

KEF11-AA

KEF11-AA DIAG #1
CJKDCBO

AH-F242B-MC
FICHE 1 OF 2

MAY 1980
COPYRIGHT © 1979
MADE IN USA



A large grid of 10 columns and 20 rows of small, illegible text or data points, likely representing a technical specification or diagnostic data table. The text is too small to be read accurately.

KEF11-AA

KEF11-AA DIAG #1
CJKDCBO

AH-F242B-MC
FICHE 2 OF 2

MAY 1980
COPYRIGHT © 1979
MADE IN USA



Microfiche grid containing multiple frames of data, likely diagnostic information.



IDENTIFICATION

PRODUCT CODE: AC-F241B-MC
PRODUCT NAME: CJKDCBO KEF11-AA DIAG #1
DATE CREATED: NOV-79
MAINTAINER: DIAGNOSTIC ENGINEERING

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSIDERED 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 BY DIGITAL EQUIPMENT CORPORATION

PROGRAM HISTORY

<u>DATE</u>	<u>REVISION</u>	<u>REASON FOR REVISION</u>
JUNE-79	A	FIRST RELEASE
NOVEMBER-79	B	CORRECTIONS WERE MADE TO MULTI-TESTER SUPPORT CODE.

CONTENTS

1. ABSTRACT
2. REQUIREMENTS
 - 2.1 EQUIPMENT
 - 2.2 STORAGE
 - 2.3 PRELIMINARY PROGRAMS
3. LOADING PROCEDURE
4. STARTING PROCEDURE
 - 4.1 CONTROL SWITCH SETTINGS
 - 4.2 STARTING ADDRESS
 - 4.3 PROGRAM 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 ACT, APT AND XXDP COMPATIBILITY
9. PROGRAM DESCRIPTION
 - 9.1 CJKDCB
10. LISTING
 - 10.1 CJKDCB

1.

ABSTRACT

THE TWO PROGRAMS:

CJKDCB, CJKDDB

ARE DESIGN TO DETECT AND REPORT LOGIC FAULTS IN THE F-11 MMU AND FLOATING POINT CHIP SET. THE DESIGN IS AN ATTEMPT TO REACH ALL MICRO-CODE LOCATIONS. TESTS ARE PARTITIONED INTO TWO STAND-ALONE PROGRAMS DESCRIBED BELOW.

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

A. CJKDCB

CJKDCB TESTS: (FLOATING POINT TEST 1)

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)
 ADDF, ADDD AND SUBD (ALL CONDITIONS NOT TESTED IN DFFPA)

 CMPD AND CMPF
 DIVD AND DIVF
 MULD AND MULF
 MODD AND MODF

B. CJKDDA

CJKDDA TESTS: (FLOATING POINT TEST 2)

STF AND STD (ALL MODES)
 STCFD AND STCDF
 CLRD AND CLRF
 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 PROCESSOR WITH THE DCF11-AA, KTF11-AA AND KEF11-A CHIP SET.

2.2 STORAGE

BOTH PROGRAMS REQUIRE A MEMORY SYSTEM OF AT LEAST 16K TO LOAD AND RUN.

2.3 PRELIMINARY PROGRAMS

THESE TWO DIAGNOSTICS WILL ASSUME THAT THE BASIC CENTRAL PROCESSOR IS FAULTLESS, THEREFORE WHEN IN DOUBT RUN THE DCF11-AA PROCESSOR DIAGNOSTICS BEFORE THESE FLOATING POINT DIAGNOSTICS.

3. LOADING PROCEDURE

THE PROGRAMS WILL BE SUPPLIED ON THE 11/23 DIAGNOSTIC MEDIA. REFER TO THE XXDP OPERATING MANUAL FOR FURTHER INFORMATION.

4. STARTING PROCEDURE

4.1 CONTROL SWITCH SETTINGS

SEE SECTION 5.1

4.2 PROGRAM AND OPERATOR ACTION

1. LOAD PROGRAM INTO MEMORY
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 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>

6. ERRORS

6.1 SUMMARIES

WHEN AN ERROR IS ENCOUNTERED, AN ERROR MESSAGE ACCOMPANIED BY THE ERROR PC ARE TYPED. THERE ARE FOUR STANDARD ERROR MESSAGES USED, DESCRIBING THE PROBABLE CAUSE OF FAILURE, SUCH AS: PROBABLY BAD MMU CHIP; BAD FP1 CHIP; BAD HYBRID FP CHIP; FLOATING POINT ERROR.

