076543          76543
2774 014322 076543          76543
2775 014324 021076          21076
2776 014326 054321          54321
2777 014330 054321          2$: 54321          :FSRC
2778 014332 065432          65432
2779 014334 107654          107654
2780 014336 032107          32107
2781 014340 000200          3$: 200          :FPS BEFORE EXECUTION
2782 014342 000210          210          :FPS AFTER EXECUTION
2783 014344 000200          200          :ERROR FPS
2784 014346 104015          4$: ERROR +15      :FPS ERROR
2785
2786          :TEST CMPD WITH AC NEGATIVE, FSRC NEGATIVE, EAC EQUAL TO EFSRC,
2787          :AND AC GREATER THAN FSRC
2788 014350          AAA13:
      014350 104413          LPERR          :SET UP THE LOOP ON ERROR ADDRESS.
2789 014352 004737 014412          JSR          PC,CMP SUB
2790 014356 112345          1$: 112345         :AC
      014360 043210          43210
2791 014360 043210          43210
2792 014362 076543          76543
2793 014364 021076          21076
2794 014366 112345          2$: 112345         :FSRC
2795 014370 054321          54321
2796 014372 007654          07654
2797 014374 032107          32107
2798 014376 000200          3$: 200          :FPS BEFORE EXECUTION
2799 014400 000210          210          :FPS AFTER EXECUTION
2800 014402 000200          200          :ERROR FPS
2801 014404 104016          4$: ERROR +16      :FPS ERROR
  
```

2802  
2803  
2804 014406 000137 014602

JMP AAADONE ;FINISHED CMPD TEST.

2805  
2806  
2807  
2808  
2809  
2810  
2811  
2812  
2813  
2814  
2815  
2816  
2817  
2818  
2819  
2820  
2821  
2822  
2823  
2824  
2825  
2826  
2827  
2828  
2829

: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.

2830 014412 012601  
2831  
2832 014414 016100 000020  
2833 014420 170100  
2834  
2835 014422 012737 014444 001236  
2836 014430 010100  
2837 014432 172410  
2838  
2839 014434 010100  
2840 014436 062700 000010  
2841  
2842 014442 000240  
2843 014444 173410  
2844  
2845 014446 170205  
2846  
2847 014450 016104 000022  
2848  
2849 014454 010137 001240  
2850 014460 010137 001242  
2851 014464 062737 000010 001242  
2852 014472 010537 001244  
2853 014476 010437 001246  
2854 014502 020405  
2855 014504 001410  
2856  
2857  
2858 014506 026105 000024

```

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      #1$, $TMP2     :SAVE ADDRESS OF CMPD INSTRUCTION.
      MOV      R1,R0          :GET ADDRESS OF AC OPERAND.
      LDD      (R0),ACO       :LOAD ACO OPERAND
      MOV      R1,R0          :COMPUTE FSRC OPERAND
      ADD      #10,R0         :ADDRESS
      NOP
1$:    CMPD     (R0),ACO       :FOR SCOPING.
      :EXECUTE THE TEST INSTRUCTION.
      STFPS   R5              :SAVE FPS AFTER INSTRUCTION.
      MOV      22(R1),R4      :GET EXPECTED FPS.
      :IF INCORRECT SET UP FOR
      :AN ERROR CALL.
      MOV      R1,$TMP3
      MOV      R1,$TMP4
      ADD      #10,$TMP4
      MOV      R5,$TMP5
      MOV      R4,$TMP6
      CMP      R4,R5          :WAS FPS CORRECT?
      BEQ     3$              :BRANCH IF YES.
      CMP      24(R1),R5      :WAS THE FPS THE SAME
    
```



```
2859                                     ;AS THE EXPECTED INCORRECT FPS?  
2860 014512 001003                     BNE      2$      ;BRANCH IF NO MATCH.  
2861                                     ;  
2862 014514 062701 000026              ADD      #26,R1  ;IF THE EXPECTED INCORRECT  
2863                                     ;FPS MATCHED THE RESULTANT FPS  
2864 014520 000111                     JMP      (R1)    ;RETURN TO THE ERROR CALL  
2865                                     ;IN THE CALLING ROUTINE.  
2866                                     ;  
2867 014522 104001 2$: ERROR +1        ;OTHERWISE REPORT INCORRECT FPS  
2868 014524 000411                     BR       5$  
2869                                     ;  
2870 014526 012700 014572              3$: MOV    #CMPTMP,R0  ;IF FPS WAS CORRECT MAKE SURE  
2871 014532 174010                      STD     ACO,(R0)    ;ACO WAS NOT AFFECTED BY CMPD.  
2872 014534 010102                      MOV     R1,R2  
2873 014536 012703 000004              MOV     #4,R3  
2874 014542 022220 4$: CMP     (R2)+,(R0)+  
2875 014544 001003                     BNE     6$  
2876 014546 077303                     SOB     R3,4$  
2877                                     ;  
2878 014550 000161 000030              5$: JMP    30(R1)   ;RETURN  
2879                                     ;  
2880 014554 010137 001240              6$: MOV    R1,$TMP3  ;REPORT ACO MODIFIED BY CMPD  
2881 014554 012737 014572 001242      MOV     #CMPTMP,$TMP4  
2882 014560 104002                      ERROR  +2  
2883 014566 000767                      BR      5$  
2884 014570 000767                      ;RETURN  
2885                                     ;  
2886 014572 000000 000000 000000      CMPTMP: .WORD 0,0,0,0  
2887                                     ;  
2888                                     ;  
2889                                     ;  
2890 014602 104412                     AAADONE: RSETUP  ;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?).  
  
2891  
2892  
2893  
2901
```

2902

```
.SBTTL TEST # 5 - DIVD WITH (FSRC=0) AND (BUT FD) TEST
:*****
:*TEST 5 DIVD WITH (FSRC=0) AND (BUT FD) TEST
:*
:*THIS IS A TEST OF THE DIVD INSTRUCTION WITH A
:*ZERO DIVISOR. THE CONDITION IS CHECKED WITH BOTH
:*TRAP ENABLED AND TRAPS DISABLED.
:*
:*****
```

2903 014604 000004

TST5: SCOPE

2904

:FIRST TEST DIVD WITH (FSRC=AC=0) AND TRAPS DISABLED.

2905 014606 104413

BBB0: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.

2906 014610 012704 040200

MOV #40200,R4 ;SET UP FPS

2907

WITH INTERRUPTS

2908

;DISABLED.

2909 014614 170104

LDFPS R4 ;SET UP FOR ANY FP INTERRUPTS.

2910 014616 012737 015060 000244

MOV #BBBER1,FPVECT

2911 014624 012737 014644 001236

MOV #BBB1,\$TMP2

2912 014632 012700 015264

MOV #BBBP1,R0 ;SET UP ACO = 0

2913 014636 172410

LDD (R0),ACO ;FSRC = 0

2914 014640 012701 015264

MOV #BBBP1,R1 ;TEST INSTRUCTION

2915

2916 014644 174411

BBB1: DIVD (R1),ACO ;GET FPS

2917

STFPS R5 ;GET FEC

2918 014646 170205

STST R3

2919 014650 170303

2920

2921 014652 012704 140204

MOV #140204,R4 ;EXPECTED FPS.

2922 014656 020405

CMP R4,R5 ;IS FPS CORRECT.

2923 014660 001131

BNE BBBER2 ;IF INCORRECT BRANCH.

2924

2925 014662 012702 000004

MOV #4,R2 ;EXPECTED FEC.

2926 014666 020203

CMP R2,R3 ;IS FEC CORRECT?

2927 014670 001140

BNE BBBER3 ;IF INCORRECT BRANCH.

2928

:TEST DIVD WITH (FSRC=0) AND TRAPS DISABLED.

2929

BBB2:

2930 014672 104413

LPERR ;SET UP THE LOOP ON ERROR ADDRESS.

2931 014674 012704 040200

MOV #40200,R4 ;LOAD FPS WITH TRAPS DISABLED.

2932 014700 170104

LDFPS R4

2933

2934 014702 012737 014722 001236

MOV #BBB3,\$TMP2 ;SET UP ACO OPERAND (NON ZERO).

2935 014710 012700 015274

MOV #BBBP2,R0

2936 014714 172410

LDD (R0),ACO ;FSRC=0

2937 014716 012700 015264

MOV #BBBP1,R0

2938 014722 174410

BBB3: DIVD (R0),ACO ;GET FPS.

2939

STFPS R5 ;GET FEC.

2940 014724 170205

STST R3 ;EXPECTED FPS.

2941 014726 170303

MOV #140200,R4 ;IS FPS CORRECT?

2942

CMP R4,R5 ;IF INCORRECT BRANCH.

2943 014730 012704 140200

BNE BBBER2

2944 014734 020405

MOV #4,R2 ;EXPECTED FEC.

2945 014736 001102

CMP R2,R3 ;WAS FEC CORRECT?

2946

2947 014740 012702 000004

2948 014744 020203

```

2949 014746 001111          BNE      BBBER3          ;IF INCORRECT BRANCH.
2950
2951          ;TEST DIVD WITH FSRC=0) AND TRAPS ENABLED.
2952 014750          BBB4:
      014750 104413          LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
2953 014752 012704 000200      MOV      #200,R4          ;SET UP FPS. TRAP ENABLED.
2954 014756 170104          LDFPS     R4
2955
2956 014760 012737 015006 001236      MOV      #BBB5,$TMP2
2957 014766 012700 015274          MOV      #BBBP2,R0          ;SET UP ACO OPERAND (NON ZERO).
2958 014772 172410          LDD      (R0),ACO
2959
2960 014774 012737 015014 000244      MOV      #BBB6,FPVECT          ;SET UP FOR THE EXPECTED INTERRUPT.
2961 015002 012700 015264          MOV      #BBBP1,R0          ;FSRC=0
2962
2963 015006 174410          BBB5:  DIVD     (R0),ACO          ;TEST INSTRUCTION (SHOULD RESULT IN TRAP).
2964 015010 170000          CFCC
2965
2966 015012 000502          BR       BBBER4          ;GO REPORT FAILURE, NO TRAP.
2967
2968 015014 022716 015010          BBB6:  CMP      #BBB5+2,(SP)          ;TRAP TO HERE WHEN THE DIVISION BY 0
2969                                     ;OCCURS. FIRST SEE IF THE ADDRESS OF
2970                                     ;THE TRAP IS 2+THE ADDRESS OF THE TEST
2971                                     ;DIVD INSTRUCTION.
2972 015020 001402          BEQ      1$
2973 015022 000137 037450          JMP      FPSPUR          ;IF NOT THEN REPORT AN UNEXPECTED
2974                                     ;FP TRAP.
2975 015026 170205          1$:  STFPS     R5          ;GET FPS.
2976 015030 170303          STST     R3          ;GET FEC.
2977 015032 022626          CMP      (SP)+,(SP)+          ;RESET THE STACK.
2978
2979 015034 012704 100200          MOV      #100200,R4          ;EXPECTED FPS.
2980 015040 020405          CMP      R4,R5          ;IS FPS CORRECT?
2981 015042 001040          BNE      BBBER2          ;IF INCORRECT BRANCH.
2982
2983 015044 012702 000004          MOV      #4,R2          ;EXPECTED FEC.
2984 015050 020203          CMP      R2,R3          ;IS FEC CORRECT?
2985 015052 001047          BNE      BBBER3          ;IF INCORRECT BRANCH.
2986
2987 015054 000137 015304          JMP      BBBDONE          ;OTHERWISE GO TO NEXT TEST.
2988
2989
2990          ;TRAP HERE IF AN UNEXPECTED INTERRUPT OCCURS.
2991 015060 062737 000002 001236      BBBER1: ADD     #2,$TMP2          ;SEE IF THE INTERRUPT OCCURRED
2992                                     ;DURING THE EXECUTION OF THE DIVD
2993                                     ;INSTRUCTION BEING TESTED.
2994 015066 021637 001236          CMP      (SP),$TMP2
2995 015072 001402          BEQ      1$
2996 015074 000137 037450          JMP      FPSPUR          ;IF NOT REPORT UNEXPECTED FP TRAP.
2997
2998 015100 022626          1$:  CMP      (SP)+,(SP)+          ;RESET THE STACK.
2999 015102 170303          STST     R3          ;GET FEC.
3000 015104 170205          STFPS     R5          ;GET FPS.
3001 015106 012737 000004 001240      MOV      #4,$TMP3          ;EXPECTED FEC.
3002 015114 010337 001242          MOV      R3,$TMP4
3003 015120 010537 001244          MOV      R5,$TMP5
3004 015124 010037 001250          MOV      R0,$TMP7
    
```



```

3005 015130 012737 140200 001246      MOV    #140200,$TMP6
3006 015136 104017      2$:    ERROR  +17      ;REPORT (BUT FD) FAILED RESULTING IN AN FP TRAP
3007                                     ;WITH TRAPS DISABLED.
3008 015140 000137 015304      JMP    BBBDONE
3009
3010                                     ;REPORT FPS INCORRECT:
3011 015144 010537 001242      BBBER2: MOV    R5,$TMP4
3012 015150 010437 001244      MOV    R4,$TMP5
3013 015154 010037 001246      MOV    R0,$TMP6
3014 015160 010137 001250      MOV    R1,$TMP7
3015 015164 104020      1$:    ERROR  +20
3016 015166 000137 015304      JMP    BBBDONE
3017
3018                                     ;REPORT FEC INCORRECT:
3019 015172 010337 001242      BBBER3: MOV    R3,$TMP4
3020 015176 010237 001240      MOV    R2,$TMP3
3021 015202 010037 001246      MOV    R0,$TMP6
3022 015206 010137 001250      MOV    R1,$TMP7
3023 015212 104021      1$:    ERROR  +21
3024 015214 000137 015304      JMP    BBBDONE
3025
3026                                     ;REPORT NO TRAP OCCURRED AFTER TRYING TO DIVIDE
3027                                     ;BY ZERO WITH ALL TRAPS ENABLED.
3028 015220 170303      BBBER4: STST   R3      ;GET FEC.
3029 015222 170205      STFPS  R5      ;GET FPS.
3030 015224 012737 000004 001242      MOV    #4,$TMP4
3031 015232 010337 001240      MOV    R3,$TMP3
3032 015236 010537 001244      MOV    R5,$TMP5
3033 015242 012737 100200 001246      MOV    #100200,$TMP6
3034 015250 010037 001250      MOV    R0,$TMP7
3035 015254 010137 001252      MOV    R1,$TMP10
3036 015260 104022      1$:    ERROR  +22
3037 015262 000410      BR     BBBDONE
3038
3039 015264 000000 000000 000000 BBBP1:  .WORD  0,0,0,0
3040 015274 012345 054321 023456 BBBP2:  .WORD  12345,54321,23456,76543
3041
3042
3043
3044 015304      BBBDONE:
      015304 104412      RSETUP      ;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?).
3045
3046
3054
    
```



```

3099 015462 002000 002000 5$: .WORD 2000,2000 ;ERROR RES.
3100 015466 104023 6$: ERROR +23
3101
3102 ;TEST THE DIVF INSTRUCTION:
3103 015470 CCC5: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
      015470 104413 JSR PC,DIVFSUB
3104 015472 004737 016070 1$: .WORD 12377,177777 ;AC
3105 015476 012377 177777 2$: .WORD 12300,0 ;FSRC
3106 015502 012300 000000 3$: .WORD 40252,125252 ;RES
3107 015506 040252 125252 4$: 0 ;FPS BEFORE EXECUTION.
3108 015512 000000 0 ;FPS AFTER EXECUTION.
3109 015514 000000 5$: .WORD -1,-1 ;ERROR RES.
3110 015516 177777 6$: ERROR +23
3111 015522 104023
3112
3113 ;TEST DIVIDE ALGORITHM. TEST ROUND CONSTANT.
3114 015524 CCC6: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
      015524 104413 JSR PC,DIVFSUB
3115 015526 004737 016070 1$: .WORD 64600,1 ;AC
3116 015532 064600 000001 2$: .WORD 66600,0 ;FSRC
3117 015536 066600 000000 3$: .WORD 36200,1 ;RES
3118 015542 036200 000001 4$: 0 ;FPS BEFORE EXECUTION.
3119 015546 000000 0 ;FPS AFTER EXECUTION.
3120 015550 000000 5$: .WORD 3000,3000 ;ERROR RES.
3121 015552 003000 003000 6$: ERROR +23
3122 015556 104023
3123
3124 ;TEST DIVF.
3125 015560 CCC7: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
      015560 104413 JSR PC,DIVFSUB
3126 015562 004737 016070 1$: .WORD 34577,177776 ;AC
3127 015566 034577 177776 2$: .WORD 23400,0 ;FSRC
3128 015572 023400 000000 3$: .WORD 51377,177776 ;RES
3129 015576 051377 177776 4$: 17 ;FPS BEFORE EXECUTION.
3130 015602 000017 0 ;FPS AFTER EXECUTION.
3131 015604 000000 5$: .WORD 3400,3400 ;ERROR RES.
3132 015606 003400 003400 6$: ERROR +23
3133 015612 104023
3134
3135 ;DIVF TEST.
3136 CCC8: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
      015614 104413 JSR PC,DIVFSUB
3137 015616 004737 016070 1$: .WORD 67652,125252 ;AC
3138 015616 004737 016070 2$: .WORD 56500,0 ;FSRC
3139 015622 067652 125252 3$: .WORD 51343,107070 ;RES
3140 015626 056500 000000 4$: 0 ;FPS BEFORE EXECUTION.
3141 015632 051343 107070 5$: .WORD 51543,107070 ;FPS AFTER EXECUTION.
3142 015636 000000 0 ;ERROR RES.
3143 015640 000000 6$: ERROR +26 ;DIDN'T INCREMENT THE EXPONENT
3144 015642 051543 107070 ;AFTER DIVID NORMALIZATION.
3145 015646 104026
3146
3147 ;DIVF WITH AC NEGATIVE, FSRC NEGATIVE.
3148 CCC9: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
      015650 104413 JSR PC,DIVFSUB
3149 015652 004737 016070
    
```



3151	015656	140400	000000	1\$:	.WORD	140400,0	:AC
3152	015662	140500	000000	2\$:	.WORD	140500,0	:FSRC
3153	015666	040052	125253	3\$:	.WORD	040052,125253	:RES
3154	015672	000000		4\$:	0		:FPS BEFORE EXECUTION.
3155	015674	000000			0		:FPS AFTER EXECUTION.
3156	015676	140052	125253	5\$:	.WORD	140052,125253	:ERROR RES.
3157	015702	104027		6\$:	ERROR	+27	:BAD SIGN.
3158							
3159							
3160	015704			:DIVF WITH AC NEGATIVE AND FSRC POSITIVE.			
	015704	104413		CCC10:	LPERR		:SET UP THE LOOP ON ERROR ADDRESS.
3161	015706	004737	016070		JSR	PC, DIVFSUB	
3162	015712	160077	000000	1\$:	.WORD	160077,0	:AC
3163	015716	040277	000000	2\$:	.WORD	40277,0	:FSRC
3164	015722	160000	000000	3\$:	.WORD	160000,0	:RES
3165	015726	000007		4\$:	7		:FPS BEFORE EXECUTION.
3166	015730	000010			10		:FPS AFTER EXECUTION.
3167	015732	060000	000000	5\$:	.WORD	60000,0	:ERROR RES.
3168	015736	104027		6\$:	ERROR	+27	:BAD SIGN.
3169							
3170				:DIVF WITH AC POSITIVE AND FSRC NEGATIVE.			
3171	015740			CCC11:	LPERR		:SET UP THE LOOP ON ERROR ADDRESS.
	015740	104413			JSR	PC, DIVFSUB	
3172	015742	004737	016070				
3173	015746	040400	000000	1\$:	.WORD	40400,0	:AC
3174	015752	140500	000000	2\$:	.WORD	140500,0	:FSRC
3175	015756	140052	125253	3\$:	.WORD	140052,125253	:RES
3176	015762	000017		4\$:	17		:FPS BEFORE EXECUTION.
3177	015764	000010			10		:FPS AFTER EXECUTION.
3178	015766	040052	125253	5\$:	.WORD	40052,125253	:ERROR RES.
3179	015772	104027		6\$:	ERROR	+27	:BAD SIGN.
3180							
3181							
3182				:TEST DIVF BOTH OPERANDS POSITIVE AND TRUNCATE MODE.			
3183	015774			CCC12:	LPERR		:SET UP THE LOOP ON ERROR ADDRESS.
	015774	104413			JSR	PC, DIVFSUB	
3184	015776	004737	016070				
3185	016002	060100	000001	1\$:	.WORD	60100,1	:AC
3186	016006	040300	000000	2\$:	.WORD	40300,0	:FSRC
3187	016012	060000	000000	3\$:	.WORD	60000,0	:RES
3188	016016	000052		4\$:	52		:FPS BEFORE EXECUTION.
3189	016020	000040			40		:FPS AFTER EXECUTION.
3190	016022	060000	000001	5\$:	.WORD	60000,1	:ERROR RES.
3191	016026	104030		6\$:	ERROR	+30	:TRUNCATION ERROR
3192							
3193				:DIVF WITH POSITIVE OPERANDS AND ROUND MODE.			
3194	016030			CCC13:	LPERR		:SET UP THE LOOP ON ERROR ADDRESS.
	016030	104413			JSR	PC, DIVFSUB	
3195	016032	004737	016070				
3196	016036	060100	000001	1\$:	.WORD	60100,1	:AC
3197	016042	040300	000000	2\$:	.WORD	40300,0	:FSRC
3198	016046	060000	000001	3\$:	.WORD	60000,1	:RES
3199	016052	000005		4\$:	5		:FPS BEFORE EXECUTION.
3200	016054	000000			0		:FPS AFTER EXECUTION.
3201	016056	060000	000000	5\$:	.WORD	60000,0	:ERROR RES.
3202	016062	104031		6\$:	ERROR	+31	:ROUND ERROR.
3203							



```

3261
3262 016214 021061 000010          CMP      (R0),10(R1)      ;IS THE RESULT CORRECT?
3263 016220 001011          BNE      10$             ;IF INCORRECT BRANCH.
3264 016222 026061 000002 000012  CMP      2(R0),12(R1)
3265 016230 001005          BNE      10$
3266
3267 016232 026104 000016          CMP      16(R1),R4       ;IS FPS CORRECT?
3268 016236 001020          BNE      15$             ;IF INCORRECT BRANCH.
3269 016240 000161 000026          JMP      26(R1)         ;IF NO ERRORS OCCURRED RETURN.
3270
3271 016244 021061 000020          10$:    CMP      (R0),20(R1)   ;DOES THE INCORRECT RESULT
3272 016250 001010          BNE      11$             ;MATCH THE ANTICIPATED INCORRECT RESULT.
3273 016252 026061 000002 000022  CMP      2(R0),22(R1)
3274 016260 001004          BNE      11$             ;BRANCH IF NO.
3275
3276 016262 010102          MOV      R1,R2          ;IT MATCHED SO RETURN TO THE ERROR
3277                                ;REPORT AT THE CALLING ROUTINE.
3278 016264 062702 000024          ADD      #24,R2
3279 016270 000112          JMP      (R2)
3280
3281 016272          11$:
3282 016272 104023          12$:    ERROR   +23
3283 016274 000161 000026          13$:    JMP      26(R1)
3284
3285 016300          15$:
3286 016300 104032          16$:    ERROR   +32
3287 016302 000774          BR      13$
3288
3289 016304 000000 000000 000000  DIVFT:  .WORD  0,0,0,0
3290
3291 016314          CCCDONE:
      016314 104412          RSETUP
                                ;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?).

3292
3293
3300
  
```



3301

```

.SBTTL TEST # 7 - DIVD TEST
:*****
:*TEST 7      DIVD TEST
:*
:*THIS IS A TEST OF THE DIVD INSTRUCTION. NOTE THAT A SUBROUTINE IS
:*USED TO SET UP THE OPERANDS, EXECUTE THE INSTRUCTION AND CHECK THE RESULTS.
:*
:*****

```

016316 000004

TST7: SCOPE

3302

:DIVD TEST WITH POSITIVE OPERANDS AND IN ROUND MODE.

3303

DDD1:

3304

```

016320 104413
016322 004737 017010
016326 034277 000000 000000
016336 040277 000000 000000
016346 034200 000000 000000
016356 000200
016360 000200
016362 177777 177777 177777
016372 104033

```

```

LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
JSR PC,DIVDSUB
1$: .WORD 34277,0,0,0 ;AC
2$: .WORD 40277,0,0,0 ;FSRC
3$: .WORD 34200,0,0,0 ;RES
4$: 200 ;FPS BEFORE EXECUTION.
200 ;FPS AFTER EXECUTION.
5$: .WORD -1,-1,-1,-1 ;ERROR RES.
6$: ERROR +35

```

3313

:DIVD WITH AC NEGATIVE AND FSRC POSITIVE IN TRUNCATE MODE.

3314

DDD2:

3315

```

016374 104413
016376 004737 017010
016402 134277 000000 000000
016412 040277 000000 000000
016422 134200 000000 000000
016432 000207
016434 000210
016436 177777 177777 177777
016446 104033

```

```

LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
JSR PC,DIVDSUB
1$: .WORD 134277,0,0,0 ;AC
2$: .WORD 40277,0,0,0 ;FSRC
3$: .WORD 134200,0,0,0 ;RES
4$: 207 ;FPS BEFORE EXECUTION.
210 ;FPS AFTER EXECUTION.
5$: .WORD -1,-1,-1,-1 ;ERROR RESULT.
6$: ERROR +35

```

3324

:DIVD TEST WITH OPERANDS BOTH NEGATIVE AND IN TRUNCATE MODE.

3325

DDD3:

3326

```

016450 104413
016452 004737 017010
016456 134300 000000 000000
016466 140300 000000 000000
016476 034200 000000 000000
016506 000250
016510 000240
016512 034200 000000 000000
016522 104035

```

```

LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
JSR PC,DIVDSUB
1$: .WORD 134300,0,0,1 ;AC
2$: .WORD 140300,0,0,0 ;FSRC
3$: .WORD 34200,0,0,0 ;RES
4$: 250 ;FPS BEFORE EXECUTION.
240 ;FPS AFTER EXECUTION.
5$: .WORD 34200,0,0,1 ;ERROR RES.
6$: ERROR +35 ;TRUNCATION ERROR.

```

3335

:DIVD WITH AC POSITIVE AND FSRC NEGATIVE IN ROUND MODE.

3336

DDD4:

3337

```

016524 104413
016526 004737 017010
016532 034300 000000 000000
016542 140300 000000 000000
016552 134200 000000 000000
016562 000207
016564 000210
016566 134200 000000 000000
016576 104036

```

```

LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
JSR PC,DIVDSUB
1$: .WORD 34300,0,0,1 ;AC
2$: .WORD 140300,0,0,0 ;FSRC
3$: .WORD 134200,0,0,1 ;RES
4$: 207 ;FPS BEFORE EXECUTION.
210 ;FPS AFTER EXECUTION.
5$: .WORD 134200,0,0,0 ;ERROR RES.
6$: ERROR +36 ;ROUND ERROR.

```

3345

```

3346
3347 ;DIVD TEST.
3348 016600 104413 017010 000000 000000 1$: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
016600 004737 017010 JSR PC,DIVDSUB ;AC
3349 016602 004737 017010 000000 000000 2$: .WORD 100400,0,0,0 ;FSRC
3350 016606 100400 000000 000000 3$: .WORD 500,0,0,0 ;RES
3351 016616 000500 000000 000000 4$: .WORD 140052,125252 ;FPS BEFORE EXECUTION.
3352 016626 140052 125252 125252 5$: .WORD 7647 ;FPS AFTER EXECUTION.
3353 016632 125252 125252 6$: .WORD 7650 ;ERROR RES.
3354 016636 007647 177777 177777 1$: .WORD -1,-1,-1,-1
3355 016640 007650 2$: .WORD +33
3356 016642 177777 3$: .WORD
3357 016652 104033 4$: ERROR

```

```

3358
3359 ;DIVD TEST WITH AC POSITIVE AND FSRC NEGATIVE IN ROUND MODE.
3360 016654 104413 017010 000000 000000 1$: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
3361 016654 004737 017010 JSR PC,DIVDSUB ;AC
3362 016656 004737 017010 000000 000000 2$: .WORD 400,0,0,0 ;FSRC
3363 016662 000400 000000 000000 3$: .WORD 100500,0,0,0 ;RES
3364 016672 100500 000000 000000 4$: .WORD 140052,125252 ;FPS BEFORE EXECUTION.
3365 016702 140052 125252 125253 5$: .WORD 7707 ;FPS AFTER EXECUTION.
3366 016706 125252 125253 6$: .WORD 7710 ;ERROR RES.
3367 016712 007707 177777 177777 1$: .WORD -1,-1,-1,-1
3368 016714 007710 2$: .WORD +33
3369 016716 177777 3$: .WORD
3370 016726 104033 4$: ERROR

```

```

3371 ;DIVD TEST.
3372 016730 104413 017010 000000 000000 1$: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
3373 016730 004737 017010 JSR PC,DIVDSUB ;AC
3374 016732 004737 017010 170360 170360 2$: .WORD 170360,170360 ;FSRC
3375 016736 170360 170360 170360 3$: .WORD 170360,170360 ;RES
3376 016742 170360 170360 170360 4$: .WORD 170360,170360 ;FPS BEFORE EXECUTION.
3377 016746 170360 170360 170360 5$: .WORD 40200,0,0,0 ;FPS AFTER EXECUTION.
3378 016752 170360 170360 000000 000000 6$: .WORD 7717 ;ERROR RES.
3379 016756 040200 000000 000000 1$: .WORD -1,-1,-1,-1
3380 016766 007717 2$: .WORD +33
3381 016770 007700 3$: .WORD
3382 016772 177777 177777 4$: ERROR
3383 017002 104033 5$:
3384 017004 000137 017250 6$: JMP DDDDONE ;GO TO NEXT TEST.

```

```

3385
3386 ;THIS SUBROUTINE, DIVDSUB, IS CALLED TO SET UP, EXECUTE
3387 ;AND CHECK THE RESULT OF A DIVD INSTRUCTION. IT IS CALLED THUS:
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399

```

	JSR PC,DIVDSUB	
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

3400  
3401  
3402

: THE OPERANDS ARE SET UP (USING ACO FOR THE AC OPERAND). THEN  
: FPSB IS LOADED INTO THE FPS. THE INSTRUCTION, DIVD IS EXECUTED.



```

3404                                     :AFTER THE EXECUTION THE RESULT IS CHECKED AGAINST THE
3405                                     :EXPECTED CORRECT RESULT, RES. IF IT IS CORRECT THEN THE FPS
3406                                     :IS CHECKED WITH THE EXPECTED CORRECT FPS, FPSA. IF THE FPS WAS
3407                                     :INCORRECT THEN IT IS REPORTED. IF THE RESULT WAS INCORRECT IT
3408                                     :IS COMPARED WITH ERRES IN AN ATTEMPT TO ANALYSE THE ERROR. IF
3409                                     :THE INCORRECT RESULT MATCHED ERRES THEN CONTROL IS PASSED TO
3410                                     :THE ERROR CALL AT ERR. IF THE INCORRECT RESULT DID NOT MATCH ERRES
3411                                     :THEN THE FAILURE IS REPORTED IN DIVDSUB AND CONTROL IS PASSED TO
3412                                     :CONT. IF NO ERRORS ARE DETECTED THEN DIVDSUB RETURNS CONTROL
3413                                     :TO CONT.
3414
3415 017010 012601                                DIVDSUB:  MOV      (SP)+,R1      ;GET A POINTER TO THE ARGUMENTS.
3416 017012 012700 000200                      MOV      #200,R0          ;SET FD MODE.
3417 017016 170100                                LDFPS   R0
3418
3419 017020 010100                                MOV      R1,R0           ;SET UP THE ACO OPERAND.
3420 017022 172410                                LDD     (R0),ACO
3421 017024 016100 000030                      MOV      30(R1),R0       ;LOAD THE FPS.
3422 017030 170100                                LDFPS   R0
3423
3424 017032 012737 017046 001236                MOV      #1$,STMP2
3425 017040 010100                                MOV      R1,R0           ;ESTABLISH A POINTER TO FSRC.
3426 017042 062700 000010                      ADD     #10,R0
3427
3428 017046 174410                                1$:     DIVD   (R0),ACO    ;EXECUTE THE TEST INSTRUCTION.
3429
3430 017050 170204                                STFPS   R4               ;GET THE FPS.
3431 017052 012700 000200                      MOV      #200,R0        ;SET FD MODE.
3432 017056 170100                                LDFPS   R0
3433
3434 017060 012700 017240                      MOV      #DIVDT,R0      ;GET THE RESULT.
3435 017064 174010                                STD     ACO,(R0)
3436
3437 017066 010102                                MOV      R1,R2           ;SAVE DATA IN CASE OF ERROR.
3438 017070 010237 001240                      MOV      R2,STMP3
3439 017074 062702 000010                      ADD     #10,R2
3440 017100 010237 001242                      MOV      R2,STMP4
3441 017104 062702 000010                      ADD     #10,R2
3442 017110 010237 001244                      MOV      R2,STMP5
3443 017114 012737 017240 001246                MOV      #DIVDT,STMP6
3444 017122 010437 001250                      MOV      R4,STMP7
3445 017126 016137 000032 001252                MOV      32(R1),STMP10
3446
3447 017134 010102                                MOV      R1,R2           ;CHECK THE RESULT.
3448 017136 062702 000020                      ADD     #20,R2
3449 017142 012703 017240                      MOV      #DIVDT,R3
3450 017146 012705 000004                      MOV      #4,R5
3451 017152 022223                                2$:     CMP     (R2)+,(R3)+
3452 017154 001006                                BNE    10$              ;BRANCH IF RESULT INCORRECT.
3453 017156 077503                                SOB    R5,2$
3454
3455 017160 026104 000032                      CMP     32(R1),R4        ;IS FPS CORRECT?
3456 017164 001023                                BNE    15$              ;BRANCH IF INCORRECT.
3457 017166 000161 000046                      JMP     46(R1)          ;RETURN.
3458
3459 017172 010102                                10$:    MOV     R1,R2
3460 017174 062702 000034                      ADD     #34,R2

```

3461	017200	012703	017240		MOV	#DIVDT,R3	
3462	017204	012705	000004		MOV	#4,R5	
3463	017210	022223		11\$:	CMP	(R2)+,(R3)+	
3464	017212	001005			BNE	12\$	:BRANCH IF NO.
3465	017214	077503			SOB	R5,11\$	
3466	017216	010102			MOV	R1,R2	:IF THE INCORRECT RESULT WAS
3467	017220	062702	000044		ADD	#44,R2	:ANTICIPATED RETURN TO THE
3468							:ERROR REPORT IN THE CALLING
3469	017224	000112			JMP	(R2)	:ROUTINE.
3470							
3471	017226			12\$:			:REPORT RESULT INCORRECT.
3472	017226	104033		13\$:	ERROR	+33	
3473	017230	000161	000046	14\$:	JMP	46(R1)	
3474							
3475	017234			15\$:			:REPORT FPS INCORRECT.
3476	017234	104034		16\$:	ERROR	+34	
3477	017236	000774			BR	14\$	
3478							
3479	017240	000000	000000	000000	DIVDT:	.WORD	0,0,0,0
3480							
3481	017250				DDDDONE:		
	017250	104412			RSETUP		: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?).
3482							
3483							
3491							

3492

```
.SBTTL TEST # 10 - MULF TEST  
:*****  
:TEST 10 MULF TEST  
:*****  
:THIS IS A TEST OF THE MULF INSTRUCTION. IT MAKES USE OF A SUBROUTINE  
:TO SET UP THE OPERANDS, EXECUTE THE MULF INSTRUCTION AND CHECK THE  
:RESULTS.  
:*****
```

```
3493 017252 000004  
3494  
3495 017254  
3496 017254 104413 020034  
3497 017256 004737 000000  
3498 017262 000000 000000  
3499 017266 000000 000000  
3500 017272 000000 000000  
3501 017276 007517  
3502 017300 007504  
3503 017302 177777 177777  
3504 017306 104037  
3505  
3506 017310  
3507 017310 104413 020034  
3508 017312 004737 034435  
3509 017316 071625 000000  
3510 017322 000000 000000  
3511 017326 000000 000000  
3512 017332 000013  
3513 017334 000004 177777  
3514 017336 177777 177777  
3515 017342 104037  
3516  
3517 017344  
3518 017344 104413 020034  
3519 017346 004737 000000  
3520 017352 071625 153443  
3521 017356 000000 000000  
3522 017362 000000 000000  
3523 017366 007500  
3524 017370 007504 177777  
3525 017372 177777 177777  
3526 017376 104037  
3527  
3528 017400  
3529 017400 104413 020034  
3530 017402 004737 000000  
3531 017406 040200 177777  
3532 017412 040177 177777  
3533 017416 040177 177777  
3534 017422 000017  
3535 017424 000000  
3535 017426 140177 177777
```

```
TST10: SCOPE  
:MULF WITH (FSRC=AC=0)  
EEE1: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.  
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.  
5$: .WORD -1,-1  
6$: ERROR +37  
:MULF WITH (FSRC=0).  
EEE2: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.  
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.  
5$: .WORD -1,-1 ;ERROR RES.  
6$: ERROR +37  
:MULF WITH (AC=0)  
EEE3: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.  
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.  
5$: .WORD -1,-1 ;ERROR RES.  
6$: ERROR +37  
:MULF WITH AC POSITIVE AND FSRC POSITIVE IN ROUND MODE.  
EEE4: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.  
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.  
5$: .WORD 140177,-1 ;ERROR RES.
```



```

3536 017432 104041
3537
3538
3539 017434
      017434 104413
3540 017436 004737 020034
3541 017442 040177 177777
3542 017446 040200 000000
3543 017452 040177 177777
3544 017456 000040
3545 017460 000040
3546 017462 037777 177777
3547 017466 104042
3548
3549
3550
3551 017470
      017470 104413
3552 017472 004737 020034
3553 017476 040100 000000
3554 017502 040100 000000
3555 017506 040020 000000
3556 017512 000012
3557 017514 000000
3558 017516 042040 000000
3559 017522 104043
3560
3561
3562
3563 017524
      017524 104413
3564 017526 004737 020034
3565 017532 017500 000000
3566 017536 023652 125252
3567 017542 003177 177777
3568 017546 007417
3569 017550 007400
3570 017552 177777 177777
3571 017556 104037
3572
3573
3574 017560
      017560 104413
3575 017562 004737 020034
3576 017566 040342 000000
3577 017572 176542 000000
3578 017576 176707 102000
3579 017602 000007
3580 017604 000010
3581 017606 076507 102000
3582 017612 104041
3583
3584
3585 017614
      017614 104413
3586 017616 004737 020034
3587 017622 140200 000000
    
```

```

6$:      ERROR      +41          ;BAD SIGN.
;MULF WITH AC POSITIVE AND FSRC POSITIVE IN TRUNCATE MODE.
EEE5:
      LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
      JSR            PC,MULFSUB
1$:      .WORD       40177,-1     ;AC
2$:      .WORD       40200,0     ;FSRC
3$:      .WORD       40177,-1     ;RES
4$:      40                    ;FPS BEFORE EXECUTION.
      40                    ;FPS AFTER EXECUTION.
5$:      .WORD       37777,-1     ;ERROR RES.
6$:      ERROR      +42          ;ST 252 TO 044 INTO 444 (BUT Y62)
      ;MUL. NORMALIZATION FAILURE.

;MULF WITH BOTH OPERANDS POSITIVE NORMALIZE TEST.
EEE6:
      LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
      JSR            PC,MULFSUB
1$:      .WORD       40100,0     ;AC
2$:      .WORD       40100,0     ;FSRC
3$:      .WORD       40020,0     ;RES
4$:      12                    ;FPS BEFORE EXECUTION.
      0                    ;FPS AFTER EXECUTION.
5$:      .WORD       42040,0     ;ERROR RES.
6$:      ERROR      +43          ;ST 252 TO 444 INTO 042 (BUT Y62)
      ;MUL. NORMALIZATION FAILURE.

;MULF WITH BOTH OPERANDS POSITIVE IN ROUND MODE.
EEE7:
      LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
      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.
5$:      .WORD       -1,-1
6$:      ERROR      +37

;MULF WITH AC POSITIVE AND FSRC NEGATIVE IN ROUND MODE.
EEE8:
      LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
      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.
7$:      .WORD       76507,102000 ;ERROR RES.
6$:      ERROR      +41          ;BAD SIGN.

;MULF WITH AC NEGATIVE AND FSRC POSITIVE IN ROUND MODE.
EEE9:
      LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
      JSR            PC,MULFSUB
1$:      .WORD       140200,0    ;AC
    
```

3588 017626 007417 007417  
 3589 017632 107417 007417  
 3590 017636 000000  
 3591 017640 000010  
 3592 017642 007417 007417  
 3593 017646 104041  
 3594  
 3595  
 3596 017650  
 3597 017652 104413 020034  
 3598 017656 144600 000000  
 3599 017662 154000 000000  
 3600 017666 060400 000000  
 3601 017672 000017  
 3602 017674 000000  
 3603 017676 160400 000000  
 3604 017702 104041  
 3605  
 3606  
 3607 017704  
 3608 017706 104413 020034  
 3609 017712 140300 000000  
 3610 017716 160000 000001  
 3611 017722 060100 000002  
 3612 017726 000010  
 3613 017730 000000  
 3614 017732 060100 000001  
 3615 017736 104044  
 3616  
 3617  
 3618 017740  
 3619 017742 104413 020034  
 3620 017746 060000 000001  
 3621 017752 140300 000000  
 3622 017756 160100 000001  
 3623 017762 007547  
 3624 017764 007550  
 3625 017766 160100 000001  
 3626 017772 104045  
 3627  
 3628  
 3629 017774  
 3630 017776 104413 020034  
 3631 020002 040277 000000  
 3632 020006 060000 000001  
 3633 020012 060077 000001  
 3634 020016 000014  
 3635 020020 000000  
 3636 020022 060077 000002  
 3637 020026 104044  
 3638  
 3639 020030 000137 020260  
 3640

```

2$: .WORD 7417,7417 ;FSRC
3$: .WORD 107417,7417 ;RES
4$: 0 ;FPS BEFORE EXECUTION.
      10 ;FPS AFTER EXECUTION.
5$: .WORD 7417,7417 ;ERROR RES.
6$: ERROR +41 ;BAD SIGN.

;MULF WITH BOTH OPERANDS NEGATIVE IN ROUND MODE.
EEE10: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
        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.
5$: .WORD 160400,0 ;ERROR RES.
6$: ERROR +41 ;BAD SIGN.

;MULF BOTH OPERANDS NEGATIVE IN ROUND MODE.
EEE11: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
        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.
5$: .WORD 60100,1 ;ERROR RES.
6$: ERROR +44 ;ROUND FAILURE.

;MULF WITH AC POSITIVE AND FSRC NEGATIVE IN TRUNCATE MODE.
EEE12: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
        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.
5$: .WORD 160100,1 ;ERROR RES.
6$: ERROR +45 ;TRUNCATION ERROR.

;MULF WITH AC POSITIVE AND FSRC POSITIVE IN ROUND MODE.
EEE13: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
        JSR PC,MULFSUB
1$: .WORD 40277,0 ;AC
2$: .WORD 60000,1 ;FSRC
3$: .WORD 60077,1 ;RES
4$: 14 ;FPS BEFORE EXECUTION.
      0 ;FPS AFTER EXECUTION.
5$: .WORD 60077,2 ;ERROR RES.
6$: ERROR +44 ;ROUND FAILURE. CONSTANT BAD.

        JMP EEEDONE ;GO TO THE NEXT TEST.
    