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 2 SECONDS FOR EACH PROGRAM ON ANY PASS.

8.2 STACK POINTER

THE STACK POINTER IS INITIALIZED TO 1100 IN EACH OF THE TWO 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.

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 TWO 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 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 ACT, APT AND XXDP COMPATIBILITY

THESE PROGRAMS ARE FULLY COMPATIBLE WITH:

APT
ACT
XXDP MONITOR AND CHAIN PROGRAMS.

9. PROGRAM DESCRIPTION

TEST 1 LDFPS, STFPS AND DATA PATHS TEST

THIS IS A TEST OF THE LDFPS (LOAD FLOATING POINT STATUS) AND STFPS (STORE FLOATING POINT STATUS) INSTRUCTIONS. VARIOUS PATTERN ARE USED AND RUN THROUGH THE FLOATING POINT STATUS REGISTER. ONLY DMO AND SMO ARE USED. NOTE THAT A MASK MUST BE USED BECAUSE SOME OF THE FPS BITS CANNOT BE SET.

TEST 2 CFCC TEST

THIS IS A TEST OF THE COPY CONDITION CODES INSTRUCTION, CFCC.

TEST 3 SETF, SETD, SETI AND SETL TEST

THIS IS A TEST OF THE SETF, SETD, SETI AND SETL INSTRUCTIONS. EACH INSTRUCTION IS EXECUTED WITH THE FPS CONTAINING ALL ONES AND ALSO WITH THE FPS CLEAR. THE RESULT OF EACH SITUATION IS CHECKED.

TEST 4 ILLEGAL FPP OP CODES AND STST TEST

THIS IS A TEST OF THE FPP OPERATION CODES:

170003
170004
:
170010
170013
170014
:
170077

THESE ARE ILLEGAL INSTRUCTIONS AND WITH INTERRUPTS ENABLED SHOULD CAUSE A TRAP TO 244. ALSO TESTED HERE IS THE INSTRUCTION: STST R1, WHICH SHOULD PUT THE FEC CODE 2 IN R1, AFTER ANY OF THE ABOVE OP CODES IS EXECUTED.

TEST 5 FID, INTERRUPT DISABLE, BIT TEST

THIS IS A TEST OF FPS BIT 14 (FID) OR FLOATING INTERRUPT DISABLE. AN ILLEGAL INSTRUCTION IS EXECUTED WITH FID=1. NO INTERRUPT SHOULD OCCUR.

TEST 6 LDD AND STD, WITH SRC AND DST MODE 1, TEST

THIS IS A TEST OF BOTH THE INSTRUCTION:

LDD (RO),ACO

AND THE INSTRUCTION:

STD ACO,(RO)

MOST OF THE FAILURES ARE ISOLATED TO THE SRC OR DST FLOWS. NOTE THAT THE INTEGRITY OF ACO HAS NOT BEEN ASSURED. THIS MEANS THAT IN SOME CASES IT WILL BE IMPOSSIBLE TO ISOLATE CERTAIN DATA PATTERN FAILURES TO EITHER THE FLOWS OR THIS ACCUMULATOR.

TEST 7 FSRC MODE 0 TEST

THIS IS A TEST OF FSRC MODE ZERO USING THE LDD AND LDF INSTRUCTIONS.

TEST 10 FDST MODE 0 TEST

THIS IS A TEST OF THE STORE INSTRUCTIONS, STD AND STF, WITH FDST MODE 0.

TEST 11 ACCUMULATORS DATA PATTERNS TEST

THIS IS A TEST OF THE FLOATING POINT PROCESSOR ACCUMULATORS.

EACH ACCUMULATOR IS TESTED IN TWO WAYS:

1 TEST PATTERN GENERATED BY FLOATING A ONE ACROSS A FIELD OF ZEROES.

2 TEST PATTERN GENERATED BY FLOATING A ZERO ACROSS A FIELD OF ONES.

EACH OF ACCUMULATORS ACO THROUGH AC5 IS TESTED.

TEST 12 FPP ACCUMULATORS DUAL ADDRESS TEST

THIS TEST PERFORMS A DUAL ADDRESSING TEST ON THE FLOATING ACCUMULATORS. NOTE THAT ACCUMULATOR ZERO IS USED TO ACCESS ALL THE OTHERS.

TEST 13 FSRC MODE 0 WITH ILLEGAL ACCUMULATOR TEST

THIS IS A TEST OF FSRC MODE 0 WITH ACCUMULATORS 6 AND 7. USE OF EITHER OF THESE NON-EXISTENT ACCUMULATORS SHOULD RESULT IN A TRAP TO 244 WITH FEC=2 (ILLEGAL FPP INSTRUCTION).

TEST 14 FSRC MODE 2 TEST

THIS IS A TEST OF FSRC MODE 2, AUTO INCREMENT MODE.

TEST 15 FSRC MODE 4 TEST

THIS IS A TEST OF FSRC MODE 4, AUTO DECREMENT MODE.

TEST 16 FSRC MODE 2, WITH FD=0, TEST

THIS IS A TEST OF FSRC MODE 2 WITH FD=0. (AUTO INCREMENT)

TEST 17 FSRC MODE 2 WITH GR7, IMMEDIATE MODE, TEST

THIS IS A TEST OF FSRC MODE 2 USING GR7 (THE PC). THIS IS IMMEDIATE MODE.

TEST 20 FSRC MODE 3 TEST

THIS IS A TEST OF FSRC MODE 3, AUTO INCREMENT DEFERRED

TEST 21 FSRC MODE 5 TEST

THIS IS A TEST OF FSRC MODE 5, AUTO DECREMENT DEFERRED.

TEST 22 FSRC MODE 6 TEST

THIS IS A TEST OF FSRC MODE 6, INDEX MODE

TEST 23 FSRC MODE 7 TEST

THIS IS A TEST OF FSRC MODE 7, INDEX DEFERRED MODE.

TEST 24 (BUT EZBT Y8), (BUT ENBT) AND (BUT FIUV) TEST

THIS IS A TEST OF THE (BUT EZBT Y8) FORK, THE (BUT ENBT) FORK AND (BUT FIUV) FORK IN THE LOAD INSTRUCTION FLOWS.
EACH OF THE PATTERNS:

0
+NUM
-NUM
-0

IS LOADED TWICE, ONCE WITH AC>0 THEN WITH AC=0.
AFTER EACH LOAD THE FPS IS CHECK TO INSURE THAT
CONTROL WAS PASSED THROUGH WITH THE FORKS PROPERLY.

TEST 25 ADDF, ADDD, SUBF AND SUBD WITH FSRC=AC=0 TEST

THIS IS A TEST OF ADD AND SUB WITH FSRC=AC=0

TEST 26 ADDD AND SUB WITH FSRC=0

THIS IS A TEST OF ADD AND SUB WITH FSRC=0.

TEST 27 SUBD WITH AC=0 TEST

THIS IS A TEST OF SUBD WITH AC=0. BOTH POSITIVE AND
NEGATIVE FSRC'S ARE TRIED.

TEST 30 ADDD WITH AC=0 TEST

POSITIVE AND NEGATIVE FSRC'S ARE TRIED.

TEST 31 ADDF AND ADDD WITH E(AC)=E(FSRC) AND (BUT FT) TEST

THIS IS A TEST OF THE ADD INSTRUCTION WITH THE
OPERANDS HAVING EQUAL EXPONENTS. THE (BUT FT) FORK
IN THE ROUND/TRUNK FLOWS IS ALSO TESTED.

TEST 32 ADDF AND ADDD WITH E(AC) LESS THAN E(FSRC) TEST

THIS IS A TEST OF THE ADDD AND ADDF INSTRUCTIONS AND
THE ALIGN AC ALGORITHM FLOWS. THE CONSTANT (25 FOR
FLOATING, 57 FOR DOUBLE) USED IS CHECKED. THEN
SIMPLE AND WORST CASE ALIGNMENT SITUATIONS ARE
TRIED. NOTE E(AC) IS LESS THEN E(FSRC)

TEST 33 ADDF AND ADDD WITH E(AC) GREATER THAN E(FSRC) TEST

THIS IS A TEST OF THE ADDD AND ADDF INSTRUCTIONS AND
THE ALIGN FSRC ALGORITHM FLOWS. FIRST THE CONSTANT
USED IS CHECKED. THEN SIMPLE AND WORST CASE
ALIGNMENT SITUATIONS ARE TRIED. NOTE E(AC) IS
GREATER THAN E(FSRC).

TEST 34 ADDD WITH NEGATIVE OPRANDS TEST

THIS IS A TEST OF THE ADDD INSTRUCTION WITH NEGATIVE
OPERANDS. EVERY COMBINATION OF OPERAND SIGNS IS
TRIED.