```







3698	020164	001011			BNE	10\$		:IF INCORRECT BRANCH.
3699	020166	026061	000002	000012	CMP	2(R0),12(R1)		
3700	020174	001005			BNE	10\$		
3701								
3702	020176	026104	000016		CMP	16(R1),R4		:IS FPS CORRECT?
3703	020202	001020			BNE	15\$		:IF INCORRECT BRANCH.
3704	020204	000161	000026		JMP	26(R1)		:IF NO ERRORS OCCURRED RETURN.
3705								
3706	020210	021061	000020	10\$:	CMP	(R0),20(R1)		:DOES THE INCORRECT RESULT
3707	020214	001010			BNE	11\$		:MATCH THE ANTICIPATED INCORRECT RESULT.
3708	020216	026061	000002	000022	CMP	2(R0),22(R1)		
3709	020224	001004			BNE	11\$		:BRANCH IF NO.
3710								
3711	020226	010102			MOV	R1,R2		:IT MATCHED SO RETURN TO THE ERROR
3712								:REPORT AT THE CALLING ROUTINE.
3713	020230	062702	000024		ADD	#24,R2		
3714	020234	000112			JMP	(R2)		
3715								
3716	020236			11\$:				:REPORT RESULT INCORRECT.
3717	020236	104037		12\$:	ERROR	+37		
3718	020240	000161	000026	13\$:	JMP	26(R1)		
3719								
3720	020244			15\$:				:REPORT FPS INCORRECT.
3721	020244	104040		16\$:	ERROR	+40		
3722	020246	000774			BR	13\$		
3723								
3724	020250	000000	000000	000000	MULFT:	.WORD	0,0,0,0	
3725								
3726	020260				EEEDONE:			
	020260	104412			RSETUP			: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?).
3727								
3728								
3736								

3737

```
.SBTTL TEST # 11 - MULD TEST
:*****
:*TEST 11      MULD TEST
:*
:*THIS IS A TEST OF THE MULD INSTRUCTION. NOTE THAT A SUBROUTINE IS
:*USED TO SET UP THE OPERANDS, EXECUTE THE MULD INSTRUCTION AND
:*CHECK THE RESULTS.
:*
:*****
```

020262 000004

TST11: SCOPE

3738

:MULD TEST WITH AC POSITIVE AND FSRC POSITIVE.

3739

FFF1:

3740

```
LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
JSR            PC,MULDSUB
:AC
:FSRC
:RES
:FPS BEFORE EXECUTION.
:FPS AFTER EXECUTION.
:ERROR RES.
:BAD CONSTANT USED IN ALGORITHM
:USED 24 INSTEAD OF 56.
```

020264 104413

3741 020266 004737 020550

3742 020272 040200 000000 000000

3743 020302 023777 177777 177777

3744 020312 023777 177777 177777

3745 020322 000217

3746 020324 000200

3747 020326 023777 177777 000000

3748 020336 104047

3749

3750

3751

:MULD TEST WITH BOTH OPERANDS POSITIVE TRUNCATION TEST.

3752 020340

FFF2:

020340 104413

3753 020342 004737 020550

3754 020346 065400 000000 000000

3755 020356 037577 177777 177777

3756 020366 064777 177777 177777

3757 020376 000247

3758 020400 000240

3759 020402 065000 000000 000000

3760 020412 104050

3761

3762

3763 020414

:MULD TEST WITH BOTH OPERANDS NEGATIVE IN ROUND MODE.

FFF3:

020414 104413

3764 020416 004737 020550

3765 020422 137577 177777 177777

3766 020432 165400 000000 000000

3767 020442 065000 000000 000000

3768 020452 007717

3769 020454 007700

3770 020456 064777 177777 177777

3771 020466 104051

3772

3773

3774 020470

:MULD TEST WITH AC POSITIVE AND FSRC NEGATIVE IN ROUND MODE.

FFF4:

020470 104413

3775 020472 004737 020550

3776 020476 017500 000000 000000

3777 020506 123652 125252

3778 020512 125252 125252

3779 020516 103177 177777 177777

3780 020526 000200

```
LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
JSR            PC,MULDSUB
:AC
:FSRC
:RES
:FPS BEFORE EXECUTION.
:FPS AFTER EXECUTION.
:ERROR RES.
:ROUND ERROR.
```

```

3781 020530 000210          210          :FPS AFTER EXECUTION.
3782 020532 103200 000000 000000 5$: .WORD 103200,0,0,0 :ERROR RES.
3783 020542 104052          6$: ERROR +52 :ROUND ERROR (BAD CONSTANT).
3784
3785 020544 000137 021010          JMP FFFDONE
    
```

:THIS SUBROUTINE, MULDSUB, IS CALLED TO SET UP, EXECUTE  
 :AND CHECK THE RESULT OF A MULDT INSTRUCTION. IT IS CALLED THUS:

```

:
:      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.

```

3813 020550 012601          MULDSUB: MOV (SP)+,R1 :GET A POINTER TO THE ARGUMENTS.
3814 020552 012700 000200 MOV #200,R0 :SET FD MODE.
3815 020556 170100 LDFPS R0
3816
3817 020560 010100 MOV R1,R0 :SET UP THE ACO OPERAND.
3818 020562 172410 LDD (R0),ACO
3819 020564 016100 000030 MOV 30(R1),R0 :LOAD THE FPS.
3820 020570 170100 LDFPS R0
3821
3822 020572 012737 020606 001236 MOV #1$,STMP2
3823 020600 010100 MOV R1,R0 :ESTABLISH A POINTER TO FSRC.
3824 020602 062700 000010 ADD #10,R0
3825
3826 020606 171010 1$: MULDT (R0),ACO :EXECUTE THE TEST INSTRUCTION.
3827
3828 020610 170204 STFPS R4 :GET THE FPS.
3829 020612 012700 000200 MOV #200,R0 :SET FD MODE.
3830 020616 170100 LDFPS R0
3831
3832 020620 012700 021000 MOV #MULDT,R0 :GET THE RESULT.
3833 020624 174010 STD ACO,(R0)
3834
3835 020626 010102 MOV R1,R2 :SAVE DATA IN CASE OF ERROR.
3836 020630 010237 001240 MOV R2,STMP3
3837 020634 062702 000010 ADD #10,R2
    
```



```

3838 020640 010237 001242      MOV    R2,$TMP4
3839 020644 062702 000010      ADD    #10,R2
3840 020650 010237 001244      MOV    R2,$TMP5
3841 020654 012737 021000 001246      MOV    #MULDT,$TMP6
3842 020662 010437 001250      MOV    R4,$TMP7
3843 020666 016137 000032 001252      MOV    32(R1),$TMP10
3844
3845 020674 010102      MOV    R1,R2      ;CHECK THE RESULT.
3846 020676 062702 000020      ADD    #20,R2
3847 020702 012703 021000      MOV    #MULDT,R3
3848 020706 012705 000004      MOV    #4,R5
3849 020712 022223      2$:  CMP    (R2)+,(R3)+
3850 020714 001006      BNE   10$      ;BRANCH IF RESULT INCORRECT.
3851 020716 077503      SOB   R5,2$
3852
3853 020720 026104 000032      CMP    32(R1),R4  ;IS FPS CORRECT?
3854 020724 001023      BNE   15$      ;BRANCH IF INCORRECT.
3855 020726 000161 000046      JMP    46(R1)    ;RETURN.
3856
3857 020732 010102      10$: MOV    R1,R2    ;WAS INCORRECT RESULT ANTICIPATED?
3858 020734 062702 000034      ADD    #34,R2
3859 020740 012703 021000      MOV    #MULDT,R3
3860 020744 012705 000004      MOV    #4,R5
3861 020750 022223      11$: CMP    (R2)+,(R3)+
3862 020752 001005      BNE   12$      ;BRANCH IF NO.
3863 020754 077503      SOB   R5,11$
3864 020756 010102      MOV    R1,R2    ;IF THE INCORRECT RESULT WAS
3865 020760 062702 000044      ADD    #44,R2   ;ANTICIPATED RETURN TO THE
3866                                     ;ERROR REPORT IN THE CALLING
3867 020764 000112      JMP    (R2)     ;ROUTINE.
3868
3869 020766      12$:      ;REPORT RESULT INCORRECT.
3870 020766 104246      13$:      ERROR  +246
3871 020770 000161 000046      14$:      JMP    46(R1)
3872
3873 020774      15$:      ;REPORT FPS INCORRECT.
3874 020774 104046      16$:      ERROR  +46
3875 020776 000774      BR     14$
3876
3877 021000 000000 000000 000000 MULDT: .WORD 0,0,0,0
3878
3879 021010      FFFDONE:
           021010 104412      RSETUP      ;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?).

3880
3881
3890
3891                                     ;TEST TITLE:UNDER/OVERFLOW, USING MULF WITH TRAPS DISABLED
    
```







```

3992      ;IF A TRAP OCCURS (IT SHOULD NOT) THEN OVUNFNT WILL READ THE FEC.
3993      ;SHOULD THE FEC MATCH THE ANTICIPATED FEC OVUNFNT WILL
3994      ;STORE ALL DATA AND TRANSFER CONTROL TO THE ERROR CALL AT ERR1. IF THE
3995      ;FEC IS NOT THE SAME AS THE ANTICIPATED FEC OVUNFNT WILL REPORT
3996      ;THE ERROR AND RETURN TO CONT. NOTE THAT OVUNFNT USES THE FLAG
3997      ;TO TELL WHETHER OR NOT THESE PARTICULAR OPERANDS WILL RESULT IN
3998      ;UNDERFLOW (FLAG=-1) OR OVERFLOW (FLAG=0).
3999
4000 021240 012601      OVUNFNT:      MOV      (SP)+,R1      ;GET A POINTER TO THE ARGUMENTS.
4001 021242 012700 000200      MOV      #200,R0      ;SET FD MODE.
4002 021246 170100      LDFPS   R0
4003
4004 021250 010100      MOV      R1,R0      ;LOAD ACO, OPERAND.
4005 021252 172410      LDD     (R0),ACO
4006
4007 021254 010102      MOV      R1,R2      ;SAVE THE DATA PATTERNS IN CASE OF
4008 021256 010237 001240      MOV      R2,$TMP3      ;ERROR.
4009 021262 062702 000004      ADD     #4,R2
4010 021266 010237 001242      MOV      R2,$TMP4
4011 021272 062702 000004      ADD     #4,R2
4012 021276 010237 001244      MOV      R2,$TMP5
4013 021302 016137 000022 001252      MOV     22(R1),$TMP10
4014 021310 012737 021640 001246      MOV     #OVFNIT,$TMP6
4015
4016 021316 016100 000020      MOV     20(R1),R0      ;LOAD THE FPS.
4017 021322 170100      LDFPS   R0
4018 021324 012737 021346 001236      MOV     #1,$TMP2
4019 021332 012737 021532 000244      MOV     #25$,FPVECT      ;SET UP THE FP TRAP VECTOR IN CASE
4020      ;OF ERROR.
4021 021340 010100      MOV     R1,R0      ;COMPUTE THE ADDRESS OF FSRC.
4022 021342 062700 000004      ADD     #4,R0
4023
4024 021346 171010      1$:     MULF   (R0),ACO      ;TEST INSTRUCTION.
4025
4026 021350 170204      2$:     STFPS  R4      ;GET FPS.
4027 021352 170305      STST   R5      ;GET FEC.
4028 021354 012700 000200      MOV     #200,R0      ;SET FD MODE.
4029 021360 170100      LDFPS   R0
4030 021362 012700 021640      MOV     #OVFNIT,R0      ;GET THE RESULT.
4031 021366 174010      STD    ACO,(R0)
4032 021370 010437 001250      MOV     R4,$TMP7
4033 021374 010537 001254      MOV     R5,$TMP11
4034
4035 021400 012700 021640      MOV     #OVFNIT,R0      ;CHECK THE RESULT.
4036 021404 010102      MOV     R1,R2
4037 021406 062702 000010      ADD     #10,R2
4038 021412 012703 000002      MOV     #2,R3
4039 021416 022022      3$:     CMP     (R0)+,(R2)+
4040 021420 001015      BNE    15$      ;BRANCH IF INCORRECT.
4041 021422 077303      SOB    R3,3$
4042
4043 021424 026104 000022      CMP     22(R1),R4      ;WAS FPS CORRECT?
4044 021430 001002      BNE    10$      ;BRANCH IF FPS IS INCORRECT.
4045
4046 021432 000161 000036      4$:     JMP     36(R1)      ;RETURN, TEST COMPLETED.
4047
4048      ;REPORT INCORRECT FPS.
    
```

4049 021436 005761 000026  
4050 021442 001002  
4051  
4052  
4053 021444 104111  
4054 021446 000771  
4055  
4056 021450  
4057 021450 104112  
4058 021452 000767  
4059

10\$: TST 26(R1)  
BNE 12\$  
  
11\$: ERROR +111  
BR 4\$  
  
12\$:  
13\$: ERROR +112  
BR 4\$

;WAS THE RESULT OVER OR UNDER FLOW?  
;BRANCH IF UNDERFLOW.  
  
;REPORT FPS BAD AFTER OVERFLOW.  
  
;REPORT FPS BAD AFTER UNDERFLOW.

```

4061                                     :RESULT INCORRECT.
4062 021454 012700 021640                15$:  MOV    #OVFNTT,R0      ;SEE IF FAILURE IS ANTICIPATED
4063 021460 010102                       MOV    R1,R2             ;FAILURE.
4064 021462 062702 000014                 ADD    #14,R2
4065 021466 012703 000002                 MOV    #2,R3
4066 021472 022022                       16$:  CMP    (R0)+,(R2)+
4067 021474 001007                       BNE   17$
4068 021476 077303                       SOB   R3,16$
4069
4070 021500 010102                       MOV    R1,R2             ;ERROR WAS ANTICIPATED SO RETURN
4071 021502 062702 000034                 ADD    #34,R2           ;TO THE ERROR REPORT IN THE CALLING
4072 021506 010237 001236                 MOV    R2,$TMP2        ;ROUTINE.
4073 021512 000112                       JMP    (R2)
4074
4075 021514 005761 000026                 17$:  TST    26(R1)
4076
4077
4078
4079
4080 021520 001002                       BNE   19$
4081
4082
4083 021522 104113                       18$:  ERROR  +113
4084 021524 000742                       BR    4$
4085
4086 021526
4087 021526 104114                       19$:  ERROR  +114
4088 021530 000740                       20$:  BR    4$
4089
4090                                     :IF AN FP TRAP OCCURS COME HERE.
4091 021532 011602
4092 021534 022702 021350                 25$:  MOV    (SP),R2      ;GET ADDRESS OF TRAP.
4093 021540 001402                       CMP    #2$,R2          ;WAS THE TRAP DURING THE MULF INSTRUCTION?
4094 021542 000137 037450                 BEQ   26$              ;BRANCH IF YES.
4095
4096 021546 022626
4097 021550 010237 001236                 JMP    FPSPUR ;OTHERWISE GO REPORT A SPURIOUS
4098 021554 170204
4099 021556 170305                       ;FP TRAP.
4100 021560 012700 000200                 26$:  CMP    (SP)+,(SP)+ ;RESET THE STACK.
4101 021564 170100                       MOV    R2,$TMP2        ;SAVE DATA FOR ERROR REPORT.
4102 021566 012700 021640                 STFPS  R4              ;GET FPS.
4103 021572 174010                       STST  R5               ;GET FEC.
4104 021574 010537 001254                 MOV    #200,R0         ;SET FD MODE.
4105 021600 020561 000024                 LDFPS R0
4106 021604 001004                       MOV    #OVFNTT,R0     ;GET THE RESULT.
4107
4108 021606 010102
4109 021610 062702 000030                 STD   AC0,(R0)
4110
4111 021614 000112                       MOV    R5,$TMP11
4112
4113 021616 005761 000026                 27$:  CMP    R5,24(R1)
4114
4115
4116 021622 001003                       BNE   27$
4117
;SEE IF FAILURE IS ANTICIPATED
;FAILURE.
;BRANCH IF NOT ANTICIPATED.
;ERROR WAS ANTICIPATED SO RETURN
;TO THE ERROR REPORT IN THE CALLING
;ROUTINE.
;RESULT WAS NOT ANTICIPATED
;SO ERROR MUST BE REPORTED HERE.
;FIRST SEE IF ARGUMENTS SHOULD
;HAVE RESULTED IN OVERFLOW OR UNDER
;FLOW BY LOOKING AT THE FLAG.
;BRANCH IF UNDERFLOW EXPECTED.
;REPORT RESULT INCORRECT, EXPECTING
;OVERFLOW.
;REPORT RESULT INCORRECT, EXPECTING
;UNDERFLOW.
;IF AN FP TRAP OCCURS COME HERE.
;GET ADDRESS OF TRAP.
;WAS THE TRAP DURING THE MULF INSTRUCTION?
;BRANCH IF YES.
;OTHERWISE GO REPORT A SPURIOUS
;FP TRAP.
;RESET THE STACK.
;SAVE DATA FOR ERROR REPORT.
;GET FPS.
;GET FEC.
;SET FD MODE.
;GET THE RESULT.
;WAS THE FEC ANTICIPATED?
;BRANCH IF NOT ANTICIPATED.
;ERROR WAS ANTICIPATED SO
;RETURN TO THE ERROR REPORT OF THE
;CALLING ROUTINE.
;THE ERROR WAS NOT ANTICIPATED SO
;IT MUST BE REPORTED HERE. FIRST SEE IF EXPECTED
;OVERFLOW OR UNDER FLOW.
;BRANCH IF EXPECTING UNDERFLOW
    
```



4118  
4119 021624 104115 28\$: ERROR +115  
4120 021626 000161 000036 JMP 36(R1)  
4121  
4122 021632 29\$:  
4123 021632 104116 30\$: ERROR +116  
4124 021634 000161 000036 JMP 36(R1)  
4125  
4126 021640 000000 000000 000000 OVFNTT: .WORD 0,0,0,0  
4127  
4128 021650 IIIDONE:  
021650 104412 RSETUP

;REPORT TRAPPED ON OVERFLOW WITH FIV=0

;REPORT TRAPPED ON UNDER FLOW WITH FIU=0

;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?).

4129  
4130  
4131  
4140  
4141

;TEST TITLE:UNDER/OVERFLOW, USING MULD WITH TRAP DISABLED

4142

```
.SBTTL TEST # 13 - SEE COMMENT ABOVE FOR TEST TITLE
:*****
:*TEST 13 SEE COMMENT ABOVE FOR TEST TITLE
:*
:*THIS IS A TEST OF THE OVERFLOW AND UNDERFLOW CONDITIONS THAT CAN
:*ARRISE USING THE MULD INSTRUCTION WITH TRAPS DISABLED. A SUBROUTINE IS
:*USED TO SET UP THE OPERANDS, EXECUTE THE MULD INSTRUCTION AND
:*CHECK THE RESULTS.
:*
:*****
TST13: SCOPE
```

```
021652 000004
4143
4144
4145 021654
021654 104413
4146 021656 004737 022200
4147 021662 020200 000000
4148 021666 127272 000000
4149 021672 020000 000000 000000
4150 021702 000000 000000 000000
4151 021712 000000 000000
4152 021716 127272 000000
4153 021722 000200
4154 021724 000204
4155 021726 000012
4156 021730 177777
4157 021732 104131
4158 021734 000401
4159 021736 104132
4160 021740
```

```
:UNDERFLOW, EXPONENT OF RESULT=-129
JJJ1:
LPERR JSR PC,OVUNDNT ;SET UP THE LOOP ON ERROR ADDRESS.
1$: .WORD 20200,0 ;AC
2$: .WORD 127272,0
3$: .WORD 20000,0,0,0 ;FSRC
4$: .WORD 0,0,0,0 ;RES
5$: .WORD 0,0 ;ERROR RES.
6$: 200 ;FPS BEFORE EXECUTION.
7$: 204 ;FPS AFTER EXECUTION.
8$: 12 ;FEC
-1 ;FLAG
7$: ERROR +131 ;ST 331 TO 155 INTO 115 (BUT FIU)
BR 8$
8$: ERROR +132 ;ST 115 (BUT FD)
```

```
4161
4162
4163 021740
021740 104413
4164 021742 004737 022200
4165 021746 010200 000000
4166 021752 123456 000000
4167 021756 010000 000000 000000
4168 021766 000000 000000 000000
4169 021776 000000 000000 123456
4170 022006 005213
4171 022010 005204
4172 022012 000012
4173 022014 177777
4174 022016 104133
4175 022020 000401
4176 022022 104132
4177 022024
```

```
:UNDERFLOW, EXPONENT OF RESULT = -193
JJJ2:
LPERR JSR PC,OVUNDNT ;SET UP THE LOOP ON ERROR ADDRESS.
1$: .WORD 10200,0 ;AC
2$: .WORD 123456,0
3$: .WORD 10000,0,0,0 ;FSRC
4$: .WORD 0,0,0,0 ;RES
5$: .WORD 0,0,123456,0 ;ERROR RES
6$: 5213 ;FPS BEFORE EXECUTION.
7$: 5204 ;FPS AFTER EXECUTION.
8$: 12 ;FEC
-1 ;FLAG
7$: ERROR +133 ;SETTING FIUV OR FIV BAD.
BR 8$
8$: ERROR +132 ;ST 115 (BUT FD)
```

```
4178
4179
4180 022024
022024 104413
4181 022026 004737 022200
4182 022032 060200 000000
4183 022036 065432 000000
4184 022042 060000 000000 000000
4185 022052 000000 000000 000000
```

```
:OVERFLOW, EXPONENT OF RESULT = 128
JJJ3:
LPERR JSR PC,OVUNDNT ;SET UP THE LOOP ON ERROR ADDRESS.
1$: .WORD 60200,0 ;AC
2$: .WORD 65432,0
3$: .WORD 60000,0,0,0 ;FSRC
4$: .WORD 0,0,0,0 ;RES
```

```

4186 022062 000000 000000 065432 4$: .WORD 0,0,65432,0 ;ERROR RES.
4187 022072 000200 5$: 200 ;FPS BEFORE EXECUTION.
4188 022074 000206 206 ;FPS AFTER EXECUTION.
4189 022076 000010 6$: 10 ;FEC
4190 022100 000000 0 ;FLAG
4191 022102 104134 7$: ERROR +134 ;ST 333 TO 136 INTO 116 (BUT FIV)
4192 022104 000401 BR 8$ ;
4193 022106 104135 ERROR +135 ;ST 116 (BUT FD)
4194 022110 8$:
    
```

;OVERFLOW, EXPONENT OF RESULT = 130  
 JJJ4:

```

4198 022110 104413 LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
4199 022110 004737 022200 JSR PC,OVUNDNT
4200 022116 060200 000000 1$: .WORD 60200,0 ;AC
4201 022122 125252 000000 .WORD 125252,0
4202 022126 060200 000000 000000 2$: .WORD 60200,0,0,0 ;FSRC
4203 022136 000000 000000 000000 3$: .WORD 0,0,0,0 ;RES
4204 022146 000000 000000 125252 4$: .WORD 0,0,125252,0 ;ERROR RES.
4205 022156 006211 5$: 6211 ;FPS BEFORE EXECUTION.
4206 022160 006206 6206 ;FPS AFTER EXECUTION.
4207 022162 000010 6$: 10 ;FEC
4208 022164 000000 0 ;FLAG
4209 022166 104136 7$: ERROR +136 ;SETTING FIUV OR FIV BAD.
4210 022170 000401 BR 8$ ;
4211 022172 104135 ERROR +135 ;ST 116 (BUT FD)
4212 022174 000137 022610 8$: JMP JJJDONE ;GO TO NEXT TEST.
    
```

;THIS SUBROUTINE, OVUNDNT, 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:

```

ACARG: .WORD X,X,X,X ;AC OPERAND
FSRCARG: .WORD X,X,X,X ;FSRC 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
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 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 OVUNDNT RETURNS CONTROL  
 ;TO THE CALLING ROUTINE AT CONT. IF THE FPS IS BAD OVUNDNT  
 ;REPORTS THIS FAILURE AND THEN RETURNS TO CONT. IF THE RESULT OF THE  
 ;MULD 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 OVUNDNT  
 ;WILL TRANSFER CONTROL TO THE ERROR CALL AT ERR2. OTHERWISE THE

4213  
4214  
4215  
4216  
4217  
4218  
4219  
4220  
4221  
4222  
4223  
4224  
4225  
4226  
4227  
4228  
4229  
4230  
4231  
4232  
4233  
4234  
4235  
4236  
4237  
4238  
4239  
4240  
4241



```

4242 :RESULT WAS INCORRECT BUT WAS NOT ANTICIPATED AND OVUNDNT WILL
4243 :REPORT THE FAILURE AFTER WHICH CONTROL WILL BE PASSED TO CONT.
4244 :IF A TRAP OCCURS (IT SHOULD NOT) THEN OVUNDNT WILL READ THE FEC.
4245 :SHOULD THE FEC MATCH THE ANTICIPATED FEC OVUNDNT WILL
4246 :STORE ALL DATA AND TRANSFER CONTROL TO THE ERROR CALL AT ERR1. IF THE
4247 :FEC IS NOT THE SAME AS THE ANTICIPATED FEC OVUNDNT WILL REPORT
4248 :THE ERROR AND RETURN TO CONT. NOTE THAT OVUNDNT USES THE FLAG
4249 :TO TELL WHETHER OR NOT THESE PARTICULAR OPERANDS WILL RESULT IN
4250 :UNDERFLOW (FLAG=-1) OR OVERFLOW (FLAG=0).
4251
4252 022200 012601 000200 OVUNDNT: MOV (SP)+,R1 ;GET A POINTER TO THE ARGUMENTS.
4253 022202 012700 MOV #200,R0 ;SET FD MODE.
4254 022206 170100 LDFPS R0
4255
4256 022210 010100 MOV R1,R0 ;LOAD ACO, OPERAND.
4257 022212 172410 LDD (R0),ACO
4258
4259 022214 010102 MOV R1,R2 ;SAVE THE DATA PATTERNS IN CASE OF
4260 022216 010237 MOV R2,$TMP3 ;ERROR.
4261 022222 062702 ADD #10,R2
4262 022226 010237 MOV R2,$TMP4
4263 022232 062702 ADD #10,R2
4264 022236 010237 MOV R2,$TMP5
4265 022242 016137 MOV 42(R1),$TMP10
4266 022250 012737 MOV #OVDNTT,$TMP6
4267
4268 022256 016100 MOV 40(R1),R0 ;LOAD THE FPS.
4269 022262 170100 LDFPS R0
4270 022264 012737 MOV #1,$TMP2
4271 022272 012737 MOV #25$,FPVECT ;SET UP THE FP TRAP VECTOR IN CASE
4272 :OF ERROR.
4273 022300 010100 MOV R1,R0 ;COMPUTE THE ADDRESS OF FSRC.
4274 022302 062700 ADD #10,R0
4275
4276 022306 171010 1$: MULD (R0),ACO ;TEST INSTRUCTION.
4277
4278 022310 170204 2$: STFPS R4 ;GET FPS.
4279 022312 170305 STST R5 ;GET FEC.
4280 022314 012700 MOV #200,R0 ;SET FD MODE.
4281 022320 170100 LDFPS R0
4282 022322 012700 MOV #OVDNTT,R0 ;GET THE RESULT.
4283 022326 174010 STD ACO,(R0)
4284 022330 010437 MOV R4,$TMP7
4285 022334 010537 MOV R5,$TMP11
4286
4287 022340 012700 MOV #OVDNTT,R0 ;CHECK THE RESULT.
4288 022344 010102 MOV R1,R2
4289 022346 062702 ADD #20,R2
4290 022352 012703 MOV #4,R3
4291 022356 022022 3$: CMP (R0)+,(R2)+
4292 022360 001015 BNE 15$ ;BRANCH IF INCORRECT.
4293 022362 077303 SOB R3,3$
4294
4295 022364 026104 000042 CMP 42(R1),R4 ;WAS FPS CORRECT?
4296 022370 001002 BNE 10$ ;BRANCH IF FPS IS INCORRECT.
4297
4298 022372 000161 000056 4$: JMP 56(R1) ;RETURN, TEST COMPLETED.
    
```

```

4299
4300          :REPORT INCORRECT FPS.
4301 022376 005761 000046 10$:   TST    46(R1)   ;WAS THE RESULT OVER OR UNDER FLOW?
4302 022402 001002          BNE    12$   ;BRANCH IF UNDERFLOW.
4303
4304          :REPORT FPS BAD AFTER OVERFLOW.
4305 022404 104123 11$:   ERROR  +123
4306 022406 000771          BR     4$
4307
4308 022410 12$:
4309 022410 104124 13$:   ERROR  +124   ;REPORT FPS BAD AFTER UNDERFLOW.
4310 022412 000767          BR     4$
4311
4312          :RESULT INCORRECT.
4313 022414 012700 022600 15$:   MOV    #OVDNTT,R0   ;SEE IF FAILURE IS ANTICIPATED
4314 022420 010102          MOV    R1,R2         ;FAILURE.
4315 022422 062702 000030          ADD    #30,R2
4316 022426 012703 000004          MOV    #4,R3
4317 022432 022022 16$:   CMP    (R0)+,(R2)+
4318 022434 001007          BNE    17$
4319 022436 077303          SOB   R3,16$   ;BRANCH IF NOT ANTICIPATED.
4320
4321 022440 010102          MOV    R1,R2   ;ERROR WAS ANTICIPATED SO RETURN
4322 022442 062702 000054          ADD    #54,R2 ;TO THE ERROR REPORT IN THE CALLING
4323 022446 010237 001236          MOV    R2,$TMP2 ;ROUTINE.
4324 022452 000112          JMP   (R2)
4325
4326 022454 005761 000046 17$:   TST    46(R1)   ;RESULT WAS NOT ANTICIPATED
4327          :SO ERROR MUST BE REPORTED HERE.
4328          :FIRST SEE IF ARGUMENTS SHOULD
4329          :HAVE RESULTED IN OVERFLOW OR UNDER
4330          :FLOW BY LOOKING AT THE FLAG.
4331 022460 001002          BNE    19$   ;BRANCH IF UNDERFLOW EXPECTED.
4332
4333          :REPORT RESULT INCORRECT, EXPECTING
4334 022462 104125 18$:   ERROR  +125   ;OVERFLOW.
4335 022464 000742          BR     4$
4336
4337 022466 19$:
4338 022466 104126 20$:   ERROR  +126   ;REPORT RESULT INCORRECT, EXPECTING
4339 022470 000740          BR     4$   ;UNDERFLOW.
4340
4341          :IF AN FP TRAP OCCURS COME HERE.
4342 022472 011602 25$:   MOV    (SP),R2   ;GET ADDRESS OF TRAP.
4343 022474 022702 022310          CMP    #2$,R2   ;WAS THE TRAP DURING THE MULF INSTRUCTION?
4344 022500 001402          BEQ    26$     ;BRANCH IF YES.
4345 022502 000137 037450          JMP   FPSPUR ;OTHERWISE GO REPORT A SPURIOUS
4346          :FP TRAP.
4347 022506 022626 26$:   CMP    (SP)+,(SP)+ ;RESET THE STACK.
4348 022510 010237 001236          MOV    R2,$TMP2 ;SAVE DATA FOR ERROR REPORT.
4349 022514 170204          STFPS R4       ;GET FPS.
4350 022516 170305          STST  R5       ;GET FEC.
4351 022520 012700 000200          MOV    #200,R0  ;SET FD MODE.
4352 022524 170100          LDFPS R0
4353 022526 012700 022600          MOV    #OVDNTT,R0 ;GET THE RESULT.
4354 022532 174010          STD   ACO,(R0)
4355 022534 010537 001254          MOV    R5,$TMP11
    
```

4356	022540	020561	000044		CMP	R5,44(R1)		:WAS THE FEC ANTICIPATED?
4357	022544	001004			BNE	27\$		:BRANCH IF NOT ANTICIPATED.
4358								
4359	022546	010102			MOV	R1,R2		:ERROR WAS ANTICIPATED SO
4360	022550	062702	000050		ADD	#50,R2		:RETURN TO THE ERROR REPORT OF THE
4361								:CALLING ROUTINE.
4362	022554	000112			JMP	(R2)		
4363								
4364	022556	005761	000026	27\$:	TST	26(R1)		:THE ERROR WAS NOT ANTICIPATED SO
4365								:IT MUST BE REPORTED HERE. FIRST SEE IF EXPECTED
4366								:OVERFLOW OR UNDER FLOW.
4367	022562	001003			BNE	29\$		:BRANCH IF EXPECTING UNDERFLOW
4368								
4369								:REPORT TRAPPED ON OVERFLOW WITH FIV=0
4370	022564	104127		28\$:	ERROR	+127		
4371	022566	000161	000056		JMP	56(R1)		
4372								
4373	022572			29\$:				:REPORT TRAPPED ON UNDER FLOW WITH FIU=0
4374	022572	104130		30\$:	ERROR	+130		
4375	022574	000161	000056		JMP	56(R1)		
4376								
4377	022600	000000	000000	000000	OVDNTT:	.WORD	0,0,0,0	
4378								
4379	022610				JJJDONE:			
	022610	104412			RSETUP			: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?).

4380  
 4381  
 4382  
 4394  
 4395

:TEST TITLE:UNDER/OVERFLOW, USING MULF WITH TRAPS ENABLED



4396

```
.SBTTL TEST # 14 - SEE COMMENT ABOVE FOR TEST TITLE
:*****
:*TEST 14 SEE COMMENT ABOVE FOR TEST TITLE
:*
:*THIS IS A TEST OF THE UNDERFLOW AND OVERFLOW
:*CONDITIONS THAT CAN OCCUR USING THE MULF INSTRUCTION.
:*A SUBROUTINE IS CALLED TO SET UP THE OPERANDS,
:* EXECUTE THE MULF INSTRUCTION AND CHECK
:*THE RESULTS. HERE THE PARTICULAR INTERRUPT,
:*EITHER OVERFLOW OR UNDERFLOW, IS ENABLED SO A TRAP SHOULD
:*OCCUR.
:*
:*****
TST14: SCOPE
```

```
022612 000004
4397
4398
4399 022614
022614 104413
4400 022616 004737 023040
4401 022622 020123 045676
4402 022626 020200 000000
4403 022632 000123 045676
4404 022636 177777 177777
4405 022642 002000
4406 022644 102004
4407 022646 000012
4408 022650 177777
4409 022652 104145
4410 022654 000401
4411 022656 104144
4412 022660
4413
4414
4415 022660
022660 104413
4416 022662 004737 023040
4417 022666 010127 127272
4418 022672 010200 000000
4419 022676 060127 127272
4420 022702 177777 177777
4421 022706 007017
4422 022710 107000
4423 022712 000012
4424 022714 177777
4425 022716 104146
4426 022720 000401
4427 022722 104144
4428 022724
4429
4430
4431 022724
022724 104413
4432 022726 004737 023040
4433 022732 060252 125252
4434 022736 060000 000000
4435 022742 000052 125252
4436 022746 177777 177777
```

```
:UNDERFLOW, EXPONENT OF RESULT = -129
KKK1:
LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
JSR PC,OVUNFT
1$: .WORD 20123,45676 ;AC
2$: .WORD 20200,0 ;FSRC
3$: .WORD 123,45676 ;RES
4$: .WORD -1,-1 ;ERROR RES.
5$: 2000 ;FPS BEFORE EXECUTION.
102004 ;FPS AFTER EXECUTION.
6$: 12 ;FEC
-1 ;FLAG
7$: ERROR +145 ;ST 331 (BUT FIU) NO TRAP.
BR 8$
ERROR +144
8$:

:UNDERFLOW, EXPONENT OF THE RESULT = -193
KKK3:
LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
JSR PC,OVUNFT
1$: .WORD 10127,127272 ;AC
2$: .WORD 10200,0 ;FSRC
3$: .WORD 60127,127272 ;RES
4$: .WORD -1,-1 ;ERROR RES.
5$: 7017 ;FPS BEFORE EXECUTION.
107000 ;FPS AFTER EXECUTION.
6$: 12 ;FEC
-1
7$: ERROR +146 ;ST 137 (BUT FIU) NO TRAP.
BR 8$
ERROR +144
8$:

:OVERFLOW, EXPONENT OF THE RESULT = 128
KKK4:
LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
JSR PC,OVUNFT
1$: .WORD 60252,125252 ;AC
2$: .WORD 60000,0 ;FSRC
3$: .WORD 000052,125252 ;RES
4$: .WORD -1,-1 ;ERROR RES.
```





```

4493                                     ;IF NO TRAP OCCURS CONTROL IS PASSED TO ERR1.
4494                                     ;NOTE THAT OVUNFT USES THE FLAG
4495                                     ;TO TELL WHETHER OR NOT THESE PARTICULAR OPERANDS WILL RESULT IN
4496                                     ;UNDERFLOW (FLAG=-1) OR OVERFLOW (FLAG=0).
4497
4498 023040 012601 000200 OVUNFT: MOV (SP)+,R1 ;GET A POINTER TO THE ARGUMENTS.
4499 023042 012700 000200 MOV #200,R0 ;SET FD MODE.
4500 023046 170100 LDFPS R0
4501
4502 023050 010100 MOV R1,R0 ;LOAD ACO, OPERAND.
4503 023052 172410 LDD (R0),ACO
4504
4505 023054 010102 MOV R1,R2 ;SAVE THE DATA PATTERNS IN CASE OF
4506 023056 010237 001240 MOV R2,$TMP3 ;ERROR.
4507 023062 062702 000004 ADD #4,R2
4508 023066 010237 001242 MOV R2,$TMP4
4509 023072 062702 000004 ADD #4,R2
4510 023076 010237 001244 MOV R2,$TMP5
4511 023102 016137 000022 001252 MOV 22(R1),$TMP10
4512 023110 012737 023442 001246 MOV #OVFTT,$TMP6
4513
4514 023116 016100 000020 MOV 20(R1),R0 ;LOAD THE FPS.
4515 023122 170100 LDFPS R0
4516 023124 012737 023146 001236 MOV #1,$TMP2
4517 023132 012737 023156 000244 MOV #50$,FPVECT ;SET UP THE FP TRAP VECTOR IN CASE
4518 ;OF ERROR.
4519 023140 010100 MOV R1,R0 ;COMPUTE THE ADDRESS OF FSRC.
4520 023142 062700 000004 ADD #4,R0
4521
4522 023146 171010 1$: MULF (R0),ACO ;TEST INSTRUCTION. SHOULD CAUSE TRAP.
4523 023150 170000 2$: CFCC
4524
4525 023152 000137 023402 JMP 25$ ;FAILURE, NO TRAP.
4526
4527 023156 011602 50$: MOV (SP),R2 ;TRAP TO HERE AND SEE IF THE PC OF THE
4528 023160 020227 023150 CMP R2,#2$ ;TRAP WAS THAT OF THE MULF INSTRUCTION.
4529 023164 001402 BEQ 51$ ;BRANCH IF YES.
4530 023166 000137 037450 JMP FPSPUR ;OTHERWISE REPORT SPURIOUS FP ERROR.
4531
4532 023172 022626 51$: CMP (SP)+,(SP)+ ;RESET THE STACK
4533 023174 170204 STFPS R4 ;GET FPS.
4534 023176 170305 STST R5 ;GET FEC.
4535 023200 012700 000200 MOV #200,R0 ;SET FD MODE.
4536 023204 170100 LDFPS R0
4537 023206 012700 023442 MOV #OVFTT,R0 ;GET THE RESULT.
4538 023212 174010 STD ACO,(R0)
4539 023214 010437 001250 MOV R4,$TMP7
4540 023220 010537 001254 MOV R5,$TMP11
4541
4542 023224 012700 023442 MOV #OVFTT,R0 ;CHECK THE RESULT.
4543 023230 010102 MOV R1,R2
4544 023232 062702 000010 ADD #10,R2
4545 023236 012703 000002 MOV #2,R3
4546 023242 022022 3$: CMP (R0)+,(R2)+ ;BRANCH IF INCORRECT.
4547 023244 001027 BNE 15$
4548 023246 077303 SOB R3,3$
4549
    
```



4550	023250	026104	000022		CMP	22(R1),R4		:WAS FPS CORRECT?
4551	023254	001014			BNE	10\$		:BRANCH IF FPS IS INCORRECT.
4552								
4553	023256	026105	000024		CMP	24(R1),R5		:IS FEC CORRECT?
4554	023262	001002			BNE	5\$		:IF INCORRECT BRANCH.
4555	023264	000161	000036	4\$:	JMP	36(R1)		:RETURN, TEST COMPLETED.
4556								
4557					:REPORT	INCORRECT FEC.		
4558	023270	005761	000026	5\$:	TST	26(R1)		:WAS THE RESULT OVERFLOW OR UNDERFLOW?
4559	023274	001002			BNE	7\$		:BRANCH IF UNDERFLOW.
4560								
4561								:REPORT BAD FEC ON EXPECTED OVERFLOW.
4562	023276	104137		6\$:	ERROR	+137		
4563	023300	000771			BR	4\$		
4564								
4565	023302			7\$:				:REPORT BAD FEC ON EXPECTED UNDERFLOW.
4566	023302	104140		8\$:	ERROR	+140		
4567	023304	000767			BR	4\$		
4568								
4569					:REPORT	INCORRECT FPS.		
4570	023306	005761	000026	10\$:	TST	26(R1)		:WAS THE RESULT OVER OR UNDER FLOW?
4571	023312	001002			BNE	12\$		:BRANCH IF UNDERFLOW.
4572								
4573								:REPORT FPS BAD AFTER OVERFLOW.
4574	023314	104141		11\$:	ERROR	+141		
4575	023316	000762			BR	4\$		
4576								
4577	023320			12\$:				:REPORT FPS BAD AFTER UNDERFLOW.
4578	023320	104142		13\$:	ERROR	+142		
4579	023322	000760			BR	4\$		
4580								
4581					:RESULT	INCORRECT.		
4582	023324	012700	023442	15\$:	MOV	#OVFTT,R0		:SEE IF FAILURE IS ANTICIPATED
4583	023330	010102			MOV	R1,R2		:FAILURE.
4584	023332	062702	000014		ADD	#14,R2		
4585	023336	012703	000002		MOV	#2,R3		
4586	023342	022022		16\$:	CMP	(R0)+,(R2)+		
4587	023344	001007			BNE	17\$		:BRANCH IF NOT ANTICIPATED.
4588	023346	077303			SOB	R3,16\$		
4589								
4590	023350	010102			MOV	R1,R2		:ERROR WAS ANTICIPATED SO RETURN
4591	023352	062702	000034		ADD	#34,R2		:TO THE ERROR REPORT IN THE CALLING
4592	023356	010237	001236		MOV	R2,\$TMP2		:ROUTINE.
4593	023362	000112			JMP	(R2)		
4594								
4595	023364	005761	000026	17\$:	TST	26(R1)		:RESULT WAS NOT ANTICIPATED
4596								:SO ERROR MUST BE REPORTED HERE.
4597								:FIRST SEE IF ARGUMENTS SHOULD
4598								:HAVE RESULTED IN OVERFLOW OR UNDER
4599								:FLOW BY LOOKING AT THE FLAG.
4600	023370	001002			BNE	19\$		:BRANCH IF UNDERFLOW EXPECTED.
4601								
4602								:REPORT RESULT INCORRECT, EXPECTING
4603	023372	104143		18\$:	ERROR	+143		:OVERFLOW.
4604	023374	000733			BR	4\$		
4605								
4606	023376			19\$:				:REPORT RESULT INCORRECT, EXPECTING

```
4607 023376 104144          20$:  ERROR  +144          :UNDERFLOW.
4608 023400 000731          BR      4$
4609
4610          :IF NO FP TRAP OCCURS COME HERE.
4611 023402 170204          25$:  STFPS  R4          :GET FPS.
4612 023404 170305          STST   R5          :GET FEC.
4613 023406 012700 000200  MOV    #200,R0     :SET FD MODE.
4614 023412 170100          LDFPS  R0
4615 023414 012700 023442  MOV    #OVFTT,R0   :GET THE RESULT.
4616 023420 174010          STD    ACO,(R0)
4617 023422 010437 001250  MOV    R4,$TMP7
4618 023426 010537 001254  MOV    R5,$TMP11
4619 023432 010102          MOV    R1,R2
4620 023434 062702 000030  ADD    #30,R2     :ERROR WAS ANTICIPATED SO
4621          :RETURN TO THE ERROR REPORT OF THE
4622 023440 000112          JMP    (R2)       :CALLING ROUTINE.
4623
4624 023442 000000 000000 000000  OVFTT: .WORD 0,0,0,0
4625
4626 023452 104412          KKKDONE:
          RSETUP
          :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?).

4627
4628
4629
4637
4638          :TEST TITLE:UNDER/OVERFLOW, USING MULD WITH TRAPS ENABLED
```

4639

```
.SBTTL TEST # 15 - SEE COMMENT ABOVE FOR TEST TITLE
:*****
:*TEST 15 SEE COMMENT ABOVE FOR TEST TITLE
:*
:*THIS IS A TEST OF THE OVER FLOW AND UNDER FLOW CONDITIONS USING THE
:*MULD INSTRUCTION WITH TRAPS ENABLED. A SUBROUTINE IS USED TO SET UP
:*THE OPERANDS, EXECUTE THE MULD INSTRUCTION AND CHECK THE RESULTS.
:*
:*****
```

```
023454 000004
4640
4641
4642 023456
4643 023460 104413 024002
4644 023464 004737 125252
4645 023470 020052 125252
4646 023474 125252 125252
4647 023504 020300 000000 000000
4648 023504 000177 177777 177777
4649 023514 000177 177777
4650 023520 125252 125252
4651 023524 002200
4652 023526 102204
4653 023530 000012
4654 023532 177777
4655 023534 104157
4656 023536 000401
4657 023540 104160
4658
4659
```

```
TST15: SCOPE
:UNDERFLOW, EXPONENT OF RESULT = -129
LLL1:
LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
JSR PC,OVUNDT
1$: .WORD 20052,125252 ;AC
2$: .WORD 125252,125252 ;FSRC
3$: .WORD 20300,0,0,0 ;RES
4$: .WORD 177,-1,-1,-1 ;ERROR RES.
5$: .WORD 125252,125252
6$: 2200 ;FPS BEFORE EXECUTION.
102204 ;FPS AFTER EXECUTION.
7$: 12 ;FEC
-1 ;FLAG
8$: ERROR +157 ;ST 331 (BUT FIU) NO TRAP.
BR 8$
ERROR +160 ;ST 155 (BUT FD)
8$:
```

```
4660 023542
4661 023542 104413
4662 023544 004737 024002
4663 023550 010327 127272
4664 023554 036363 045454
4665 023560 010000 000000 000000
4666 023570 060127 127272
4667 023574 036363 045454
4668 023600 177777 177777
4669 023610 007217
4670 023612 107200
4671 023614 000012
4672 023616 177777
4673 023620 104161
4674 023622 000401
4675 023624 104155
4676
4677
```

```
:UNDERFLOW, EXPONENT OF THE RESULT = -193
LLL2:
LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
JSR PC,OVUNDT
1$: .WORD 10327,127272 ;AC
2$: .WORD 36363,45454 ;FSRC
3$: .WORD 10000,0,0,0 ;RES
4$: .WORD 60127,127272
5$: .WORD 36363,45454
6$: .WORD -1,-1,-1,-1 ;ERROR RES.
7$: 7217 ;FPS BEFORE EXECUTION.
107200 ;FPS AFTER EXECUTION.
8$: 12 ;FEC
-1 ;FLAG
9$: ERROR +161 ;ST 137 (BUT FIU) NO TRAP.
BR 8$
ERROR +156
8$:
```

```
4678 023626
4679 023626 104413
4680 023630 004737 024002
4681 023634 060252 125252
4682 023640 125252 125252
4683 023644 160100 000000 000000
4683 023654 100177 177777 177777
```

```
:OVERFLOW, EXPONENT OF THE RESULT = 128
LLL3:
LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
JSR PC,OVUNDT
1$: .WORD 60252,125252 ;AC
2$: .WORD 125252,125252 ;FSRC
3$: .WORD 160100,0,0,0 ;FSRC
4$: .WORD 100177,-1,-1,-1 ;RES
```





```

4720      :
4721      :
4722      :
4723      :
4724      :
4725      :
4726      :
4727      :
4728      :
4729      :
4730      :
4731      :
4732      :
4733      :
4734      :
4735      :
4736      :
4737      :
4738      :
4739      :
4740      :
4741      :
4742      :
4743      :
4744      :
4745      :
4746      :
4747      :
4748      :
4749      :
4750 024002 012601      :
4751 024004 012700 000200 :
4752 024010 170100      :
4753      :
4754 024012 010100      :
4755 024014 172410      :
4756      :
4757 024016 010102      :
4758 024020 010237 001240 :
4759 024024 062702 000010 :
4760 024030 010237 001242 :
4761 024034 062702 000010 :
4762 024040 010237 001244 :
4763 024044 016137 000042 001252 :
4764 024052 012737 024404 001246 :
4765      :
4766 024060 016100 000040 :
4767 024064 170100      :
4768 024066 012737 024110 001236 :
4769 024074 012737 024120 000244 :
4770      :
4771 024102 010100      :
4772 024104 062700 000010 :
4773      :
4774 024110 171010      :
4775 024112 170000      :
4776      :

```

```

FSRCARG: .WORD X,X,X,X      :FSRC 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
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      R1,R2        :SAVE DATA PATTERNS IN CASE OF
        MOV      R2,$TMP3     :ERROR.
        ADD     #10,R2
        MOV      R2,$TMP4
        ADD     #10,R2
        MOV      R2,$TMP5
        MOV      42(R1),$TMP10
        MOV      #OVDTT,$TMP6
        MOV      40(R1),R0     :LOAD THE FPS.
        LDFPS   R0
        MOV      #1,$TMP2
        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

```

```

4777 024114 000137 024344          JMP      25$          ;FAILURE, NO TRAP.
4778
4779 024120 011602          50$:  MOV      (SP),R2      ;TRAP TO HERE AND SEE IF THE PC OF THE
4780 024122 020227 024112          CMP      R2,#2$        ;TRAP WAS THAT OF THE MULF INSTRUCTION.
4781 024126 001402          BEQ      51$          ;BRANCH IF YES.
4782 024130 000137 037450          JMP      FPSPUR ;OTHERWISE REPORT SPURIOUS FP ERROR.
4783
4784 024134 022626          51$:  CMP      (SP)+,(SP)+  ;RESET THE STACK
4785 024136 170204          STFPS   R4            ;GET FPS.
4786 024140 170305          STST    R5            ;GET FEC.
4787 024142 012700 000200          MOV      #200,R0      ;SET FD MODE.
4788 024146 170100          LDFPS   R0
4789 024150 012700 024404          MOV      #OVDTT,R0    ;GET THE RESULT.
4790 024154 174010          STD     ACO,(R0)
4791 024156 010437 001250          MOV      R4,$TMP7
4792 024162 010537 001254          MOV      R5,$TMP11
4793
4794 024166 012700 024404          MOV      #OVDTT,R0    ;CHECK THE RESULT.
4795 024172 010102          MOV      R1,R2
4796 024174 062702 000020          ADD     #20,R2
4797 024200 012703 000004          MOV      #4,R3
4798 024204 022022          3$:  CMP      (R0)+,(R2)+  ;BRANCH IF INCORRECT.
4799 024206 001027          BNE     15$
4800 024210 077303          SOB     R3,3$
4801
4802 024212 026104 000042          CMP     42(R1),R4    ;WAS FPS CORRECT?
4803 024216 001014          BNE     10$          ;BRANCH IF FPS IS INCORRECT.
4804
4805 024220 026105 000044          CMP     44(R1),R5    ;IS FEC CORRECT?
4806 024224 001002          BNE     5$          ;IF INCORRECT BRANCH.
4807 024226 000161 000056          4$:  JMP     56(R1)      ;RETURN. TEST COMPLETED.
4808
4809          ;REPORT INCORRECT FEC.
4810 024232 005761 000046          5$:  TST     46(R1)      ;WAS THE RESULT OVERFLOW OR UNDERFLOW?
4811 024236 001002          BNE     7$          ;BRANCH IF UNDERFLOW.
4812
4813          ;REPORT BAD FEC ON EXPECTED OVERFLOW.
4814 024240 104151          6$:  ERROR   +151
4815 024242 000771          BR     4$
4816
4817 024244          ;REPORT BAD FEC ON EXPECTED UNDERFLOW.
4818 024244 104152          7$:  ERROR   +152
4819 024246 000767          8$:  BR     4$
4820
4821          ;REPORT INCORRECT FPS.
4822 024250 005761 000046          10$: TST     46(R1)     ;WAS THE RESULT OVER OR UNDER FLOW?
4823 024254 001002          BNE     12$          ;BRANCH IF UNDERFLOW.
4824
4825          ;REPORT FPS BAD AFTER OVERFLOW.
4826 024256 104153          11$: ERROR   +153
4827 024260 000762          BR     4$
4828
4829 024262          ;REPORT FPS BAD AFTER UNDERFLOW.
4830 024262 104154          12$: ERROR   +154
4831 024264 000760          13$: BR     4$
4832
4833          ;RESULT INCORRECT.

```



```

4834 024266 012700 024404      15$:  MOV    #OVDTT,R0      ;SEE IF FAILURE IS ANTICIPATED
4835 024272 010102              MOV    R1,R2          ;FAILURE.
4836 024274 062702 000030      ADD    #30,R2
4837 024300 012703 000004      MOV    #4,R3
4838 024304 022022      16$:  CMP    (R0)+,(R2)+    ;BRANCH IF NOT ANTICIPATED.
4839 024306 001007          BNE    17$
4840 024310 077303          SOB    R3,16$
4841
4842 024312 010102              MOV    R1,R2          ;ERROR WAS ANTICIPATED SO RETURN
4843 024314 062702 000054      ADD    #54,R2        ;TO THE ERROR REPORT IN THE CALLING
4844 024320 010237 001236      MOV    R2,$TMP2     ;ROUTINE.
4845 024324 000112          JMP    (R2)
4846
4847 024326 005761 000046      17$:  TST    46(R1)      ;RESULT WAS NOT ANTICIPATED
4848                          ;SO ERROR MUST BE REPORTED HERE.
4849                          ;FIRST SEE IF ARGUMENTS SHOULD
4850                          ;HAVE RESULTED IN OVERFLOW OR UNDER
4851                          ;FLOW BY LOOKING AT THE FLAG.
4852 024332 001002          BNE    19$          ;BRANCH IF UNDERFLOW EXPECTED.
4853
4854                          ;REPORT RESULT INCORRECT, EXPECTING
4855 024334 104155      18$:  ERROR  +155        ;OVERFLOW.
4856 024336 000733          BR     4$
4857
4858 024340      19$:
4859 024340 104156      20$:  ERROR  +156        ;REPORT RESULT INCORRECT, EXPECTING
4860 024342 000731          BR     4$          ;UNDERFLOW.
4861
4862                          ;IF NO FP TRAP OCCURS COME HERE.
4863 024344 170204      25$:  STFPS  R4          ;GET FPS.
4864 024346 170305          STST  R5          ;GET FEC.
4865 024350 012700 000200      MOV    #200,R0      ;SET FD MODE.
4866 024354 170100          LDFPS R0
4867 024356 012700 024404      MOV    #OVDTT,R0    ;GET THE RESULT.
4868 024362 174010          STD   ACO,(R0)
4869 024364 010437 001250      MOV    R4,$TMP7
4870 024370 010537 001254      MOV    R5,$TMP11
4871 024374 010102          MOV    R1,R2
4872 024376 062702 000050      ADD    #50,R2
4873                          ;ERROR WAS ANTICIPATED SO
4874 024402 000112          JMP    (R2)        ;RETURN TO THE ERROR REPORT OF THE
4875                          ;CALLING ROUTINE.
4876 024404 000000 000000 000000 OVDTT: .WORD 0,0,0,0
4877
4878 024414      LLLDONE:
         024414 104412          RSETUP
                          ;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 CONTRGL G?).

4879
4880
4881
4882
4890
    
```

4891

```
.SBTTL TEST # 16 - MODF TEST  
:*****  
:*TEST 16 MODF TEST  
:*  
:*THIS IS A TEST OF THE MODF INSTRUCTION, WHICH MAKES USE OF  
:*A SUBROUTINE TO SET UP THE OPERANDS, EXECUTE THE MODF INSTRUCTION  
:*AND CHECK THE RESULTS.  
:*  
:*****
```

```
024416 000004  
4892  
4893  
4894 024420  
024420 104413  
4895 024422 004737 025504  
4896 024426 000000 000000  
4897 024432 000000 000000  
4898 024436 000000 000000  
4899 024442 000000 000000  
4900 024446 177777 177777  
4901 024452 177777 177777  
4902 024456 000013  
4903 024460 000004  
4904 024462 104056  
4905 024464 000401  
4906 024466 104057  
4907 024470  
4908  
4909  
4910 024470  
024470 104413  
4911 024472 004737 025504  
4912 024476 123456 076543  
4913 024502 000000 000000  
4914 024506 000000 000000  
4915 024512 000000 000000  
4916 024516 123456 076543  
4917 024522 177777 177777  
4918 024526 000000  
4919 024530 000004  
4920 024532 104056  
4921 024534 000401  
4922 024536 104057  
4923 024540  
4924  
4925  
4926 024540  
024540 104413  
4927 024542 004737 025504  
4928 024546 000000 000000  
4929 024552 076543 021234  
4930 024556 000000 000000  
4931 024562 000000 000000  
4932 024566 000000 000000  
4933 024572 177777 177777  
4934 024576 000003  
4935 024600 000004
```

```
TST16: SCOPE  
:MODF WITH (FSRC=AC=0)  
GGG1:  
LPERR JSR PC,MODFSUB ;SET UP THE LOOP ON ERROR ADDRESS.  
1$: .WORD 0,0 ;AC  
2$: .WORD 0,0 ;FSRC  
3$: .WORD 0,0 ;FRACTIONAL RES.  
4$: .WORD 0,0 ;INTEGER RES.  
5$: .WORD -1,-1 ;ERROR FRACTIONAL RES.  
6$: .WORD -1,-1 ;ERROR INGETER RES.  
7$: 13 ;FPS BEFORE EXECUTION.  
4 ;FPS AFTER EXECUTION.  
8$: ERROR +56 ;STORE SINGLE ZERO BAD.  
BR 9$  
ERROR +57 ;AC V 1 <= ZERO FAILED.  
9$:  
:MODF TEST, WITH (FSRC=0)  
GGG2:  
LPERR JSR PC,MODFSUB ;SET UP THE LOOP ON ERROR ADDRESS.  
1$: .WORD 123456,76543 ;AC  
2$: .WORD 0,0 ;FSRC  
3$: .WORD 0,0 ;FRACTIONAL RES.  
4$: .WORD 0,0 ;INTEGER RESULT.  
5$: .WORD 123456,76543 ;ERROR FRACTIONAL RES.  
6$: .WORD -1,-1 ;ERROR INTEGER RES.  
7$: 0 ;FPS BEFORE EXECUTION.  
4 ;FPS AFTER EXECUTION.  
8$: ERROR +56 ;STORE ZERO FAILURE.  
BR 9$  
ERROR +57  
9$:  
:MODF TEST WITH (AC=0)  
GGG3:  
LPERR JSR PC,MODFSUB ;SET UP THE LOOP ON ERROR ADDRESS.  
1$: .WORD 0,0 ;AC  
2$: .WORD 76543,21234 ;FSRC  
3$: .WORD 0,0 ;FRACTIONAL RES.  
4$: .WORD 0,0 ;INTEGER RES.  
5$: .WORD 0,0 ;ERROR FRACTIONAL RES.  
6$: .WORD -1,-1 ;ERROR INTEGER RES.  
7$: 3 ;FPS BEFORE EXECUTION.  
4 ;FPS AFTER EXECUTION.
```

```

4936 024602 104053      8$:      ERROR      +53      ;RES.BAD
4937 024604 000401      BR          9$
4938 024606 104057      ERROR      +57
4939 024610
4940
4941
4942 024610      ;MODF TEST WITH EXPONENT OF THE RESULT = 25
GGG4:      LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
           JSR          PC,MODFSUB
4943 024612 004737 025504      .WORD      46252,125252      ;AC
4944 024616 046252 125252      1$:      .WORD      40300,0      ;FSRC
4945 024622 040300 000000      2$:      .WORD      0,0      ;FRACTIONAL RES.
4946 024626 000000 000000      3$:      .WORD      0,0      ;INTEGER RES.
4947 024632 046377 177777      4$:      .WORD      46377,-1      ;ERROR FRACTIONAL RES.
4948 024636 046252 125252      5$:      .WORD      46252,125252      ;ERROR INTEGER RES.
4949 024642 040300 000000      6$:      .WORD      40300,0      ;FPS BEFORE EXECUTION.
4950 024646 000013      7$:      13      ;FPS AFTER EXECUTION.
4951 024650 000004      4
4952 024652 104053      8$:      ERROR      +53      ;ST 134
4953 024654 000401      BR          9$
4954 024656 104060      ERROR      +60
4955 024660
4956
4957
4958 024660      ;MODF TEST WITH EXPONENT OF THE RESULT = 127
GGG5:      LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
           JSR          PC,MODFSUB
4959 024662 004737 025504      .WORD      77652,125252      ;AC
4960 024666 077652 125252      1$:      .WORD      40300,0      ;FSRC
4961 024672 040300 000000      2$:      .WORD      0,0      ;FRACTIONAL RES.
4962 024676 000000 000000      3$:      .WORD      0,0      ;INTEGER RES.
4963 024702 077777 177777      4$:      .WORD      77777,-1      ;ERROR FRACTIONAL RES.
4964 024706 077652 125252      5$:      .WORD      77652,125252      ;ERROR INTEGER RES.
4965 024712 040300 000000      6$:      .WORD      40300,0      ;FPS BEFORE EXECUTION.
4966 024716 000000      7$:      0      ;FPS AFTER EXECUTION.
4967 024720 000004      4
4968 024722 104053      8$:      ERROR      +53
4969 024724 000401      BR          9$
4970 024726 104060      ERROR      +60
4971 024730
4972
4973
4974 024730      ;MODF TEST WITH EXPONENT OF RESULT = 25
GGG6:      LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
           JSR          PC,MODFSUB
4975 024732 004737 025504      .WORD      46200,1      ;AC
4976 024736 046200 000001      1$:      .WORD      40340,0      ;FSRC
4977 024742 040340 000000      2$:      .WORD      0,0      ;FRACTIONAL RES.
4978 024746 000000 000000      3$:      .WORD      0,0      ;INTEGER RES.
4979 024752 046340 000001      4$:      .WORD      46340,1      ;ERROR FRACTIONAL RES.
4980 024756 040000 000000      5$:      .WORD      40000,0      ;ERROR INTEGER RES.
4981 024762 177777 177777      6$:      .WORD      -1,-1      ;FPS BEFORE EXECUTION.
4982 024766 000013      7$:      13      ;FPS AFTER EXECUTION.
4983 024770 000004      4
4984 024772 104061      8$:      ERROR      +61
4985
4986 024774 000401      BR          9$
4987 024776 104054      ERROR      +54
4988 025000
4989

```



```

4990                                     :MODF TEST WITH EXPONENT OF THE RESULT = 24
4991 025000                               GGG7:
      025000 104413                       LPERR
4992 025002 004737 025504                 JSR PC,MODFSUB ;SET UP THE LOOP ON ERROR ADDRESS.
4993 025006 046000 000001                 1$: .WORD 46000,1 ;AC
4994 025012 040340 000000                 2$: .WORD 40340,0 ;FSRC
4995 025016 040100 000000                 3$: .WORD 40100,0 ;FRACTIONAL RES.
4996 025022 046140 000001                 4$: .WORD 46140,1 ;INTEGER RESULT.
4997 025026 000000 000000                 5$: .WORD 0,0 ;ERROR FRACTIONAL RES.
4998 025032 177777 177777                 6$: .WORD -1,-1 ;ERROR INTEGER RES.
4999 025036 000000                       7$: 0 ;FPS BEFORE EXECUTION.
5000 025040 000000                       8$: 0 ;FPS AFTER EXECUTION.
5001 025042 104062                       8$: ERROR +62 ;BAD CONSTANT USED (NOT 24)
5002                                     ;OR ST 525 TO 150 INTO 050
5003 025044 000401                       BR 9$
5004 025046 104054                       ERROR +54
5005 025050                               9$:
5006
5007                                     :MODF TEST WITH EXPONENT OF THE RESULT = 10
5008 025050                               GGG8:
      025050 104413                       LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
5009 025052 004737 025504                 JSR PC,MODFSUB
5010 025056 042577 177777                 1$: .WORD 42577,-1 ;AC
5011 025062 040200 000000                 2$: .WORD 40200,0 ;FSRC
5012 025066 040177 176000                 3$: .WORD 40177,176000 ;FRACTIONAL RES.
5013 025072 042577 140000                 4$: .WORD 42577,140000 ;INTEGER RES.
5014 025076 177777 177777                 5$: .WORD -1,-1 ;ERROR FRACTIONAL RES.
5015 025102 177777 177777                 6$: .WORD -1,-1 ;ERROR INTEGER RES.
5016 025106 000000                       7$: 0 ;FPS BEFORE EXECUTION.
5017 025110 000000                       8$: 0 ;FPS AFTER EXECUTION.
5018 025112 104053                       8$: ERROR +53
5019 025114 000401                       BR 9$
5020 025116 104054                       ERROR +54
5021 025120                               9$:
5022
5023                                     :MODF TEST WITH THE EXPONENT OF THE RESULT = 10
5024 025120                               GGG9:
      025120 104413                       LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
5025 025122 004737 025504                 JSR PC,MODFSUB
5026 025126 042577 140001                 1$: .WORD 42577,140001 ;AC
5027 025132 040200 000000                 2$: .WORD 40200,0 ;FSRC
5028 025136 034600 000000                 3$: .WORD 34600,0 ;FRACTIONAL RES.
5029 025142 042577 140000                 4$: .WORD 42577,140000 ;INTEGER RES.
5030 025146 000000 000000                 5$: .WORD 0,0 ;ERROR FRACTIONAL RES.
5031 025152 177777 177777                 6$: .WORD -1,-1 ;ERROR INTEGER RES.
5032 025156 000000                       7$: 0 ;FPS BEFORE EXECUTION.
5033 025160 000000                       8$: 0 ;FPS AFTER EXECUTION.
5034 025162 104063                       8$: ERROR +63 ;ST 532 TO 122 INTO NORMALIZE.
5035 025164 000401                       BR 9$
5036 025166 104054                       ERROR +54
5037 025170                               9$:
5038
5039                                     :MODF TEST WITH EXPONENT OF THE RESULT = 9
5040 025170                               GGG10:
      025170 104413                       LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
5041 025172 004737 025504                 JSR PC,MODFSUB
5042 025176 042377 100000                 1$: .WORD 42377,100000 ;AC

```

5043 025202 040200 000000  
 5044 025206 000000 000000  
 5045 025212 042377 100000  
 5046 025216 177777 177777  
 5047 025222 177777 177777  
 5048 025226 000013  
 5049 025230 000004  
 5050 025232 104053  
 5051 025234 000401  
 5052 025236 104054  
 5053 025240  
 5054  
 5055  
 5056 025240  
 025240 104413  
 5057 025242 004737 025504  
 5058 025246 040177 177777  
 5059 025252 040200 000000  
 5060 025256 040177 177777  
 5061 025262 000000 000000  
 5062 025266 000000 000000  
 5063 025272 040177 177777  
 5064 025276 000017  
 5065 025300 000000  
 5066 025302 104064  
 5067 025304 000401  
 5068 025306 104064  
 5069 025310  
 5070  
 5071  
 5072 025310  
 025310 104413  
 5073 025312 004737 025504  
 5074 025316 034377 177777  
 5075 025322 040200 000000  
 5076 025326 034377 177777  
 5077 025332 000000 000000  
 5078 025336 000000 000000  
 5079 025342 034377 177777  
 5080 025346 000000  
 5081 025350 000000  
 5082 025352 104064  
 5083 025354 000401  
 5084 025356 104064  
 5085 025360  
 5086  
 5087  
 5088 025360  
 025360 104413  
 5089 025362 004737 025504  
 5090 025366 020000 000001  
 5091 025372 040300 000000  
 5092 025376 020100 000002  
 5093 025402 000000 000000  
 5094 025406 020100 000001  
 5095 025412 000000 000000  
 5096 025416 000000

```

2$: .WORD 40200,0      :FSRC
3$: .WORD 0,0          :FRACTIONAL RES.
4$: .WORD 42377,100000 :INTEGER RES.
5$: .WORD -1,-1       :ERROR FRACTIONAL RES.
6$: .WORD -1,-1       :ERROR INTEGER RES.
7$: 13                 :FPS BEFORE EXECUTION.
      4                 :FPS AFTER EXECUTION.
8$: ERROR +53
      BR 9$
      ERROR +54
9$:

:MODF TEST WITH EXPONENT OF THE RESULT = 0
GGG11:
      LPERR                :SET UP THE LOOP ON ERROR ADDRESS.
      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.
5$: .WORD 0,0          :ERROR FRACTIONAL RES.
6$: .WORD 40177,-1      :ERROR INTEGER RES.
7$: 17                 :FPS BEFORE EXECUTION.
      0                 :FPS AFTER EXECUTION.
8$: ERROR +64
      BR 9$
      ERROR +64
9$:

:MODF TEST WITH EXPONENT OF THE RESULT = -15
GGG12:
      LPERR                :SET UP THE LOOP ON ERROR ADDRESS.
      JSR PC,MODFSUB
1$: .WORD 34377,-1      :AC
2$: .WORD 40200,0      :FSRC
3$: .WORD 34377,-1      :FRACTIONAL RES.
4$: .WORD 0,0          :INTEGER RES.
5$: .WORD 0,0          :ERROR FRACTIONAL RES.
6$: .WORD 34377,-1      :ERROR INTEGER RES.
7$: 0                 :FPS BEFORE EXECUTION.
      0                 :FPS AFTER EXECUTION.
8$: ERROR +64
      BR 9$
      ERROR +64
9$:

:MODF TEST WITH EXPONENT OF RESULT = -64, IN ROUND MODE
GGG13:
      LPERR                :SET UP THE LOOP ON ERROR ADDRESS.
      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.
5$: .WORD 20100,1      :ERROR FRACTIONAL RES.
6$: .WORD 0,0          :ERROR INTEGER RES.
7$: 0                 :FPS BEFORE EXECUTION.

```



5097 025420 000000  
5098 025422 104065  
5099 025424 000401  
5100 025426 104054  
5101 025430  
5102  
5103  
5104 025430 104413  
5105 025432 004737 025504  
5106 025436 142777 170000  
5107 025442 040200 000000  
5108 025446 140000 000000  
5109 025452 142777 160000  
5110 025456 040000 000000  
5111 025462 042777 160000  
5112 025466 000007  
5113 025470 000010  
5114 025472 104066  
5115 025474 000401  
5116 025476 104067  
5117 025500 000137 026072  
5118  
5119  
5120  
5121  
5122  
5123  
5124  
5125  
5126  
5127  
5128  
5129  
5130  
5131  
5132  
5133  
5134  
5135  
5136  
5137  
5138  
5139  
5140  
5141  
5142  
5143  
5144  
5145  
5146  
5147  
5148  
5149  
5150  
5151  
5152

```

0                               :FPS AFTER EXECUTION.
8$: ERROR +65                   :ROUND TRUNK, ST 126 INTO ROUND.
   BR 9$
   ERROR +54
9$:
:MODF TEST WITH EXPONENT OF RESULT = 11
GGG14:
   LPERR                         :SET UP THE LOOP ON ERROR ADDRESS.
   JSR PC,MODFSUB
1$: .WORD 142777,170000          :AC
2$: .WORD 40200,0                :FSRC
3$: .WORD 140000,0              :FRACTIONAL RES.
4$: .WORD 142777,160000         :INTEGER RES.
5$: .WORD 40000,0               :ERROR FRACTIONAL RES.
6$: .WORD 42777,160000         :ERROR INTEGER RES.
7$: 7                            :FPS BEFORE EXECUTION.
   10                            :FPS AFTER EXECUTION.
8$: ERROR +66                   :SIGN OF FRACTION.
   BR 9$
   ERROR +67                   :SIGN OF INTEGER.
9$: JMP GGGDONE                 :GO TO NEXT TEST.

```

```

: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
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
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 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 MODFSUB 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 MODFSUB 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, MODFSUB WILL RETURN CONTROL TO THE ERROR
:CALL AT ERR2.

```



5153	025504	012601			MODFSUB:	MOV	(SP)+,R1		;GET A POINTER TO THE ARGUMENTS
5154	025506	012700	000200			MOV	#200,R0		;SET FD MODE.
5155	025512	170100				LDFPS	R0		
5156	025514	010100				MOV	R1,R0		;SET UP ACO
5157	025516	172410				LDD	(R0),ACO		
5158	025520	012700	026062			MOV	#MODP1,R0		;PUT A BACKGROUND PATTERN INTO AC1.
5159	025524	172510				LDD	(R0),AC1		
5160	025526	016100	000030			MOV	30(R1),R0		;SET UP THE FPS.
5161	025532	170100				LDFPS	R0		
5162	025534	012737	025550	001236		MOV	#1\$,STMP2		;COMPUTE THE ADDRESS OF THE FSRC.
5163	025542	010100				MOV	R1,R0		
5164	025544	062700	000004			ADD	#4,R0		
5165									
5166	025550	171410			1\$:	MODF	(R0),ACO		;EXECUTE THE TEST INSTRUCTION.
5167									
5168	025552	170204				STFPS	R4		;GET THE FPS.
5169	025554	012700	000200			MOV	#200,R0		;SET FD MODE.
5170	025560	170100				LDFPS	R0		
5171	025562	012700	026042			MOV	#MODFT0,R0		;GET THE FRACTIONAL RESULT.
5172	025566	174010				STD	ACO,(R0)		
5173	025570	012700	026052			MOV	#MODFT1,R0		;GET THE INTEGER RESULT.
5174	025574	174110				STD	AC1,(R0)		
5175									
5176	025576	010102				MOV	R1,R2		;SAVE THE DATA IN CASE OF ERROR.
5177	025600	010237	001240			MOV	R2,\$TMP3		
5178	025604	062702	000004			ADD	#4,R2		
5179	025610	010237	001242			MOV	R2,\$TMP4		
5180	025614	062702	000004			ADD	#4,R2		
5181	025620	010237	001244			MOV	R2,\$TMP5		
5182	025624	062702	000004			ADD	#4,R2		
5183	025630	010237	001246			MOV	R2,\$TMP6		
5184	025634	012737	026042	001250		MOV	#MODFT0,\$TMP7		
5185	025642	012737	026052	001252		MOV	#MODFT1,\$TMP10		
5186	025650	010437	001254			MOV	R4,\$TMP11		
5187	025654	016137	000032	001256		MOV	32(R1),\$TMP12		
5188									
5189	025662	012702	026042			MOV	#MODFT0,R2		;CHECK THE FRACTIONAL RESULT.
5190	025666	026112	000010			CMP	10(R1),(R2)		
5191	025672	001022				BNE	10\$		;BRANCH IF INCORRECT.
5192	025674	026162	000012	000002		CMP	12(R1),2(R2)		
5193	025702	001016				BNE	10\$		
5194									
5195	025704	012702	026052			MOV	#MODFT1,R2		;CHECK THE INTEGER RESULT.
5196	025710	026112	000014			CMP	14(R1),(R2)		
5197	025714	001026				BNE	15\$		;BRANCH IF INCORRECT.
5198	025716	026162	000016	000002		CMP	16(R1),2(R2)		
5199	025724	001022				BNE	15\$		
5200									
5201	025726	026104	000032			CMP	32(R1),R4		;CHECK THE FPS.
5202	025732	001034				BNE	20\$		;BRANCH IF INCORRECT.
5203									
5204	025734	000161	000042		9\$:	JMP	42(R1)		;RETURN.
5205									
5206									
5207	025740	026112	000020			CMP	20(R1),(R2)		;WAS THE ERROR ANTICIPATED?
5208	025744	001010			10\$:	BNE	11\$		;BRANCH IF NOT ANTICIPATED.
5209	025746	026162	000022	000002		CMP	22(R1),2(R2)		

```

5210 025754 001004          BNE      11$
5211 025756 010102          MOV     R1,R2
5212 025760 062702 000034  ADD     #34,R2          ;THE ERROR WAS ANTICIPATED SO
5213                                     ;RETURN TO THE ERROR REPORT AT THE
5214 025764 000112          JMP     (R2)           ;CALLING ROUTINE.
5215
5216 025766          11$:      ;THE ERROR WAS NOT ANTICIPATED SO
5217 025766 104053 12$:      ERROR  +53      ;REPORT THE INCORRECT FRACTION HERE.
5218 025770 000761          BR     9$
5219
5220          ;INTEGER ERROR.
5221 025772 026112 000024 15$:      CMP     24(R1),(R2)   ;WAS THIS ERROR ANTICIPATED?
5222 025776 001010          BNE     16$           ;BRANCH IF NOT.
5223 026000 026162 000026 000002  CMP     26(R1),2(R2)
5224 026006 001004          BNE     16$
5225 026010 010102          MOV     R1,R2
5226 026012 062702 000040  ADD     #40,R2          ;THE ERROR WAS ANTICIPATED SO RETURN
5227                                     ;TO THE ERROR REPORT IN THE CALLING
5228 026016 000112          JMP     (R2)           ;ROUTINE.
5229
5230 026020          16$:      ;THE ERROR WAS NOT ANTICIPATED SO REPORT
5231 026020 104054 17$:      ERROR  +54      ;THE INTEGER FAILURE HERE.
5232 026022 000744          BR     9$
5233
5234          ;FPS INCORRECT.
5235 026024 010437 001254 20$:      MOV     R4,$TMP11   ;REPORT INCORRECT FPS.
5236 026030 016137 000032 001256  MOV     32(R1),$TMP12
5237 026036 104055 21$:      ERROR  +55
5238 026040 000735          BR     9$
5239
5240 026042 000000 000000 000000 MODFT0: .WORD 0,0,0,0
5241
5242 026052 000000 000000 000000 MODFT1: .WORD 0,0,0,0
5243
5244 026062 177777 177777 177777 MODP1:  .WORD -1,-1,-1,-1
5245
5246 026072          GGGDONE:
5247 026072 104412          RSETUP      ;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?).

```

5247  
5248  
5249  
5257

5258

```
.SBTTL TEST # 17 - MODD TEST  
:*****  
:*TEST 17 MODD TEST  
:*  
:*THIS IS A TEST OF THE MODD INSTRUCTION. IT MAKES USE OF A SUBROUTINE  
:*TO SET UP THE ARGUMENTS, EXECUTE THE INSTRUCTION AND CHECK THE  
:*RESULTS.  
:*  
:*****
```

026074 000004

TST17: SCOPE

5259

:MODD WITH (FSRC=AC=0)

5260

HHH1:

5261

026076 104413

LPEFR

:SET UP THE LOOP ON ERROR ADDRESS.

5262

026100 004737

027602

JSR PC,MODDSUB

5263

026104 000000

000000

000000

1\$:

.WORD 0,0,0,0

:AC

5264

026114 000000

000000

000000

2\$:

.WORD 0,0,0,0

:FSRC

5265

026124 000000

000000

000000

3\$:

.WORD 0,0,0,0

:FRACTIONAL RES.

5266

026134 000000

000000

000000

4\$:

.WORD 0,0,0,0

:INTEGER RES.

5267

026144 000000

000000

000000

5\$:

.WORD 0,0,0,0

:ERROR FRACTIONAL RES.

5268

026154 000000

000000

177777

6\$:

.WORD 0,0,-1,-1

:ERROR INTEGER RES.

5269

026164 000200

7\$:

200

:FPS BEFORE EXECUTION.

5270

026166 000204

8\$:

ERROR +70

:FPS AFTER EXECUTION.

5271

026170 104070

9\$:

BR 9\$

:ST 231 TO 142 INTO 143

5272

026172 000401

9\$:

ERROR +74

5273

026174 104074

5274

026176

9\$:

5275

:MODD TEST WITH FSRC=0

5276

HHH2:

5277

026176 104413

LPERR

:SET UP THE LOOP ON ERROR ADDRESS.

5278

026200 004737

027602

JSR PC,MODDSUB

:AC

5279

026204 012345

067012

1\$:

.WORD 012345,67012

:AC

5280

026210 034567

012345

2\$:

.WORD 34567,012345

:FSRC

5281

026214 000000

000000

000000

3\$:

.WORD 0,0,0,0

:FRACTIONAL RES.

5282

026224 000000

000000

000000

4\$:

.WORD 0,0,0,0

:INTEGER RES.

5283

026234 000000

000000

000000

5\$:

.WORD 012345,67012

:ERROR FRACTIONAL RES.

5284

026244 012345

067012

6\$:

.WORD 34567,012345

:ERROR INTEGER RES.

5285

026250 034567

012345

177777

7\$:

.WORD -1,-1,-1,-1

:FPS BEFORE EXECUTION.

5286

026254 177777

177777

177777

8\$:

ERROR +75

:FPS AFTER EXECUTION.

5287

026264 000213

9\$:

213

:STORE DOUBLE ZERO

5288

026266 000204

9\$:

BR 9\$

:AC V 1 <= ZERO ST 143

5289

026270 104075

5290

026272 000401

5291

026274 104076

5292

026276

5293

:MODD TEST WITH (AC=0)

5294

HHH3:

5295

026276 104413

LPERR

:SET UP THE LOOP ON ERROR ADDRESS.

5296

026300 004737

027602

JSR PC,MODDSUB

:AC

5297

026304 000000

000000

000000

1\$:

.WORD 0,0,0,0

:FSRC

5298

026314 072727

127272

2\$:

.WORD 72727,127272

:FRACTIONAL RES.

5299

026320 072727

127272

3\$:

.WORD 72727,127272

:INTEGER RES.

5300

026324 000000

000000

000000

4\$:

.WORD 0,0,0,0

:ERROR FRACTIONAL RES.

5301

026334 000000

000000

000000

5\$:

.WORD -1,-1,-1,-1

5302

026344 177777

177777

177777



5303	026354	177777	177777	177777	6\$:	.WORD	-1,-1,-1,-1		:ERROR INTEGER RES.
5304	026364	000213			7\$:	213			:FPS BEFORE EXECUTION.
5305	026366	000204				204			:FPS AFTER EXECUTION.
5306	026370	104070			8\$:	ERROR	+70		
5307	026372	000401				BR	9\$		
5308	026374	104071				ERROR	+71		
5309	026376				9\$:				
5310									
5311									
5312	026376								
5313	026400	104413				LPERR			:SET UP THE LOOP ON ERROR ADDRESS.
5314	026404	004737	027602			JSR	PC,MODDSUB		
5315	026410	056252	125252		1\$:	.WORD	56252,125252		:AC
5316	026414	125252	125250			.WORD	125252,125250		
5317	026424	040300	000000	000000	2\$:	.WORD	40300,0,0,0		:FSRC
5318	026434	000000	000000	000000	3\$:	.WORD	0,0,0,0		:FRACTIONAL RES.
5319	026444	056377	177777	177777	4\$:	.WORD	56377,-1,-1,-4		:INTEGER RES.
5320	026450	000000	000000		5\$:	.WORD	0,0		:ERROR FRACTIONAL RES.
5321	026454	125252	125252			.WORD	125252,125252		
5322	026464	056377	177777	177777	6\$:	.WORD	56377,-1,-1,-1		:ERROR INTEGER RES.
5323	026466	000213			7\$:	213			:FPS BEFORE EXECUTION.
5324	026470	000204				204			:FPS AFTER EXECUTION.
5325	026472	104077			8\$:	ERROR	+77		:ST 526 TO 134 INTO 135
5326	026474	000401				BR	9\$		
5327	026476	104077				ERROR	+77		
5328					9\$:				
5329									
5330	026476								
5331	026500	104413				LPERR			:SET UP THE LOOP ON ERROR ADDRESS.
5332	026504	004737	027602			JSR	PC,MODDSUB		
5333	026514	140240	000000	000000	1\$:	.WORD	140240,0,0,0		:AC
5334	026520	063714	146314		2\$:	.WORD	63714,146314		:FSRC
5335	026524	133572	167737			.WORD	133572,167737		
5336	026534	000000	000000	000000	3\$:	.WORD	0,0,0,0		:FRACTIONAL RES.
5337	026540	163777	177777		4\$:	.WORD	163777,-1		:INTEGER RES.
5338	026544	162531	125726			.WORD	162531,125726		
5339	026554	177777	177777	177777	5\$:	.WORD	-1,-1,-1,-1		:ERROR FRACTIONAL RES.
5340	026560	063777	177777		6\$:	.WORD	63777,-1		:ERROR INTEGER RES.
5341	026564	162531	125726			.WORD	162531,125726		
5342	026566	000210			7\$:	210			:FPS BEFORE EXECUTION.
5343	026570	000204				204			:FPS AFTER EXECUTION.
5344	026572	104070			8\$:	ERROR	+70		
5345	026574	000401				BR	9\$		
5346	026576	104100				ERROR	+100		:ST 526 BAD SIGN
5347					9\$:				
5348									
5349	026576								
5350	026600	104413				LPERR			:SET UP THE LOOP ON ERROR ADDRESS.
5351	026604	004737	027602			JSR	PC,MODDSUB		
5352	026614	056200	000000	000000	1\$:	.WORD	56200,0,0,1		:AC
5353	026624	040340	000000	000000	2\$:	.WORD	40340,0,0,0		:FSRC
5354	026634	000000	000000	000000	3\$:	.WORD	0,0,0,0		:FRACTIONAL RES.
5355	026644	056340	000000	000000	4\$:	.WORD	56340,0,0,1		:INTEGER RES.
5356	026654	040000	000000	000000	5\$:	.WORD	40000,0,0,0		:ERROR FRACTIONAL RES.
		056340	000000	000000	6\$:	.WORD	56340,0,0,1		:ERROR INTEGER RES.





```

5411 027170 104103      8$:      ERROR      +103      ;(NORMALIZE) ST 532 TO 122
5412                                     ;INTO NORM.
5413 027172 000401      BR          9$
5414 027174 104104      ERROR      +104      ;AC V 1 <= X14
5415                                     ;OR ST 733 TO 156 INTO 157.
5416 027176      9$:
5417
5418      ;MODD TEST WITH EXPONENT OF THE RESULT = 31
5419 027176      HHH10:
      027176 104413      LPERR      ;SET UP THE LOOP ON ERROR ADDRESS.
5420 027200 004737 027602      JSR      PC,MODDSUB
5421 027204 047777 177777      1$:      .WORD      47777,-1      ;AC
5422 027210 177000 000000      .WORD      177000,0
5423 027214 040200 000000 000000 2$:      .WORD      40200,0,0,0      ;FSRC
5424 027224 000000 000000 000000 3$:      .WORD      0,0,0,0      ;FRACTIONAL RES.
5425 027234 047777 177777      4$:      .WORD      47777,-1      ;INTEGER RES.
5426 027240 177000 000000      .WORD      177000,0
5427 027244 000000 000000 177000 5$:      .WORD      0,0,177000,0      ;ERROR FRACTIONAL RES.
5428 027254 177777 177777 177777 6$:      .WORD      -1,-1,-1,-1      ;ERROR INTEGER RES.
5429 027264 000213      7$:      213      ;FPS BEFORE EXECUTION.
5430 027266 000204      204      ;FPS AFTER EXECUTION.
5431 027270 104105      8$:      ERROR      +105      ;(BUT FD) STORE X10
5432 027272 000401      BR          9$
5433 027274 104071      ERROR      +71
5434 027276      9$:
5435
5436      ;MODD TEST WITH EXPONENT OF THE RESULT = 0
5437 027276      HHH11:
      027276 104413      LPERR      ;SET UP THE LOOP ON ERROR ADDRESS.
5438 027300 004737 027602      JSR      PC,MODDSUB
5439 027304 040200 000000 000000 1$:      .WORD      40200,0,0,0      ;AC
5440 027314 040177 072727      2$:      .WORD      40177,72727      ;FSRC
5441 027320 127272 072727      .WORD      127272,72727
5442 027324 040177 072727      3$:      .WORD      40177,72727      ;FRACTIONAL RES.
5443 027330 127272 072727      .WORD      127272,72727
5444 027334 000000 000000 000000 4$:      .WORD      0,0,0,0      ;INTEGER RES.
5445 027344 177777 177777 177777 5$:      .WORD      -1,-1,-1,-1      ;ERROR FRACTIONAL RES.
5446 027354 000000 000000 177777 6$:      .WORD      0,0,-1,-1      ;ERROR INTEGER RES.
5447 027364 000200      7$:      200      ;FPS BEFORE EXECUTION.
5448 027366 000200      200      ;FPS AFTER EXECUTION.
5449 027370 104070      8$:      ERROR      +70
5450 027372 000401      BR          9$
5451 027374 104106      ERROR      +106      ;ST 246 TO 126 INTO 127 (BUT FD)
5452 027376      9$:
5453
5454      ;MODD TEST WITH EXPONENT OF THE RESULT = -115
5455 027376      HHH12:
      027376 104413      LPERR      ;SET UP THE LOOP ON ERROR ADDRESS.
5456 027400 004737 027602      JSR      PC,MODDSUB
5457 027404 003377 177777      1$:      .WORD      3377,-1      ;AC
5458 027410 177777 052525      .WORD      -1,52525
5459 027414 040200 000000 000000 2$:      .WORD      40200,0,0,0      ;FSRC
5460 027424 003377 177777      3$:      .WORD      3377,-1      ;FRACTIONAL RES.
5461 027430 177777 052525      .WORD      -1,52525
5462 027434 000000 000000 000000 4$:      .WORD      0,0,0,0      ;INTEGER RES.
5463 027444 177777 177777 177777 5$:      .WORD      -1,-1,-1,-1      ;ERROR FRACTIONAL RES.
5464 027454 000000 000000 177777 6$:      .WORD      0,0,-1,-1      ;ERROR INTEGER RES.

```





```

5521 027602 012601          MODDSUB:  MOV      (SP)+,R1      ;GET A POINTER TO THE ARGUMENTS
5522 027604 012700 000200      MOV      #200,R0      ;SET FD MODE.
5523 027610 170100          LDFPS   R0
5524 027612 010100          MOV      R1,R0        ;SET UP ACO
5525 027614 172410          LDD     (R0),ACO
5526 027616 012700 026062      MOV      #MODP1,R0    ;PUT A BACKGROUND PATTERN INTO AC1.
5527 027622 172510          LDD     (R0),AC1
5528 027624 016100 000060      MOV      60(R1),R0    ;SET UP THE FPS.
5529 027630 170100          LDFPS   R0
5530 027632 012737 027646 001236  MOV      #1$,STMP2
5531 027640 010100          MOV      R1,R0        ;COMPUTE THE ADDRESS OF THE FSRC.
5532 027642 062700 000010      ADD     #10,R0
5533
5534 027646 171410          1$:    MODD     (R0),ACO      ;EXECUTE THE TEST INSTRUCTION.
5535
5536 027650 170204          STFPS   R4            ;GET THE FPS.
5537 027652 012700 000200      MOV      #200,R0      ;SET FD MODE.
5538 027656 170100          LDFPS   R0
5539 027660 012700 030160      MOV      #MODDT0,R0   ;GET THE FRACTIONAL RESULT.
5540 027664 174010          STD     ACO,(R0)
5541 027666 012700 030170      MOV      #MODDT1,R0   ;GET THE INTEGER RESULT.
5542 027672 174110          STD     AC1,(R0)
5543
5544 027674 010102          MOV      R1,R2        ;SAVE THE DATA IN CASE OF ERROR.
5545 027676 010237 001240      MOV      R2,STMP3
5546 027702 062702 000010      ADD     #10,R2
5547 027706 010237 001242      MOV      R2,STMP4
5548 027712 062702 000010      ADD     #10,R2
5549 027716 010237 001244      MOV      R2,STMP5
5550 027722 062702 000010      ADD     #10,R2
5551 027726 010237 001246      MOV      R2,STMP6
5552 027732 012737 030160 001250  MOV      #MODDT0,STMP7
5553 027740 012737 030170 001252  MOV      #MODDT1,STMP10
5554 027746 016137 000062 001256  MOV      62(R1),STMP12
5555 027754 010437 001254      MOV      R4,STMP11
5556
5557 027760 012702 030160      MOV      #MODDT0,R2    ;CHECK THE FRACTIONAL RESULT.
5558 027764 010103          MOV      R1,R3
5559 027766 062703 000020      ADD     #20,R3
5560 027772 012705 000004      MOV      #4,R5
5561 027776 022223          2$:    CMP      (R2)+,(R3)+
5562 030000 001020          BNE     10$           ;BRANCH IF INCORRECT.
5563 030002 077503          SOB     R5,2$
5564
5565 030004 012702 030170      MOV      #MODDT1,R2    ;CHECK THE INTEGER RESULT.
5566 030010 010103          MOV      R1,R3
5567 030012 062703 000030      ADD     #30,R3
5568 030016 012705 000004      MOV      #4,R5
5569 030022 022223          3$:    CMP      (R2)+,(R3)+
5570 030024 001026          BNE     15$           ;BRANCH IF INCORRECT.
5571 030026 077503          SOB     R5,3$
5572
5573
5574 030030 026104 000062          CMP     62(R1),R4     ;CHECK THE FPS.
5575 030034 001042          BNE     20$           ;BRANCH IF INCORRECT.
5576
5577 030036 000161 000072          9$:    JMP     72(R1)      ;RETURN.

```



```

5578
5579
5580 030042 012702 030160
5581 030046 010103
5582 030050 062703 000040
5583 030054 012705 000004
5584 030060 022223
5585 030062 001005
5586 030064 077503
5587 030066 010102
5588 030070 062702 000064
5589
5590 030074 000112
5591
5592 030076
5593 030076 104070
5594 030100 000756
5595
5596
5597 030102 012702 030170
5598 030106 010103
5599 030110 062703 000050
5600 030114 012705 000004
5601 030120 022223
5602 030122 001005
5603 030124 077503
5604 030126 010102
5605 030130 062702 000070
5606
5607 030134 000112
5608
5609 030136
5610 030136 104071
5611 030140 000736
5612
5613
5614 030142 010437 001254
5615 030146 016137 000062 001256
5616 030154 104072
5617 030156 000727
5618
5619 030160 000000 000000 000000
5620
5621 030170 000000 000000 000000
5622
5623 030200
030200 104412

: FRACTIONAL ERROR.
10$: MOV #MODDT0,R2 ;WAS THE FRACTIONAL ERROR ANTICIPATED?
MOV R1,R3
ADD #40,R3
MOV #4,R5
50$: CMP (R2)+,(R3)+ ;BRANCH IF NOT ANTICIPATED.
BNE 11$
SOB R5,50$
MOV R1,R2 ;THE ERROR WAS ANTICIPATED SO
ADD #64,R2 ;RETURN TO THE ERROR REPORT AT THE
;CALLING ROUTINE.
JMP (R2)

11$:
12$: ERROR +70 ;THE ERROR WAS NOT ANTICIPATED SO
BR 9$ ;REPORT THE INCORRECT FRACTION HERE.