TEST 35 SUBD TEST

THIS IS A TEST OF THE SUBD INSTRUCTION. BOTH A POSITIVE AND A NEGATIVE NUMBER IS SUBTRACTED FROM IT SELF

TEST 36 NORMALIZE ALGORITHM TEST

THIS IS A TEST OF THE NORMALIZE FLOW ALGORITHM. TWO PATTERNS ARE USED, FIRST THE MINIMUM SITUATION REQUIRING ONE LEFT SHIFT AND THEN THE MAXIMUM SITUATION REQUIRING 56 SHIFTS.

TEST 37 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

TEST 40 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 41 LDCFD AND LDCDF TEST

THIS IS A TEST OF LDCFD AND LDCDF.

TEST 42 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 43 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 44 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 45 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 46 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.

TEST 47 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 50 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 51 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 52 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 53 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 54 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 55 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.

TEST 56 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 57 UNDER\OVER FLOW, USING MODD WITH TRAPS DISABLED, TESTTEST 60 MORE MICROCODES COVERAGE, TEST

10.

LISTING

&

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

.LIST ME
.NLIST MD,MC,CND

.ENABL ABS

.TITLE CJKDCB-A KEF11-A FP DIAG PART 1
:*COPYRIGHT (C) NOV 1979
:*DIGITAL EQUIPMENT CORP.
:*MAYNARD, MASS. 01754
:*
:*PROGRAM BY DIAGNOSTIC ENGINEERING
:*
:*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
:*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
:*
\$TN=1
\$SWR=160000 ;:HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYP0UT

000001
160000

000244
177400
000200
000011
000015

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

.SBTTL BASIC DEFINITIONS

001100

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

```

57
58      : *MISCELLANEOUS DEFINITIONS
59      000011 HT= 11      :: CODE FOR HORIZONTAL TAB
60      000012 LF= 12      :: CODE FOR LINE FEED
61      000015 CR= 15      :: CODE FOR CARRIAGE RETURN
62      000200 CRLF= 200    :: CODE FOR CARRIAGE RETURN-LINE FEED
63      177776 PS= 177776  :: PROCESSOR STATUS WORD
64      .EQUIV PS,PSW
65      177774 STKLMT= 177774 :: STACK LIMIT REGISTER
66      177772 PIRQ= 177772 :: PROGRAM INTERRUPT REQUEST REGISTER
67      177570 DSWR= 177570 :: HARDWARE SWITCH REGISTER
68      177570 DDISP= 177570 :: HARDWARE DISPLAY REGISTER
69
70      : *GENERAL PURPOSE REGISTER DEFINITIONS
71      000000 R0= %0      :: GENERAL REGISTER
72      000001 R1= %1      :: GENERAL REGISTER
73      000002 R2= %2      :: GENERAL REGISTER
74      000003 R3= %3      :: GENERAL REGISTER
75      000004 R4= %4      :: GENERAL REGISTER
76      000005 R5= %5      :: GENERAL REGISTER
77      000006 R6= %6      :: GENERAL REGISTER
78      000007 R7= %7      :: GENERAL REGISTER
79      000006 SP= %6      :: STACK POINTER
80      000007 PC= %7      :: PROGRAM COUNTER
81
82      : *PRIORITY LEVEL DEFINITIONS
83      000000 PR0= 0      :: PRIORITY LEVEL 0
84      000040 PR1= 40     :: PRIORITY LEVEL 1
85      000100 PR2= 100    :: PRIORITY LEVEL 2
86      000140 PR3= 140    :: PRIORITY LEVEL 3
87      000200 PR4= 200    :: PRIORITY LEVEL 4
88      000240 PR5= 240    :: PRIORITY LEVEL 5
89      000300 PR6= 300    :: PRIORITY LEVEL 6
90      000340 PR7= 340    :: PRIORITY LEVEL 7
91
92      : *'SWITCH REGISTER' SWITCH DEFINITIONS
93      100000 SW15= 100000
94      040000 SW14= 40000
95      020000 SW13= 20000
96      010000 SW12= 10000
97      004000 SW11= 4000
98      002000 SW10= 2000
99      001000 SW09= 1000
100     000400 SW08= 400
101     000200 SW07= 200
102     000100 SW06= 100
103     000040 SW05= 40
104     000020 SW04= 20
105     000010 SW03= 10
106     000004 SW02= 4
107     000002 SW01= 2
108     000001 SW00= 1
109     .EQUIV SW09,SW9
110     .EQUIV SW08,SW8
111     .EQUIV SW07,SW7
112     .EQUIV SW06,SW6
    
```

```

113      .EQUIV SW05,SW5
114      .EQUIV SW04,SW4
115      .EQUIV SW03,SW3
116      .EQUIV SW02,SW2
117      .EQUIV SW01,SW1
118      .EQUIV SW00,SW0
119
120      ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
121      100000 BIT15= 100000
122      040000 BIT14= 40000
123      020000 BIT13= 20000
124      010000 BIT12= 10000
125      004000 BIT11= 4000
126      002000 BIT10= 2000
127      001000 BIT09= 1000
128      000400 BIT08= 400
129      000200 BIT07= 200
130      000100 BIT06= 100
131      000040 BIT05= 40
132      000020 BIT04= 20
133      000010 BIT03= 10
134      000004 BIT02= 4
135      000002 BIT01= 2
136      000001 BIT00= 1
137      .EQUIV BIT09,BIT9
138      .EQUIV BIT08,BIT8
139      .EQUIV BIT07,BIT7
140      .EQUIV BIT06,BIT6
141      .EQUIV BIT05,BIT5
142      .EQUIV BIT04,BIT4
143      .EQUIV BIT03,BIT3
144      .EQUIV BIT02,BIT2
145      .EQUIV BIT01,BIT1
146      .EQUIV BIT00,BIT0
147
148      ;*BASIC 'CPU' TRAP VECTOR ADDRESSES
149      000004 ERRVEC= 4          ;; TIME OUT AND OTHER ERRORS
150      000010 RESVEC= 10       ;; RESERVED AND ILLEGAL INSTRUCTIONS
151      000014 TBITVEC=14       ;; 'T' BIT
152      000014 TRTVEC= 14       ;; TRACE TRAP
153      000014 BPTVEC= 14       ;; BREAKPOINT TRAP (BPT)
154      000020 IOTVEC= 20       ;; INPUT/OUTPUT TRAP (IOT) **SCOPE**
155      000024 PWRVEC= 24       ;; POWER FAIL
156      000030 EMTVEC= 30       ;; EMULATOR TRAP (EMT) **ERROR**
157      000034 TRAPVEC=34       ;; 'TRAP' TRAP
158      000060 TKVEC= 60        ;; TTY KEYBOARD VECTOR
159      000064 TPVEC= 64        ;; TTY PRINTER VECTOR
160      000240 PIRQVEC=240      ;; PROGRAM INTERRUPT REQUEST VECTOR
161      .SBTTL FPP REGISTER DEFINITIONS
162      000000 AC0 =%0
163      000001 AC1 =%1
164      000002 AC2 =%2
165      000003 AC3 =%3
166      000004 AC4 =%4
167      000005 AC5 =%5
168      000006 AC6 =%6
    
```

```
169          000007          AC7      =%7
170
171          .SBTTL TRAP CATCHER
172
173          000000          .=0
174          ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
175          ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
176          ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
177          .=174
178 000174 000000  DISPREG: .WORD 0          ;;SOFTWARE DISPLAY REGISTER
179 000176 000000  SWREG:   .WORD 0          ;;SOFTWARE SWITCH REGISTER
180          .SBTTL STARTING ADDRESS(ES)
181 000200 000137 001466  JMP      @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
```