: INTEGER ERROR.
15$: MOV #MODDT1,R2 ;WAS THE INTEGER ERROR ANTICIPATED?
MOV R1,R3
ADD #50,R3
MOV #4,R5
60$: CMP (R2)+,(R3)+ ;BRANCH IF NOT ANTICIPATED.
BNE 17$
SOB R5,60$
MOV R1,R2 ;THE ERROR WAS ANTICIPATED SO RETURN
ADD #70,R2 ;TO THE ERROR REPORT IN THE CALLING
;ROUTINE.
JMP (R2)

16$:
17$: ERROR +71 ;THE ERROR WAS NOT ANTICIPATED SO REPORT
BR 9$ ;THE INTEGER FAILURE HERE.

: FPS INCORRECT.
20$: MOV R4,$TMP11 ;REPORT INCORRECT FPS.
MOV 62(R1),$TMP12
21$: ERROR +72
BR 9$

MODDT0: .WORD 0,0,0,0
MODDT1: .WORD 0,0,0,0

HHHDONE:
RSETUP ;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?).

```

5624  
5625  
5626  
5634  
5635

;TEST TITLE:UNDER/OVERFLOW, USING MODF WITH TRAPS DISABLED



5636

.SBTTL TEST # 20 - SEE COMMENT ABOVE FOR TEST TITLE  
\*\*\*\*\*  
\*TEST 20 SEE COMMENT ABOVE FOR TEST TITLE  
\*  
\*THIS IS A TEST OF THE MODF OVERFLOW AND UNDERFLOW CONDITIONS. IT MAKES  
\*USE OF A SUBROUTINE TO SETUP THE OPERANDS, EXECUTE THE MODF INSTRUCTION  
\*AND CHECK THE RESULTS. TRAPS ARE DISABLED DURING THIS TEST.  
\*  
\*\*\*\*\*

5637 030202 000004  
5638  
5639 030204 104413  
5640 030206 004737 030532  
5641 030212 020123 045676  
5642 030216 020200 000000  
5643 030222 000123 045676  
5644 030226 000000 000000  
5645 030232 177777 177777  
5646 030236 177777 177777  
5647 030242 042000  
5648 030244 142004  
5649 030246 000012  
5650 030250 104170  
5651 030252 000401  
5652 030254 104171  
5653 030256

TST20: SCOPE  
;UNDERFLOW TEST, WITH EXPONENT OF THE RESULT = -129, FIU = 1, FID = 1  
MM1:  
LPERR ;SET UP THE LOOP ON ERROR ADDRESS.  
JSR PC,MODFOV  
1\$: .WORD 20123,45676 ;AC  
2\$: .WORD 20200,0 ;FSRC  
3\$: .WORD 123,45676 ;FRACTIONAL RES.  
4\$: .WORD 0,0 ;INTEGER RES.  
5\$: .WORD -1,-1 ;ERROR FRACTIONAL RES.  
6\$: .WORD -1,-1 ;ERROR INTEGER RES.  
7\$: 42000 ;FPS BEFORE EXECUTION.  
142004 ;FPS AFTER EXECUTION.  
12 ;FEC  
8\$: ERROR +170 ;FEC INCORRECT, UNDERFLOW.  
BR 9\$  
9\$: ERROR +171 ;AC V 1 (2,3) <= ZERO, ST 126.

5654  
5655  
5656 030256 104413  
5657 030260 004737 030532  
5658 030264 010200 000000  
5659 030270 010000 000000  
5660 030274 000000 000000  
5661 030300 000000 000000  
5662 030304 177777 177777  
5663 030310 177777 177777  
5664 030314 005013  
5665 030316 005004  
5666 030320 000012  
5667 030322 000240  
5668 030324 000401  
5669 030326 104171  
5670 030330

;UNDERFLOW EXP OF RESULT = -193, FIU = 0, FID = 1  
MM2:  
LPERR ;SET UP THE LOOP ON ERROR ADDRESS.  
JSR PC,MODFOV  
1\$: .WORD 10200,0 ;AC  
2\$: .WORD 10000,0 ;FSRC  
3\$: .WORD 0,0 ;FRACTIONAL RES.  
4\$: .WORD 0,0 ;INTEGER RES.  
5\$: .WORD -1,-1 ;ERROR FRACTIONAL RES.  
6\$: .WORD -1,-1 ;ERROR INTEGER RES.  
7\$: 5013 ;FPS BEFORE EXECUTION.  
5004 ;FPS AFTER EXECUTION.  
12 ;FEC  
8\$: NOP  
BR 9\$  
9\$: ERROR +171

5671  
5672  
5673 030330 104413  
5674 030332 004737 030532  
5675 030336 060052 125252  
5676 030342 060200 000000  
5677 030346 000000 000000  
5678 030352 000052 125252  
5679 030356 000000 000000  
5680 030362 000000 000000

;OVERFLOW TEST WITH EXPONENT OF THE RESULT = 128, FIV = 1, FID = 1  
MM3:  
LPERR ;SET UP THE LOOP ON ERROR ADDRESS.  
JSR PC,MODFOV  
1\$: .WORD 60052,125252 ;AC  
2\$: .WORD 60200,0 ;FSRC  
3\$: .WORD 0,0 ;FRACTIONAL RES.  
4\$: .WORD 52,125252 ;INTEGER RES.  
5\$: .WORD 0,0 ;ERROR FRACTIONAL RES.  
6\$: .WORD 0,0 ;ERROR INTEGER RES.

```

5681 030366 041000      7$:      41000      :FPS BEFORE EXECUTION.
5682 030370 141006      :FPS AFTER EXECUTION.
5683 030372 000010      :FEC
5684 030374 104172      8$:      ERROR +172      :BAD FEC ON OVERFLOW.
5685 030376 000401      BR 9$
5686 030400 104173      ERROR +173      :ST 520 TO STORE ZERO TWICE
5687                                     :INTO 162
5688 030402      9$:
5689
5690      :OVERFLOW TEST WITH EXPONENT OF THE RESULT = 130, FIV = 0, FID = 1
5691 030402      M44:
      LPERR      :SET UP THE LOOP ON ERROR ADDRESS.
      JSR      PC,MODFOV
5692 030404 104413 030532      1$:      .WORD 60345,67654      :AC
5693 030410 060345 067654      2$:      .WORD 60200,0      :FSRC
5694 030414 060200 000000      3$:      .WORD 0,0      :FRACTIONAL RES.
5695 030420 000000 000000      4$:      .WORD 0,0      :INTEGER RES.
5696 030424 000000 000000      5$:      .WORD 0,0      :ERROR FRACTIONAL RES.
5697 030430 000000 000000      6$:      .WORD 345,67654      :ERROR INTEGER RES.
5698 030434 000345 067654      7$:      6011      :FPS BEFORE EXECUTION.
5699 030440 006011      6006      :FPS AFTER EXECUTION.
5700 030442 006006      10      :FEC
5701 030444 000010
5702 030446 000240      8$:      NOP
5703 030450 000401      BR 9$
5704 030452 104174      ERROR +174      :ST 520 TO 162 INTO STORE ZERO TWICE.
5705 030454      9$:
5706
5707      :OVERFLOW TEST WITH EXPONENT OF THE RESULT = 128, RESULT NEGATIVE
5708      :AND FIV = 1, FID = 1
5709 030454      M45:
      LPERR      :SET UP THE LOOP ON ERROR ADDRESS.
      JSR      PC,MODFOV
5710 030456 104413 030532      1$:      .WORD 160252,125252      :AC
5711 030462 160252 125252      2$:      .WORD 60000,0      :FSRC
5712 030466 060000 000000      3$:      .WORD 0,0      :FRACTIONAL RES.
5713 030472 000000 000000      4$:      .WORD 100052,125252      :INTEGER RES.
5714 030476 100052 125252      5$:      .WORD 0,0      :ERROR FRACTIONAL RES.
5715 030502 000000 000000      6$:      .WORD 52,125252      :ERROR INTEGER RES.
5716 030506 000052 125252      7$:      41000      :FPS BEFORE EXECUTION.
5717 030512 041000      141006      :FPS AFTER EXECUTION.
5718 030514 141006      10      :FEC
5719 030516 000010
5720 030520 104172      8$:      ERROR +172
5721 030522 000401      BR 9$
5722 030524 104175      ERROR +175      :ST 517, BAD SIGN.
5723 030526 000137 031126      9$:      JMP M45DONE ;GO TO THE NEXT TEST.
5724
5725      :THIS SUBROUTINE, MODFOV, IS CALLED TO SETUP THE
5726      :OPERANDS, EXECUTE THE MODF INSTRUCTION AND CHECK THE RESULTS.
5727      :IT IS CALLED THUS:
5728      :
5729      :ACARG: .WORD X,X      :AC OPERAND
5730      :FSRCARG: .WORD X,X      :FSRC OPERAND
5731      :FRES: .WORD X,X      :FRACTIONAL RESULT
5732      :INTRES: .WORD X,X      :INTEGER RESULT
5733      :ERFRES: .WORD X,X      :ERROR FRACTION RESULT
5734      :ERINTRES: .WORD X,X      :ERROR INTEGER RESULT
5735      :FPSB: .WORD X      :FPS BEFORE EXECUTION
    
```



5736  
5737  
5738  
5739  
5740  
5741  
5742  
5743  
5744  
5745  
5746  
5747  
5748  
5749  
5750  
5751  
5752  
5753  
5754  
5755  
5756  
5757  
5758  
5759 030532 012601  
5760 030534 012700 000200  
5761 030540 170100  
5762 030542 010100  
5763 030544 172410  
5764 030546 012700 026062  
5765 030552 172510  
5766 030554 016100 000030  
5767 030560 170100  
5768 030562 012737 030576 001236  
5769 030570 010100  
5770 030572 062700 000004  
5771  
5772 030576 171410  
5773  
5774 030600 170204  
5775 030602 170305  
5776 030604 012700 000200  
5777 030610 170100  
5778 030612 012700 031106  
5779 030616 174010  
5780 030620 012700 031116  
5781 030624 174110  
5782  
5783 030626 010102  
5784 030630 010237 001240  
5785 030634 062702 000004  
5786 030640 010237 001242  
5787 030644 062702 000004  
5788 030650 010237 001244  
5789 030654 062702 000004  
5790 030660 010237 001246  
5791 030664 012737 031106 001250  
5792 030672 012737 031116 001252

```

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.
        LDFPS R0
        MOV R1,R0        ;SET UP ACO
        LDD (R0),ACO
        MOV #MODP1,R0    ;PUT A BACKGROUND PATTERN INTO AC1.
        LDD (R0),AC1
        MOV 30(R1),R0    ;SET UP THE FPS.
        LDFPS R0
        MOV #1$,STMP2
        MOV R1,R0        ;COMPUTE THE ADDRESS OF THE FSRC.
        ADD #4,R0

1$:     MODF (R0),ACO    ;EXECUTE THE TEST INSTRUCTION.

        STFPS R4        ;GET THE FPS.
        STST R5         ;GET FEC.
        MOV #200,R0     ;SET FD MODE.
        LDFPS R0
        MOV #MODFDO,R0 ;GET THE FRACTIONAL RESULT.
        STD ACO,(R0)
        MOV #MODFD1,R0 ;GET THE INTEGER RESULT.
        STD AC1,(R0)

        MOV R1,R2      ;SAVE THE DATA IN CASE OF ERROR.
        MOV R2,STMP3
        ADD #4,R2
        MOV R2,STMP4
        ADD #4,R2
        MOV R2,STMP5
        ADD #4,R2
        MOV R2,STMP6
        MOV #MODFDO,STMP7
        MOV #MODFD1,STMP10
    
```



5793	030700	010437	001254		MOV	R4,\$TMP11	
5794	030704	016137	000032	001256	MOV	32(R1),\$TMP12	
5795	030712	010537	001260		MOV	R5,\$TMP13	
5796	030716	016137	000034	001262	MOV	34(R1),\$TMP14	
5797							
5798	030724	012702	031106		MOV	#MODFD0,R2	:CHECK THE FRACTIONAL RESULT.
5799	030730	026112	000010		CMP	10(R1),(R2)	
5800	030734	001025			BNE	10\$	:BRANCH IF INCORRECT.
5801	030736	026162	000012	000002	CMP	12(R1),2(R2)	
5802	030744	001021			BNE	10\$	
5803							
5804	030746	012702	031116		MOV	#MODFD1,R2	:CHECK THE INTEGER RESULT.
5805	030752	026112	000014		CMP	14(R1),(R2)	
5806	030756	001016			BNE	15\$	:BRANCH IF INCORRECT.
5807	030760	026162	000016	000002	CMP	16(R1),2(R2)	
5808	030766	001012			BNE	15\$	
5809							
5810	030770	026104	000032		CMP	32(R1),R4	:CHECK THE FPS.
5811	030774	001024			BNE	20\$	:BRANCH IF INCORRECT.
5812							
5813	030776	026105	000034		CMP	34(R1),R5	:CHECK THE FEC.
5814	031002	001030			BNE	25\$	:BRANCH IF INCORRECT.
5815							
5816	031004	000161	000044		9\$: JMP	44(R1)	:RETURN.
5817							
5818							:FRACTIONAL ERROR.
5819	031010				10\$:		:THE ERROR WAS NOT ANTICIPATED SO
5820	031010	104165			12\$: ERROR	+165	:REPORT THE INCORRECT FRACTION HERE.
5821	031012	000774			BR	9\$	
5822							
5823							:INTEGER ERROR.
5824	031014	026112	000024		15\$: CMP	24(R1),(R2)	:WAS THIS ERROR ANTICIPATED?
5825	031020	001010			BNE	16\$	:BRANCH IF NOT.
5826	031022	026162	000026	000002	CMP	26(R1),2(R2)	
5827	031030	001004			BNE	16\$	
5828	031032	010102			MOV	R1,R2	:THE ERROR WAS ANTICIPATED SO RETURN
5829	031034	062702	000042		ADD	#42,R2	:TO THE ERROR REPORT IN THE CALLING
5830							:ROUTINE.
5831	031040	000112			JMP	(R2)	
5832							
5833	031042				16\$:		:THE ERROR WAS NOT ANTICIPATED SO REPORT
5834	031042	104166			17\$: ERROR	+166	:THE INTEGER FAILURE HERE.
5835	031044	000757			BR	9\$	
5836							
5837							:FPS INCORRECT.
5838	031046	010437	001254		20\$: MOV	R4,\$TMP11	:REPORT INCORRECT FPS.
5839	031052	016137	000032	001256	MOV	32(R1),\$TMP12	
5840	031060	104167			21\$: ERROR	+167	
5841	031062	000750			BR	9\$	
5842							
5843							:REPORT FEC ERROR.
5844	031064	010537	001260		25\$: MOV	R5,\$TMP13	
5845	031070	016137	000034	001262	MOV	34(R1),\$TMP14	
5846	031076	010102			MOV	R1,R2	
5847	031100	062702	000036		ADD	#36,R2	
5848	031104	000112			JMP	(R2)	
5849							

5850 031106 000000 000000 000000 MODFD0: .WORD 0,0,0,0  
5851  
5852 031116 000000 000000 000000 MODFD1: .WORD 0,0,0,0  
5853  
5854 031126 MMDONE:  
031126 104412 RSETUP

: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?).

5855  
5856  
5857  
5865  
5866

;TEST TITLE:UNDER/OVERFLOW, USING MODD WITH TRAPS DISABLED

5867

```
.SBTTL TEST # 21 - SEE COMMENT ABOVE FOR TEST TITLE
*****
:TEST 21 SEE COMMENT ABOVE FOR TEST TITLE
*
*THIS IS A TEST OF THE MODD INSTRUCTION'S OVER FLOW AND UNDER FLOW
*CONDITIONS. A SUBROUTINE IS USED TO SET UP THE OPERANDS, EXECUTE THE
*MODD INSTRUCTION AND CHECK THE RESULTS.
*
*****
```

5868 031130 000004

TST21: SCOPE

5869

:UNDERFLOW TEST WITH EXPONENT OF THE RESULT = -129, FIU = 1, FID = 1

5870 031132

NNN1:

```
031132 104413
5871 031134 004737 031546
5872 031140 020252 125252
5873 031144 125252 125252
5874 031150 020100 000000 000000
5875 031160 000177 177777 177777
5876 031170 000000 000000 000000
5877 031200 020252 125252
5878 031204 125252 125252
5879 031210 000000 000000 177777
5880 031220 042200
5881 031222 142204
5882 031224 000012
5883 031226 104201
5884 031230 000401
5885 031232 104202
5886 031234
```

```
LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
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.
5$: .WORD 20252,125252 ;ERROR FRACTIONAL RES.
.WORD 125252,125252
6$: .WORD 0,0,-1,-1 ;ERROR INTEGER RES.
7$: 42200 ;FPS BEFORE EXECUTION.
142204 ;FPS AFTER EXECUTION.
12 ;FEC
8$: ERROR +201 ;FEC INCORRECT ON UNDERFLOW.
BR 9$
ERROR +202 ;ST 155 (BUT FD)
9$:
```

5887

:UNDERFLOW TEST WITH EXPONENT OF THE RESULT = -193, FIU = 0, FID = 1

5888

NNN2:

```
031234
5889 031234 104413
5890 031236 004737 031546
5891 031242 010000 000000
5892 031246 12 000000
5893 031252 010200 000000 000000
5894 031262 000000 000000 000000
5895 031272 000000 000000 000000
5896 031302 000000 000000 000000
5897 031312 000000 000000
5898 031316 123456 000000
5899 031322 005213
5900 031324 005204
5901 031326 000012
5902 031330 000240
5903 031332 000401
5904 031334 104203
5905 031336
```

```
LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
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.
5$: .WORD 0,0,0,0 ;ERROR FRACTIONAL RES.
6$: .WORD 0,0 ;ERROR INTEGER RES.
.WORD 123456,0
7$: 5213 ;FPS BEFORE EXECUTION.
5204 ;FPS AFTER EXECUTION.
12
8$: NOP
BR 9$
ERROR +203 ;ST 047 (BUT FD).
9$:
```

5906

:OVERFLOW TEST WITH EXPONENT OF THE RESULT = 128, FIV = 1, FID = 1

5907

NNN3:

```
031336
5908 031336 104413
5909 031340 004737 031546
5910 031344 060252 125252
5911 031350 125252 125252
```

```
LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
JSR PC,MODDOV
1$: .WORD 60252,125252 ;AC
.WORD 125252,125252
```



```

5912 031354 060100 000000 000000 2$: .WORD 60100,0,0,0 :FSRC
5913 031364 000000 000000 000000 3$: .WORD 0,0,0,0 :FRACTIONAL RES.
5914 031374 000177 177777 177777 4$: .WORD 177,-1,-1,-1 :INTEGER RES.
5915 031404 000000 000000 000000 5$: .WORD 0,0,0,0 :ERROR FRACTIONAL RES.
5916 031414 000177 177777 177777 6$: .WORD 177,-1 :ERROR INTEGER RES.
5917 031420 125252 125252 .WORD 125252,125252
5918 031424 041200 7$: 41200 :FPS BEFORE EXECUTION.
5919 031426 141206 141206 :FPS AFTER EXECUTION.
5920 031430 000010 10 :FEC
5921 031432 104204 8$: ERROR +204 :FEC BAD ON OVERFLOW.
5922 031434 000401 BR 9$
5923 031436 104205 9$: ERROR +205 :ST 520 TO 162 INTO 163 (BUT FD).
5924 031440
5925
5926 :OVERFLOW TEST WITH EXPONENT OF THE RESULT = 130, FIV = 0, FID = 1
5927 031440 NNN4:
031440 104413 LPERR :SET UP THE LOOP ON ERROR ADDRESS.
031442 004737 031546 JSR PC,MODDOV
5928 031442 004737 031546 1$: .WORD 60200,0 :AC
5929 031446 060200 000000 .WORD 125252,0
5930 031452 125252 000000 2$: .WORD 60200,0,0,0 :FSRC
5931 031456 060200 000000 000000 3$: .WORD 0,0,0,0 :FRACTIONAL RES.
5932 031466 000000 000000 000000 4$: .WORD 0,0,0,0 :INTEGER RES.
5933 031476 000000 000000 000000 5$: .WORD 0,0,0,0 :ERROR FRACTIONAL RES.
5934 031506 000000 000000 000000 6$: .WORD 400,0 :ERROR INTEGER RES.
5935 031516 000400 000000 .WORD 125252,0
5936 031522 125252 000000 7$: 6211 :FPS BEFORE EXECUTION.
5937 031526 006211 6206 :FPS AFTER EXECUTION.
5938 031530 006206 10 :FEC
5939 031532 000010 8$: NOP
5940 031534 000240 BR 9$
5941 031536 000401 ERROR +206 :ST 520 TO 162 INTO STORE ZERO TWICE.
5942 031540 104206 9$: JMP NNNDONE ;GO TO NEXT TEST.
5943 031542 000137 032154
5944
5945
5946 :THIS SUBROUTINE, MODDOV, IS CALLED TO SETUP THE
5947 :OPERANDS, EXECUTE THE MODD INSTRUCTION AND CHECK THE RESULTS.
5948 :IT IS CALLED THUS:
5949
5950 ACARG: .WORD X,X,X,X :AC OPERAND
5951 FSRCARG: .WORD X,X,X,X :FSRC OPERAND
5952 FRES: .WORD X,X,X,X :FRACTIONAL RESULT
5953 INTRES: .WORD X,X,X,X :INTEGER RESULT
5954 ERFRES: .WORD X,X,X,X :ERROR FRACTION RESULT
5955 ERINTRES: .WORD X,X,X,X :ERROR INTEGER RESULT
5956 FPSB: .WORD X :FPS BEFORE EXECUTION
5957 FPSA: .WORD X :FPS AFTER EXECUTION
5958 ERR1: ERROR +X :FRACTION ERROR
5959 BR CONT
5960 ERR2: ERROR +X :INTEGER ERROR
5961 CONT: :RETURN ADDRESS
5962
5963 :THE OPERANDS ARE SET UP (USING ACO FOR THE AC ARGUMENT). THE MODD
5964 :INSTRUCTION IS EXECUTED, THEN THE RESULTS ARE RETRIEVED.
5965 :THE FRACTION PART OF THE RESULT IS COMPARED WITH FRES. IF THIS IS CORRECT
5966 :THEN THE INTEGER PART IS COMPARED WITH INTRES. IF BOTH OF THESE ARE CORRECT
5967 :THEN THE FPS IS COMPARED WITH FPSA. AFTER EXECUTION IF NO ERRORS OCCURRED
    
```

```

5968 ;IT IS REPORTED HERE. IF THE FRACTION IS INCORRECT IT IS COMPARED WITH
5969 ;THE ANTICIPATED BAD FRACTION, ERFRES. IF THIS DOESN'T MATCH
5970 ;THE TRUE RESULT THEN THE ERROR IS REPORTED HERE. IF THE ANTICIPATED
5971 ;FAILURE MATCHES THE TRUE RESULT THEN MODDOV PASSES CONTROL TO THE
5972 ;ERROR CALL AT ERR1. LIKEWISE IF THE INTEGER PART OF THE RESULT IS
5973 ;NOT CORRECT THEN IT IS COMPARED WITH THE ANTICIPATED INTEGER
5974 ;FAILURE. IF THIS DOESN'T MATCH THEN THE ERROR IS REPORTED HERE.
5975 ;IF A MATCH IS MADE HOWEVER, MODDOV WILL RETURN CONTROL TO THE ERROR
5976 ;CALL AT ERR2.
5977
5978 031546 012601 MODDOV: MOV (SP)+,R1 ;GET A POINTER TO THE ARGUMENTS
5979 031550 012700 000200 MOV #200,R0 ;SET FD MODE.
5980 031554 170100 LDFPS R0
5981 031556 010100 MOV R1,R0 ;SET UP ACO
5982 031560 172410 LDD (R0),ACO
5983 031562 012700 026062 MOV #MODP1,R0 ;PUT A BACKGROUND PATTERN INTO AC1.
5984 031566 172510 LDD (R0),AC1
5985 031570 016100 000060 MOV 60(R1),R0 ;SET UP THE FPS.
5986 031574 170100 LDFPS R0
5987 031576 012737 031612 001236 MOV #1$,STMP2
5988 031604 010100 MOV R1,R0 ;COMPUTE THE ADDRESS OF THE FSRC.
5989 031606 062700 000010 ADD #10,R0
5990
5991 031612 171410 1$: MODD (R0),ACO ;EXECUTE THE TEST INSTRUCTION.
5992
5993 031614 170305 STST R5 ;GET THE FPS.
5994 031616 170204 STFPS R4 ;GET THE FPS.
5995 031620 012700 000200 MOV #200,R0 ;SET FD MODE.
5996 031624 170100 LDFPS R0
5997 031626 012700 032134 MOV #MODDD0,R0 ;GET THE FRACTIONAL RESULT.
5998 031632 174010 STD ACO,(R0)
5999 031634 012700 032144 MOV #MODDD1,R0 ;GET THE INTEGER RESULT.
6000 031640 174110 STD AC1,(R0)
6001
6002 031642 010102 MOV R1,R2 ;SAVE THE DATA IN CASE OF ERROR.
6003 031644 010237 001240 MOV R2,$TMP3
6004 031650 062702 000010 ADD #10,R2
6005 031654 010237 001242 MOV R2,$TMP4
6006 031660 062702 000010 ADD #10,R2
6007 031664 010237 001244 MOV R2,$TMP5
6008 031670 062702 000010 ADD #10,R2
6009 031674 010237 001246 MOV R2,$TMP6
6010 031700 012737 032134 001250 MOV #MODDD0,$TMP7
6011 031706 012737 032144 001252 MOV #MODDD1,$TMP10
6012 031714 010437 001254 MOV R4,$TMP11
6013 031720 016137 000062 001256 MOV 62(R1),$TMP12
6014 031726 010537 001260 MOV R5,$TMP13
6015 031732 016137 000064 001262 MOV 64(R1),$TMP14
6016
6017 031740 012702 032134 MOV #MODDD0,R2 ;CHECK THE FRACTIONAL RESULT.
6018 031744 010103 MOV R1,R3
6019 031746 062703 000020 ADD #20,R3
6020 031752 012700 000004 MOV #4,R0
6021 031756 022223 2$: CMP (R2)+,(R3)+ ;BRANCH IF INCORRECT.
6022 031760 001023 BNE 10$
6023 031762 077003 SOB R0,2$
6024
    
```

6025	031764	012702	032144	MOV	#MODDD1,R2	;CHECK THE INTEGER RESULT.
6026	031770	010103		MOV	R1,R3	
6027	031772	062703	000030	ADD	#30,R3	
6028	031776	012700	000004	MOV	#4,R0	
6029	032002	022223		3\$: CMP	(R2)+,(R3)+	
6030	032004	001013		BNE	15\$	;BRANCH IF INCORRECT.
6031	032006	077003		SOB	R0,3\$	



```

6033
6034
6035 032010 026104 000062          CMP    62(R1),R4      ;CHECK THE FPS.
6036 032014 001027                BNE    20$           ;BRANCH IF INCORRECT.
6037
6038 032016 026105 000064          CMP    64(R1),R5      ;CHECK THE FEC.
6039 032022 001033                BNE    25$
6040
6041 032024 000161 000074          9$:    JMP    74(R1)   ;RETURN.
6042
6043          ;FRACTIONAL ERROR.
6044 032030
6045 032030 104176          10$:   ;THE ERROR WAS NOT ANTICIPATED SO
6046 032032 000774          12$:   ERROR    +176   ;REPORT THE INCORRECT FRACTION HERE.
6047          BR      9$
6048          ;INTEGER ERROR.
6049 032034 012702 032144          15$:   MOV    #MODDD1,R2   ;WAS THE INTEGER ERROR ANTICIPATED?
6050 032040 010103                MOV    R1,R3
6051 032042 062703 000050                ADD    #50,R3
6052 032046 012705 000004                MOV    #4,R5
6053 032052 022223          60$:   CMP    (R2)+,(R3)+   ;BRANCH IF NOT ANTICIPATED.
6054 032054 001005                BNE    17$
6055 032056 077503                SOB    R5,60$
6056 032060 010102                MOV    R1,R2
6057 032062 062702 000072                ADD    #72,R2
6058          ;THE ERROR WAS ANTICIPATED SO RETURN
6059 032066 000112                JMP    (R2)         ;TO THE ERROR REPORT IN THE CALLING
6060          ;ROUTINE.
6061 032070
6062 032070 104267          16$:   ;THE ERROR WAS NOT ANTICIPATED SO REPORT
6063 032072 000754          17$:   ERROR    +267   ;THE INTEGER FAILURE HERE.
6064          BR      9$
6065          ;FPS INCORRECT.
6066 032074 010437 001254          20$:   MOV    R4,$TMP11    ;REPORT INCORRECT FPS.
6067 032100 016137 000062 001256          MOV    62(R1),$TMP12
6068 032106 104200          21$:   ERROR    +200
6069 032110 000745          BR      9$
6070
6071          ;REPORT FEC ERROR.
6072 032112 010537 001260          25$:   MOV    R5,$TMP13
6073 032116 016137 000064 001262          MOV    64(R1),$TMP14
6074 032124 010102                MOV    R1,R2
6075 032126 062702 000066                ADD    #66,R2
6076 032132 000112                JMP    (R2)
6077
6078 032134 000000 000000 000000 MODDD0: .WORD 0,0,0,0
6079
6080 032144 000000 000000 000000 MODDD1: .WORD 0,0,0,0
6081
6082 032154
        NNNDONE:
        032154 104412          RSETUP
        ;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?).
    
```

6083  
6125

6126

```

.SBTTL TEST # 22 - INTERRUPT CORRECT FLOWS TEST
:*****
:*TEST 22      INTERRUPT CORRECT FLOWS TEST
:*
:*THIS IS A TEST OF THE 'CORRECT' FLOWS. THIS PART OF THE MICRO CODE
:*HAS AS ITS PURPOSE INSURING THAT INTERRUPT REQUESTS MADE DURING
:*CERTAIN LENGTHY FPP INSTRUCTIONS GET HONORED. THIS IS DONE
:*IN A WAY SUCH THAT IF AN INTERRUPT REQUEST OCCURS DURING ONE
:*OF THESE INSTRUCTIONS THE STATE OF THAT INSTRUCTION'S
:*EXECUTION WILL BE THE SAME AS IF THAT INSTRUCTION HAD NEVER
:*BEEN FETCHED AND ITS EXECUTION NEVER STARTED. THUS THE MICRO CODE
:*WILL RESTORE ALL REGISTERS, BACK UP THE PC AND LEAVE THE
:*FPS AND ACO THROUGH AC5 UNMODIFIED.
:*THE INSTRUCTIONS FOR WHICH THIS IS NECESSARY ARE:
:*
:*   ADD (OR SUB)
:*   DIV
:*   MUL
:*   MOD
:*(BOTH DOUBLE AND FLOATING)
:*ALL ADDRESSING MODES WILL BE TRIED WITH THE ADDD INSTRUCTION. THEN
:*EACH OF THE OTHER INSTRUCTIONS WILL BE TRIED USING MODE 1.
:*NOTE THAT THIS TEST NEEDS A SPECIAL INTERRUPT MODULE,
:*WHICH WILL PROBABLY ONLY BE PRESENT IN DEC'S MANUFACTURING ENVIRONMENT,
:*TO RUN. THIS SPECIAL EQUIPMENT IS DESIGNED TO RAISE AN
:*INTERRUPT REQUEST IN THE PROCESSOR IF A BIT IS SET IN ITS STATUS
:*REGISTER AND ONLY WHEN AN FPP INSTRUCTION IS ENCOUNTERED.
:*THEREFORE THIS TEST WILL BE RUN CONDITIONALLY (DEPENDENT UPON WHETHER
:*OR NOT THE STATUS REGISTER OF THE TEST EQUIPMENT TIMES OUT WHEN REFERENCED).
:*THIS TEST CAN ALSO BE DESELECTED BY TURNING SWITCH 7 OF THE
:*SWITCH REGISTER (PHYSICAL OR VIRTUAL) ON.
:*THE TEST ASSUMES THAT THE TEST EQUIPMENT'S STATUS REGISTER IS AT
:*LOCATION 77774 (NOTE THAT ALL REFERENCES TO THIS LOCATION ARE
:*MADE INDIRECT THROUGH THIS PROGRAMS LOCATION CORINT, SO THAT
:*IF THE USER HAS MODIFIED THE TEST EQUIPMENT'S STATUS REGISTER TO
:*RESPOND TO A DIFFERENT ADDRESS LOCATION CORINT MUST BE
:*MADE TO CONTAIN THAT STATUS REGISTER'S NEW ADDRESS).
:*THIS PROGRAM ASSUMES THAT THE TRAP VECTOR FOR THE TEST EQUIPMENT IS 110.
:*AGAIN NOTE THAT ALL REFERENCES TO THIS TRAP VECTOR ARE INDIRECT, THROUGH
:*THIS PROGRAM'S LOCATION CORTRP (IF THE TEST EQUIPMENT IS
:*MADE TO TRAP TO A DIFFERENT VECTOR LOCATION CORTRP MUST CONTAIN THE
:*ADDRESS OF THIS VECTOR).
:*
:*****

```

```

032156 000004
6127
6128 032160 132737 000200 001337
6129 032166 001406
6130 032170 032777 000200 146742
6131
6132 032176 001022
6133 032200 000137 033336
6134
6135 032204 012737 032230 000004
6136 032212 012777 000000 001112
6137 032220 012737 037502 000004
6138 032226 000406
6139

```

```

TST22: SCOPE
        BITB   #200,$ENVM      ;SEE IF APT IS SELECTING TEST.
        BEQ    COR1           ;BRANCH TO AUTOSIZE IF NOT.
        BIT    #200,@SWR      ;SEE IF THE USER HAS SELECTED THIS
                                ;TEST USING THE SWITCH REGISTER.
        BNE    COR3           ;IF SO, PERFORM TEST.
        JMP    CORDONE ;ELSE DO NOT RUN TEST.
COR1:   MOV    #COR2,ERRVECT  ;SEE IF THE TEST EQUIPMENT'S STATUS
        MOV    #0,@CORINT     ;REGISTER TIMES OUT.
        MOV    #CPSPUR,ERRVECT
        BR     COR3           ;DIDN'T TIME OUT SO START TEST.

```



```

6140 032230 022626          COR2:  CMP      (SP)+,(SP)+      ;IF THE REFERENCE TIMES OUT DO
6141 032232 012737 037502 000004      MOV      #CPSPUR,ERRVECT
6142 032240 000137 033336          JMP      CORDONE ;NOT RUN TEST.
6143
6144          ;TEST ADDD MODE 0
6145 032244          COR3:
6146 032244 005227 177777          INC      #-1
6147 032250 001002          BNE     COR33
6148 032252 104401          TYPE
6149 032254 040455          .WORD  CORMES
6150 032256          COR33:
6151 032256 104413          LPERR   ;SET UP THE LOOP ON ERROR ADDRESS.
6152 032260 004737 033036          JSR     PC,CORSUB
6153 032264 040200 000100 000200 1$:  .WORD  40200,100,200,300 ;ACO
6154 032274 123456          2$:  .WORD  123456 ;RO
6155 032276 000200          3$:  200 ;FPS
6156 032300 172000          4$:  ADDD  ACO,ACO ;TEST INSTRUCTION.
6157 032302 000240          NOP
6158 032304 005037 033320          CLR     CORFLG ;RESET INTERRUPT FLAG
6159 032310 104252          ERROR  +252 ;NO INTERRUPT! TEST EQUIPMENT FAILED.
6160 032312 000401          BR      11$
6161 032314 104253          5$:  ERROR  +253 ;INCORRECT STATE AT INTERRUPT.
6162 032316
6163          ;TEST ADDD MODE 1
6164 032316          COR4:
6165 032316 104413          LPERR   ;SET UP THE LOOP ON ERROR ADDRESS.
6166 032320 004737 033036          JSR     PC,CORSUB
6167 032324 040201 000555 077007 1$:  .WORD  40201,555,77007,111111 ;ACO
6168 032334 032324          2$:  .WORD  1$ ;RO
6169 032340 172010          3$:  217 ;FPS
6170 032342 000240          4$:  ADDD  (RO),ACO ;TEST INSTRUCTION
6171 032344 005037 033320          NOP
6172 032350 104252          CLR     CORFLG ;RESET INTERRUPT FLAG
6173 032352 000401          ERROR  +252 ;REPORT FAILURE. NO INTERRUPT.
6174 032354 104254          BR      11$
6175 032356          11$:  ERROR  +254
6176
6177          ;TEST ADDD MODE 2
6178 032356          COR5:
6179 032356 104413          LPERR   ;SET UP THE LOOP ON ERROR ADDRESS.
6180 032360 004737 033036          JSR     PC,CORSUB
6181 032364 040202 111333 052525 1$:  .WORD  40202,111333,52525,70707 ;ACO
6182 032374 032364          2$:  .WORD  1$ ;RO
6183 032376 000205          3$:  205 ;FPS
6184 032400 172020          4$:  ADDD  (RO)+,ACO ;TEST INSTRUCTION
6185 032402 000240          NOP
6186 032404 005037 033320          CLR     CORFLG ;RESET THE INTERRUPT FLAG
6187 032410 104252          ERROR  +252 ;REPORT FAILURE. NO INTERRUPT.
6188 032412 000401          BR      11$
6189 032414 104255          11$:  ERROR  +255 ;CORRECT FLOWS FAILED.
6190
6191          ;TEST ADDD MODE 3
6192 032416          COR6:
6193 032416 104413          LPERR   ;SET UP THE LOOP ON ERROR ADDRESS.
6193 032420 004737 033036          JSR     PC,CORSUB
    
```



```

6194 032424 040203 071735 072746 1$: .WORD 40203,71735,72746,1 ;ACO
6195 032434 032460 2$: .WORD 10$ ;RO
6196 032436 000206 3$: 206 ;FPS
6197 032440 172030 4$: ADDD @ (RO)+,ACO ;TEST INSTRUCTION
6198 032442 000240 NOP
6199 032444 005037 033320 CLR CORFLG ;RESET THE INTERRUPT FLAG
6200 032450 104252 ERROR +252 ;REPORT FAILURE, NO INTERRUPT.
6201 032452 000403 BR 11$
6202 032454 104256 5$: ERROR +256 ;CORRECT FLOWS FAILED.
6203 032456 000401 BR 11$
6204 032460 032424 10$: .WORD 1$ ;USED FOR THE ADDRESSING OF THE OPERAND
6205 ;IN THIS MODE.
6206 032462 11$:
6207 ;TEST ADDD MODE 4
6208 032462 COR7:
6209 032462 104413 LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
6210 032470 004737 033036 JSR PC,CORSUB
6211 032470 040204 123456 070123 1$: .WORD 40204,123456,70123,45671 ;ACO
6212 032500 032500 2$: .WORD 1$+10 ;RO
6213 032502 000212 3$: 212 ;FPS
6214 032504 172040 4$: ADDD -(RO),ACO ;TEST INSTRUCTION
6215 032506 000240 NOP
6216 032510 005037 033320 CLR CORFLG ;RESET THE INTERRUPT FLAG
6217 032514 104252 ERROR +252 ;REPORT FAILURE, NO INTERRUPT.
6218 032516 000401 BR 11$
6219 032520 104257 ERROR +257 ;CORRECT FLOWS FAILED
6220 11$:
6221 ;TEST ADDD MODE 5
6222 032522 COR8:
6223 032522 104413 LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
6224 032524 004737 033036 JSR PC,CORSUB
6225 032530 040205 076543 021076 1$: .WORD 40205,76543,21076,54321 ;ACO
6226 032540 032566 2$: .WORD 10$+2 ;RO
6227 032542 000213 3$: 213 ;FPS
6228 032544 172050 4$: ADDD @-(RO),ACO ;TEST INSTRUCTION
6229 032546 000240 NOP
6230 032550 005037 033320 CLR CORFLG ;RESET THE INTERRUPT FLAG
6231 032554 104252 ERROR +252 ;REPORT ERROR, NO INTERRUPT.
6232 032560 104260 5$: ERROR +260 ;CORRECT FLOWS FAILED.
6233 032562 000401 BR 11$
6234 032564 032530 10$: .WORD 1$
6235 032566 11$:
6236 ;TEST ADDD MODE 6
6237 COR9:
6238 032566 104413 LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
6239 032570 004737 033036 JSR PC,CORSUB
6240 032574 040206 034353 063730 1$: .WORD 40206,34353,63730,31323 ;ACO
6241 032604 032573 2$: .WORD 1$-1 ;RO
6242 032606 000214 3$: 214 ;FPS
6243 032610 172060 000001 4$: ADDD 1 (RO),ACO ;TEST INSTRUCTION
6244 032614 005037 033320 CLR CORFLG
6245 032620 104252 ERROR +252 ;REPORT FAILURE NO TRAP.
6246 032622 000401 BR 11$
6247 032624 104261 5$: ERROR +261
  
```

```

6248 032626          11$:
6249
6250                ;TEST ADDD MODE 7
6251 032626          COR10:
        032626 104413          LPERR                ;SET UP THE LOOP ON ERROR ADDRESS.
        032630 004737 033036          JSR          PC,CORSUB
        032634 040210 070107 062426 1$:          .WORD 40210,70107,62426,55555 ;ACO
        032644 032667          2$:          .WORD 10$-1          ;RO
        032646 000204          3$:          204          ;FPS
        032650 172070 000001          4$:          ADDD @1(R0),ACO          ;TEST INSTRUCTION
        032654 005037 033320          CLR          CORFLG
        032660 104252          ERROR +252          ;REPORT FAILURE NO TRAP
        032662 000403          BR          11$
        032664 104262          5$:          ERROR +262          ;CORRECT FLOWS FAILED.
        032666 000401          BR          11$
        032670 032634          10$:         .WORD 1$
        032672          11$:
6264
6265                ;TEST DIVD MODE 1
6266 032672          COR11:
        032672 104413          LPERR                ;SET UP THE LOOP ON ERROR ADDRESS.
        032674 004737 033036          JSR          PC,CORSUB
        032700 040211 033445 056677 1$:          .WORD 40211,33445,56677,001122          ;ACO
        032710 032700          2$:          .WORD 1$          ;RO
        032712 000205          3$:          205          ;FPS
        032714 174410          4$:          DIVD (R0),ACO          ;TEST INSTRUCTION
        032716 000240          NOP
        032720 005037 033320          CLR          CORFLG
        032724 104252          ERROR +252          ;REPORT FAILURE, NO TRAP.
        032726 000401          BR          11$
        032730 104263          5$:          ERROR +263          ;CORRECT FLOWS FAILED.
        032732          11$:
6278
6279                ;TEST MULD MODE 1
6280 032732          COR12:
        032732 104413          LPERR                ;SET UP THE LOOP ON ERROR ADDRESS.
        032734 004737 033036          JSR          PC,CORSUB
        032740 040212 165411 046252 1$:          .WORD 40212,165411,46252,63650          ;ACO
        032750 032740          2$:          .WORD 1$          ;RO
        032752 000210          3$:          .WORD 210          ;FPS
        032754 171010          4$:          MULD (R0),ACO          ;TEST INSTRUCTION
        032756 000240          NOP
        032760 005037 033320          CLR          CORFLG
        032764 104252          ERROR +252          ;REPORT FAILURE, NO TRAP.
        032766 000401          BR          11$
        032770 104264          5$:          ERROR +264          ;CORRECT FLOWS FAILED.
        032772          11$:
6292
6293                ;TEST MODD MODE 1
6294 032772          COR13:
        032772 104413          LPERR                ;SET UP THE LOOP ON ERROR ADDRESS.
        032774 004737 033036          JSR          PC,CORSUB
        033000 040213 045654 054542 1$:          .WORD 40213,45654,54542,171623          ;ACO
        033010 033000          2$:          .WORD 1$          ;RO
        033012 000412          3$:          .WORD 412          ;FPS
        033014 171410          4$:          MODD (R0),ACO          ;TEST INSTRUCTION.
        033016 000240          NOP
    
```



6301 033020 005037 033320  
 6302 033024 104252  
 6303 033026 000401  
 6304 033030 104265  
 6305 033032 000137 033336  
 6306  
 6307  
 6308  
 6309  
 6310  
 6311  
 6312  
 6313  
 6314  
 6315  
 6316  
 6317  
 6318  
 6319  
 6320  
 6321  
 6322  
 6323  
 6324  
 6325  
 6326  
 6327  
 6328  
 6329  
 6330  
 6331  
 6332  
 6333  
 6334  
 6335  
 6336 033036 005037 033320  
 6337  
 6338 033042 012601  
 6339 033044 010102  
 6340 033046 012700 000200  
 6341 033052 170100  
 6342 033054 172412  
 6343 033056 016100 000012  
 6344 033062 170100  
 6345 033064 016100 000010  
 6346 033070 010102  
 6347 033072 062702 000014  
 6348 033076 010237 001236  
 6349  
 6350 033102 005037 177776  
 6351 033106 012777 033134 000220  
 6352 033114 012737 177777 033320  
 6353  
 6354 033122 012777 177777 000202  
 6355  
 6356 033130 000161 000014  
 6357

```

CLR          CORFLG          ;REPORT FAILURE NO TRAP.
ERROR      +252
BR         11$
5$: ERROR    +265          ;CORRECT FLOWS FAILED.
11$: JMP     CORDONE ;FINISHED TEST!