Line	Address	Value	Label	Description
182			.SBTTL	COMMON TAGS
183				
184			::*****	
185			::*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS	
186			::*USED IN THE PROGRAM.	
187				
188	001100		SCMTAG: . =1100	:::START OF COMMON TAGS
189	001100	000000	.WORD 0	
190	001100	000000	\$TSTNM: .BYTE 0	:::CONTAINS THE TEST NUMBER
191	001102	000	\$ERFLG: .BYTE 0	:::CONTAINS ERROR FLAG
192	001103	000	\$ICNT: .WORD 0	:::CONTAINS SUBTEST ITERATION COUNT
193	001104	000000	\$LPADR: .WORD 0	:::CONTAINS SCOPE LOOP ADDRESS
194	001106	000000	\$LPERR: .WORD 0	:::CONTAINS SCOPE RETURN FOR ERRORS
195	001110	000000	\$ERTTL: .WORD 0	:::CONTAINS TOTAL ERRORS DETECTED
196	001112	000000	\$ITEMB: .BYTE 0	:::CONTAINS ITEM CONTROL BYTE
197	001114	000	\$ERMAX: .BYTE 1	:::CONTAINS MAX. ERRORS PER TEST
198	001115	001	\$ERRPC: .WORD 0	:::CONTAINS PC OF LAST ERROR INSTRUCTION
199	001116	000000	\$GDADR: .WORD 0	:::CONTAINS ADDRESS OF 'GOOD' DATA
200	001120	000000	\$BDADR: .WORD 0	:::CONTAINS ADDRESS OF 'BAD' DATA
201	001122	000000	\$GDDAT: .WORD 0	:::CONTAINS 'GOOD' DATA
202	001124	000000	\$BDDAT: .WORD 0	:::CONTAINS 'BAD' DATA
203	001126	000000	.WORD 0	:::RESERVED--NOT TO BE USED
204	001130	000000	.WORD 0	
205	001132	000000	.WORD 0	
206	001134	000	\$AUTOB: .BYTE 0	:::AUTOMATIC MODE INDICATOR
207	001135	000	\$INTAG: .BYTE 0	:::INTERRUPT MODE INDICATOR
208	001136	000000	.WORD 0	
209	001140	177570	SWR: .WORD DSWR	:::ADDRESS OF SWITCH REGISTER
210	001142	177570	DISPLAY: .WORD DDISP	:::ADDRESS OF DISPLAY REGISTER
211	001144	177560	\$TKS: 177560	:::TTY KBD STATUS
212	001146	177562	\$TKB: 177562	:::TTY KBD BUFFER
213	001150	177564	\$TPS: 177564	:::TTY PRINTER STATUS REG. ADDRESS
214	001152	177566	\$TPB: 177566	:::TTY PRINTER BUFFER REG. ADDRESS
215	001154	000	\$NULL: .BYTE 0	:::CONTAINS NULL CHARACTER FOR FILLS
216	001155	002	\$FILLS: .BYTE 2	:::CONTAINS # OF FILLER CHARACTERS REQUIRED
217	001156	012	\$FILLC: .BYTE 12	:::INSERT FILL CHARS. AFTER A 'LINE FEED'
218	001157	000	\$TPFLG: .BYTE 0	:::'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
219	001160	000000	\$REGAD: .WORD 0	:::CONTAINS THE ADDRESS FROM WHICH (\$REGO) WAS OBTAINED
220				
221	001162	000000	\$REG0: .WORD 0	:::CONTAINS ((\$REGAD)+0)
222	001164	000000	\$REG1: .WORD 0	:::CONTAINS ((\$REGAD)+2)
223	001166	000000	\$REG2: .WORD 0	:::CONTAINS ((\$REGAD)+4)
224	001170	000000	\$REG3: .WORD 0	:::CONTAINS ((\$REGAD)+6)
225	001172	000000	\$REG4: .WORD 0	:::CONTAINS ((\$REGAD)+10)
226	001174	000000	\$REG5: .WORD 0	:::CONTAINS ((\$REGAD)+12)
227	001176	000000	\$REG6: .WORD 0	:::CONTAINS ((\$REGAD)+14)
228	001200	000000	\$REG7: .WORD 0	:::CONTAINS ((\$REGAD)+16)
229	001202	000000	\$REG10: .WORD 0	:::CONTAINS ((\$REGAD)+20)
230	001204	000000	\$REG11: .WORD 0	:::CONTAINS ((\$REGAD)+22)
231	001206	000000	\$REG12: .WORD 0	:::CONTAINS ((\$REGAD)+24)
232	001210	000000	\$REG13: .WORD 0	:::CONTAINS ((\$REGAD)+26)
233	001212	000000	\$REG14: .WORD 0	:::CONTAINS ((\$REGAD)+30)
234	001214	000000	\$REG15: .WORD 0	:::CONTAINS ((\$REGAD)+32)
235	001216	000000	\$REG16: .WORD 0	:::CONTAINS ((\$REGAD)+34)
236	001220	000000	\$REG17: .WORD 0	:::CONTAINS ((\$REGAD)+36)
237	001222	000000	\$REG20: .WORD 0	:::CONTAINS ((\$REGAD)+40)

```

238 001224 000000 $REG21: .WORD 0 ::CONTAINS (($REGAD)+42)
239 001226 000000 $REG22: .WORD 0 ::CONTAINS (($REGAD)+44)
240 001230 000000 $REG23: .WORD 0 ::CONTAINS (($REGAD)+46)
241 001232 000000 $TMP0: .WORD 0 ::USER DEFINED
242 001234 000000 $TMP1: .WORD 0 ::USER DEFINED
243 001236 000000 $TMP2: .WORD 0 ::USER DEFINED
244 001240 000000 $TMP3: .WORD 0 ::USER DEFINED
245 001242 000000 $TMP4: .WORD 0 ::USER DEFINED
246 001244 000000 $TMP5: .WORD 0 ::USER DEFINED
247 001246 000000 $TMP6: .WORD 0 ::USER DEFINED
248 001250 000000 $TMP7: .WORD 0 ::USER DEFINED
249 001252 000000 $TMP10: .WORD 0 ::USER DEFINED
250 001254 000000 $TMP11: .WORD 0 ::USER DEFINED
251 001256 000000 $TMP12: .WORD 0 ::USER DEFINED
252 001260 000000 $TMP13: .WORD 0 ::USER DEFINED
253 001262 000000 $TMP14: .WORD 0 ::USER DEFINED
254 001264 000000 $TMP15: .WORD 0 ::USER DEFINED
255 001266 000000 $TMP16: .WORD 0 ::USER DEFINED
256 001270 000000 $TMP17: .WORD 0 ::USER DEFINED
257 001272 000000 $TMP20: .WORD 0 ::USER DEFINED
258 001274 000000 $TMP21: .WORD 0 ::USER DEFINED
259 001276 000000 $TMP22: .WORD 0 ::USER DEFINED
260 001300 000000 $TMP23: .WORD 0 ::USER DEFINED
261 001302 000000 $TIMES: 0 ::MAX. NUMBER OF ITERATIONS
262 001304 000000 $ESCAPE: 0 ::ESCAPE ON ERROR ADDRESS
263 001306 177607 000377 $BELL: .ASCIZ <207><377><377> ::CODE FOR BELL
264 001312 077 $QUES: .ASCII /?/ ::QUESTION MARK
265 001313 015 $CRLF: .ASCII <15> ::CARRIAGE RETURN
266 001314 000012 $LF: .ASCIZ <12> ::LINE FEED
267 *****
268 .SBTTL APT MAILBOX-ETABLE
269 *****
270
271 .EVEN
272 001316 $MAIL: ::APT MAILBOX
273 001316 000000 $MSGTY: .WORD AMSGTY ::MESSAGE TYPE CODE
274 001320 000000 $FATAL: .WORD AFATAL ::FATAL ERROR NUMBER
275 001322 000000 $TESTN: .WORD ATESTN ::TEST NUMBER
276 001324 000000 $PASS: .WORD APASS ::PASS COUNT
277 001326 000000 $DEVCT: .WORD ADEVCT ::DEVICE COUNT
278 001330 000000 $UNIT: .WORD AUNIT ::I/O UNIT NUMBER
279 001332 000000 $MSGAD: .WORD AMSGAD ::MESSAGE ADDRESS
280 001334 000000 $MSGLG: .WORD AMSGLG ::MESSAGE LENGTH
281 001336 $ETABLE: ::APT ENVIRONMENT TABLE
282 001336 000 $ENV: .BYTE AENV ::ENVIRONMENT BYTE
283 001337 000 $ENVM: .BYTE AENVM ::ENVIRONMENT MODE BITS
284 001340 000000 $SWREG: .WORD ASWREG ::APT SWITCH REGISTER
285 001342 000000 $USWR: .WORD AUSWR ::USER SWITCHES
286 001344 000000 $CPUOP: .WORD ACPUOP ::CPU TYPE, OPTIONS
287 * BITS 15-11=CPU TYPE
288 * 11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
289 * 11/70=06,PDQ=07,Q=10
290 * BIT 10=REAL TIME CLOCK
291 * BIT 9=FLOATING POINT PROCESSOR
292 * BIT 8=MEMORY MANAGEMENT
293 001346 000 $MAMS1: .BYTE AMAMS1 ::HIGH ADDRESS,M.S. BYTE
    
```

```

294 001347 000 $MTYP1: .BYTE AMTYP1 ;;MEM. TYPE,BLK#1
295 : * MEM. TYPE BYTE -- (HIGH BYTE)
296 : * 900 NSEC CORE=001
297 : * 300 NSEC BIPOLAR=002
298 : * 500 NSEC MOS=003
299 001350 000000 $MADR1: .WORD AMADR1 ;;HIGH ADDRESS,BLK#1
300 : * MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF 'TYPE' ABOVE