;THIS SUBROUTINE, CORSUB, IS CALLED TO SET UP THE OPERANDS
;AND CHECK THE RESULTS IN THIS TEST. IT IS CALLED THUS:

          JSR     PC,CORSUB
          1$: .WORD X,X,X,X      ;ACO OPERAND
          2$: .WORD X            ;RO
          3$: .WORD X            ;FPS
          4$: INST              ;TEST INSTRUCTION TO BE
                                ;EXECUTED.
                                ;AN ADDRESS OFFSET FOR
                                ;CERTAIN MODES OR NOP.
                                ;NO TRAP ERROR.
          ADR
          ERROR  +252
          BR    11$
          5$: ERROR +N          ;CORRECT FLOWS FAILURE.
          BR    11$          ;OPTIONAL FOR CERTAIN MODES.
          10$: .WORD ADDRESS  ;OPTIONAL FOR CERTAIN MODES.
          11$:

;CORSUB WILL PICK UP A POINTER TO THE ARGUMENTS, IN R1. ACO, RO AND
;THE FPS WILL BE SET TO THE DESIGNATED VALUES. THEN THE TEST MODULE
;WILL BE SET UP TO INTERRUPT AND THE INSTRUCTION AT 4$ EXECUTED. IF
;NO TRAP OCCURS THEN THE TEST MODULE IS FAULTY. WHEN THE TRAP OCCURS
;THE PC ON THE STACK SHOULD BE 4$, AND ACO, RO AND THE FPS SHOULD NOT
;HAVE BEEN MODIFIED. IF EVERYTHING IS CORRECT CORSUB WILL RETURN TO
;5$ PLUS TWO. IF AN ERROR IS DETECTED THEN CORSUB WILL RETURN TO THE
;ERROR REPORT AT 5$.
;NOTE THAT A FLAG, CORFLG, IS SET TO -1 WHEN AN INTERRUPT IS PENDING.
;CORFLG IS ZERO OTHERWISE.

CORSUB: CLR          CORFLG ;SET FLAG TO INDICATE NO INTERRUPT
                                ;PENDING.
          MOV        (SP)+,R1  ;GET A POINTER TO THE ARGUMENTS.
          MOV        R1,R2    ;SET ACO.
          MOV        #200,R0
          LDFPS     R0
          LDD        (R2),ACO
          MOV        12(R1),R0 ;SET UP THE FPS.
          LDFPS     R0
          MOV        10(R1),R0 ;SET UP RO.
          MOV        R1,R2
          ADD        #14,R2
          MOV        R2,$TMP2 ;SAVE ADDRESS OF INSTRUCTION IN CASE
                                ;OF ERROR.
          CLR        PSW      ;CLEAR THE PRIORITY TO ALLOW INTERRUPTS.
          MOV        #CORTV,@CORTRP ;SET UP THE INTERRUPT VECTOR.
          MOV        #-1,CORFLG ;SET THE FLAG TO INDICATE
                                ;AN INTERRUPT IS PENDING.
          MOV        #-1,@CORINT ;ENABLE THE TEST EQUIPMENT'S
                                ;TRAP FUNCTION AND GO
          JMP        14(R1)    ;EXECUTE THE INSTRUCTION.
    
```



```

6358                                     :TRAP TO HERE WHEN THE INTERRUPT OCCURS.
6359 033134 005137 033320          CORTV: COM   CORFLG ;FIRST SEE IF AN INTERRUPT WAS PENDING.
6360 033140 001060                BNE   CORTV1 ;IF NOT GO REPORT AN ERROR.
6361 033142 012777 000000 000162  MOV   #0,@CORINT ;MAKE SURE THE TEST EQUIPMENT
6362                                     ;IS NOT INTERRUPT ENABLED.
6363
6364 033150 170204                STFPS  R4      ;GET THE FPS.
6365 033152 012702 000200          MOV   #200,R2 ;GET ACO
6366 033156 170102                LDFPS  R2
6367 033160 012702 033322          MOV   #CORTMP,R2
6368 033164 174012                STD   ACO,(R2)
6369 033166 012737 033322 001240  MOV   #CORTMP,$TMP3
6370 033174 010037 001244          MOV   R0,$TMP5
6371 033200 010437 001250          MOV   R4,$TMP7
6372 033204 011637 001254          MOV   (SP),$TMP11
6373 033210 010102                MOV   R1,R2
6374 033212 010237 001242          MOV   R2,$TMP4
6375 033216 062702 000010          ADD   #10,R2
6376 033222 012237 001246          MOV   (R2)+,$TMP6
6377 033226 012237 001252          MOV   (R2)+,$TMP10
6378 033232 010237 001256          MOV   R2,$TMP12
6379 033236 021602                CMP   (SP),R2 ;SEE IF THE TRAP OCCURRED
6380                                     ;AT THE CORRECT ADDRESS.
6381 033240 001016                BNE   CORTV0
6382 033242 022626                CMP   (SP)+,(SP)+ ;RESET THE STACK.
6383 033244 020061 000010          CMP   R0,10(R1) ;SEE IF R0 IS CORRECT.
6384 033250 001012                BNE   CORTV0 ;BR IF NOT CORRECT.
6385 033252 010102                MOV   R1,R2 ;SEE IF ACO WAS CORRECT
6386 033254 012703 033322          MOV   #CORTMP,R3
6387 033260 012705 000004          MOV   #4,R5
6388 033264 022223 1$:          CMP   (R2)+,(R3)+ ;BRANCH IF INCORRECT.
6389 033266 001003                BNE   CORTV0
6390 033270 077503                SOB   R5,1$
6391 033272 000161 000032          JMP   32(R1) ;IF EVERYTHING IS CORRECT THEN RETURN.
6392
6393 033276 000161 000030          CORTV0: JMP   30(R1) ;CORRECT FLOWS FAILED SO GO REPORT ERROR.
6394
6395 033302 011637 001236          CORTV1: MOV   (SP),$TMP2 ;AN INTERRUPT OCCURRED WHEN THE FLAG
6396                                     ;CORFLG, DID NOT INDICATE THAT ONE WAS
6397                                     ;PENDING SO REPORT SPURIOUS TRAP.
6398 033306 005037 033320          CLR   CORFLG
6399 033312 022626                CMP   (SP)+,(SP)+
6400 033314 104266                ERROR +266
6401 033316 000407                BR    CORDONE
6402
6403 033320 000000 000000 000000  CORFLG: .WORD 0
6404 033322 000000 000000 000000  CORTMP: .WORD 0,0,0,0
6405
6406 033332 177774                CORINT: .WORD 177774 ;THIS IS THE ADDRESS, 177774, OF THE
6407                                     ;TEST EQUIPMENT'S STATUS REGISTER.
6408                                     ;THE CONTENTS OF CORINT CAN BE MODIFIED
6409                                     ;IF THIS STATUS REGISTER'S ADDRESS IS
6410                                     ;CHANGED.
6411 033334 000110                CORTRP: .WORD 110 ;THIS IS THE ADDRESS OF THE TEST EQUIPMENTS
6412                                     ;TRAP VECTOR. LIKE THE STATUS REGISTER'S ADDRESS
6413                                     ;DESCRIBED IMMEDIATELY ABOVE
6414                                     ;THIS VECTOR CAN BE CHANGED, BUT THE
    
```

6415  
 6416  
 6417  
 6418 033336 104412  
 033336

CORDONE:  
 RSETUP

:CONTENTS OF CORTRP MUST INDICATE THE  
 :CHANGE.

: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?).

6419  
 6420  
 6421  
 6422 033340  
 6423  
 6424  
 6425  
 6426

TST23:

.SBTTL END OF PASS ROUTINE

::\*\*\*\*\*  
 :\*INCREMENT THE PASS NUMBER (\$PASS)  
 :\*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM  
 :\*IF SW12=1 INHIBIT TRACE TRAP  
 :\*IF THERES A MONITOR GO TO IT  
 :\*IF THERE ISN'T JUMP TO LOOP  
 \$EOP:

033340  
 033340 000004  
 033342 005037 001102  
 033346 005037 001302  
 033352 005237 001324  
 033356 100004  
 033360 005037 001324  
 033364 005237 034106  
 033370 005327  
 033372 000001  
 033374 003402  
 033376 000137 034034  
 033402 012737  
 033404 000001  
 033406 033372  
 033410 005737 001112  
 033414 001007  
 033416 005737 034104  
 033422 001120  
 033424 032737 001777 001324  
 033432 001114  
 033434  
 033434 104401 033442  
 033440 000407  
 033460  
 033460 005737 034106  
 033464 001440  
 033466 013746 034106  
 033472 104403  
 033474 006  
 033475 000

SCOPE  
 CLR \$TSTNM ;:ZERO THE TEST NUMBER  
 CLR \$TIMES ;:ZERO THE NUMBER OF ITERATIONS  
 INC \$PASS ;:INCREMENT THE PASS NUMBER  
 BPL 1000\$ ;:BRANCH IF STILL PLUS ;:DPM002  
 CLR \$PASS ;:CLEAR THE PASS COUNTER ;:DPM002  
 INC \$PASS2 ;:INCREMENT OVERFLOW PASS COUNTER ;:DPM002  
 1000\$: DEC (PC)+ ;:LOOP?  
 \$EOPCT: .WORD 1  
 BLE 999\$ ;:NO ;:DPM002  
 JMP \$DOAGN ;:YES  
 999\$: MOV (PC)+,a(PC)+ ;:RESTORE COUNTER  
 \$ENDCT: .WORD 1  
 \$EOPCT  
 TST \$ERTTL ;:SEE IF ANY ERRORS THIS PASS ;:DPM002  
 BNE 5000\$ ;:BRANCH IF SO TO PRINT THE EOP ;:DPM002  
 TST \$PENDS ;:SEE IF EOP MSGS ARE DISABLED ;:DPM002  
 BNE \$GET42 ;:BRANCH IF SO ;:DPM002  
 BIT #1777,\$PASS ;:PRINT EOP EVERY 2000TH PASS ;:DPM002  
 BNE \$GET42 ;:BRANCH IF NOT MULTIPLE OF 1000 ;:DPM002  
 5000\$: TYPE ,65\$ ;:TYPE ASCIZ STRING  
 BR 64\$ ;:GET OVER THE ASCIZ  
 ;:65\$: .ASCIZ <12><15>/END PASS # /  
 64\$: TST \$PASS2 ;:SEE IF OVERFLOW HAS NON-ZERO VALUE ;:DPM002  
 BEQ 4900\$ ;:BRANCH IF ZERO ;:DPM002  
 MOV \$PASS2,-(SP) ;:SAVE \$PASS2 FOR TYPEOUT  
 ;:TYPE OVERFLOW PASS NUMBER IN OCTAL ;:DPM002  
 TYPOS ;:GO TYPE--OCTAL ASCII  
 .BYTE 6 ;:TYPE 6 DIGITS  
 .BYTE 0 ;:SUPPRESS LEADING ZEROS



```

033476 005737 001324      TST      $PASS      ;SEE IF PASS COUNT IS ZERO      ;DPM002
033502 001007              BNE      3000$      ;BRANCH IF NOT                  ;DPM002
033504 104401 033512      TYPE    ,67$       ;:TYPE ASCIZ STRING
033510 000403              BR       66$        ;:GET OVER THE ASCIZ
                                ;:67$: .ASCIZ !00000!
033520 000426              BR       4910$      ;GO TEST $ERTTL                  ;DPM002
033522 012737 070000 001244 5000$: MOV     #70000,$TMP5 ;CHECK 5TH OCTAL DIGIT FIRST    ;DPM002
033530 033737 001244 001324 4000$: BIT     $TMP5,$PASS ;CHECK TO SEE IF OCTAL DIGIT IS ZERO ;DPM002
033536 001013              BNE     4900$      ;BRANCH OUT IF ZERO             ;DPM002
033540 104401 033546      TYPE    ,69$       ;:TYPE ASCIZ STRING
033544 000401              BR       68$        ;:GET OVER THE ASCIZ
                                ;:69$: .ASCIZ !0!
033550 006237 001244      ASR     $TMP5      ;SHIFT THE THREE BITS RIGHT 3 PLACES ;DPM002
033554 006237 001244      ASR     $TMP5      ;                                ;DPM002
033560 006237 001244      ASR     $TMP5      ;                                ;DPM002
033564 000761              BR       4000$      ;BRANCH BACK TO CHECK $PASS    ;DPM002
033566 013746 001324      MOV     $PASS,-(SP) ;:SAVE $PASS FOR TYPEOUT
                                ;:TYPE PASS NUMBER IN OCTAL
033572 104403              TYPOS   ;:GO TYPE--OCTAL ASCII
033574 006                .BYTE  6           ;:TYPE 6 DIGITS
033575 000                .BYTE  0           ;:SUPPRESS LEADING ZEROS
033576 005737 001112      TST     $ERTTL     ;SEE IF ANY ERRORS THIS PASS    ;DPM002
033602 001426              BEQ     5001$      ;BRANCH AROUND REPORT IF NONE  ;DPM002
033604 104401 033612      TYPE    ,71$       ;:TYPE ASCIZ STRING
033610 000415              BR       70$        ;:GET OVER THE ASCIZ
                                ;:71$: .ASCIZ / TOTAL ERRORS THIS PASS /
033644 013746 001112      MOV     $ERTTL,-(SP) ;:SAVE $ERTTL FOR TYPEOUT
033650 104403              TYPOS   ;:TOTAL NUMBER OF ERRORS IN OCTAL
033652 006                .BYTE  6           ;:GO TYPE--OCTAL ASCII
033653 000                .BYTE  0           ;:TYPE 6 DIGITS
033654 005037 001112      CLR     $ERTTL     ;:SUPPRESS LEADING ZEROS
033660 104401 001313      TYPE    ,SCRLF     ;:CLEAR ERROR TOTAL
033664 105777 145254      $GET42: TSTB    @STKS ;:TYPE CARRIAGE RETURN, LINE FEED
033670 100042              BPL     $GT42C     ;IS A CHARACTER WAITING?      ;DPM002
033672 013737 001146 001244 MOV     $TKB,$TMP5 ;BRANCH IF NOT                  ;DPM002
033700 005737 034104      TST     EPENDS     ;WASTE THE CHARACTER, CLEARING READY ;DPM002
033704 001017              BNE     $DISAB     ;SEE WHICH STATE ENABLE/DISABLE IS IN ;DPM002
033706 005237 034104      INC     EPENDS     ;BRANCH IF EOP'S DISABLED      ;DPM002
033712 104401 033720      TYPE    ,65$       ;:SET FLAG DISABLING PRINTOUTS
033716 000411              BR       64$        ;:TYPE ASCIZ STRING
                                ;:GET OVER THE ASCIZ
                                ;:65$: .ASCIZ <CRLF>!EOP'S DISABLED!<CRLF>
033742 000415              BR       $GT42C     ;BRANCH OVER ENABLE ROUTINE    ;DPM002
033744 005037 034104      $DISAB: CLR     EPENDS ;CLEAR FLAG ENABLING PRINTOUTS ;DPM002
033750 104401 033756      TYPE    ,65$       ;:TYPE ASCIZ STRING
033754 000410              BR       64$        ;:GET OVER THE ASCIZ
                                ;:65$: .ASCIZ <CRLF>!EOP'S ENABLED!<CRLF>
033776 013700 000042      SGT42C: MOV     @#42,R0 ;:GET MONITOR ADDRESS
034002 001414              BEQ     $DOAGN     ;:BRANCH IF NO MONITOR
034004 005046              CLR     -(SP)      ;:INSURE THE 'T' BIT IS CLEAR
034006 012746 034014      MOV     #$CLR.T,-(SP) ;:SETUP FOR AN RTI OR RTT

```



```

034012 000426          BR      $RTRN          ;;GO DO AN RTI OR RTT TO LOAD THE PSW
                                           ;;WITH A CLEARED 'T' BIT
034014 013700 000042  $CLR.T:  MOV      @#42,R0          ;;INSURE R0 CONTAINS THE MONITORS
034014 001405          BEQ      $DOAGN          ;;RETURN ADDRESS
034022 000005          RESET          ;;CLEAR THE WORLD
034024 004710          SENDAD: JSR     PC,(R0)       ;;GO TO MONITOR
034026 000240          NOP          ;;SAVE ROOM
034030 000240          NOP          ;;FOR
034032 000240          NOP          ;;ACT11
034034 104400          SDOAGN: TRAP          ;;PUSH OLD PSW AND PC ON STACK
034036 042716 000020  BIC      #20,(SP)       ;;CLEAR THE 'T' BIT
034042 032777 010000 145070 BIT      #BIT12,@SWR     ;;RUN WITH TRACE TRAP?
034050 001005          BNE      1$          ;;BR IF NO
034052 005137 034076  COM      $TBIT          ;;IS IT TIME FOR TRACE TRAP
034056 100402          BMI      1$          ;;BR IF NO
034060 052716 000020  BIS      #20,(SP)       ;;SET TRACE TRAP
034064 012746 034072  1$:     MOV      #$LOOP,-(SP)    ;;JUMP TO START OF TEST
034070 000002          SRTNAD: RTI          ;;RETURN--THIS IS CHANGED TO
                                           ;;AN 'RTT' IF 'RTT' IS A LEGAL
                                           ;;INSTRUCTION

034072 000137          SLOOP:   JMP      @(PC)+          ;;RETURN
034074 005214          SRTNAD: .WORD   LOOP          ;;RETURN
034076 000000          STBIT:  .WORD   0           ;;'T' BIT STATE INDICATOR
034100 377 000 377 000  SNULL:  .BYTE   -1,-1,0       ;;NULL CHARACTER STRING
                                           .EVEN
034104 000000          EPENDS: .WORD   0           ;;LOCATION FOR EOP PRINT FLAG
034106 000000          $PASS2: .WORD   0           ;;LOCATION FOR PASS COUNT OVERFLOW
                                           :DPM002
                                           :DPM002

```

6427  
6428

.SBTTL SCOPE HANDLER ROUTINE

```

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

```

```

034110 104406          $SCOPE:  CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
034110 032777 040000 145020 1$:     BIT      #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
034112 001131          BNE      $OVER          ;;YES IF SW14=1
034122 000416          :#####START OF CODE FOR THE XOR TESTER#####
          $XTSTR: BR      6$          ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
                                           ;;THIS INSTRUCTION TO A 'NOP' (NOP=240)
034124 013746 000004          MOV      @#ERRVEC,-(SP)    ;;SAVE THE CONTENTS OF THE ERROR VECTOR
034130 012737 034150 000004          MOV      #5$,@#ERRVEC      ;;SET FOR TIMEOUT
034136 005737 177060          TST      @#177060          ;;TIME OUT ON XOR?
034142 012637 000004          MOV      (SP)+,@#ERRVEC    ;;RESTORE THE ERROR VECTOR
034146 000500          BR      $$VLAD          ;;GO TO THE NEXT TEST

```

```

034150 022626          5$:    CMP      (SP)+,(SP)+      ;;CLEAR THE STACK AFTER A TIME OUT
034152 012637 000004    MOV      (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
034156 000440          BR       7$           ;;LOOP ON THE PRESENT TEST
034160          6$:;#####END OF CODE FOR THE XOR TESTER#####
034160 032777 000400 144752    BIT      #BIT08,@SWR    ;;LOOP ON SPEC. TEST?
034166 001404          BEQ      2$           ;;BR IF NO
034170 127737 144744 001102    CMPB    @SWR,$STNM     ;;ON THE RIGHT TEST?   SWR<7:0>
034176 001502          BEQ      $OVER        ;;BR IF YES
034200 013737 177766 034422    2$:    MOV      177766,CPSAVE ;;MOVE CPU ERR REG VALUE TO LOC FOR TST ;DPM001
034206 032737 000001 034422    BIT      #BIT00,CPSAVE ;;SEE IF THE POWER MONITOR BIT IS ON  ;DPM001
034214 001406          BEQ      2000$        ;;BRANCH TO CONTINUE ROUTINE IF CLEAR ;DPM001
034216 042737 000001 177766    BIC      #BIT00,177766 ;;CLEAR THE BIT FOUND TO BE SET      ;DPM001
034224 104177          EMT      +177        ;;CALL SPECIAL POWER FAIL BIT ERROR CALL ;DPM001
034226 105037 001103          CLRB    $ERFLG       ;;CLEAR THE ERROR FLAG FOR NEXT TEST  ;DPM001
034232 105737 001103    2000$: TSTB    $ERFLG       ;;WAS THERE AN ERROR?
034236 001421          BEQ      3$           ;;BR IF NO
034240 123737 001115 001103    CMPB    $ERMAX,$ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
034246 101015          BHI      3$           ;;BR IF NO
034250 032777 001000 144662    BIT      #BIT09,@SWR   ;;LOOP ON ERROR?
034256 001404          BEQ      4$           ;;BR IF NO
034260 013737 001110 001106    7$:    MOV      $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
034266 000446          BR       $OVER
034270 105037 001103    4$:    CLRB    $ERFLG       ;;ZERO THE ERROR FLAG
034274 005037 001302    CLR     $TIMES        ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
034300 000415          BR       1$           ;;ESCAPE TO THE NEXT TEST
034302 032777 004000 144630    3$:    BIT      #BIT11,@SWR ;;INHIBIT ITERATIONS?
034310 001011          BNE     1$           ;;BR IF YES
034312 005737 001324          TST     $PASS        ;;IF FIRST PASS OF PROGRAM
034316 001406          BEQ     1$           ;;      INHIBIT ITERATIONS
034320 005237 001104          INC     $ICNT        ;;INCREMENT ITERATION COUNT
034324 023737 001302 001104    CMP     $TIMES,$ICNT  ;;CHECK THE NUMBER OF ITERATIONS MADE
034332 002024          BGE     $OVER        ;;BR IF MORE ITERATION REQUIRED
034334 012737 000001 001104    1$:    MOV      #1,$ICNT    ;;REINITIALIZE THE ITERATION COUNTER
034342 013737 034420 001302    MOV     $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
034350 105237 001102    $SVLAD: INCB   $STNM     ;;COUNT TEST NUMBERS
034354 113737 001102 001322    MOVB   $STNM,$STESTN ;;SET TEST NUMBER IN APT MAILBOX
034362 011637 001106          MOV     (SP),$LPADR  ;;SAVE SCOPE LOOP ADDRESS
034366 011637 001110          MOV     (SP),$LPERR  ;;SAVE ERROR LOOP ADDRESS
034372 005037 001304          CLR     $ESCAPE     ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
034376 112737 000001 001115    MOVB   #1,$ERMAX     ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
034404 013777 001102 144530    $OVER: MOV     $STNM,@DISPLAY ;;DISPLAY TEST NUMBER
034412 013716 001106          MOV     $LPADR,(SP) ;;FUDGE RETURN ADDRESS
034416 000002          RTI
034420 000001    $MXCNT: 1           ;;FIXES PS
034422 000000    CPSAVE: .WORD 0    ;;MAX. NUMBER OF ITERATIONS
                                ;;LOCATION TO SAVE CPU ERR REG CONTENTS ;DPM001

```

6429  
6430

.SBTTL ERROR HANDLER ROUTINE

```

;*****
;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
;*AND GO TO ERTYPE ON ERROR
;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
;*SW15=1      HALT ON ERROR
;*SW13=1      INHIBIT ERROR TYPEOUTS
;*SW10=1      BELL ON ERROR
;*SW09=1      LOOP ON ERROR

```



```

      ;*CALL
      ;*
      IBSAVE: .WORD 0          ;:LOC'N TO HOLD $ERRPC DURING DUAL ERR ;DPM001
      $ERROR:
      CKSWR          ;:TEST FOR CHANGE IN SOFT-SWR
      7$: INCB        $ERFLG    ;:SET THE ERROR FLAG
      BEQ 7$         ;:DON'T LET THE FLAG GO TO ZERO
      MOV $TSTNM,@DISPLAY ;:DISPLAY TEST NUMBER AND ERROR FLAG
      BIT #BIT10,@SWR  ;:BELL ON ERROR?
      BEQ 1$        ;:NO - SKIP
      TYPE ,SBELL    ;:RING BELL
      1$: INC $ERTTL  ;:COUNT THE NUMBER OF ERRORS
      MOV (SP),$ERRPC ;:GET ADDRESS OF ERROR INSTRUCTION
      SUB #2,$ERRPC
      MOVB @SERRPC,$ITEMB ;:STRIP AND SAVE THE ERROR ITEM CODE
      CMPB #177,$ITEMB ;:SEE IF THIS IS THE POWER FAIL CALL ;DPM001
      BEQ 1000$      ;:BRANCH AROUND ROUTINE IF IT IS ;DPM001
      MOV 177766,CPSAVE ;:MOVE CPU ERR REG TO CPSAVE FOR TEST ;DPM001
      BIT #BIT00,CPSAVE ;:SEE IF POWER MONITOR BIT IS SET ;DPM001
      BEQ 1000$      ;:BRANCH IF OK ;DPM001
      BIC #BIT00,177766 ;:CLEAR THE BIT FOUND SET ;DPM001
      MOV $ERRPC,IBSAVE ;:SAVE $ERRPC ;DPM001
      ERROR +177     ;:CALL SPECIAL POWER MON BIT ERROR ;DPM001
      MOV IBSAVE,$ERRPC ;:RESTORE $ERRPC
      1000$:
      BIT #BIT13,@SWR ;:SKIP TYPEOUT IF SET
      BNE 20$        ;:SKIP TYPEOUTS
      PC,ERTYPE      ;:GO TO USER ERROR ROUTINE
      TYPE ,SCLRF
      20$:
      CMPB #APTENV,$ENV ;:RUNNING IN APT MODE
      BNE 2$         ;:NO,SKIP APT ERROR REPORT
      MOVB $ITEMB,21$ ;:SET ITEM NUMBER AS ERROR NUMBER
      JSR PC,$SATY4  ;:REPORT FATAL ERROR TO APT
      21$:
      .BYTE 0
      .BYTE 0
      22$:
      BR 22$         ;:APT ERROR LOOP
      2$:
      TST IBSAVE     ;:SEE IF POWER FAIL ERROR CALL ;DPM001
      BNE 3$         ;:BRANCH IF NOT - HALT NOT ALLOWED ;DPM001
      TST @SWR
      BPL 3$         ;:HALT ON ERROR
      HALT           ;:SKIP IF CONTINUE
      CKSWR         ;:HALT ON ERROR!
      3$:
      BIT #BIT09,@SWR ;:TEST FOR CHANGE IN SOFT-SWR
      BEQ 4$         ;:LOOP ON ERROR SWITCH SET?
      TST IBSAVE     ;:BR IF NO
      BNE 4$         ;:SEE IF THIS IS THE PWR MNTR BIT ERROR ;DPM001
      MOV $LPERR,(SP) ;:BRANCH IF SO - NO FUDGING ALLOWED ;DPM001
      4$:
      TST $ESCAPE    ;:FUDGE RETURN FOR LOOPING
      BEQ 5$         ;:CHECK FOR AN ESCAPE ADDRESS
      TST IBSAVE     ;:BR IF NONE
      BNE 5$         ;:BR IF NONE
      MOV $ESCAPE,(SP) ;:SEE IF THIS IS THE PWR MNTR BIT ERROR ;DPM001
      5$:
      CMP #SENDAD,42 ;:BRANCH IF SO - NO FUDGING ALLOWED ;DPM001
      BNE 6$         ;:FUDGE RETURN ADDRESS FOR ESCAPE
      ;:ACT-11 AUTO-ACCEPT?
      ;:BRANCH IF NO
  
```



6431  
6432

```
034716 000000          HALT          ;;YES
034720
034720 032777 001000 144212 6$: BIT      #BIT09,@SWR
034726 001013          BNE      ERM10
034730 011637 001162          MOV      (SP), $REGO          ;SEE IF ERROR #377
034734 062737 177776 001162  ADD      #-2,$REGO
034742 122777 000377 144212  CMPB    #377,@$REGO
034750 001002          BNE      ERM10
034752 062716 000002  ADD      #2,(SP)
034756 000002  ERM10: RTI
```

.SBTTL SAVE AND RESTORE R0-R5 ROUTINES  
:\*\*\*\*\*

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

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

:\*RESTORE R0-R5

:\*CALL:  
:\* RESREG  
:\$RESREG:  
MOV (SP)+,22(SP) ;;RESTORE PC OF CALL  
MOV (SP)+,22(SP) ;;RESTORE PS OF CALL  
MOV (SP)+,22(SP) ;;RESTORE PC OF MAIN FLOW  
MOV (SP)+,22(SP) ;;RESTORE PS OF MAIN FLOW  
MOV (SP)+,R5 ;;POP STACK INTO R5  
MOV (SP)+,R4 ;;POP STACK INTO R4  
MOV (SP)+,R3 ;;POP STACK INTO R3  
MOV (SP)+,R2 ;;POP STACK INTO R2  
MOV (SP)+,R1 ;;POP STACK INTO R1  
MOV (SP)+,R0 ;;POP STACK INTO R0  
RTI

6433  
6434

.SBTTL TYPE ROUTINE  
:\*\*\*\*\*  
:\*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.

```

; *THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
; *NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
; *NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
; *NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
; *
; *CALL:
; *1) USING A TRAP INSTRUCTION
; *      TYPE      ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
; *OR
; *      TYPE
; *      MESADR
; *
035054 105737 001157 $TYPE: TSTB $TPFLG      ;; IS THERE A TERMINAL?
035060 100002      BPL 1$      ;; BR IF YES
035062 000000      HALT      ;; HALT HERE IF NO TERMINAL
035064 000430      BR 3$      ;; LEAVE
035066 010046      1$: MOV R0,-(SP)      ;; SAVE R0
035070 017600 000002 MOV @2(SP),R0      ;; GET ADDRESS OF ASCIZ STRING
035074 122737 000001 001336 CMPB #APTENV,$ENV      ;; RUNNING IN APT MODE
035102 001011      BNE 62$      ;; NO, GO CHECK FOR APT CONSOLE
035104 132737 000100 001337 BITB #APTSPOOL,$ENVM      ;; SPOOL MESSAGE TO APT
035112 001405      BEQ 62$      ;; NO, GO CHECK FOR CONSOLE
035114 010037 035124 MOV R0,61$      ;; SETUP MESSAGE ADDRESS FOR APT
035120 004737 035662 JSR PC,$ATY3      ;; SPOOL MESSAGE TO APT
035124 000000      61$: .WORD 0      ;; MESSAGE ADDRESS
035126 132737 000040 001337 62$: BITB #APTCSUP,$ENVM      ;; APT CONSOLE SUPPRESSED
035134 001003      BNE 60$      ;; YES, SKIP TYPE OUT
035136 112046      2$: MOVB (R0)+,-(SP)      ;; PUSH CHARACTER TO BE TYPED ONTO STACK
035140 001005      BNE 4$      ;; BR IF IT ISN'T THE TERMINATOR
035142 005726      TST (SP)+      ;; IF TERMINATOR POP IT OFF THE STACK
035144 012600      60$: MOV (SP)+,R0      ;; RESTORE R0
035146 062716 000002 3$: ADD #2,(SP)      ;; ADJUST RETURN PC
035152 000002      RTI      ;; RETURN
035154 122716 000011 4$: CMPB #HT,(SP)      ;; BRANCH IF <HT>
035160 001430      BEQ 8$      ;; BRANCH IF NOT <CRLF>
035162 122716 000200 CMPB #CRLF,(SP)
035166 001006      BNE 5$      ;; POP <CR><LF> EQUIV
035170 005726      TST (SP)+      ;; TYPE A CR AND LF
035172 104401      TYPE
035174 001313      $CRLF
035176 105037 035414 CLRB $CHARCNT      ;; CLEAR CHARACTER COUNT
035202 000755      BR 2$      ;; GET NEXT CHARACTER
035204 004737 035266 5$: JSR PC,$TYPEC      ;; GO TYPE THIS CHARACTER
035210 123726 001156 6$: CMPB $FILLC,(SP)+      ;; IS IT TIME FOR FILLER CHARS.?
035214 001350      BNE 2$      ;; IF NO GO GET NEXT CHAR.
035216 013746 001154 MOV $NULL,-(SP)      ;; GET # OF FILLER CHARS. NEEDED
                                ;; AND THE NULL CHAR.
035222 105366 000001 7$: DECB 1(SP)      ;; DOES A NULL NEED TO BE TYPED?
035226 002770      BLT 6$      ;; BR IF NO--GO POP THE NULL OFF OF STACK
035230 004737 035266 JSR PC,$TYPEC      ;; GO TYPE A NULL
035234 105337 035414 DECB $CHARCNT      ;; DO NOT COUNT AS A COUNT
035240 000770      BR 7$      ;; LOOP
; HORIZONTAL TAB PROCESSOR
035242 112716 000040 8$: MOVB #' ,(SP)      ;; REPLACE TAB WITH SPACE
035246 004737 035266 9$: JSR PC,$TYPEC      ;; TYPE A SPACE
035252 132737 000007 035414 BITB #7,$CHARCNT      ;; BRANCH IF NOT AT
035260 001372      BNE 9$      ;; TAB STOP

```

```

035262 005726          TST      (SP)+      ;;POP SPACE OFF STACK
035264 000724          BR        2$          ;;GET NEXT CHARACTER
035266          $TYPEC: TSTB     @STKS      ;;CHAR IN KYBD BUFFER?
035266 105777 143652   BPL      10$          ;;BR IF NOT
035272 100022          MOV     @STKB,-(SP)  ;;GET CHAR
035274 017746 143646   RIC      #177600,(SP) ;;STRIP EXTRANEIOUS BITS
035300 042716 177600   CMPB    #SXOFF,(SP)  ;;WAS CHAR XOFF
035304 122716 000023   BNE     102$         ;;BR IF NOT
035310 001012          101$: TSTB     @STKS      ;;WAIT FOR CHAR
035312          BPL      101$          ;;
035312 105777 143626   MOVB    @STKB,(SP)  ;;GET CHAR
035316 100375          BIC     #177600,(SP) ;;STRIP IT
035320 117716 143622   CMPB    #SXON,(SP)  ;;WAS IT XON?
035324 042716 177600   BNE     101$         ;;BR IF NOT
035330 122716 000021          102$: TST      (SP)+      ;;FIX STACK
035334 001366          10$: TSTB     @STPS      ;;WAIT UNTIL PRINTER IS READY
035336 005726          BPL      10$          ;;
035340          CMPB    2(SP),#$XON  ;;IS CHARACTER A RANDOM XON?
035340 105777 143604   BEQ     $TYPEX       ;;BRANCH IF YES
035344 100375          MOVB    2(SP),@STPB  ;;LOAD CHAR TO BE TYPED INTO DATA REG.
035346 126627 000002 000021  CMPB    #CR,2(SP)    ;;IS CHARACTER A CARRIAGE RETURN?
035354 001420          BNE     1$           ;;BRANCH IF NO
035356 116677 000002 143566  CLRB    $CHARCNT     ;;YES--CLEAR CHARACTER COUNT
035364 122766 000015 000002  BR      $TYPEX       ;;EXIT
035372 001003          1$:  CMPB    #LF,2(SP)  ;;IS CHARACTER A LINE FEED?
035374 105037 035414   BEQ     $TYPEX       ;;BRANCH IF YES
035400 000406          INCB    (PC)+       ;;COUNT THE CHARACTER
035402 122766 000012 000002  $CHARCNT: .WORD 0    ;;CHARACTER COUNT STORAGE
035410 001402          $TYPEX: RTS      PC
035412 105227
035414 000000
035416 000207

```

6435  
6436

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
*OCTAL (ASCII) NUMBER AND TYPE IT.
*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
*   MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
*   TYPOS   ;;CALL FOR TYPEOUT
*   .BYTE  N              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
*   .BYTE  M              ;;M=1 OR 0
*                               ;;1=TYPE LEADING ZEROS
*                               ;;0=SUPPRESS LEADING ZEROS
*
*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*$TYPOS OR $TYPOC
*CALL:
*   MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
*   TYPON   ;;CALL FOR TYPEOUT
*
*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
*CALL:
*   MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED

```



```

        :*      TYPOC      ::CALL FOR TYPEOUT
035420 017646 000000      $TYPOS: MOV      @ (SP), -(SP)      ::PICKUP THE MODE
035424 116637 000001 035651 MOVB     1 (SP), $OFILL      ::LOAD ZERO FILL SWITCH
035432 112637 035653      MOVB     (SP)+, $SOMODE+1      ::NUMBER OF DIGITS TO TYPE
035436 062716 000002      ADD      #2, (SP)      ::ADJUST RETURN ADDRESS
035442 000406      BR      $TYPON
035444 112737 000001 035651 $TYPOC: MOVB     #1, $OFILL      ::SET THE ZERO FILL SWITCH
035452 112737 000006 035653      MOVB     #6, $SOMODE+1      ::SET FOR SIX(6) DIGITS
035460 112737 000005 035650 $TYPON: MOVB     #5, $SOCNT      ::SET THE ITERATION COUNT
035466 010346      MOV      R3, -(SP)      ::SAVE R3
035470 010446      MOV      R4, -(SP)      ::SAVE R4
035472 010546      MOV      R5, -(SP)      ::SAVE R5
035474 113704 035653      MOVB     $SOMODE+1, R4      ::GET THE NUMBER OF DIGITS TO TYPE
035500 005404      NEG      R4
035502 062704 000006      ADD      #6, R4      ::SUBTRACT IT FOR MAX. ALLOWED
035506 110437 035652      MOVB     R4, $SOMODE      ::SAVE IT FOR USE
035512 113704 035651      MOVB     $OFILL, R4      ::GET THE ZERO FILL SWITCH
035516 016605 000012      MOV      12 (SP), R5      ::PICKUP THE INPUT NUMBER
035522 005003      CLR      R3      ::CLEAR THE OUTPUT WORD
035524 006105      1$:     ROL      R5      ::ROTATE MSB INTO 'C'
035526 000404      BR      3$
035530 006105      2$:     ROL      R5      ::GO DO MSB
035532 006105      ROL      R5      ::FORM THIS DIGIT
035534 006105      ROL      R5
035536 010503      MOV      R5, R3
035540 006103      3$:     ROL      R3      ::GET LSB OF THIS DIGIT
035542 105337 035652      DECB     $SOMODE      ::TYPE THIS DIGIT?
035546 100021      BPL      7$
035550 042703 177770      BIC      #177770, R3      ::BR IF NO
035554 001002      BNE      4$      ::GET RID OF JUNK
035556 005704      TST      R4      ::TEST FOR 0
035560 001403      BEQ      5$      ::SUPPRESS THIS 0?
035562 005204      4$:     INC      R4      ::BR IF YES
035564 052703 000060      BIS      #'0, R3      ::DON'T SUPPRESS ANYMORE 0'S
035570 052703 000040      5$:     BIS      #' , R3      ::MAKE THIS DIGIT ASCII
035574 122703 000040      CMPB     #' , R3      ::MAKE ASCII IF NOT ALREADY
035600 001404      BEQ      7$      ::IS THIS A SPACE CHARACTER?
035602 110337 035646      MOVB     R3, 8$      ::BRANCH IF SO - DON'T TYPE
035606 104401 035646      TYPE     8$      ::SAVE FOR TYPING
035612 105337 035650      7$:     DECB     $SOCNT      ::GO TYPE THIS DIGIT
035616 003344      BGT      2$      ::COUNT BY 1
035620 002402      BLT      6$      ::BR IF MORE TO DO
035622 005204      INC      R4      ::BR IF DONE
035624 000741      BR      2$      ::INSURE LAST DIGIT ISN'T A BLANK
035626 012605      6$:     MOV      (SP)+, R5      ::GO DO THE LAST DIGIT
035630 012604      MOV      (SP)+, R4      ::RESTORE R5
035632 012603      MOV      (SP)+, R3      ::RESTORE R4
035634 016666 000002 000004      MOV      2 (SP), 4 (SP)      ::RESTORE R3
035642 012616      MOV      (SP)+, (SP)      ::SET THE STACK FOR RETURNING
035644 000002      RTI
035646 000      8$:     .BYTE 0      ::RETURN
035647 000      .BYTE 0      ::STORAGE FOR ASCII DIGIT
035650 000      $SOCNT: .BYTE 0      ::TERMINATOR FOR TYPE ROUTINE
035651 000      $OFILL: .BYTE 0      ::OCTAL DIGIT COUNTER
035652 000000      $SOMODE: .WORD 0      ::ZERO FILL SWITCH
        ::NUMBER OF DIGITS TO TYPE
    
```

:DPM002  
:DPM002

6438

```
.SBTTL APT COMMUNICATIONS ROUTINE
*****
035654 112737 000001 036120 $ATY1: MOVB #1,$FFLG ::TO REPORT FATAL ERROR
035662 112737 000001 036116 $ATY3: MOVB #1,$MFLG ::TO TYPE A MESSAGE
035670 000403
035672 112737 000001 036120 $ATY4: MOVB #1,$FFLG ::TO ONLY REPORT FATAL ERROR
035700 $ATYC:
035700 010046 MOV RO,-(SP) ::PUSH R0 ON STACK
035702 010146 MOV R1,-(SP) ::PUSH R1 ON STACK
035704 105737 036116 TSTB $MFLG ::SHOULD TYPE A MESSAGE?
035710 001450 BEQ 5$ ::IF NOT: BR
035712 122737 000001 001336 CMPB #APTENV,$ENV ::OPERATING UNDER APT?
035720 001031 BNE 3$ ::IF NOT: BR
035722 132737 000100 001337 BITB #APTPOOL,$ENVM ::SHOULD SPOOL MESSAGES?
035730 001425 BEQ 3$ ::IF NOT: BR
035732 017600 000004 MOV @4(SP),R0 ::GET MESSAGE ADDR.
035736 062766 000002 000004 ADD #2,4(SP) ::BUMP RETURN ADDR.
035744 005737 001316 1$: TST $MSGTYPE ::SEE IF DONE W/ LAST XMISSION?
035750 001375 BNE 1$ ::IF NOT: WAIT
035752 010037 001332 MOV RO,$MSGAD ::PUT ADDR IN MAILBOX
035756 105720 2$: TSTB (R0)+ ::FIND END OF MESSAGE
035760 001376 BNE 2$
035762 163700 001332 SUB $MSGAD,R0 ::SUB START OF MESSAGE
035766 006200 ASR RO ::GET MESSAGE LNTH IN WORDS
035770 010037 001334 MOV RO,$MSGGLT ::PUT LENGTH IN MAILBOX
035774 012737 000004 001316 MOV #4,$MSGTYPE ::TELL APT TO TAKE MSG.
036002 000413 BR 5$
036004 017637 000004 036030 3$: MOV @4(SP),4$ ::PUT MSG ADDR IN JSR LINKAGE
036012 062766 000002 000004 ADD #2,4(SP) ::BUMP RETURN ADDRESS
036020 013746 177776 MOV 177776,-(SP) ::PUSH 177776 ON STACK
036024 004737 035054 JSR PC,$TYPE ::CALL TYPE MACRO
036030 000000 4$: .WORD 0
036032 5$:
036032 105737 036120 10$: TSTB $FFLG ::SHOULD REPORT FATAL ERROR?
036036 001416 BEQ 12$ ::IF NOT: BR
036040 005737 001336 TST $ENV ::RUNNING UNDER APT?
036044 001413 BEQ 12$ ::IF NOT: BR
036046 005737 001316 11$: TST $MSGTYPE ::FINISHED LAST MESSAGE?
036052 001375 BNE 11$ ::IF NOT: WAIT
036054 017637 000004 001320 MOV @4(SP),$FATAL ::GET ERROR #
036062 062766 000002 000004 ADD #2,4(SP) ::BUMP RETURN ADDR.
036070 005237 001316 INC $MSGTYPE ::TELL APT TO TAKE ERROR
036074 105037 036120 12$: CLRB $FFLG ::CLEAR FATAL FLAG
036100 105037 036117 CLRB $LFLG ::CLEAR LOG FLAG
036104 105037 036116 CLRB $MFLG ::CLEAR MESSAGE FLAG
036110 012601 MOV (SP)+,R1 ::POP STACK INTO R1
036112 012600 MOV (SP)+,R0 ::POP STACK INTO R0
036114 000207 RTS PC ::RETURN
036116 000 $MFLG: .BYTE 0 ::MESSG. FLAG
036117 000 $LFLG: .BYTE 0 ::LOG FLAG
036120 000 $FFLG: .BYTE 0 ::FATAL FLAG
.EVEN
APTSIZE=200
APTENV=001
APTPOOL=100
APTCSUP=040
```

6439



6440

.SBTTL TTY INPUT ROUTINE

```

*****
. ENABL LSE
*****
*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
*WHEN OPERATING IN TTY FLAG MODE.
036122 022737 000176 001140 $CKSWR: CMP #SWREG,SWR ;; IS THE SOFT-SWR SELECTED?
036130 001074 BNE 15$ ;; BRANCH IF NO
036132 105777 143006 TSTB @STKS ;; CHAR THERE?
036136 100071 BPL 15$ ;; IF NO, DON'T WAIT AROUND
036140 117746 143002 MOVB @STKB,-(SP) ;; SAVE THE CHAR
036144 042716 177600 BIC #^C177,(SP) ;; STRIP-OFF THE ASCII
036150 022726 000007 CMP #7,(SP)+ ;; IS IT A CONTROL G?
036154 001062 BNE 15$ ;; NO, RETURN TO USER
036156 123727 001134 000001 CMPB $AUTOB,#1 ;; ARE WE RUNNING IN AUTO-MODE?
036164 001456 BEQ 15$ ;; BRANCH IF YES
036166 104401 036541 TYPE ,SCNTLG ;; ECHO THE CONTROL-G (^G)
036172 104401 036546 $GTSWR: TYPE ,SMSWR ;; TYPE CURRENT CONTENTS
036176 013746 000176 MOV SWREG,-(SP) ;; SAVE SWREG FOR TYPEOUT
036202 104402 TYPOC ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
036204 104401 036557 TYPE ,SMNEW ;; PROMPT FOR NEW SWR
036210 005046 19$: CLR -(SP) ;; CLEAR COUNTER
036212 005046 CLR -(SP) ;; THE NEW SWR
036214 105777 142724 7$: TSTB @STKS ;; CHAR THERE?
036220 100375 BPL 7$ ;; IF NOT TRY AGAIN
036222 117746 142720 MOVB @STKB,-(SP) ;; PICK UP CHAR
036226 042716 177600 BIC #^C177,(SP) ;; MAKE IT 7-BIT ASCII
036232 021627 000025 9$: CMP (SP),#25 ;; IS IT A CONTROL-U?
036236 001005 BNE 10$ ;; BRANCH IF NOT
036240 104401 036534 TYPE ,SCNTLU ;; YES, ECHO CONTROL-U (^U)
036244 062706 000006 20$: ADD #6,SP ;; IGNORE PREVIOUS INPUT
036250 000757 BR 19$ ;; LET'S TRY IT AGAIN
036252 021627 000015 10$: CMP (SP),#15 ;; IS IT A <CR>?
036256 001022 BNE 16$ ;; BRANCH IF NO
036260 005766 000004 TST 4(SP) ;; YES, IS IT THE FIRST CHAR?
036264 001403 BEQ 11$ ;; BRANCH IF YES
036266 016677 000002 142644 MOV 2(SP),@SWR ;; SAVE NEW SWR
036274 062706 000006 11$: ADD #6,SP ;; CLEAR UP STACK
036300 104401 001313 14$: TYPE ,SCRLF ;; ECHO <CR> AND <LF>
036304 123727 001135 000001 CMPB $INTAG,#1 ;; RE-ENABLE TTY KBD INTERRUPTS?
036312 001003 BNE 15$ ;; BRANCH IF NOT
036314 012777 000100 142622 MOV #100,@STKS ;; RE-ENABLE TTY KBD INTERRUPTS
036322 000002 15$: RTI ;; RETURN
036324 004737 035266 16$: JSR PC,$TYPEC ;; ECHO CHAR
036330 021627 000060 CMP (SP),#60 ;; CHAR < 0?
036334 002420 BLT 18$ ;; BRANCH IF YES
036336 021627 000067 CMP (SP),#67 ;; CHAR > 7?
036342 003015 BGT 18$ ;; BRANCH IF YES
036344 042726 000060 BIC #60,(SP)+ ;; STRIP-OFF ASCII
036350 005766 000002 TST 2(SP) ;; IS THIS THE FIRST CHAR
036354 001403 BEQ 17$ ;; BRANCH IF YES
036356 006316 ASL (SP) ;; NO, SHIFT PRESENT
036360 006316 ASL (SP) ;; CHAR OVER TO MAKE
036362 006316 ASL (SP) ;; ROOM FOR NEW ONE.
036364 005266 000002 17$: INC 2(SP) ;; KEEP COUNT OF CHAR

```



036370 056616 177776  
036374 000707  
036376 104401 001312  
036402 000720

BIS -2(SP),(SP) ::SET IN NEW CHAR  
BR 7\$ ::GET THE NEXT ONE  
18\$: TYPE \$QUES ::TYPE ?<CR><LF>  
BR 20\$ ::SIMULATE CONTROL-U

.DSABL LSB  
\*\*\*\*\*  
\*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY  
\*CALL:

\* RDCHR ::INPUT A SINGLE CHARACTER FROM THE TTY  
\* RETURN HERE ::CHARACTER IS ON THE STACK  
\* ::WITH PARITY BIT STRIPPED OFF

036404 011646  
036406 016666 000004 000002  
036414 105777 142524  
036420 100375  
036422 117766 142520 000004  
036430 042766 177600 000004  
036436 026627 000004 000023  
036444 001013  
036446 105777 142472  
036452 100375  
036454 117746 142466  
036460 042716 177600  
036464 022627 000021  
036470 001366  
036472 000750  
036474 026627 000004 000021  
036502 001744  
036504 026627 000004 000140  
036512 002407  
036514 026627 000004 000175  
036522 003003  
036524 042766 000040 000004  
036532 000002  
036534 136 125 015  
036541 136 107 015  
036546 015 012 123  
036557 040 040 116

\$RDCHR: MOV (SP),-(SP) ::PUSH DOWN THE PC  
MU. 4(SP),2(SP) ::SAVE THE PS  
1\$: TSTB @\$TKS ::WAIT FOR  
BPL 1\$ ::A CHARACTER  
MOVB @\$TKB,4(SP) ::READ THE TTY  
BIC #^C<177>,4(SP) ::GET RID OF JUNK IF ANY  
CMP 4(SP),#23 ::IS IT A CONTROL-S?  
BNE 3\$ ::BRANCH IF NO  
2\$: TSTB @\$TKS ::WAIT FOR A CHARACTER  
BPL 2\$ ::LOOP UNTIL ITS THERE  
MOVB @\$TKB,-(SP) ::GET CHARACTER  
BIC #^C177,(SP) ::MAKE IT 7-BIT ASCII  
CMP (SP)+,#21 ::IS IT A CONTROL-Q?  
BNE 2\$ ::IF NOT DISCARD IT  
BR 1\$ ::YES, RESUME  
3\$: CMP 4(SP),#\$XON ::IS IT A RANDOM XON?  
BEQ 1\$ ::BRANCH IF YES  
CMP 4(SP),#140 ::IS IT UPPER CASE?  
BLT 4\$ ::BRANCH IF YES  
CMP 4(SP),#175 ::IS IT A SPECIAL CHAR?  
BGT 4\$ ::BRANCH IF YES  
BIC #40,4(SP) ::MAKE IT UPPER CASE  
4\$: RTI ::GO BACK TO USER  
\$CNTLU: .ASCIZ /^U/<15><12> ::CONTROL 'U'  
\$CNTLG: .ASCIZ /^G/<15><12> ::CONTROL 'G'  
\$MSWR: .ASCIZ <15><12>/SWR = /  
\$MNEW: .ASCIZ / NEW = /

:RAN001  
:RAN001

6441  
6442

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

036570 010046  
036572 016600 000002  
036576 005740  
036600 111000  
036602 006300  
036604 016000 036624  
036610 000200  
  
036612 011646  
036614 016666 000004 000002  
036622 000002

\$TRAP: MOV RO,-(SP) ::SAVE RO  
MOV 2(SP),RO ::GET TRAP ADDRESS  
TST -(RO) ::BACKUP BY 2  
MOVB (RO),RO ::GET RIGHT BYTE OF TRAP  
ASL RO ::POSITION FOR INDEXING  
MOV \$TRPAD(RO),RO ::INDEX TO TABLE  
RTS RO ::GO TO ROUTINE  
  
::THIS IS USE TO HANDLE THE "GETPRI" MACRO  
\$TRAP2: MOV (SP),-(SP) ::MOVE THE PC DOWN  
MOV 4(SP),2(SP) ::MOVE THE PSW DOWN  
RTI ::RESTORE THE PSW





037030 000002  
037032 000000  
037034 000776  
037036 000000

RTI  
\$ILLUP: HALT  
BR -2  
\$SAVR6: 0

::THE POWER UP SEQUENCE WAS STARTED  
:: BEFORE THE POWER DOWN WAS COMPLETE  
::PUT THE SP HERE

6448  
6449  
6450  
6451

6452  
6453  
6454  
6455  
6456  
6457

6458	037040	104401	001313	
6459	037044	113737	001102	001232
6460	037052	042737	177400	001232
6461	037060	013737	001116	001234
6462	037066	010046		
6463	037070	113700	001114	
6464	037074	042700	177400	
6465	037100	001005		
6466	037102	013746	001116	
6467	037106	104402		
6468	037110	000137	037410	
6469	037114	022700	000377	
6470	037120	001005		
6471	037122	016600	000004	
6472	037126	011000		
6473	037130	062700	000400	
6474	037134	010037	001320	
6475	037140	005300		
6476	037142	006300		
6477	037144	006300		
6478	037146	006300		
6479	037150	062700	001442	
6480	037154	012037	037164	
6481	037160	001404		
6482	037162	104401		
6483	037164	000000		
6484	037166	104401	001313	
6485	037172	012037	037202	
6486	037176	001404		
6487	037200	104401		
6488	037202	000000		
6489	037204	104401	001313	
6490	037210	010146		
6491	037212	010246		
6492	037214	010346		
6493	037216	012001		
6494	037220	001503		
6495	037222	011000		
6496	037224	105710		
6497	037226	001003		
6498	037230	013146		
6499	037232	104402		

.SBTTL ERROR TYPE OUT ROUTINE

\*\*\*\*\*  
\*\*\*\*\*

\*THIS ROUTINE IS CALLED TO TYPE AN ERROR MESSAGE WHICH IS INCLUDED  
\*IN THE ERROR MESSAGE DATA TABLE. IT IS CALLED BY THE \$ERROR ROUTINE  
\*OR BY FIRST SETTING \$ITEMB EQUAL TO THE ERROR TABLE ITEM TO BE PRINTED  
\*OUT AND THEN EXECUTING A:

\* JSR PC,ERTYPE

ERTYPE:	TYPE	\$CRLF	:TYPE A CRLF
	MOVB	\$TSTNM,\$TMP0	:MOVE TEST NUMBER TO \$TMP0
	BIC	#177400,\$TMP0	:CLEAR THE UPPER BYTE
	MOV	\$ERRPC,\$TMP1	:GET PC OF CALL
	MOV	R0,-(SP)	:SAVE R0
	MOVB	\$ITEMB,R0	:GET THE ITEM NUMBER.
	BIC	#177400,R0	:CLEAR THE UPPER BYTE
	BNE	1\$	:BRANCH IF ITEM IS ZERO
	MOV	\$ERRPC,-(SP)	:MOVE THE ERROR PC TO THE STACK FOR PRINTING
	TYPOC		:PRINT THE PC
	JMP	22\$	:JUMP TO EXIT - ALL DONE
1\$:	CMP	#377,R0	:SEE IF ERROR # IS 377
	BNE	3\$	:BRANCH IF NOT
	MOV	4(SP),R0	:MOVE ITEM ADDRESS ON STACK TO R0
	MOV	(R0),R0	:MOVE ITEM TO R0
	ADD	#400,R0	:ADD 400 TO R0
3\$:	MOV	R0,\$FATAL	:SET ITEM NUMBER IN \$FATAL FOR APT ;DPM001
	DEC	R0	:DECREMENT R0 AND
	ASL	R0	:SHIFT THREE TIMES
	ASL	R0	:TO FORM AN INDEX
	ASL	R0	:FOR THE TABLE.
	ADD	#\$ERRTB,R0	:ADD ERROR TABLE START TO R0 - FORMS STARTING ADDRESS
4\$:	MOV	(R0)+,\$5	:PICK UP THE ADDRESS OF THE ERROR MESSAGE
	BEQ	6\$	:BRANCH IF NONE
	TYPE		:TYPE THE MESSAGE
5\$:	.WORD	0	:LOCATION FOR ASCII MESSAGE ADDRESS
	TYPE	\$CRLF	:TYPE A <CRLF>
6\$:	MOV	(R0)+,\$7	:GET THE DATA HEADER
	BEQ	8\$	:BRANCH IF NO HEADER
	TYPE		:TYPE THE HEADER
7\$:	.WORD	0	:LOCATION FOR HEADER ADDRESS
	TYPE	\$CRLF	:TYPE A <CRLF>
8\$:	MOV	R1,-(SP)	:SAVE R1
	MOV	R2,-(SP)	:SAVE R2
	MOV	R3,-(SP)	:SAVE R3
	MOV	(R0)+,R1	:GET VHE ADDRESS OF THE DATA TABLE.
	BEQ	23\$	:BRANCH TO RETURN IF NO DATA.
	MOV	(R0),R0	:GET A POINTER TO THE DATA FORMAT TABLE.
9\$:	TSTB	(R0)	:IS THE FORMAT ZERO?
	BNE	10\$	:BRANCH TO CHECK FOR FORMAT 2 IF NOT
	MOV	@(R1)+,-(SP)	:FORMAT ZERO SO TYPE
	TYPOC		:AN OCTAL NUMBER.



ADDRESS	OPCODE	OPERAND	COMMENT	DESCRIPTION
6500	037234	000463		BR 21\$ ;BRANCH TO PRINT A TAB CHARACTER AFTER TABLE ENTRY
6501	037236	122710	000002	10\$: CMPB #2,(R0) ;IS THE FORMAT TWO?
6502	037242	001005		BNE 11\$ ;BRANCH TO CHECK FOR FORMAT 3 IF NOT
6503	037244	004737	037434	JSR PC,TOCTNM ;TYPE 1ST OCTAL NUMBER
6504	037250	004737	037434	JSR PC,TOCTNM ;TYPE 2ND OCTAL NUMBER
6505	037254	000453		BR 21\$ ;BRANCH TO PRINT A TAB CHARACTER AFTER TABLE ENTRY
6506	037256	122710	000003	11\$: CMPB #3,(R0) ;IS THE FORMAT THREE?
6507	037262	001006		BNE 13\$ ;BRANCH TO CHECK FOR FORMAT 4 IF NOT
6508	037264	012703	000004	MOV #4,R3 ;SET LOOP COUNTER TO TYPE 4 OCTAL NUMBERS
6509	037270	004737	037434	12\$: JSR PC,TOCTNM ;TYPE AN OCTAL NUMBER
6510	037274	077303		SOB R3,12\$ ;SUBTRACT 1 AND BRANCH IF NOT DONE
6511	037276	000442		BR 21\$ ;BRANCH TO PRINT A TAB CHARACTER AFTER TABLE ENTRY
6512				
6513	037300	122710	000004	13\$: CMPB #4,(R0) ;IS THE FORMAT FOUR?
6514	037304	001004		BNE 14\$ ;BRANCH TO CHECK FOR FORMAT 5 IF NOT
6515	037306	013146		MOV @ (R1)+,-(SP) ;MOVE THE DATA TO THE STACK FOR TYPING
6516	037310	104403		TYPOS ;GO TYPE AN OCTAL NUMBER
6517	037312	016		.BYTE 16 ;TYPE UP TO 16 DIGITS
6518	037313	000		.BYTE 0 ;SUPPRESSING LEADING ZEROES.
6519	037314	000433		BR 21\$ ;BRANCH TO PRINT A TAB CHARACTER AFTER TABLE ENTRY
6520	037316	122710	000005	14\$: CMPB #5,(R0) ;IS THE FORMAT FIVE?
6521	037322	001005		BNE 16\$ ;BRANCH TO CHECK FOR FORMAT 11 IF NOT
6522	037324	012137	037332	MOV (R1)+,15\$ ;PUT THE ADDRESS OF THE ASCII STRING IN THE LOCATION
6523	037330	104401		TYPE ;TYPE THE ASCII STRING.
6524	037332	000000		.WORD 0 ;LOCATION FOR THE ADDRESS OF THE ASCII STRING
6525	037334	000425		BR 22\$ ;BRANCH TO INCREMENT R0 AND CONTINUE
6526	037336	122710	000011	16\$: CMPB #11,(R0) ;IS THE FORMAT ELEVEN?
6527	037342	001005		BNE 18\$ ;BRANCH TO CHECK FOR FORMAT 12 IF NOT
6528	037344	013137	037352	MOV @ (R1)+,17\$ ;MOVE THE DATA TO THE STACK FOR TYPING
6529	037350	104401		TYPE ;TYPE THE ASCII STRING.
6530	037352	000000		.WORD 0 ;LOCATION FOR THE ADDRESS OF THE ASCII STRING
6531	037354	000415		BR 22\$ ;BRANCH TO INCREMENT R0 AND CONTINUE
6532	037356	122710	000012	18\$: CMPB #12,(R0) ;IS THE FORMAT TWELVE?
6533	037362	001007		BNE 20\$ ;BRANCH TO HALT IF NOT - FORMAT NOT RECOGNIZED
6534	037364	013102		MOV @ (R1)+,R2 ;MOVE THE DATA TO THE STACK FOR TYPING
6535	037366	012703	000006	MOV #6,R3 ;TYPE SIX OCTAL NUMBERS
6536	037372	004737	037434	19\$: JSR PC,TOCTNM ;TYPE AN OCTAL NUMBER
6537	037376	077303		SOB R3,19\$ ;SUBTRACT 1 AND BRANCH IF NOT DONE YET
6538	037400	000401		BR 21\$ ;BRANCH TO PRINT A TAB CHARACTER AFTER TABLE ENTRY
6539	037402	000000		HALT ;UNDEFINED FORMAT FOR DATA?????
6540	037404	104401	040005	21\$: TYPE ,STAB ;PRINT A TAB AFTER TYPING A DATA TABLE ENTRY
6541				
6542	037410	005200		22\$: INC R0 ;POINT TO THE NEXT FORMAT
6543	037412	005711		TST (R1) ;HAS THE END OF THE DATA TABLE BEEN REACHED?
6544	037414	001303		BNE 9\$ ;BRANCH BACK IF NOT
6545	037416	104401	001313	TYPE ,SRLF ;DONE.
6546	037422	012603		MOV (SP)+,R3 ;RESTORE R1,R2 AND R3
6547	037424	012602		MOV (SP)+,R2
6548	037426	012601		MOV (SP)+,R1
6549	037430	012600		23\$: MOV (SP)+,R0 ;RESTORE R0.
6550	037432	000207		RTS PC ;AND RETURN.
6551				
6552	037434	013102		TOCTNM: MOV @ (R1)+,R2 ;FORMAT TWO SO TYPE TWO
6553	037436	012246		MOV (R2)+,-(SP) ;OCTAL NUMBERS.
6554	037440	104402		TYPOC ;TYPE THE OCTAL NUMBER
6555	037442	104401	040007	TYPE ,SPACE ;TYPE A SPACE CHARACTER
6556	037446	000207		RTS PC ;EXIT

6557  
6558  
6559  
6560  
6561  
6562  
6563 037450 011637 001236  
6564 037454 022626  
6565 037456 170200  
6566 037460 010037 001240  
6567 037464 170300  
6568 037466 010037 001242  
6569 037472 104247  
6570 037474 104412

```
.SBTTL FPP SPURIOUS TRAP TO 244 HANDLER
:*****
:*****
:THIS ROUTINE HANDLES UNEXPECTED TRAPS TO THE FPP TRAP VECTOR AT 244.
:THE LAST FPP INSTRUCTION EXECUTED AND ITS ADDRESS HAS BEEN RECORDED
:THESE ALONG WITH THE FEC, FPS AND PC OF TRAP ARE REPORTED.
:*
FPPSPUR: MOV (SP), $TMP2 ;SAVE PC OF TRAP.
          CMP (SP)+, (SP)+ ;RESTORE SP.
          STFPS R0 ;GET FPS
          MOV R0, $TMP3
          STST R0 ;GET FEC
          MOV R0, $TMP4
1$:      ERROR +247
          RSETUP ;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?).

          JMP $EOP
```

6571 037476 000137 033340  
6572  
6573  
6574  
6575

```
.SBTTL CPU SPURIOUS TRAP TO 4 HANDLER
:*****
:*****
:THIS ROUTINE REPORTS UNEXPECTED CPU TRAPS TO VECTOR 4.
:*
CPSPUR: MOV (SP), $TMP2 ;SAVE PC OF TRAP.
          CMP (SP)+, (SP)+
1$:      ERROR +250
          RSETUP ;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?).

          JMP $EOP
```

6582 037514 000137 033340  
6583  
6584  
6585  
6586

```
.SBTTL CPU SPURIOUS TRAP TO 10 HANDLER
:*****
:*****
:THIS ROUTINE REPORTS UNEXPECTED CPU TRAPS TO VECTOR 10.
:*
CPTWO: MOV (SP), $TMP2 ;SAVE PC OF TRAP.
          CMP (SP)+, (SP)+
1$:      ERROR +251
          RSETUP ;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?).

          JMP $EOP
```

6593 037532 000137 033340  
6594  
6595  
6596  
6597  
6598



```

6599
6600
6601
6602 037536 011637 001110
6603 037542 000002
6604
6605
6606
6607
6608
6609
6610
6611
6612
6613
6614 037544 023727 001140 177570
6615
6616 037552 001001
6617 037554 104406
6618
6619
6620
6621 037556 012737 037450 000244 1$: MOV #FPSPUR,FPVECT
6622 037564 012737 037502 000004 MOV #CPSPUR,ERRVECT
6623 037572 012737 037520 000010 MOV #CPTWO,10
6624 037600 011600 MOV (SP),R0
6625 037602 012706 001100 MOV #STACK,SP
6626 037606 005004 CLR R4
6627 037610 170104 LDFPS R4
6628 037612 000110 JMP (R0)
6629
6630
6631
6632
6633 037614 124 122 101 MSA1: .ASCIZ 'TRAPPED AT:'<TAB><TAB>
6634 037632 105 130 120 MSA2: .ASCIZ 'EXPECTED TRAP AT:'<TAB>
6635 037655 107 117 124 MSA3: .ASCIZ 'GOT R0:'<TAB><TAB>
6636 037667 105 130 120 MSA4: .ASCIZ 'EXPECTED R0:'<TAB>
6637 037705 107 117 124 MSA5: .ASCIZ 'GOT ACO:'<TAB><TAB>
6638 037720 105 130 120 MSA6: .ASCIZ 'EXPECTED ACO:'<TAB><TAB>
6639 037740 200 120 117 POWERM: .ASCIZ <CRLF>'POWER FAILURE. PROGRAM RESTARTING.'<CRLF>
6640 040005 011 000 $TAB: .ASCIZ <TAB>
6641 040007 040 040 000 SPACE: .ASCIZ ' '
6642 040012 101 103 040 MS1: .ASCIZ 'AC OPERAND:'<TAB><TAB>
6643 040030 106 123 122 MS2: .ASCIZ 'FSRC OPERAND:'<TAB><TAB>
6644 040050 101 103 060 MS3: .ASCIZ 'ACO BEFORE EXECUTION:'<TAB>
6645 040077 101 103 060 MS4: .ASCIZ 'ACO AFTER EXECUTION:'<TAB>
6646 040125 105 130 120 MS5: .ASCIZ 'EXPECTED RESULT:'<TAB>
6647 040147 107 117 124 MS6: .ASCIZ 'GOT RESULT:'<TAB><TAB>
6648 040165 106 122 101 MS7: .ASCIZ 'FRACTIONAL RESULT:'<TAB>
6649 040211 111 116 124 MS10: .ASCIZ 'INTEGER RESULT:'<TAB>
6650 040233 105 130 120 MS11: .ASCIZ 'EXPECTED FRACTION:'<TAB>
6651 040257 105 130 120 MS12: .ASCIZ 'EXPECTED INTEGER:'<TAB>
6652 040302 114 117 101 MS37: .ASCIZ 'LOADED DATA:'
6653 040320 122 105 101 MS40: .ASCIZ 'READ DATA:'
    
```

```

.SBTTL SET LOOP ON ERROR ADDRESS ROUTINE
*****
*****
    
```

```

.LPER: MOV (SP),SLPERR
RTI
    
```

```

.SBTTL FLAG RESET AND CONSOLE TEST ROUTINE
*****
*****
    
```

```

*THIS ROUTINE WILL BE CALLED AT THE END OF EACH TEST TO
*RESET THE STACK, CLEAR THE FPS AND SEE IF THE USER HAS TYPED
* CONTROL G ON THE TERMINAL. IF THE USER HAS TYPED CONTROL G AND
*THERE IS NO PHYSICAL CONSOLE SWITCH REGISTER THEN THE CONTENTS
*OF THE SOFTWARE SWITCH REGISTER WILL BE TYPED IN OCTAL ON THE
*TELETYPE AND THE USER CAN MODIFY IT.
.RSET: CMP SWR,#177570 ;SEE IF THERE IS A PHYSICAL
;CONSOLE SWITCH REGISTER.
BNE 1$ ;BRANCH IF NO.
CKSWR ;OTHERWISE TYPE THE CONTENTS
;OF THE PROGRAM VIRTUAL SWITCH REGISTER
;AND GIVE THE USER A CHANCE TO
;MODIFY IT.
    
```

```

1$: MOV #FPSPUR,FPVECT
MOV #CPSPUR,ERRVECT
MOV #CPTWO,10
MOV (SP),R0
MOV #STACK,SP
CLR R4
LDFPS R4
JMP (R0)
;SAVE RETURN ADDRESS.
;RESET THE STACK POINTER.
;CLEAR THE FPS.
;RETURN.
    
```

```

.SBTTL SPECIAL MESSAGES
.ASCIZ 'TRAPPED AT:'<TAB><TAB>
.ASCIZ 'EXPECTED TRAP AT:'<TAB>
.ASCIZ 'GOT R0:'<TAB><TAB>
.ASCIZ 'EXPECTED R0:'<TAB>
.ASCIZ 'GOT ACO:'<TAB><TAB>
.ASCIZ 'EXPECTED ACO:'<TAB><TAB>
.ASCIZ <CRLF>'POWER FAILURE. PROGRAM RESTARTING.'<CRLF>
.ASCIZ <TAB>
.ASCIZ ' '
.ASCIZ 'AC OPERAND:'<TAB><TAB>
.ASCIZ 'FSRC OPERAND:'<TAB><TAB>
.ASCIZ 'ACO BEFORE EXECUTION:'<TAB>
.ASCIZ 'ACO AFTER EXECUTION:'<TAB>
.ASCIZ 'EXPECTED RESULT:'<TAB>
.ASCIZ 'GOT RESULT:'<TAB><TAB>
.ASCIZ 'FRACTIONAL RESULT:'<TAB>
.ASCIZ 'INTEGER RESULT:'<TAB>
.ASCIZ 'EXPECTED FRACTION:'<TAB>
.ASCIZ 'EXPECTED INTEGER:'<TAB>
.ASCIZ 'LOADED DATA:'
.ASCIZ 'READ DATA:'
    
```



SPECIAL MESSAGES

6654	040334	105	130	120	MS415:	.ASCIZ	'EXPECTED DATA: '
6655	040354	104	101	124	MS41:	.ASCIZ	'DATA IN (RO) FSRC: '
6656	040400	104	101	124	MS42:	.ASCIZ	'DATA IN ACO: '
6657	040416	107	117	124	MS43:	.ASCIZ	'GOT RESULT: '
6658	040433	105	130	120	MS44:	.ASCIZ	'EXPECTED RESULT: '
6659	040455	200	124	105	CORMES:	.ASCIZ	<CRLF>'TEST 22, TESTING INTERRUPTS.'<CRLF>

Line No.	Code	Page	Line	Column	Msg No.	Msg Text
6660						.SBTTL ERROR MESSAGES
6661	040514	106	120	123	EM1:	.ASCIZ 'FPS BAD AFTER CMPD (R),A.'
6662	040546	101	103	060	EM2:	.ASCIZ 'ACO MODIFIED BY CMPD (R),A.'
6666	040602				EM3:	
6667	040626	106	120	123		.ASCII 'FPS BAD AFTER CMPD.'<CRLF>
6668	040707	050	102	125		.ASCIZ '(BUT ENBT) STATE 225 WENT TO 475 INSTEAD OF 075.'
6669	040733	106	120	123	EM4:	.ASCII 'FPS BAD AFTER CMPD.'<CRLF>
6670	041014	050	102	125		.ASCIZ '(BUT ENBT) STATE 225 WENT TO 075 INSTEAD OF 475.'
6671	041040	106	120	123	EM5:	.ASCII 'FPS BAD AFTER CMPD.'<CRLF>
6672	041121	050	102	125		.ASCIZ '(BUT ENBT) STATE 035 WENT TO 075 INSTEAD OF 475.'
6673	041145	106	120	123	EM6:	.ASCII 'FPS BAD AFTER CMPD.'<CRLF>
6674	041226	050	102	125		.ASCIZ '(BUT ENBT) STATE 035 WENT TO 475 INSTEAD OF 075.'
6675	041252	106	120	123	EM7:	.ASCII 'FPS BAD AFTER CMPD.'<CRLF>
6676	041333	050	102	125		.ASCIZ '(BUT ENBT Y8) STATE 777 SHOULD HAVE GONE TO 007.'
6677	041357	106	120	123	EM10:	.ASCII 'FPS BAD AFTER CMPD.'<CRLF>
6678	041440	050	102	125		.ASCIZ '(BUT ENBT Y8) STATE 777 SHOULD HAVE GONE TO 405.'
6679	041464	106	120	123	EM11:	.ASCII 'FPS BAD AFTER CMPD.'<CRLF>
6680	041547	050	102	125		.ASCIZ '(BUT NBIT ZBIT) STATE 456 SHOULD HAVE GONE TO 010.'
6681	041573	106	120	123	EM12:	.ASCII 'FPS BAD AFTER CMPD.'<CRLF>
6682	041656	050	102	125		.ASCIZ '(BUT NBIT ZBIT) STATE 456 SHOULD HAVE GONE TO 110.'
6683	041702	106	120	123	EM13:	.ASCII 'FPS BAD AFTER CMPD.'<CRLF>
6684	041772	104	111	104		.ASCIZ '/DIDN'T TAKE THE PATH: STATE 456, TO 012, TO 363 TO 120./
6685	042016	106	120	123	EM14:	.ASCII 'FPS BAD AFTER CMPD.'<CRLF>
6686	042104	050	102	125		.ASCIZ '(BUT XNBT XZBT) STATE 363 WENT TO 140 INSTEAD OF 100.'
6687	042130	106	120	123	EM15:	.ASCII 'FPS BAD AFTER CMPD.'<CRLF>
6688	042216	050	102	125		.ASCIZ '(BUT XNBT XZBT) STATE 363 WENT TO 100 INSTEAD OF 140.'
6689	042242	106	120	123	EM16:	.ASCII 'FPS BAD AFTER CMPD.'<CRLF>
6690	042313	104	111	104		.ASCIZ '/DIDN'T TAKE THE PATH: STATE 777, TO 407./
6691	042370	106	120	123	EM17:	.ASCIZ 'DIVD (R),A TRAPPED TO 244. FSRC=0 AND FID=1.'
6692	042422	106	105	103	EM20:	.ASCIZ 'FPS BAD AFTER DIVD (R),A.'
6693	042454	104	111	126	EM21:	.ASCIZ 'FEC BAD AFTER DIVD (R),A.'
6694	042535	104	111	126	EM22:	.ASCIZ '/DIVD (R),A DIDN'T TRAP TO 244. FSRC=0 AND FID=0./
6695	042560	106	120	123	EM23:	.ASCIZ 'DIVF (R),A FAILED.'
6699	042612				EM32:	.ASCIZ 'FPS BAD AFTER DIVF (R),A.'
6700	042634	104	111	126	EM24:	.ASCII 'DIVF (R),A FAILED.'
6701	042710	050	102	125		.ASCIZ '(BUT Y61) WENT TO STATE 006 INSTEAD OF 206.'
6702	042732	104	111	126	EM25:	.ASCII 'DIVF (R),A FAILED.'
6703	042775	130	117	122		.ASCIZ 'XOR OF SIGN BITS FAILED STATE 470.'
6704	043017	104	111	126	EM26:	.ASCII 'DIVF (R),A FAILED.'
6705	043073	050	102	125		.ASCIZ '(BUT Y61) WENT TO STATE 206 INSTEAD OF 006.'
6706	043073	104	111	126	EM27=EM25 EM30:	.ASCII 'DIVF (R),A FAILED.'

6707	043115	124	122	125		.ASCIZ	'TRUNCATION ERROR. FT=1.'
6708	043145				EM31:		
	043145	104	111	126		.ASCII	'DIVF (R),A FAILED.'
6709	043167	122	117	125		.ASCIZ	'ROUND ERROR. FT=0.'
6710	043212	104	111	126	EM33:	.ASCIZ	'DIVD (R),A FAILED.'
6711	043235	106	120	123	EM34:	.ASCIZ	'FPS BAD AFTER DIVD (R),A.'
6715	043267				EM35:		
	043267	104	111	126		.ASCII	'DIVD (R),A FAILED.'<CRLF>
6716	043312	124	122	125		.ASCIZ	'TRUNCATION ERROR. FT=1.'
6717	043342				EM36:		
	043342	104	111	126		.ASCII	'DIVD (R),A FAILED.'<CRLF>
6718	043365	122	117	125		.ASCIZ	'ROUND ERROR. FT=0.'
6719	043410	115	125	114	EM37:	.ASCIZ	'MULF (R),A FAILED.'
6723	043433	106	120	123	EM40:	.ASCIZ	'FPS BAD AFTER MULF (R),A.'
6724	043465				EM41:		
	043465	115	125	114		.ASCII	'MULF (R),A FAILED.'<CRLF>
6725	043510	123	111	107		.ASCIZ	'SIGN BIT BAD STATE 511.'
6726	043540				EM42:		
	043540	115	125	114		.ASCII	'MULF (R),A FAILED.'<CRLF>
6727	043563	116	117	122		.ASCII	'NORMALIZATION FAILED.'<CRLF>
6728	043611	050	102	125		.ASCIZ	'(BUT Y62) STATE 252 WENT TO 044 INSTEAD OF 444.'
6729	043671				EM43:		
	043671	115	125	114		.ASCII	'MULF (R),A FAILED.'<CRLF>
6730	043714	116	117	122		.ASCII	'NORMALIZATION FAILED.'<CRLF>
6731	043742	050	102	125		.ASCIZ	'(BUT Y62) STATE 252 WENT TO 444 INSTEAD OF 044.'
6732	044022				EM44:		
	044022	115	125	114		.ASCII	'MULF (R),A FAILED.'<CRLF>
6733	044045	122	117	125		.ASCIZ	'ROUND ERROR. FT=0.'
6734	044070				EM45:		
	044070	115	125	114		.ASCII	'MULF (R),A FAILED.'<CRLF>
6735	044113	124	122	125		.ASCIZ	'TRUNCATION ERROR. FT=1.'
6736	044143	106	120	123	EM46:	.ASCIZ	'FPS BAD AFTER MULD (R),A.'
6740	044175	115	125	114	EM246:	.ASCIZ	'MULD (R),A FAILED.'
6741	044220				EM47:		
	044220	115	125	114		.ASCII	'MULD (R),A FAILED.'<CRLF>
6742	044243	102	101	104		.ASCII	'BAD CONSTANT USED IN THE MUL ALGORITHM.'
6743	044312	200	125	123		.ASCIZ	<CRLF>'USED 24 INSTEAD OF 56 STATE 020.'
6744	044354				EM50:		
	044354	115	125	114		.ASCII	'MULD (R),A FAILED.'<CRLF>
6745	044377	124	122	125		.ASCIZ	'TRUNCATION ERROR. FT=1.'
6746	044427				EM51:		
	044427	115	125	114		.ASCII	'MULD (R),A FAILED.'<CRLF>
6747	044452	122	117	125		.ASCIZ	'ROUND ERROR. FT=0.'
6748	044475				EM52:		
	044475	115	125	114		.ASCII	'MULD (R),A FAILED.'<CRLF>
6749	044520	102	101	104		.ASCIZ	'BAD CONSTANT USED IN ROUNDING, FT=0.'
6750	044565	106	120	123	EM111:	.ASCIZ	'FPS BAD AFTER MULF (R),A. EXPECTED OVERFLOW.'
6751	044642	106	120	123	EM112:	.ASCIZ	'FPS BAD AFTER MULF (R),A. EXPECTED UNDERFLOW.'
6752	044720				EM113:		
	044720	115	125	114		.ASCII	'MULF (R),A FAILED.'<CRLF>
6753	044743	105	130	120		.ASCIZ	'EXPECTING OVERFLOW, FIV=0.'
6754	044776				EM114:		
	044776	115	125	114		.ASCII	'MULF (R),A FAILED.'<CRLF>
6755	045021	105	130	120		.ASCIZ	'EXPECTING UNDERFLOW, FIU=0.'
6756	045055	115	125	114	EM115:	.ASCIZ	'MULF (R),A TRAPPED TO 244 ON OVERFLOW. FIV=0.'
6757	045133	115	125	114	EM116:	.ASCIZ	'MULF (R),A TRAPPED TO 244 ON UNDERFLOW. FIU=0.'
6764	045212				EM117:		



6765	045212	115	125	114	.ASCII	'MULF (R),A TRAPPED TO 244 ON UNDERFLOW. FIU=0.'
	045270	050	102	125	.ASCIIZ	'(BUT FIU) STATE 331 WENT TO 155 INSTEAD OF 115.'
6766	045350				EM120:	
6767	045350	115	125	114	.ASCII	'MULF (R),A TRAPPED TO 244 ON UNDERFLOW. FIU=0.'
	045426	050	102	125	.ASCIIZ	'(BUT FIU) STATE 137 WENT TO 155 INSTEAD OF 115.'
6768	045506				EM121:	
6769	045506	115	125	114	.ASCII	'MULF (R),A TRAPPED TO 244 ON OVERFLOW. FIV=0.'
	045563	050	102	125	.ASCIIZ	'(BUT FIV) STATE 333 WENT TO 136 INSTEAD OF 116.'
6770	045643				EM122:	
	045643	115	125	114	.ASCII	'MULF (R),A TRAPPED TO 244 ON OVERFLOW. FIV=0.'
6771	045720	050	102	125	.ASCIIZ	'(BUT FIV) STATE 133 WENT TO 136 INSTEAD OF 116.'
6772	046000	106	120	123	EM123:	'FPS BAD AFTER MULF (R),A. EXPECTING OVERFLOW.'
6773	046056	106	120	123	EM124:	'FPS BAD AFTER MULF (R),A. EXPECTING UNDERFLOW.'
6774	046135				EM125:	
	046135	115	125	114	.ASCII	'MULD (R),A FAILED.'<CRLF>
6775	046160	105	130	120	.ASCIIZ	'EXPECTING OVERFLOW, FIV=0.'
6776	046213				EM126:	
	046213	115	125	114	.ASCII	'MULD (R),A FAILED.'<CRLF>
6777	046236	105	130	120	.ASCIIZ	'EXPECTING UNDERFLOW, FIU=0.'
6778	046272	115	125	114	EM127:	'MULD (R),A TRAPPED TO 244 ON OVERFLOW. FIV=0.'
6779	046350	115	125	114	EM130:	'MULD (R),A TRAPPED TO 244 ON UNDERFLOW. FIU=0.'
6786	046427				EM131:	
	046427	115	125	114	.ASCII	'MULF (R),A TRAPPED TO 244 ON UNDERFLOW. FIU=0.'
6787	046505	050	102	125	.ASCIIZ	'(BUT FIU) STATE 331 WENT TO 155 INSTEAD OF 115.'
6788	046565				EM132:	
	046565	115	125	114	.ASCII	'MULD (R),A FAILED.'<CRLF>
6789	046610	105	130	120	.ASCII	'EXPECTING UNDERFLOW, FIU=0.'
6790	046643	200	050	102	.ASCIIZ	<CRLF>'(BUT FD) STATE 115 WENT TO 424 INSTEAD OF 425.'
6791	046723				EM133:	
	046723	115	125	114	.ASCII	'MULF (R),A TRAPPED TO 244 ON UNDERFLOW. FIU=0.'
6792	047001	050	102	125	.ASCIIZ	'(BUT FIU) STATE 137 WENT TO 155 INSTEAD OF 115.'
6793	047061				EM134:	
	047061	115	125	114	.ASCII	'MULD (R),A TRAPPED TO 244 ON OVERFLOW. FIV=0.'
6794	047136	050	102	125	.ASCIIZ	'(BUT FIV) STATE 333 WENT TO 136 INSTEAD OF 116.'
6795	047216				EM135:	
	047216	115	125	114	.ASCII	'MULD (R),A FAILED.'<CRLF>
6796	047241	105	130	120	.ASCII	'EXPECTING OVERFLOW, FIV=0.'
6797	047273	200	050	102	.ASCIIZ	<CRLF>'(BUT FD) STATE 116 WENT TO 424 INSTEAD OF 425.'
6798	047353				EM136:	
	047353	115	125	114	.ASCII	'MULD (R),A TRAPPED TO 244 ON OVERFLOW. FIV=0.'
6799	047430	050	102	125	.ASCIIZ	'(BUT FIV) STATE 133 WENT TO 136 INSTEAD OF 116.'
6800	047510	106	105	103	EM137:	'FEC BAD AFTER MULF (R),A. EXPECTING OVERFLOW, FEC=10.'
6801	047576	106	105	103	EM140:	'FEC BAD AFTER MULF (R),A. EXPECTING UNDERFLOW, FEC=12.'
6802		044565			EM141=EM111	
6803		044642			EM142=EM112	
6804	047665				EM143:	
	047665	115	125	114	.ASCII	'MULF (R),A FAILED.'<CRLF>
6805	047710	105	130	120	.ASCIIZ	'EXPECTING OVERFLOW, FIV=1.'
6806	047743				EM144:	
	047743	115	125	114	.ASCII	'MULF (R),A FAILED.'<CRLF>
6807	047766	105	130	120	.ASCIIZ	'EXPECTING UNDERFLOW, FIU=1.'
6814	050022				EM145:	
	050022	115	125	114	.ASCII	'MULF (R),A FAILED TO TRAP TO 244 ON UNDERFLOW. FIU=1.'
6815	050107	050	102	125	.ASCIIZ	'(BUT FIU) STATE 331 WENT TO 115 INSTEAD OF 155.'
6816	050167				EM146:	
	050167	115	125	114	.ASCII	'MULF (R),A FAILED TO TRAP TO 244 ON UNDERFLOW. FIU=1.'
6817	050254	050	102	125	.ASCIIZ	'(BUT FIU) STATE 137 WENT TO 115 INSTEAD OF 155.'

6818	050334				EM147:		
	050334	115	125	114	.ASCII	'MULF (R),A FAILED TO TRAP TO 244 ON OVERFLOW, FIV=1.'	
6819	050420	050	102	125	.ASCIZ	'(BUT FIV) STATE 333 WENT TO 116 INSTEAD OF 136.'	
6820	050500				EM150:		
	050500	115	125	114	.ASCII	'MULF (R),A FAILED TO TRAP TO 244 ON OVERFLOW, FIV=1.'	
6821	050564	050	102	125	.ASCIZ	'(BUT FIV) STATE 133 WENT TO 116 INSTEAD OF 136.'	
6822	050644	106	105	103	EM151:	'FEC BAD AFTER MULF (R),A. EXPECTING OVERFLOW, FEC=10.'	
6823	050732	106	105	103	EM152:	'FEC BAD AFTER MULF (R),A. EXPECTING UNDERFLOW, FEC=12.'	
6824		046000			EM153=EM123		
6825		046056			EM154=EM124		
6826	051021				EM155:		
	051021	115	125	114	.ASCII	'MULF (R),A FAILED.'<CRLF>	
6827	051044	105	130	120	.ASCIZ	'EXPECTING OVERFLOW, FIV=1.'	
6828	051077				EM156:		
	051077	115	125	114	.ASCII	'MULF (R),A FAILED.'<CRLF>	
6829	051122	105	130	120	.ASCIZ	'EXPECTING UNDERFLOW, FIU=1.'	
6836	051156				EM157:		
	051156	115	125	114	.ASCII	'MULF (R),A FAILED TO TRAP TO 244 ON UNDERFLOW, FIU=1.'	
6837	051243	050	102	125	.ASCIZ	'(BUT FIU) STATE 331 WENT TO 115 INSTEAD OF 155.'	
6838	051323				EM160:		
	051323	115	125	114	.ASCII	'MULF (R),A FAILED.'<CRLF>	
6839	051346	105	130	120	.ASCII	'EXPECTING UNDERFLOW, FIU=1.'	
6840	051401	200	050	102	.ASCIZ	<CRLF>'(BUT FD) STATE 155 WENT TO 426 INSTEAD OF 427.'	
6841	051461				EM161:		
	051461	115	125	114	.ASCII	'MULF (R),A FAILED TO TRAP TO 244 ON UNDERFLOW, FIU=1.'	
6842	051546	050	102	125	.ASCIZ	'(BUT FIU) STATE 137 WENT TO 115 INSTEAD OF 155.'	
6843	051626				EM162:		
	051626	115	125	114	.ASCII	'MULF (R),A FAILED TO TRAP TO 244 ON OVERFLOW, FIV=1.'	
6844	051712	050	102	125	.ASCIZ	'(BUT FIV) STATE 333 WENT TO 116 INSTEAD OF 136.'	
6845	051772				EM163:		
	051772	115	125	114	.ASCII	'MULF (R),A FAILED.'<CRLF>	
6846	052015	105	130	120	.ASCII	'EXPECTING OVERFLOW, FIV=1.'	
6847	052047	200	050	102	.ASCIZ	<CRLF>'(BUT FD) STATE 700 WENT TO 426 INSTEAD OF 427.'	
6848	052127				EM164:		
	052127	115	125	114	.ASCII	'MULF (R),A FAILED TO TRAP TO 244 ON OVERFLOW, FIV=1.'	
6849	052213	050	102	125	.ASCIZ	'(BUT FIV) STATE 133 WENT TO 116 INSTEAD OF 136.'	
6850	052273	106	120	123	EM55:	'FPS BAD AFTER MODF (R),A.'	
6851	052325	115	117	104	EM53:	'MODF (R),A FRACTION BAD.'	
6852	052356	115	117	104	EM54:	'MODF (R),A INTEGER BAD.'	
6859	052406				EM56:		
	052406	115	117	104	.ASCII	'MODF (R),A FRACTION BAD.'<CRLF>	
6860	052437	101	103	060	.ASCIZ	'ACO DID NOT GET 0 IN STATE 424.'	
6861	052477				EM57:		
	052477	115	117	104	.ASCII	'MODF (R),A INTEGER BAD.'<CRLF>	
6862	052527	101	103	061	.ASCIZ	'AC1 DID NOT GET 0 IN STATE 142.'	
6863	052567				EM60:		
	052567	115	117	104	.ASCII	'MODF (R),A INTEGER BAD.'<CRLF>	
6864	052617	101	103	061	.ASCIZ	'AC1 DID NOT GET THE INTEGER IN STATE 134.'	
6865	052671				EM61:		
	052671	115	117	104	.ASCII	'MODF (R),A FRACTION BAD.'<CRLF>	
6866	052722	101	040	102	.ASCII	'A BAD CONSTANT WAS USED (NOT 24) IN STATE 046.'	
6867	053000	200	117	122	.ASCIZ	<CRLF>'OR (BUT NBIT) STATE 525 WENT TO 050 INSTEAD OF 150.'	
6868	053065				EM62:		
	053065	115	117	104	.ASCII	'MODF (R),A FRACTION BAD.'<CRLF>	
6869	053116	101	040	102	.ASCII	'A BAD CONSTANT WAS USED (NOT 24) IN STATE 046.'	
6870	053174	200	117	122	.ASCIZ	<CRLF>'OR (BUT NBIT) STATE 525 WENT TO 150 INSTEAD OF 050.'	
6871	053261				EM63:		



6872	053261	115	117	104		.ASCII	'MODF (R),A FRACTION BAD.'	<CRLF>
6873	053312	050	102	125		.ASCIZ	'(BUT ZBT) STATE 532 WENT TO 102 INSTEAD OF 122.'	
6874	053372	115	117	104	EM64:	.ASCII	'MODF (R),A FRACTION BAD.'	<CRLF>
6875	053423	050	102	125		.ASCIZ	'(BUT ENBT EZBT) STATE 041 WENT TO 046 INSTEAD OF 246.'	
6876	053511	115	117	104	EM65:	.ASCII	'MODF (R),A FRACTION BAD.'	<CRLF>
6877	053542	050	102	125		.ASCIZ	'(BUT FT) STATE 126 SHOULD HAVE GONE TO 133. FT=0.'	
6878	053624	115	117	104	EM66:	.ASCII	'MODF (R),A FRACTION BAD.'	<CRLF>
6879	053655	123	111	107		.ASCIZ	'SIGN BIT BAD.'	
6880	053673	115	117	104	EM67:	.ASCII	'MODF (R),A FRACTION BAD.'	<CRLF>
6881	053723	123	111	107		.ASCIZ	'SIGN BIT BAD IN STATE 733.'	
6882	053756	115	117	104	EM70:	.ASCIZ	'MODD (R),A FRACTION BAD.'	
6883	054007	115	117	104	EM71:	.ASCIZ	'MODD (R),A INTEGER BAD.'	
6884	054037	106	120	123	EM72:	.ASCIZ	'FPS BAD AFTER MODD (R),A.'	
6891	054071	054037			EM73=EM72			
6892	054071	115	117	104	EM74:	.ASCII	'MODD (R),A INTEGER BAD.'	<CRLF>
6893	054121	050	102	125		.ASCIZ	'(BUT FD) STATE 231 WENT TO 142 INSTEAD OF 143.'	
6894	054200	115	117	104	EM75:	.ASCII	'MODD (R),A FRACTION BAD.'	<CRLF>
6895	054231	101	103	060		.ASCIZ	'ACO GETS 0 IN STATE 425 FAILED.'	
6896	054271	115	117	104	EM76:	.ASCII	'MODD (R),A INTEGER BAD.'	<CRLF>
6897	054321	101	103	061		.ASCIZ	'AC1 GETS 0 IN STATE 143 FAILED.'	
6898	054361	115	117	104	EM77:	.ASCII	'MODD (R),A FRACTION BAD.'	<CRLF>
6899	054412	050	102	125		.ASCIZ	'(BUT FD) STATE 526 WENT TO 134 INSTEAD OF 135.'	
6900	054471	115	117	104	EM100:	.ASCII	'MODD (R),A INTEGER BAD.'	<CRLF>
6901	054521	123	111	107		.ASCIZ	'SIGN BIT BAD IN STATE 526.'	
6902	054554	115	117	104	EM101:	.ASCII	'MODD (R),A FRACTION BAD.'	<CRLF>
6903	054605	101	040	102		.ASCII	'A BAD CONSTANT WAS USED (NOT 56) IN STATE 046.'	
6904	054663	200	117	122		.ASCIZ	<CRLF>'OR (BUT NBIT) STATE 525 WENT TO 050 INSTEAD OF 151.'	
6905	054750	115	117	104	EM102:	.ASCII	'MODD (R),A FRACTION BAD.'	<CRLF>
6906	055001	101	040	102		.ASCII	'A BAD CONSTANT WAS USED (NOT 56) IN STATE 046.'	
6907	055057	200	117	122		.ASCIZ	<CRLF>'OR (BUT NBIT) STATE 525 WENT TO 150 INSTEAD OF 050.'	
6908	055144	115	117	104	EM103:	.ASCII	'MODD (R),A FRACTION BAD.'	<CRLF>
6909	055175	050	102	125		.ASCIZ	'(BUT ZBIT) STATE 532 WENT TO 122 INSTEAD OF 102.'	
6910	055256	115	117	104	EM104:	.ASCII	'MODD (R),A INTEGER BAD.'	<CRLF>
6911	055306	123	105	124		.ASCII	'SET INTEGER IN AC1 FAILED.'	
6912	055340	200	117	122		.ASCIZ	<CRLF>'OR (BUT FD) STATE 733 WENT TO 156 INSTEAD OF 157.'	
6913	055423	115	117	104	EM105:	.ASCII	'MODD (R),A INTEGER BAD.'	<CRLF>
6914	055453	050	102	125		.ASCIZ	'(BUT FD) STATE 122 WENT TO 424 INSTEAD OF 425.'	
6915	055532	115	117	104	EM106:	.ASCII	'MODD (R),A INTEGER BAD.'	<CRLF>
6916	055562	050	102	125		.ASCIZ	'(BUT FD) STATE 246 WENT TO 126 INSTEAD OF 127.'	
6917	055641	115	117	104	EM107:	.ASCII	'MODD (R),A INTEGER BAD.'	<CRLF>
	055671	050	102	125		.ASCIZ	'(BUT FD) STATE 446 WENT TO 126 INSTEAD OF 127.'	



6918	055750				EM110:	.ASCII	'MODD (R),A FRACTION BAD.' <CRLF>
6919	056001	115	117	104		.ASCIIZ	'(BUT FT) STATE 127 WENT TO 313 INSTEAD OF 113.'
6932	056060	050	102	125	EM165:	.ASCII	/MODF (R),A FRACTION BAD. RESULT OVER OR UNDER FLOW./
6933	056143	115	117	104		.BYTE	0
6934	056144				EM166:	.ASCII	/MODF (R),A INTEGER BAD. RESULT OVER OR UNDER FLOW./
6935	056226	115	117	104		.BYTE	0
6936	056227	000			EM167:	.ASCII	/FPS BAD AFTER MODF (R),A./
6937	056260	106	120	123		.BYTE	0
6938	056261	000			EM170:	.ASCII	/FEC BAD AFTER MODF (R),A./
6939	056312	106	105	103		.ASCIIZ	'EXPECTING UNDERFLOW, FEC=12.'
6940	056347	105	130	120	EM171:	.ASCII	/MODF (R),A INTEGER BAD. RESULT OVER OR UNDER FLOW./
6941	056431	115	117	104		.ASCIIZ	<CRLF>'AC1 GETS 0 IN STATE 126 FAILED.'
6942	056472	200	101	103	EM172:	.ASCII	/FEC BAD AFTER MODF (R),A./
6943	056523	106	105	103		.ASCIIZ	'EXPECTING OVERFLOW, FEC=10.'
6944	056557	105	130	120	EM173:	.ASCII	/MODF (R),A INTEGER BAD. RESULT OVER OR UNDER FLOW./
6945	056641	115	117	104		.ASCIIZ	<CRLF>'(BUT FIV FD) STATE 520 WENT TO 142 INSTEAD OF 162.'
6946	056725	200	050	102	EM174:	.ASCII	/MODF (R),A INTEGER BAD. RESULT OVER OR UNDER FLOW./
6947	057007	115	117	104		.ASCIIZ	<CRLF>'(BUT FIV FD) STATE 520 WENT TO 162 INSTEAD OF 142.'
6948	057073	200	050	102	EM175:	.ASCII	/MODF (R),A INTEGER BAD. RESULT OVER OR UNDER FLOW./
6949	057155	115	117	104		.ASCIIZ	<CRLF>'SIGN BAD IN STATE 517.'
6950	057205	200	123	111	EM176:	.ASCII	/MODD (R),A FRACTION BAD. RESULT OVER OR UNDER FLOW./
6951	057270	115	117	104		.BYTE	0
6952	057271	000			EM177:	.ASCIIZ	!POWER MONITOR BIT FOUND SET!
6953	057325	120	117	127	EM200:	.ASCII	/FPS BAD AFTER MODD (R),A./
6954	057325	106	120	123		.BYTE	0
6955	057357	000			EM201:	.ASCII	/FEC BAD AFTER MODD (R),A./
6956	057410	106	105	103		.ASCIIZ	<CRLF>'EXPECTING UNDERFLOW, FEC=12.'
6957	057446	200	105	130	EM202:	.ASCII	/MODD (R),A INTEGER BAD. RESULT OVER OR UNDER FLOW./
6958	057530	115	117	104		.ASCIIZ	<CRLF>'(BUT FD) STATE 241 WENT TO 126 INSTEAD OF 127.'
6959	057610	200	050	102	EM203:	.ASCII	/MODD (R),A INTEGER BAD. RESULT OVER OR UNDER FLOW./
6960	057672	115	117	104		.ASCIIZ	<CRLF>'(BUT FD) STATE 047 WENT TO 126 INSTEAD OF 127.'
6961	057752	200	050	102	EM204:	.ASCII	/FEC BAD AFTER MODD (R),A./
6962	060003	106	105	103		.ASCIIZ	<CRLF>'EXPECTING OVERFLOW, FEC=10.'
6963	060040	200	105	130	EM205:	.ASCII	/MODD (R),A INTEGER BAD. RESULT OVER OR UNDER FLOW./
6964	060122	115	117	104		.ASCIIZ	<CRLF>'(BUT FIV FD) STATE 520 WENT TO 162 INSTEAD OF 163.'
6965	060206	200	050	102	EM206:	.ASCII	/MODD (R),A INTEGER BAD. RESULT OVER OR UNDER FLOW./
6966	060270	115	117	104		.ASCIIZ	<CRLF>'(BUT FIV FD) STATE 520 WENT TO 162 INSTEAD OF 143.'
6967		200	050	102			
6968							

;\*

7061	060354				EM207:	.ASCIZ /ADD (R),A PRODUCED A BAD RESULT./
	060354	101	104	104		
7062	060416				EM210:	.ASCIZ /THE FPS WAS BAD AFTER ADD (R),A./
	060416	124	110	105		
7063	060460	101	104	104	EM211:	.ASCII 'ADD (R),A FAILED IN THE ROUND\TRUNK FLOWS.'
7064	060533	200	127	105		.ASCII <CRLF>'WENT FROM STATE 663 TO 313,'<CRLF>
7065	060570	111	116	123		.ASCIZ 'INSTEAD OF FROM 663 TO 353.'
7090	060624				EM212:	.ASCII 'ADD (R),A FAILED IN THE ROUND\TRUNK FLOWS.'
	060624	101	104	104		.ASCII <CRLF>'THE FPS WAS BAD.'<CRLF>
	060677	200	124	110		.ASCII 'DID NOT TAKE THE PATH:'<CRLF>
	060721	104	111	104		.ASCIZ /FROM STATE 664, TO 505, TO 251./
	060750	106	122	117		
7091	061010				EM213:	.ASCII 'ADD (R),A FAILED IN THE ROUND\TRUNK FLOWS.'
	061010	101	104	104		.ASCII <CRLF>'THE FPS WAS BAD.'<CRLF>
	061063	200	124	110		.ASCII 'DID NOT TAKE THE PATH:'<CRLF>
	061105	104	111	104		.ASCIZ /FROM STATE 664, TO 505, TO 253./
	061134	106	122	117		
7092	061174				EM214:	.ASCII 'ADD (R),A FAILED IN THE ROUND\TRUNK FLOWS.'
	061174	101	104	104		.ASCII <CRLF>'THE FPS WAS BAD.'<CRLF>
	061247	200	124	110		.ASCII 'DID NOT TAKE THE PATH:'<CRLF>
	061271	104	111	104		.ASCIZ /FROM STATE 664, TO 705, TO 735./
	061320	106	122	117		
7093	061360				EM215:	.ASCII 'ADD (R),A FAILED IN THE ROUND\TRUNK FLOWS.'
	061360	101	104	104		.ASCII <CRLF>'THE FPS WAS BAD.'<CRLF>
	061433	200	124	110		.ASCII 'DID NOT TAKE THE PATH:'<CRLF>
	061455	104	111	104		.ASCIZ /FROM STATE 664, TO 705, TO 737./
	061504	106	122	117		
7094	061544	124	110	105	EM216:	.ASCII 'THE (BUT FIU FORK IN THE OVER\UNDER FLOWS FAILED. FIU =1.'
7095	061635	200	127	105		.ASCII <CRLF>'WENT FROM STATE 331 TO 115.'<CRLF>
7096	061672	111	116	123		.ASCIZ 'INSTEAD OF FROM 331 TO 155.'
7097	061726	101	104	104	EM217:	.ASCIZ 'ADD (R)A TRAPPED TO 244,.. FID=1.'
7098	061770				EM220:	.ASCII /ADD (R),A TRAPPED TO 244./<CRLF>
	061770	101	104	104		.ASCII 'THE RESULT WAS AN OVERFLOW CONDITION BUT FIV= 0.'
	062023	124	110	105		.ASCIZ <CRLF>/(BUT FIV) STATE 133 WENT TO 136 INSTEAD OF 116./
	062103	200	050	102		.ASCIZ /INSTEAD OF FROM 133 TO 116./
	062164	111	116	123		
7099	062220				EM221:	.ASCII /ADD (R),A FAILED TO TRAP TO 244./<CRLF>
	062220	101	104	104		.ASCII 'THE RESULT WAS A OVERFLOW CONDITION AND FIV=1.'<CRLF>
	062262	124	110	105		.ASCII /THE (BUT FIV) FORK FAILED./<CRLF>
	062341	124	110	105		.ASCII /WENT FROM STATE 133 TO 116./<CRLF>
	062374	127	105	116		.ASCIZ /INSTEAD OF FROM 133 TO 136./
	062430	111	116	123		
7100	062464				EM222:	.ASCII /ADD (R),A TRAPPED TO 244./<CRLF>
	062464	101	104	104		.ASCII 'THE RESULT WAS AN UNDERFLOW CONDITION BUT FIU= 0.'
	062517	124	110	105		.ASCIZ <CRLF>/(BUT FIU) STATE 331 WENT TO 155 INSTEAD OF 115./
	062600	200	050	102		.ASCIZ /INSTEAD OF FROM 331 TO 115./
	062661	111	116	123		
7101	062715				EM223:	.ASCII /ADD (R),A FAILED TO TRAP TO 244./<CRLF>
	062715	101	104	104		.ASCII 'THE RESULT WAS A UNDERFLOW CONDITION AND FIU=1.'<CRLF>
	062757	124	110	105		.ASCII /THE (BUT FIU) FORK FAILED./<CRLF>
	063037	124	110	105		.ASCII /WENT FROM STATE 331 TO 115./<CRLF>
	063072	127	105	116		.ASCIZ /INSTEAD OF FROM 331 TO 155./
	063126	111	116	123		
7102	063162				EM224:	.ASCII /ADD (R),A TRAPPED TO 244./<CRLF>
	063162	101	104	104		.ASCII 'THE RESULT WAS AN UNDERFLOW CONDITION BUT FIU= 0.'
	063215	124	110	105		.ASCIZ <CRLF>/(BUT FIU) STATE 137 WENT TO 155 INSTEAD OF 115./
	063276	200	050	102		



ERROR MESSAGES

7103	063357	111	116	123	EM225:	.ASCIZ /INSTEAD OF FROM 137 TO 115./
	063413					.ASCII /ADDD (R),A FAILED TO TRAP TO 244./<CRLF>
	063413	101	104	104		.ASCII 'THE RESULT WAS A UNDERFLOW CONDITION AND FIU=1.'<CRLF>
	063455	124	110	105		.ASCII /THE (BUT FIU) FORK FAILED./<CRLF>
	063535	124	110	105		.ASCII /WENT FROM STATE 137 TO 115./<CRLF>
	063570	127	105	116		.ASCIZ /INSTEAD OF FROM 137 TO 155./
7104	063624	111	116	123	EM226:	.ASCII /ADDD (R),A TRAPPED TO 244./
	063660					<CRLF>'BECAUSE OF AN EXPECTED OVERFLOW CONDITION,'<CRLF>
	063660	101	104	104		.ASCII 'BUT THE FEC WAS BAD.'
	063712	200	102	105		.ASCIZ /ADDD (R),A TRAPPED TO 244./
	063766	102	125	124		<CRLF>'BECAUSE OF AN EXPECTED OVERFLOW CONDITION,'<CRLF>
7105	064013				EM227:	.ASCII 'BUT THE FPS WAS BAD.'
	064013	101	104	104		.ASCII /ADDD (R),A TRAPPED TO 244./
	064045	200	102	105		<CRLF>'BECAUSE OF AN EXPECTED OVERFLOW CONDITION,'<CRLF>
	064121	102	125	124		.ASCII 'BUT THE FPS WAS BAD.'
7106	064146				EM230:	.ASCII /ADDD (R),A TRAPPED TO 244./
	064146	101	104	104		<CRLF>'BECAUSE OF AN EXPECTED UNDERFLOW CONDITION,'<CRLF>
	064200	200	102	105		.ASCII 'BUT THE FEC WAS BAD.'
	064255	102	125	124		.ASCIZ /ADDD (R),A TRAPPED TO 244./
7107	064302				EM231:	.ASCII <CRLF>'BECAUSE OF AN EXPECTED UNDERFLOW CONDITION,'<CRLF>
	064302	101	104	104		.ASCII 'BUT THE FPS WAS BAD.'
	064334	200	102	105		.ASCIZ /ADDD (R),A TRAPPED TO 244./
	064411	102	125	124		<CRLF>'BECAUSE OF AN EXPECTED UNDERFLOW CONDITION,'<CRLF>
7108		060416			EM232=EM210	.ASCII 'BUT THE FPS WAS BAD.'
7109						
7110						
7111						
7116						
7120						
7127						
7137						
7138	064436				EM233:	.ASCIZ \LDCFD (R)+,A RESULT INCORRECT.\
	064436	114	104	103		.ASCIZ \LDCFD (R)+,A RESULT INCORRECT.\
7139	064475				EM234:	.ASCII \RO BAD AFTER LDCFD (R)+,A.\
	064475	122	060	040		.ASCIZ <CRLF>'A BAD CONSTANT WAS USED.'
	064527	200	101	040		.ASCIZ \PC BAD AFTER LDCFD #NUM,A. TRAP TO 4.\
7140	064561				EM235:	.ASCIZ \PC BAD AFTER LDCFD #NUM,A.\
	064561	120	103	040		.ASCIZ \PC BAD AFTER LDCFD #NUM,A.\
7141	064627				EM236:	.ASCII \RO BAD AFTER LDCDF (R)+,A.\
	064627	120	103	040		.ASCIZ <CRLF>'A BAD CONSTANT WAS USED.'
7142	064662				EM237:	.ASCIZ \FPS BAD AFTER LDCFD (R)+,A.\
	064662	122	060	040		.ASCII \LDCFD (R)+,A FAILED.\
	064714	200	101	040		.ASCII <CRLF>'THE FD '
7143	064746				EM240:	.ASCII 'BIT WAS NOT COMPLIMENTED '
	064746	106	120	123		.ASCIZ 'IN STATE 017.'
7144	065002				EM241:	.ASCIZ \FPS BAD AFTER LDCDF (R)+,A.\
	065002	114	104	103		.ASCII \LDCFD (R)+,A FAILED.\
	065026	200	124	110		.ASCII <CRLF>'THE FD '
	065036	102	111	124		.ASCII 'BIT WAS NOT COMPLIMENTED '
	065067	111	116	040		.ASCIZ 'IN STATE 017.'
7145	065105				EM242:	.ASCIZ \FPS BAD AFTER LDCDF (R)+,A.\
	065105	106	120	123		.ASCIZ \FPS BAD AFTER LDCDF (R)+,A.\
7146	065141				EM243:	.ASCII \LDCDF (R)+,A FAILED.\
	065141	114	104	103		.ASCII <CRLF>'THE FD '
	065165	200	124	110		.ASCII 'BIT WAS NOT COMPLIMENTED '
	065175	102	111	124		.ASCIZ 'IN STATE 017.'
	065226	111	116	040		.ASCIZ 'IN STATE 017.'



7147	065244				EM244:	.ASCIZ	\LDCDF (R)+,A RESULT INCORRECT.\
	065244	114	104	103			
7148							
7149	065303	114	104	103	EM245:	.ASCII	'LDCFD (R),A FAILED.'
7150	065326	200	123	105		.ASCII	<CRLF>'SET SIGN FAILED '
7151	065347	111	116	040		.ASCIZ	'IN STATE 512.'
7152	065365	125	116	105	EM247:	.ASCIZ	'UNEXPECTED FPP TRAP TO 244.'
7153	065421	125	116	105	EM250:	.ASCIZ	'UNEXPECTED CPU TRAP TO 4.'
7154	065453	125	116	105	EM251:	.ASCIZ	'UNEXPECTED CPU TRAP TO 10.'
7155							
7156	065506	103	117	122	EM252:	.ASCIZ	'CORRECT FLOWS INTERRUPT TEST MODULE FAILED TO INTERRUPT.'
7157							
7161							
7162	065577				EM253:		
	065577	101	104	104		.ASCIZ	/ADDD ACO,ACO FAILED IN THE INTERRUPT CORRECT FLOWS./
7163	065663				EM254:		
	065663	101	104	104		.ASCIZ	/ADDD (RO),ACO FAILED IN THE INTERRUPT CORRECT FLOWS./
7164	065750				EM255:		
	065750	101	104	104		.ASCIZ	/ADDD (RO)+,ACO FAILED IN THE INTERRUPT CORRECT FLOWS./
7165	066036				EM256:		
	066036	101	104	104		.ASCIZ	/ADDD @ (RO)+,ACO FAILED IN THE INTERRUPT CORRECT FLOWS./
7166	066125				EM257:		
	066125	101	104	104		.ASCIZ	/ADDD -(RO),ACO FAILED IN THE INTERRUPT CORRECT FLOWS./
7167	066213				EM260:		
	066213	101	104	104		.ASCIZ	/ADDD @-(RO),ACO FAILED IN THE INTERRUPT CORRECT FLOWS./
7168	066302				EM261:		
	066302	101	104	104		.ASCIZ	/ADDD NUM(RO),ACO FAILED IN THE INTERRUPT CORRECT FLOWS./
7169	066372				EM262:		
	066372	101	104	104		.ASCIZ	/ADDD @NUM(RO),ACO FAILED IN THE INTERRUPT CORRECT FLOWS./
7170	066463				EM263:		
	066463	104	111	126		.ASCIZ	/DIVD (RO),ACO FAILED IN THE INTERRUPT CORRECT FLOWS./
7171	066550				EM264:		
	066550	115	125	114		.ASCIZ	/MULD (RO),ACO FAILED IN THE INTERRUPT CORRECT FLOWS./
7172	066635				EM265:		
	066635	115	117	104		.ASCIZ	/MODD (RO),ACO FAILED IN THE INTERRUPT CORRECT FLOWS./
7173							
7174	066722	103	117	122	EM266:	.ASCIZ	'CORRECT FLOWS INTERRUPT TEST MODULE CAUSED UNEXPECTED INTERRUPT.'
7175	067023				EM267:		
	067023	115	117	104		.ASCII	/MODD (R),A INTEGER BAD. RESULT OVER OR UNDER FLOW./
7176	067105	000				.BYTE	0



7234	067447	DH122=DH115
7235	067106	DH123=DH1
7236	067106	DH124=DH1
7237	067106	DH125=DH1
7238	067106	DH126=DH1
7239	067447	DH127=DH115
7240	067447	DH130=DH115
7241	067447	DH131=DH115
7242	067106	DH132=DH1
7243	067447	DH133=DH115
7244	067447	DH134=DH115
7245	067106	DH135=DH1
7246	067447	DH136=DH115
7247	067447	DH137=DH115
7248	067447	DH140=DH115
7249	067447	DH141=DH115
7250	067447	DH142=DH115
7251	067447	DH143=DH115
7252	067447	DH144=DH115
7253	067106	DH145=DH1
7254	067106	DH146=DH1
7255	067106	DH147=DH1
7256	067106	DH150=DH1
7257	067447	DH151=DH115
7258	067447	DH152=DH115
7259	067447	DH153=DH115
7260	067447	DH154=DH115
7261	067447	DH155=DH115
7262	067447	DH156=DH115
7263	067106	DH157=DH1
7264	067447	DH160=DH115
7265	067106	DH161=DH1
7266	067106	DH162=DH1
7267	067447	DH163=DH115
7268	067106	DH164=DH1
7269	067106	DH53=DH1
7270	067106	DH54=DH1
7271	067106	DH55=DH1
7272	067106	DH56=DH1
7273	067106	DH57=DH1
7274	067106	DH60=DH1
7275	067106	DH61=DH1
7276	067106	DH62=DH1
7277	067106	DH63=DH1
7278	067106	DH64=DH1
7279	067106	DH65=DH1
7280	067106	DH66=DH1
7281	067106	DH67=DH1
7282	067106	DH70=DH1
7283	067106	DH71=DH1
7284	067106	DH72=DH1
7285	067106	DH73=DH1
7286	067106	DH74=DH1
7287	067106	DH75=DH1
7288	067106	DH76=DH1
7289	067106	DH77=DH1
7290	067106	DH100=DH1



7291	067106				DH101=DH1	
7292	067106				DH102=DH1	
7293	067106				DH103=DH1	
7294	067106				DH104=DH1	
7295	067106				DH105=DH1	
7296	067106				DH106=DH1	
7297	067106				DH107=DH1	
7298	067106				DH110=DH1	
7299	067447				DH165=DH115	
7300	067447				DH166=DH115	
7301	067447				DH167=DH115	
7302	067447				DH170=DH115	
7303	067447				DH171=DH115	
7304	067447				DH172=DH115	
7305	067447				DH173=DH115	
7306	067447				DH174=DH115	
7307	067447				DH175=DH115	
7308	067447				DH176=DH115	
7309	067546	040	124		DH177: .ASCIZ	' TEST.<TAB>IERR PC<TAB>ICPU ERROR REGISTER'
7310	067447				DH200=DH115	
7311	067447				DH201=DH115	
7312	067447				DH202=DH115	
7313	067447				DH203=DH115	
7314	067447				DH204=DH115	
7315	067447				DH205=DH115	
7316	067447				DH206=DH115	
7317	067607	040	124		DH220: .ASCIZ	' TEST.<TAB>PC OF CALL.<TAB>PC OF TRAP.'
7318	067647	040	124		DH207: .ASCII	' TEST.<TAB>PC OF CALL.<TAB>PC OF ERROR.'
7319	067707	011	107		.ASCIZ	<TAB>'GOT FPS.<TAB>'EXPECTED FPS.'
7320	067737	040	124		DH210: .ASCIZ	' TEST.<TAB>PC OF CALL.<TAB>PC OF ERROR.'
7321	067737				DH211=DH210	
7322	067647				DH212=DH207	
7323	067647				DH213=DH207	
7324	067647				DH214=DH207	
7325	067647				DH215=DH207	
7326	067737				DH216=DH210	
7327	070000	040	124		DH217: .ASCII	' TEST.<TAB>PC OF CALL.<TAB>PC OF ERROR.'
7328	070040	011	106		.ASCIZ	<TAB>'FPS.<TAB>'FEC.'
7329	067737				DH221=DH210	
7330	067607				DH222=DH220	
7331	067737				DH223=DH210	
7332	067607				DH224=DH220	
7333	067737				DH225=DH210	
7334	070053	040	124		DH226: .ASCII	' TEST.<TAB>PC OF CALL.<TAB>PC OF TRAP.'
7335	070112	011	107		.ASCIZ	<TAB>'GOT FEC.<TAB>'EXPECTED FEC.'
7336	070142	040	124		DH227: .ASCII	' TEST.<TAB>PC OF CALL.<TAB>PC OF TRAP.'
7337	070201	011	107		.ASCIZ	<TAB>'GOT FPS.<TAB>'EXPECTED FPS.'
7338	070053				DH230=DH226	
7339	070142				DH231=DH227	
7340	070142				DH232=DH227	
7341						
7342						
7343	070231	040	124		DH233: .ASCIZ	' TEST.<TAB>PC OF CALL.<TAB>PC OF ERROR.'
7344						
7345	070272	040	124		DH234: .ASCII	' TEST.<TAB>PC OF CALL.<TAB>PC OF ERROR.'
7346	070332	011	107		.ASCIZ	<TAB>'GOT RO.<TAB>'EXPECTED RO.'
7347						

7348	070360	040	040	124	DH235:	.ASCII	' TEST.' <tab>'PC OF CALL.'<tab></tab></tab>
7349	070404	120	103	040		.ASCII	'PC OF TRAP.' <tab></tab>
7350							
7351	070420	040	040	124	DH236:	.ASCII	' TEST.' <tab>'PC OF CALL.'<tab></tab></tab>
7352	070444	120	103	040		.ASCII	'PC OF ERROR.' <tab>'GOT PC.'</tab>
7353	070470	011	105	130		.ASCII	<TAB>'EXPECTED PC.'
7354							
7355	070272						DH237=DH234
7356							
7357	070506	040	040	124	DH240:	.ASCII	' TEST.' <tab>'PC OF CALL.'<tab></tab></tab>
7358	070532	120	103	040		.ASCII	'PC OF ERROR.' <tab></tab>
7359	070547	107	117	124		.ASCII	'GOT FPS.' <tab>'EXPECTED FPS.'</tab>
7360							
7361	070231						DH241=DH233
7362	070506						DH242=DH240
7363	070231						DH243=DH233
7364	070231						DH244=DH233
7365	070231						DH245=DH233
7366	070576	040	040	124	DH247:	.ASCII	' TEST.' <tab>'PC OF CALL.'<tab>'PC OF TRAP.'<tab>'FEC.'</tab></tab></tab>
7367	070643	040	040	124	DH250:	.ASCII	' TEST.' <tab>'PC OF CALL.'<tab>'PC OF TRAP.'</tab></tab>
7368	070643						DH251=DH250
7369							
7370	067176						DH252=DH2
7371	067106						DH253=DH1
7372	067106						DH254=DH1
7373	067106						DH255=DH1
7374	067106						DH256=DH1
7375	067106						DH257=DH1
7376	067106						DH260=DH1
7377	067106						DH261=DH1
7378	067106						DH262=DH1
7379	067106						DH263=DH1
7380	067106						DH264=DH1
7381	067106						DH265=DH1
7382	070703	040	040	124	DH266:	.ASCII	' TEST.' <tab>'PC OF CALL.'<tab>'PC OF TRAP.'</tab></tab>
7383	067447						DH267=DH115

Line	Code	Address	Value	DF	Format	Specifiers
7384					.SBTTL	DATA FORMAT SPECIFIERS
7385	070743	004	000	005	DF1:	.BYTE 4,0,5,0,5,0,5,0,5,5,3,5,5,3
7386	070761	004	000	005	DF2:	.BYTE 4,0,5,0,5,5,3,5,5,3
7387		070743			DF3=DF1	
7388		070743			DF4=DF1	
7389		070743			DF5=DF1	
7390		070743			DF6=DF1	
7391		070743			DF7=DF1	
7392		070743			DF10=DF1	
7393		070743			DF11=DF1	
7394		070743			DF12=DF1	
7395		070743			DF13=DF1	
7396		070743			DF14=DF1	
7397		070743			DF15=DF1	
7398		070743			DF16=DF1	
7399	070773	004	000	005	DF17:	.BYTE 4,0,5,0,5,0,0
7400	071002	004	000	005	DF20:	.BYTE 4,0,5,0,5,0,5,0
7401		071002			DF21=DF20	
7402		071002			DF22=DF20	
7403	071012	004	000	005	DF23:	.BYTE 4,0,5,0,5,0,5,0,5,5,2,5,5,2,5,5,2,5,5,2
7404		071012			DF24=DF23	
7405		071012			DF32=DF23	
7406		071012			DF25=DF23	
7407		071012			DF26=DF23	
7408		071012			DF27=DF23	
7409		071012			DF30=DF23	
7410		071012			DF31=DF23	
7411	071036	004	000	005	DF33:	.BYTE 4,0,5,0,5,0,5,0,5,5,3,5,5,3,5,5,3,5,5,3
7412		071036			DF34=DF33	
7413		071036			DF35=DF33	
7414		071036			DF36=DF33	
7415		071012			DF37=DF23	
7416		071012			DF40=DF23	
7417		071012			DF41=DF23	
7418		071012			DF42=DF23	
7419		071012			DF43=DF23	
7420		071012			DF44=DF23	
7421		071012			DF45=DF23	
7422		071036			DF246=DF33	
7423		071036			DF46=DF33	
7424		071036			DF47=DF33	
7425		071036			DF50=DF33	
7426		071036			DF51=DF33	
7427		071036			DF52=DF33	
7428		071012			DF111=DF23	
7429		071012			DF112=DF23	
7430		071012			DF113=DF23	
7431		071012			DF114=DF23	
7432	071062	004	000	005	DF115:	.BYTE 4,0,5,0,5,0,0,0,5,5,2,5,5,2,5,5,2,5,5,2
7433		071062			DF116=DF115	
7434		071062			DF117=DF115	
7435		071062			DF120=DF115	
7436		071062			DF121=DF115	
7437		071062			DF122=DF115	
7438		071036			DF123=DF33	
7439		071036			DF124=DF33	
7440		071036			DF125=DF33	



DATA FORMAT SPECIFIERS

7441		071036			DF126=DF33	
7442	071106	004	000	005	DF127: .BYTE	4,0,5,0,5,0,0,0,5,5,3,5,5,3,5,5,3,5,5,3
7443		071106			DF130=DF127	
7444		071106			DF131=DF127	
7445		071036			DF132=DF33	
7446		071106			DF133=DF127	
7447		071106			DF134=DF127	
7448		071036			DF135=DF33	
7449		071106			DF136=DF127	
7450		071062			DF137=DF115	
7451		071062			DF140=DF115	
7452		071062			DF141=DF115	
7453		071062			DF142=DF115	
7454		071062			DF143=DF115	
7455		071062			DF144=DF115	
7456		071012			DF145=DF23	
7457		071012			DF146=DF23	
7458		071012			DF147=DF23	
7459		071012			DF150=DF23	
7460		071106			DF151=DF127	
7461		071106			DF152=DF127	
7462		071106			DF153=DF127	
7463		071106			DF154=DF127	
7464		071106			DF155=DF127	
7465		071106			DF156=DF127	
7466		071036			DF157=DF33	
7467		071106			DF160=DF127	
7468		071036			DF161=DF33	
7469		071036			DF162=DF33	
7470		071106			DF163=DF127	
7471		071036			DF164=DF33	
7472	071132	004	000	005	DF53: .BYTE	4,0,5,0,5,0,5,0,5,5,2,5,5,2,5,5,2,5,5,2,5,5,2
7473		071132			DF54=DF53	
7474		071132			DF55=DF53	
7475		071132			DF56=DF53	
7476		071132			DF57=DF53	
7477		071132			DF60=DF53	
7478		071132			DF61=DF53	
7479		071132			DF62=DF53	
7480		071132			DF63=DF53	
7481		071132			DF64=DF53	
7482		071132			DF65=DF53	
7483		071132			DF66=DF53	
7484		071132			DF67=DF53	
7485	071164	004	000	005	DF70: .BYTE	4,0,5,0,5,0,5,0,5,5,3,5,5,3,5,5,3,5,5,3,5,5,3
7486		071164			DF71=DF70	
7487		071164			DF72=DF70	
7488		071164			DF73=DF70	
7489		071164			DF74=DF70	
7490		071164			DF75=DF70	
7491		071164			DF76=DF70	
7492		071164			DF77=DF70	
7493		071164			DF100=DF70	
7494		071164			DF101=DF70	
7495		071164			DF102=DF70	
7496		071164			DF103=DF70	
7497		071164			DF104=DF70	

7498		071164			DF105=DF70	
7499		071164			DF106=DF70	
7500		071164			DF107=DF70	
7501		071164			DF110=DF70	
7502	071216	004	000	005	DF165: .BYTE	4,0,5,0,5,0,0,0,5,5,2,5,5,2,5,5,2,5,5,2,5,5,2,5,5,2
7503		071216			DF166=DF165	
7504		071216			DF167=DF165	
7505		071216			DF170=DF165	
7506		071216			DF171=DF165	
7507		071216			DF172=DF165	
7508		071216			DF173=DF165	
7509		071216			DF174=DF165	
7510		071216			DF175=DF165	
7511	071250	004	000	005	DF176: .BYTE	4,0,5,0,5,0,0,0,5,5,3,5,5,3,5,5,3,5,5,3,5,5,3,5,5,3
7512	071302	004	000	000	DF177: .BYTE	4,0,0
7513		071250			DF200=DF176	
7514		071250			DF201=DF176	
7515		071250			DF202=DF176	
7516		071250			DF203=DF176	
7517		071250			DF204=DF176	
7518		071250			DF205=DF176	
7519		071250			DF206=DF176	
7520	071305	004	000	005	DF210: .BYTE	4,0,5,0,5,0,5,0
7521	071315	004	000	005	DF207: .BYTE	4,0,5,0,5,5,5,3,5,5,5,3,5,5,5,3,5,5,5,3
7522		071315			DF211=DF207	
7523	071341	004	000	005	DF212: .BYTE	4,0,5,0,5,0,5,0,5,5,5,3,5,5,5,3,5,5,5,3,5,5,5,3
7524		071341			DF213=DF212	
7525		071341			DF214=DF212	
7526		071341			DF215=DF212	
7527		071315			DF216=DF207	
7528	071371	004	000	005	DF217: .BYTE	4,0,5,0,5,0,0
7529		071315			DF220=DF207	
7530		071315			DF221=DF207	
7531		071315			DF222=DF207	
7532		071315			DF223=DF207	
7533		071315			DF224=DF207	
7534		071315			DF225=DF207	
7535		071341			DF226=DF212	
7536		071341			DF227=DF212	
7537		071341			DF230=DF212	
7538		071341			DF231=DF212	
7539		071341			DF232=DF212	
7540	071400	004	000	005	DF233: .BYTE	4,0,5,0,5,5,3,5,5,3,5,5,3
7541	071415	004	000	005	DF234: .BYTE	4,0,5,0,5,0,0
7542						
7543	071424	004	000	005	DF235: .BYTE	4,0,5,0
7544		071415			DF236=DF234	
7545		071415			DF237=DF234	
7546		071415			DF240=DF234	
7547		071400			DF241=DF233	
7548		071415			DF242=DF234	
7549		071400			DF243=DF233	
7550		071400			DF244=DF233	
7551		071400			DF245=DF233	
7552	071430	004	000	005	DF247: .BYTE	4,0,5,0,5,0
7553		071430			DF250=DF247	
7554		071430			DF251=DF247	

7555							
7556	071436	004	000	005	DF252: .BYTE	4,0,5,0	
7557	071442	004	000	005	DF253: .BYTE	4,0,5,0,5,0,5,0,5,5,0,5,5,0,5,5,0,5,5,0,5,5,0,5,5,3,5,5,3	
7558		071442			DF254=DF253		
7559		071442			DF255=DF253		
7560		071442			DF256=DF253		
7561		071442			DF257=DF253		
7562		071442			DF260=DF253		
7563		071442			DF261=DF253		
7564		071442			DF262=DF253		
7565		071442			DF263=DF253		
7566		071442			DF264=DF253		
7567		071442			DF265=DF253		
7568		071436			DF266=DF252		
7569		071250			DF267=DF176		
7570					.EVEN		





7628	071734				DT64=DT53	
7629	071734				DT65=DT53	
7630	071734				DT66=DT53	
7631	071734				DT67=DT53	
7632	071734				DT70=DT53	
7633	071734				DT71=DT53	
7634	071734				DT72=DT53	
7635	071734				DT73=DT53	
7636	071734				DT74=DT53	
7637	071734				DT75=DT53	
7638	071734				DT76=DT53	
7639	071734				DT77=DT53	
7640	071734				DT100=DT53	
7641	071734				DT101=DT53	
7642	071734				DT102=DT53	
7643	071734				DT103=DT53	
7644	071734				DT104=DT53	
7645	071734				DT105=DT53	
7646	071734				DT106=DT53	
7647	071734				DT107=DT53	
7648	071734				DT110=DT53	
7649	071662				DT111=DT23	
7650	071662				DT112=DT23	
7651	071662				DT113=DT23	
7652	071662				DT114=DT23	
7653	072022	001234	040005		DT115: .WORD	STMP0,STMP1,STAB,STMP2,STAB,STMP11,STMP7,STAB,STMP10
7654	072044	001313	040012	001240	.WORD	\$CRLF,MS1,STMP3,\$CRLF,MS2,STMP4
7655	072060	001313	040125	001244	.WORD	\$CRLF,MS5,STMP5,\$CRLF,MS6,STMP6,0
7656	072022				DT116=DT115	
7657	072022				DT117=DT115	
7658	072022				DT120=DT115	
7659	072022				DT121=DT115	
7660	072022				DT122=DT115	
7661	071662				DT123=DT23	
7662	071662				DT124=DT23	
7663	071662				DT125=DT23	
7664	071662				DT126=DT23	
7665	072022				DT127=DT115	
7666	072022				DT130=DT115	
7667	072022				DT131=DT115	
7668	071662				DT132=DT23	
7669	072022				DT133=DT115	
7670	072022				DT134=DT115	
7671	071662				DT135=DT23	
7672	072022				DT136=DT115	
7673	072022				DT137=DT115	
7674	072022				DT140=DT115	
7675	072022				DT141=DT115	
7676	072022				DT142=DT115	
7677	072022				DT143=DT115	
7678	072022				DT144=DT115	
7679	071662				DT145=DT23	
7680	071662				DT146=DT23	
7681	071662				DT147=DT23	
7682	071662				DT150=DT23	
7683	072022				DT151=DT115	
7684	072022				DT152=DT115	

7685		072022			DT153=DT115	
7686		072022			DT154=DT115	
7687		072022			DT155=DT115	
7688		072022			DT156=DT115	
7689		071662			DT157=DT23	
7690		072022			DT160=DT115	
7691		071662			DT161=DT23	
7692		071662			DT162=DT23	
7693		072022			DT163=DT115	
7694		071662			DT164=DT23	
7695	072076	001232	001234	040005	DT165: .WORD	\$TMP0,\$TMP1,\$TAB,\$TMP2,\$TAB,\$TMP13,\$TMP11,\$TMP12
7696	072116	001313	040012	001240	.WORD	\$CRLF,\$MS1,\$TMP3,\$CRLF,\$MS2,\$TMP4
7697	072132	001313	040233	001244	.WORD	\$CRLF,\$MS11,\$TMP5,\$CRLF,\$MS12,\$TMP6
7698	072146	001313	040165	001250	.WORD	\$CRLF,\$MS7,\$TMP7,\$CRLF,\$MS10,\$TMP10,0
7699		072076			DT166=DT165	
7700		072076			DT167=DT165	
7701		072076			DT170=DT165	
7702		072076			DT171=DT165	
7703		072076			DT172=DT165	
7704		072076			DT173=DT165	
7705		072076			DT174=DT165	
7706		072076			DT175=DT165	
7707		072076			DT176=DT165	
7708	072164	001232	001234	034422	DT177: .WORD	\$TMP0,\$TMP1,\$CPSAVE,0
7709		072076			DT200=DT165	
7710		072076			DT201=DT165	
7711		072076			DT202=DT165	
7712		072076			DT203=DT165	
7713		072076			DT204=DT165	
7714		072076			DT205=DT165	
7715		072076			DT206=DT165	
7716	072174	001232	001234	040005	DT207: .WORD	\$TMP0,\$TMP1,\$TAB,\$TMP2,\$CRLF,\$MS41,\$CRLF,\$TMP3
7717	072214	001313	040400	001313	.WORD	\$CRLF,\$MS42,\$CRLF,\$TMP4,\$CRLF,\$MS43,\$CRLF,\$TMP5
7718	072234	001313	040433	001313	.WORD	\$CRLF,\$MS44,\$CRLF,\$TMP6,0
7719	072246	001232	001234	040005	DT210: .WORD	\$TMP0,\$TMP1,\$TAB,\$TMP2,\$TAB,\$TMP3
7720	072262	040005	001242	000000	.WORD	\$TAB,\$TMP4,0
7721		072174			DT211=DT207	
7722	072270	001232	001234	040005	DT212: .WORD	\$TMP0,\$TMP1,\$TAB,\$TMP2,\$TAB,\$TMP10,\$TAB,\$TMP11
7723	072310	001313	040354	001313	.WORD	\$CRLF,\$MS41,\$CRLF,\$TMP3,\$CRLF,\$MS42,\$CRLF,\$TMP4
7724	072330	001313	040416	001313	.WORD	\$CRLF,\$MS43,\$CRLF,\$TMP5,\$CRLF,\$MS44,\$CRLF,\$TMP6,0
7725		072270			DT213=DT212	
7726		072270			DT214=DT212	
7727		072270			DT215=DT212	
7728		072174			DT216=DT207	
7729	072352	001232	001234	040005	DT217: .WORD	\$TMP0,\$TMP1,\$TAB,\$TMP2,\$TAB,\$TMP3,\$TMP4,0
7730		072174			DT220=DT207	
7731		072174			DT221=DT207	
7732		072174			DT222=DT207	
7733		072174			DT223=DT207	
7734		072174			DT224=DT207	
7735		072174			DT225=DT207	
7736		072270			DT226=DT212	
7737		072270			DT227=DT212	
7738		072270			DT230=DT212	
7739		072246			DT231=DT210	
7740		072174			DT232=DT207	
7741	072372	001232	001234	040005	DT233: .WORD	\$TMP0,\$TMP1,\$TAB,\$TMP2,\$CRLF



```

7742 072404 040302 001244 001313 .WORD MS37,$TMP5,$CRLF,MS40,$TMP6
7743 072416 001313 040354 001250 .WORD $CRLF,MS41,$TMP7,0
7744 072426 001232 001234 040005 DT234: .WORD $TMP0,$TMP1,$TAB,$TMP2,$TAB,$TMP3,$TMP4,0
7745 072446 001232 001234 040005 DT235: .WORD $TMP0,$TMP1,$TAB,$TMP2,0
7746 072426 DT236=DT234
7747 072426 DT237=DT234
7748 072426 DT240=DT234
7749 072372 DT241=DT233
7750 072426 DT242=DT234
7751 072372 DT243=DT233
7752 072372 DT244=DT233
7753 072372 DT245=DT233
7754 071662 DT246=DT23
7755 072460 001232 001234 040005 DT247: .WORD $TMP0,$TMP1,$TAB,$TMP2,$TAB,$TMP3,0
7756 072476 001232 001234 040005 DT250: .WORD $TMP0,$TMP1,$TAB,$TMP2,0
7757 072476 DT251=DT250
7758 072510 001232 001234 040005 DT252: .WORD $TMP0,$TMP1,$TAB,$TMP2,0
7759 072522 001232 001234 040005 DT253: .WORD $TMP0,$TMP1,$TAB,$TMP2,$TAB,$TMP7,$TAB,$TMP10
7760 072542 001313 037614 001254 .WORD $CRLF,MSA1,$TMP11,$CRLF,MSA2,$TMP12
7761 072556 001313 037655 001244 .WORD $CRLF,MSA3,$TMP5,$CRLF,MSA4,$TMP6
7762 072572 001313 037705 001240 .WORD $CRLF,MSA5,$TMP3,$CRLF,MSA6,$TMP4,0
7763 072522 DT254=DT253
7764 072522 DT255=DT253
7765 072522 DT256=DT253
7766 072522 DT257=DT253
7767 072522 DT260=DT253
7768 072522 DT261=DT253
7769 072522 DT262=DT253
7770 072522 DT263=DT253
7771 072522 DT264=DT253
7772 072522 DT265=DT253
7773 072510 DT266=DT252
7774 072076 DT267=DT165
7775 072610 007070 007070 .WORD 7070,7070,7070,7070 :KLUDGE FOR APT LOAD PROBLEM
7776 000001 .END
    
```

SYMBOL TABLE

AAADON 014602  
 AAA1 013600  
 AAA10 014216  
 AAA11 014254  
 AAA12 014312  
 AAA13 014350  
 AAA2 013636  
 AAA3 013674  
 AAA4 013732  
 AAA5 013770  
 AAA6 014026  
 AAA7 014064  
 AAA8 014122  
 AAA9 014160  
 ABASE = 000000  
 ACDW1 = 000000  
 ACDW2 = 000000  
 ACPUOP= 000000  
 AC0 =%000000  
 AC1 =%000001  
 AC2 =%000002  
 AC3 =%000003  
 AC4 =%000004  
 AC5 =%000005  
 AC6 =%000006  
 AC7 =%000007  
 ADDW0 = 000000  
 ADDW1 = 000000  
 ADDW10= 000000  
 ADDW11= 000000  
 ADDW12= 000000  
 ADDW13= 000000  
 ADDW14= 000000  
 ADDW15= 000000  
 ADDW2 = 000000  
 ADDW3 = 000000  
 ADDW4 = 000000  
 ADDW5 = 000000  
 ADDW6 = 000000  
 ADDW7 = 000000  
 ADDW8 = 000000  
 ADDW9 = 000000  
 ADEVCT= 000000  
 ADEVN = 000000  
 AENV = 000000  
 AENVN = 000000  
 AFATAL= 000000  
 AMADR1= 000000  
 AMADR2= 000000  
 AMADR3= 000000  
 AMADR4= 000000  
 AMAMS1= 000000  
 AMAMS2= 000000  
 AMAMS3= 000000  
 AMAMS4= 000000  
 AMSGAD= 000000  
 AMSGLG= 000000

AMSGTY= 000000  
 AMTYP1= 000000  
 AMTYP2= 000000  
 AMTYP3= 000000  
 AMTYP4= 000000  
 APASS = 000000  
 APRIOR= 000000  
 APTCSU= 000040  
 APTENV= 000001  
 APTSIZ= 000200  
 APTSPO= 000100  
 ASWREG= 000000  
 ATESTN= 000000  
 AUNIT = 000000  
 AUSWR = 000000  
 AVECT1= 000000  
 AVECT2= 000000  
 BBBDON 015304  
 BBBER1 015060  
 BBBER2 015144  
 BBBER3 015172  
 BBBER4 015220  
 BBBP1 015264  
 BBBP2 015274  
 BBB0 014610  
 BBB1 014644  
 BBB2 014672  
 BBB3 014722  
 BBB4 014750  
 BBB5 015006  
 BBB6 015014  
 BIT0 = 000001  
 BIT00 = 000001  
 BIT01 = 000002  
 BIT02 = 000004  
 BIT03 = 000010  
 BIT04 = 000020  
 BIT05 = 000040  
 BIT06 = 000100  
 BIT07 = 000200  
 BIT08 = 000400  
 BIT09 = 001000  
 BIT1 = 000002  
 BIT10 = 002000  
 BIT11 = 004000  
 BIT12 = 010000  
 BIT13 = 020000  
 BIT14 = 040000  
 BIT15 = 100000  
 BIT2 = 000004  
 BIT3 = 000010  
 BIT4 = 000020  
 BIT5 = 000040  
 BIT6 = 000100  
 BIT7 = 000200  
 BIT8 = 000400  
 BIT9 = 001000

BPTVEC= 000014  
 CCCDON 016314  
 CCC1 015310  
 CCC10 015704  
 CCC11 015740  
 CCC12 015774  
 CCC13 016030  
 CCC2 015344  
 CCC3 015400  
 CCC4 015434  
 CCC5 015470  
 CCC6 015524  
 CCC7 015560  
 CCC8 015614  
 CCC9 015650  
 CKSWR = 104406  
 CMPSUB 014412  
 CMPTMP 014572  
 CNT = 000270  
 CORDON 033336  
 CORFLG 033320  
 CORINT 033332  
 CORMES 040455  
 CORSUB 033036  
 CORTMP 033322  
 CORTRP 033334  
 CORTV 033134  
 CORTV0 033276  
 CORTV1 033302  
 COR1 032204  
 COR10 032626  
 COR11 032672  
 COR12 032732  
 COR13 032772  
 COR2 032230  
 COR3 032244  
 COR33 032256  
 COR4 032316  
 COR5 032356  
 COR6 032416  
 COR7 032462  
 COR8 032522  
 COR9 032566  
 CPSAVE 034422  
 CPSPUR 037502  
 CPTWO 037520  
 CR = 000015  
 CRLF = 000200  
 DDDDON 017250  
 DDD1 016320  
 DDD2 016374  
 DDD3 016450  
 DDD4 016524  
 DDD5 016600  
 DDD6 016654  
 DDD7 016730  
 DDISP = 177570

DF1 = 070743  
 DF10 = 070743  
 DF100 = 071164  
 DF101 = 071164  
 DF102 = 071164  
 DF103 = 071164  
 DF104 = 071164  
 DF105 = 071164  
 DF106 = 071164  
 DF107 = 071164  
 DF11 = 070743  
 DF110 = 071164  
 DF111 = 071012  
 DF112 = 071012  
 DF113 = 071012  
 DF114 = 071012  
 DF115 = 071062  
 DF116 = 071062  
 DF117 = 071062  
 DF12 = 070743  
 DF120 = 071062  
 DF121 = 071062  
 DF122 = 071062  
 DF123 = 071036  
 DF124 = 071036  
 DF125 = 071036  
 DF126 = 071036  
 DF127 = 071106  
 DF13 = 070743  
 DF130 = 071106  
 DF131 = 071106  
 DF132 = 071036  
 DF133 = 071106  
 DF134 = 071106  
 DF135 = 071036  
 DF136 = 071106  
 DF137 = 071062  
 DF14 = 070743  
 DF140 = 071062  
 DF141 = 071062  
 DF142 = 071062  
 DF143 = 071062  
 DF144 = 071062  
 DF145 = 071012  
 DF146 = 071012  
 DF147 = 071012  
 DF15 = 070743  
 DF150 = 071012  
 DF151 = 071106  
 DF152 = 071106  
 DF153 = 071106  
 DF154 = 071106  
 DF155 = 071106  
 DF156 = 071106  
 DF157 = 071036  
 DF16 = 070743  
 DF160 = 071106

DF161 = 071036  
 DF162 = 071036  
 DF163 = 071106  
 DF164 = 071036  
 DF165 = 071216  
 DF166 = 071216  
 DF167 = 071216  
 DF17 = 070773  
 DF170 = 071216  
 DF171 = 071216  
 DF172 = 071216  
 DF173 = 071216  
 DF174 = 071216  
 DF175 = 071216  
 DF176 = 071250  
 DF177 = 071302  
 DF2 = 070761  
 DF20 = 071002  
 DF200 = 071250  
 DF201 = 071250  
 DF202 = 071250  
 DF203 = 071250  
 DF204 = 071250  
 DF205 = 071250  
 DF206 = 071250  
 DF207 = 071315  
 DF21 = 071002  
 DF210 = 071305  
 DF211 = 071315  
 DF212 = 071341  
 DF213 = 071341  
 DF214 = 071341  
 DF215 = 071341  
 DF216 = 071315  
 DF217 = 071371  
 DF22 = 071002  
 DF220 = 071315  
 DF221 = 071315  
 DF222 = 071315  
 DF223 = 071315  
 DF224 = 071315  
 DF225 = 071315  
 DF226 = 071341  
 DF227 = 071341  
 DF23 = 071012  
 DF230 = 071341  
 DF231 = 071341  
 DF232 = 071341  
 DF233 = 071400  
 DF234 = 071415  
 DF235 = 071424  
 DF236 = 071415  
 DF237 = 071415  
 DF24 = 071012  
 DF240 = 071415  
 DF241 = 071400  
 DF242 = 071415



DF243 = 071400  
DF244 = 071400  
DF245 = 071400  
DF246 = 071036  
DF247 = 071430  
DF248 = 071012  
DF250 = 071430  
DF251 = 071430  
DF252 = 071436  
DF253 = 071442  
DF254 = 071442  
DF255 = 071442  
DF256 = 071442  
DF257 = 071442  
DF26 = 071012  
DF260 = 071442  
DF261 = 071442  
DF262 = 071442  
DF263 = 071442  
DF264 = 071442  
DF265 = 071442  
DF266 = 071436  
DF267 = 071250  
DF27 = 071012  
DF3 = 070743  
DF30 = 071012  
DF31 = 071012  
DF32 = 071012  
DF33 = 071036  
DF34 = 071036  
DF35 = 071036  
DF36 = 071036  
DF37 = 071012  
DF4 = 070743  
DF40 = 071012  
DF41 = 071012  
DF42 = 071012  
DF43 = 071012  
DF44 = 071012  
DF45 = 071012  
DF46 = 071036  
DF47 = 071036  
DF5 = 070743  
DF50 = 071036  
DF51 = 071036  
DF52 = 071036  
DF53 = 071132  
DF54 = 071132  
DF55 = 071132  
DF56 = 071132  
DF57 = 071132  
DF6 = 070743  
DF60 = 071132  
DF61 = 071132  
DF62 = 071132  
DF63 = 071132  
DF64 = 071132

DF65 = 071132  
DF66 = 071132  
DF67 = 071132  
DF7 = 070743  
DF70 = 071164  
DF71 = 071164  
DF72 = 071164  
DF73 = 071164  
DF74 = 071164  
DF75 = 071164  
DF76 = 071164  
DF77 = 071164  
DH1 = 067106  
DH10 = 067106  
DH100 = 067106  
DH101 = 067106  
DH102 = 067106  
DH103 = 067106  
DH104 = 067106  
DH105 = 067106  
DH106 = 067106  
DH107 = 067106  
DH11 = 067106  
DH110 = 067106  
DH111 = 067106  
DH112 = 067106  
DH113 = 067106  
DH114 = 067106  
DH115 = 067447  
DH116 = 067447  
DH117 = 067447  
DH12 = 067106  
DH120 = 067447  
DH121 = 067447  
DH122 = 067447  
DH123 = 067106  
DH124 = 067106  
DH125 = 067106  
DH126 = 067106  
DH127 = 067447  
DH13 = 067106  
DH130 = 067447  
DH131 = 067447  
DH132 = 067106  
DH133 = 067447  
DH134 = 067447  
DH135 = 067106  
DH136 = 067447  
DH137 = 067447  
DH14 = 067106  
DH140 = 067447  
DH141 = 067447  
DH142 = 067447  
DH143 = 067447  
DH144 = 067447  
DH145 = 067106  
DH146 = 067106

DH147 = 067106  
DH15 = 067106  
DH150 = 067106  
DH151 = 067447  
DH152 = 067447  
DH153 = 067447  
DH154 = 067447  
DH155 = 067447  
DH156 = 067447  
DH157 = 067106  
DH16 = 067106  
DH160 = 067447  
DH161 = 067106  
DH162 = 067106  
DH163 = 067447  
DH164 = 067106  
DH165 = 067447  
DH166 = 067447  
DH167 = 067447  
DH17 = 067237  
DH170 = 067447  
DH171 = 067447  
DH172 = 067447  
DH173 = 067447  
DH174 = 067447  
DH175 = 067447  
DH176 = 067447  
DH177 = 067546  
DH2 = 067176  
DH20 = 067106  
DH200 = 067447  
DH201 = 067447  
DH202 = 067447  
DH203 = 067447  
DH204 = 067447  
DH205 = 067447  
DH206 = 067447  
DH207 = 067647  
DH21 = 067312  
DH210 = 067737  
DH211 = 067737  
DH212 = 067647  
DH213 = 067647  
DH214 = 067647  
DH215 = 067647  
DH216 = 067737  
DH217 = 070000  
DH22 = 067401  
DH220 = 067607  
DH221 = 067737  
DH222 = 067607  
DH223 = 067737  
DH224 = 067607  
DH225 = 067737  
DH226 = 070053  
DH227 = 070142  
DH23 = 067106

DH230 = 070053  
DH231 = 070142  
DH232 = 070142  
DH233 = 070231  
DH234 = 070272  
DH235 = 070360  
DH236 = 070420  
DH237 = 070272  
DH24 = 067106  
DH240 = 070506  
DH241 = 070231  
DH242 = 070506  
DH243 = 070231  
DH244 = 070231  
DH245 = 070231  
DH246 = 067106  
DH247 = 070576  
DH25 = 067106  
DH250 = 070643  
DH251 = 070643  
DH252 = 067176  
DH253 = 067106  
DH254 = 067106  
DH255 = 067106  
DH256 = 067106  
DH257 = 067106  
DH26 = 067106  
DH260 = 067106  
DH261 = 067106  
DH262 = 067106  
DH263 = 067106  
DH264 = 067106  
DH265 = 067106  
DH266 = 070703  
DH267 = 067447  
DH27 = 067106  
DH3 = 067106  
DH30 = 067106  
DH31 = 067106  
DH32 = 067106  
DH33 = 067106  
DH34 = 067106  
DH35 = 067106  
DH36 = 067106  
DH37 = 067106  
DH4 = 067106  
DH40 = 067106  
DH41 = 067106  
DH42 = 067106  
DH43 = 067106  
DH44 = 067106  
DH45 = 067106  
DH46 = 067106  
DH47 = 067106  
DH5 = 067106  
DH50 = 067106  
DH51 = 067106

DH52 = 067106  
DH53 = 067106  
DH54 = 067106  
DH55 = 067106  
DH56 = 067106  
DH57 = 067106  
DH6 = 067106  
DH60 = 067106  
DH61 = 067106  
DH62 = 067106  
DH63 = 067106  
DH64 = 067106  
DH65 = 067106  
DH66 = 067106  
DH67 = 067106  
DH7 = 067106  
DH70 = 067106  
DH71 = 067106  
DH72 = 067106  
DH73 = 067106  
DH74 = 067106  
DH75 = 067106  
DH76 = 067106  
DH77 = 067106  
DISPLA = 001142  
DISPRE = 000174  
DIVDSU = 017010  
DIVDT = 017240  
DIVFSU = 016070  
DIVFT = 016304  
DSWR = 177570  
DT1 = 071474  
DT10 = 071474  
DT100 = 071734  
DT101 = 071734  
DT102 = 071734  
DT103 = 071734  
DT104 = 071734  
DT105 = 071734  
DT106 = 071734  
DT107 = 071734  
DT11 = 071474  
DT110 = 071734  
DT111 = 071662  
DT112 = 071662  
DT113 = 071662  
DT114 = 071662  
DT115 = 072022  
DT116 = 072022  
DT117 = 072022  
DT12 = 071474  
DT120 = 072022  
DT121 = 072022  
DT122 = 072022  
DT123 = 071662  
DT124 = 071662  
DT125 = 071662



DT126 = 071662  
DT127 = 072022  
DT13 = 071474  
DT130 = 072022  
DT131 = 072022  
DT132 = 071662  
DT133 = 072022  
DT134 = 072022  
DT135 = 071662  
DT136 = 072022  
DT137 = 072022  
DT14 = 071474  
DT140 = 072022  
DT141 = 072022  
DT142 = 072022  
DT143 = 072022  
DT144 = 072022  
DT145 = 071662  
DT146 = 071662  
DT147 = 071662  
DT15 = 071474  
DT150 = 071662  
DT151 = 072022  
DT152 = 072022  
DT153 = 072022  
DT154 = 072022  
DT155 = 072022  
DT156 = 072022  
DT157 = 071662  
DT16 = 071474  
DT160 = 072022  
DT161 = 071662  
DT162 = 071662  
DT163 = 072022  
DT164 = 071662  
DT165 = 072076  
DT166 = 072076  
DT167 = 072076  
DT17 = 071560  
DT170 = 072076  
DT171 = 072076  
DT172 = 072076  
DT173 = 072076  
DT174 = 072076  
DT175 = 072076  
DT176 = 072076  
DT177 = 072164  
DT2 = 071532  
DT20 = 071600  
DT200 = 072076  
DT201 = 072076  
DT202 = 072076  
DT203 = 072076  
DT204 = 072076  
DT205 = 072076  
DT206 = 072076  
DT207 = 072174

DT21 = 071622  
DT210 = 072246  
DT211 = 072174  
DT212 = 072270  
DT213 = 072270  
DT214 = 072270  
DT215 = 072270  
DT216 = 072174  
DT217 = 072352  
DT22 = 071644  
DT220 = 072174  
DT221 = 072174  
DT222 = 072174  
DT223 = 072174  
DT224 = 072174  
DT225 = 072174  
DT226 = 072270  
DT227 = 072270  
DT23 = 071662  
DT230 = 072270  
DT231 = 072246  
DT232 = 072174  
DT233 = 072372  
DT234 = 072426  
DT235 = 072446  
DT236 = 072426  
DT237 = 072426  
DT24 = 071662  
DT240 = 072426  
DT241 = 072372  
DT242 = 072426  
DT243 = 072372  
DT244 = 072372  
DT245 = 072372  
DT246 = 071662  
DT247 = 072460  
DT25 = 071662  
DT250 = 072476  
DT251 = 072476  
DT252 = 072510  
DT253 = 072522  
DT254 = 072522  
DT255 = 072522  
DT256 = 072522  
DT257 = 072522  
DT26 = 071662  
DT260 = 072522  
DT261 = 072522  
DT262 = 072522  
DT263 = 072522  
DT264 = 072522  
DT265 = 072522  
DT266 = 072510  
DT267 = 072076  
DT27 = 071662  
DT3 = 071474  
DT30 = 071662

DT31 = 071662  
DT32 = 071662  
DT33 = 071662  
DT34 = 071662  
DT35 = 071662  
DT36 = 071662  
DT37 = 071662  
DT4 = 071474  
DT40 = 071662  
DT41 = 071662  
DT42 = 071662  
DT43 = 071662  
DT44 = 071662  
DT45 = 071662  
DT46 = 071662  
DT47 = 071662  
DT5 = 071474  
DT50 = 071662  
DT51 = 071662  
DT52 = 071662  
DT53 = 071734  
DT54 = 071734  
DT55 = 071734  
DT56 = 071734  
DT57 = 071734  
DT6 = 071474  
DT60 = 071734  
DT61 = 071734  
DT62 = 071734  
DT63 = 071734  
DT64 = 071734  
DT65 = 071734  
DT66 = 071734  
DT67 = 071734  
DT7 = 071474  
DT70 = 071734  
DT71 = 071734  
DT72 = 071734  
DT73 = 071734  
DT74 = 071734  
DT75 = 071734  
DT76 = 071734  
DT77 = 071734  
EEEDON 020260  
EEE1 017254  
EEE10 017650  
EEE11 017704  
EEE12 017740  
EEE13 017774  
EEE2 017310  
EEE3 017344  
EEE4 017400  
EEE5 017434  
EEE6 017470  
EEE7 017524  
EEE8 017560  
EEE9 017614

EMTVEC= 000030  
EM1 040514  
EM10 041333  
EM100 054471  
EM101 054554  
EM102 054750  
EM103 055144  
EM104 055256  
EM105 055423  
EM106 055532  
EM107 055641  
EM11 041440  
EM110 055750  
EM111 044565  
EM112 044642  
EM113 044720  
EM114 044776  
EM115 045055  
EM116 045133  
EM117 045212  
EM12 041547  
EM120 045350  
EM121 045506  
EM122 045643  
EM123 046000  
EM124 046056  
EM125 046135  
EM126 046213  
EM127 046272  
EM13 041656  
EM130 046350  
EM131 046427  
EM132 046565  
EM133 046723  
EM134 047061  
EM135 047216  
EM136 047353  
EM137 047510  
EM14 041772  
EM140 047576  
EM141 = 044565  
EM142 = 044642  
EM143 047665  
EM144 047743  
EM145 050022  
EM146 050167  
EM147 050334  
EM15 042104  
EM150 050500  
EM151 050644  
EM152 050732  
EM153 = 046000  
EM154 = 046056  
EM155 051021  
EM156 051077  
EM157 051156  
EM16 042216

EM160 051323  
EM161 051461  
EM162 051626  
EM163 051772  
EM164 052127  
EM165 056060  
EM166 056144  
EM167 056227  
EM17 042313  
EM170 056261  
EM171 056347  
EM172 056472  
EM173 056557  
EM174 056725  
EM175 057073  
EM176 057205  
EM177 057271  
EM2 040546  
EM20 042370  
EM200 057325  
EM201 057357  
EM202 057446  
EM203 057610  
EM204 057752  
EM205 060040  
EM206 060206  
EM207 060354  
EM21 042422  
EM210 060416  
EM211 060460  
EM212 060624  
EM213 061010  
EM214 061174  
EM215 061360  
EM216 061544  
EM217 061726  
EM22 042454  
EM220 061770  
EM221 062220  
EM222 062464  
EM223 062715  
EM224 063162  
EM225 063413  
EM226 063660  
EM227 064013  
EM23 042535  
EM230 064146  
EM231 064302  
EM232 = 060416  
EM233 064436  
EM234 064475  
EM235 064561  
EM236 064627  
EM237 064662  
EM24 042612  
EM240 064746  
EM241 065002

EM242	065105	EM64	053372	GGG4	024610	HHER9	006602	HXER10	013342
EM243	065141	EM65	053511	GGG5	024660	HHDON	030200	HXER11	013374
EM244	065244	EM66	053624	GGG6	024730	HH1	026076	HXER12	013414
EM245	065303	EM67	053673	GGG7	025000	HH10	027176	HXER13	013444
EM246	044175	EM7	041226	GGG8	025050	HH11	027276	HXER2	013112
EM247	065365	EM70	053756	GGG9	025120	HH12	027376	HXER22	013126
EM25	042710	EM71	054007	GGP1	011744	HH13	027476	HXER3	013142
EM250	065421	EM72	054037	GGP2	011754	HH2	026176	HXER33	013156
EM251	065453	EM73 =	054037	GGP3	011764	HH3	026276	HXER4	013172
EM252	065506	EM74	054071	GGP4	011774	HH4	026376	HXER5	013072
EM253	065577	EM75	054200	GGP5	012004	HH5	026476	HXER6	013226
EM254	065663	EM76	054271	GGP6	012014	HH6	026576	HXER66	013242
EM255	065750	EM77	054361	GGP7	012024	HH7	026676	HXER7	013256
EM256	066036	EPENDS	034104	GGP8	012034	HH8	026776	HXER8	013210
EM257	066125	ERM10	034756	GGP9	012044	HH9	027076	HXER9	013306
EM26	042775	ERROR =	104000	GG1	007142	HHP0	006776	HXP1	013474
EM260	066213	ERRVEC =	000004	GG10	007436	HHP1	007006	HXP2	013504
EM261	066302	ERTYPE	037040	GG11	007474	HHP10	007116	HXP3	013514
EM262	066372	FFFDON	021010	GG12	007516	HHP11	007126	HXP4	013524
EM263	066463	FFF1	020264	GG13	007526	HHP2	007016	HXP5	013534
EM264	066550	FFF2	020340	GG14	007544	HHP3	007026	HXP6	013544
EM265	066635	FFF3	020414	GG15	007602	HHP4	007036	HXP7	013554
EM266	066722	FFF4	020470	GG16	007620	HHP5	007046	HXP8	013564
EM267	067023	FPSPUR	037450	GG17	007666	HHP6	007056	HX1	012060
EM27 =	042710	FPVECT =	000244	GG18	007676	HHP7	007066	HX10	012266
EM3	040602	GGDATO	011734	GG19	007732	HHP8	007076	HX11	012310
EM30	043073	GGDONE	012054	GG2	007200	HHP9	007106	HX12	012330
EM31	043145	GGERO	010232	GG20	007770	HHTRAP	006716	HX13	012340
EM32	042560	GGER1	010330	GG21	010012	HH1	005216	HX14	012346
EM33	043212	GGER10	011042	GG22	010022	HH10	005430	HX15	012364
EM34	043235	GGER11	011110	GG23	010040	HH11	005446	HX16	012412
EM35	043267	GGER12	011156	GG24	010076	HH12	005476	HX165	012416
EM36	043342	GGER13	011224	GG25	010114	HH13	005520	HX17	012444
EM37	043410	GGER14	011272	GG26	010162	HH14	005540	HX18	012502
EM4	040707	GGER15	011370	GG27	010172	HH15	005550	HX19	012522
EM40	043433	GGER16	011436	GG28	010226	HH16	005556	HX2	012110
EM41	043465	GGER17	011504	GG3	007222	HH17	005574	HX20	012532
EM42	043540	GGER18	011552	GG4	007232	HH18	005632	HX21	012534
EM43	043671	GGER19	011620	GG5	007250	HH19	005654	HX22	012564
EM44	044022	GGER2	010376	GG6	007306	HH2	005246	HX23	012604
EM45	044070	GGER20	011666	GG7	007324	HH20	005664	HX24	012624
EM46	044143	GGER3	010444	GG8	007372	HH21	005702	HX25	012634
EM47	044220	GGER4	010512	GG9	007402	HH22	005732	HX26	012642
EM5	041014	GGER5	010514	GTSWR =	104405	HH23	005754	HX27	012644
EM50	044354	GGER6	010562	HHDATO	006766	HH24	005764	HX28	012676
EM51	044427	GGER7	010630	HHDONE	007136	HH25	006002	HX29	012720
EM52	044475	GGER8	010726	HHER0	006006	HH3	005270	HX3	012124
EM53	052325	GGER9	010774	HHER00	006122	HH4	005310	HX30	012730
EM54	052356	GGGDON	026072	HHER1	006054	HH5	005320	HX31	012746
EM55	052273	GGG1	024420	HHER10	006650	HH6	005326	HX32	012776
EM56	052406	GGG10	025170	HHER2	006170	HH7	005340	HX33	013020
EM57	052477	GGG11	025240	HHER3	006236	HH8	005376	HX34	013030
EM6	041121	GGG12	025310	HHER4	006304	HH9	005420	HX35	013046
EM60	052567	GGG13	025360	HHER5	006352	HT =	000011	HX4	012144
EM61	052671	GGG14	025430	HHER6	006420	HXDATO	013464	HX5	012164
EM62	053065	GGG2	024470	HHER7	006466	HXDONE	013574	HX6	012174
EM63	053261	GGG3	024540	HHER8	006534	HXER1	013052	HX7	012202



HX8 012220  
HX9 012252  
IBSAVE 034424  
IIIDON 021650  
III1 021014  
III2 021060  
III3 021124  
III4 021170  
IOTVEC= 000020  
JJJDON 022610  
JJJ1 021654  
JJJ2 021740  
JJJ3 022024  
JJJ4 022110  
KKKDON 023452  
KKK1 022614  
KKK3 022660  
KKK4 022724  
KKK5 022770  
LF = 000012  
LLLDON 024414  
LLL1 023456  
LLL2 023542  
LLL3 023626  
LLL4 023712  
LOOP 005214  
LPERR = 104413  
MMMDON 031126  
MMM1 030204  
MMM2 030256  
MMM3 030330  
MMM4 030402  
MMM5 030454  
MNUMBE= 000267  
MODDD0 032134  
MODDD1 032144  
MODDOV 031546  
MODDSU 027602  
MODDT0 030160  
MODDT1 030170  
MODFDO 031106  
MODFD1 031116  
MODFOV 030532  
MODFSU 025504  
MODFT0 026042  
MODFT1 026052  
MODP1 026062  
MSA1 037614  
MSA2 037632  
MSA3 037655  
MSA4 037667  
MSA5 037705  
MSA6 037720  
MS1 040012  
MS10 040211  
MS11 040233  
MS12 040257

MS2 040030  
MS3 040050  
MS37 040302  
MS4 040077  
MS40 040320  
MS41 040354  
MS415 040334  
MS42 040400  
MS43 040416  
MS44 040433  
MS5 040125  
MS6 040147  
MS7 040165  
MULDSU 020550  
MULDT 021000  
MULFSU 020034  
MULFT 020250  
NNNDON 032154  
NNN1 031132  
NNN2 031234  
NNN3 031336  
NNN4 031440  
OVDNTT 022600  
OVDTT 024404  
OVFNTT 021640  
OVFTT 023442  
OVUNDN 022200  
OVUNDT 024002  
OVUNFN 021240  
OVUNFT 023040  
PIRQ = 177772  
PIRQVE= 000240  
POWERM 037740  
PROGNUM= 000002  
PRO = 000000  
PR1 = 000040  
PR2 = 000100  
PR3 = 000140  
PR4 = 000200  
PR5 = 000240  
PR6 = 000300  
PR7 = 000340  
PS = 177776  
PSW = 177776  
PWRVEC= 000024  
RDCHR = 104407  
RESREG= 104411  
RESVEC= 000010  
RSETUP= 104412  
R6 = X000006  
R7 = X000007  
SAVREG= 104410  
SCOPE = 000004  
SPACE 040007  
STACK = 001100  
START 004346  
STKLMT= 177774

SWR 001140  
SWREG 000176  
SW0 = 000001  
SW00 = 000001  
SW01 = 000002  
SW02 = 000004  
SW03 = 000010  
SW04 = 000020  
SW05 = 000040  
SW06 = 000100  
SW07 = 000200  
SW08 = 000400  
SW09 = 001000  
SW1 = 000002  
SW10 = 002000  
SW11 = 004000  
SW12 = 010000  
SW13 = 020000  
SW14 = 040000  
SW15 = 100000  
SW2 = 000004  
SW3 = 000010  
SW4 = 000020  
SW5 = 000040  
SW6 = 000100  
SW7 = 000200  
SW8 = 000400  
SW9 = 001000  
TAB = 000011  
TBITVE= 000014  
TKVEC = 000060  
TOCTNM 037434  
TPVEC = 000064  
TRAPVE= 000034  
TRTVEC= 000014  
TST1 005214  
TST10 017252  
TST11 020262  
TST12 021012  
TST13 021652  
TST14 022612  
TST15 023454  
TST16 024416  
TST17 026074  
TST2 007140  
TST20 030202  
TST21 031130  
TST22 032156  
TST23 033340  
TST3 012056  
TST4 013576  
TST5 014604  
TST6 015306  
TST7 016316  
TYPE = 104401  
TYPOC = 104402  
TYPON = 104404

TYPOS = 104403  
\$APTHD 004332  
\$ATYC 035700  
\$ATY1 035654  
\$ATY3 035662  
\$ATY4 035672  
\$AUTOB 001134  
\$BASE 001372  
\$BDADR 001122  
\$BDDAT 001126  
\$BELL 001306  
\$CDW1 001376  
\$CDW2 001400  
\$CHARC 035414  
\$CKSWR 036122  
\$CLR.T 034014  
\$CMTAG 001100  
\$CM1 = 000024  
\$CM2 = 000050  
\$CM3 = 000024  
\$CM4 = 000024  
\$CNTLG 036541  
\$CNTLU 036534  
\$CPUOP 001344  
\$CRLF 001313  
\$DDW0 001402  
\$DDW1 001404  
\$DDW10 001426  
\$DDW11 001430  
\$DDW12 001432  
\$DDW13 001434  
\$DDW14 001436  
\$DDW15 001440  
\$DDW2 001406  
\$DDW3 001410  
\$DDW4 001412  
\$DDW5 001414  
\$DDW6 001416  
\$DDW7 001420  
\$DDW8 001422  
\$DDW9 001424  
\$DEVCT 001326  
\$DEVM 001374  
\$DISAB 033744  
\$DOAGN 034034  
\$ENDAD 034024  
\$ENDCT 033404  
\$ENULL 034100  
\$ENV 001336  
\$ENVM 001337  
\$EOP 033340  
\$EOPCT 033372  
\$ERFLG 001103  
\$ERMAX 001115  
\$ERROR 034426  
\$ERRPC 001116  
\$ERRTB 001442

\$ERTTL 001112  
\$ESCAP 001304  
\$ETABL 001336  
\$ETEND 001442  
\$FATAL 001320  
\$FFLG 036120  
\$FILLC 001156  
\$FILLS 001155  
\$GDADR 001120  
\$GDDAT 001124  
\$GET42 033664  
\$GTSWR 036172  
\$GT42C 033776  
\$HD = 000003  
\$HIBTS 004332  
\$ICNT 001104  
\$ILLUP 037032  
\$INTAG 001135  
\$ITEMB 001114  
\$LF 001314  
\$LFLG 036117  
\$LOOP 034072  
\$LPADR 001106  
\$LPERR 001110  
\$MADR1 001350  
\$MADR2 001354  
\$MADR3 001360  
\$MADR4 001364  
\$MAIL 001316  
\$MAMS1 001345  
\$MAMS2 001352  
\$MAMS3 001356  
\$MAMS4 001362  
\$MBADR 004334  
\$MFLC 036116  
\$MNEW 036557  
\$MSGAD 001332  
\$MSGLG 001334  
\$MSGTY 001316  
\$MSWR 036546  
\$MTYP1 001347  
\$MTYP2 001353  
\$MTYP3 001357  
\$MTYP4 001363  
\$MXCNT 034420  
\$NULL 001154  
\$NWTST= 000001  
\$OCNT 035650  
\$OMODE 035652  
\$OVER 034404  
\$PASS 001324  
\$PASS2 034106  
\$PASTM 004340  
\$PWRAD 037014  
\$PWRDN 036654  
\$PWRMG 037010  
\$PWRUP 036726



SYMBOL TABLE

\$QUES	001312	\$REG3	001170	\$TAB	040005	\$TMP21	001274	\$TYPEC	035266
\$RDCHR	036404	\$REG4	001172	\$TBIT	034076	\$TMP22	001276	\$TYPEX	035416
\$RDSZ =	000001	\$REG5	001174	\$TERM =	000030	\$TMP23	001300	\$TYPOC	035444
\$REGAD	001160	\$REG6	001176	\$TESTN	001322	\$TMP3	001240	\$TYPON	035460
\$REGO	001162	\$REG7	001200	\$TIMES	001302	\$TMP4	001242	\$TYPOS	035420
\$REG1	001164	\$RESRE	035016	\$TKB	001146	\$TMP5	001244	\$UNIT	001330
\$REG10	001202	\$RTNAD	034074	\$TKS	001144	\$TMP6	001246	\$UNITM	004342
\$REG11	001204	\$RTRN	034070	\$TMP0	001232	\$TMP7	001250	\$USWR	001342
\$REG12	001206	\$\$AVRE	034760	\$TMP1	001234	\$TN =	000023	\$VECT1	001366
\$REG13	001210	\$\$AVR6	037036	\$TMP10	001252	\$TPB	001152	\$VECT2	001370
\$REG14	001212	\$\$SCOPE	034110	\$TMP11	001254	\$TPFLG	001157	\$XOFF =	000023
\$REG15	001214	\$\$SETUP=	000137	\$TMP12	001256	\$TPS	001150	\$XON =	000021
\$REG16	001216	\$\$STUP =	177777	\$TMP13	001260	\$TRAP	036570	\$XTSTR	034122
\$REG17	001220	\$\$VLAD	034350	\$TMP14	001262	\$TRAP2	036612	\$\$GET4=	000001
\$REG2	001166	\$\$VPC =	004332	\$TMP15	001264	\$TRP =	000014	\$OFILL	035651
\$REG20	001222	\$\$WR =	177400	\$TMP16	001266	\$TRPAD	036624	.LPER	037536
\$REG21	001224	\$\$WREG	001340	\$TMP17	001270	\$TSTM	004336	.RSET	037544
\$REG22	001226	\$\$WRMK=	000000	\$TMP2	001236	\$TSTNM	001102	.\$X =	004332
\$REG23	001230	\$\$WRMS=	000200	\$TMP20	001272	\$TYPE	035054		

. ABS. 072620 000  
000000 001

ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 60776 WORDS ( 238 PAGES)

DYNAMIC MEMORY: 20034 WORDS ( 77 PAGES)

ELAPSED TIME: 03:05:19

CKFPBC.BIN,CKFPBC/CR/-SP/NL:TOC=CKFPBC.MLB/ML,CKFPBC.P11