301 001352 000 $MAMS2: .BYTE AMAMS2 ;;HIGH ADDRESS,M.S. BYTE
302 001353 000 $MTYP2: .BYTE AMTYP2 ;;MEM. TYPE,BLK#2
303 001354 000000 $MADR2: .WORD AMADR2 ;;MEM.LAST ADDRESS,BLK#2
304 001356 000 $MAMS3: .BYTE AMAMS3 ;;HIGH ADDRESS,M.S.BYTE
305 001357 000 $MTYP3: .BYTE AMTYP3 ;;MEM. TYPE,BLK#3
306 001360 000000 $MADR3: .WORD AMADR3 ;;MEM.LAST ADDRESS,BLK#3
307 001362 000 $MAMS4: .BYTE AMAMS4 ;;HIGH ADDRESS,M.S.BYTE
308 001363 000 $MTYP4: .BYTE AMTYP4 ;;MEM. TYPE,BLK#4
309 001364 000000 $MADR4: .WORD AMADR4 ;;MEM.LAST ADDRESS,BLK#4
310 001366 000000 $VECT1: .WORD AVECT1 ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
311 001370 000000 $VECT2: .WORD AVECT2 ;;INTERRUPT VECTOR#2BUS PRIORITY#2
312 001372 000000 $BASE: .WORD ABASE ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
313 001374 000000 $DEVN: .WORD ADEVN ;;DEVICE MAP
314 001376 000000 $CDW1: .WORD ACDW1 ;;CONTROLLER DESCRIPTION WORD#1
315 001400 000000 $CDW2: .WORD ACDW2 ;;CONTROLLER DESCRIPTION WORD#2
316 001402 000000 $DDW0: .WORD ADDW0 ;;DEVICE DESCRIPTOR WORD#0
317 001404 000000 $DDW1: .WORD ADDW1 ;;DEVICE DESCRIPTOR WORD#1
318 001406 000000 $DDW2: .WORD ADDW2 ;;DEVICE DESCRIPTOR WORD#2
319 001410 000000 $DDW3: .WORD ADDW3 ;;DEVICE DESCRIPTOR WORD#3
320 001412 000000 $DDW4: .WORD ADDW4 ;;DEVICE DESCRIPTOR WORD#4
321 001414 000000 $DDW5: .WORD ADDW5 ;;DEVICE DESCRIPTOR WORD#5
322 001416 000000 $DDW6: .WORD ADDW6 ;;DEVICE DESCRIPTOR WORD#6
323 001420 000000 $DDW7: .WORD ADDW7 ;;DEVICE DESCRIPTOR WORD#7
324 001422 000000 $DDW8: .WORD ADDW8 ;;DEVICE DESCRIPTOR WORD#8
325 001424 000000 $DDW9: .WORD ADDW9 ;;DEVICE DESCRIPTOR WORD#9
326 001426 000000 $DDW10: .WORD ADDW10 ;;DEVICE DESCRIPTOR WORD#10
327 001430 000000 $DDW11: .WORD ADDW11 ;;DEVICE DESCRIPTOR WORD#11
328 001432 000000 $DDW12: .WORD ADDW12 ;;DEVICE DESCRIPTOR WORD#12
329 001434 000000 $DDW13: .WORD ADDW13 ;;DEVICE DESCRIPTOR WORD#13
330 001436 000000 $DDW14: .WORD ADDW14 ;;DEVICE DESCRIPTOR WORD#14
331 001440 000000 $DDW15: .WORD ADDW15 ;;DEVICE DESCRIPTOR WORD#15
332
333
334 001442 $ETEND:
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
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391

```
.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:
:ITEM NUMBER
.WORD EM1
.WORD EM2
.WORD EM3
.WORD EM4

.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

.SBTTL APT PARAMETER BLOCK

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

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

$APTHD:
$HIBTS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MBADR: .WORD $MAIL  ;;ADDRESS OF APT MAILBOX (BITS 0-15)
$STMT:  .WORD 10     ;;RUN TIM OF LONGEST TEST
$PASTM: .WORD 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   .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

```
001442
001442 062743
001444 062771
001446 063025
001450 063053

001452
000046 057322
000052 000000
001452

001452
000024 000200
000044 001452
001452

001452 000000
001454 001316
001456 000010
001460 000040
001462 000000
001464 000052
```



```

392 001466 START:
393 .SBTTL INITIALIZE THE COMMON TAGS
394 ::CLEAR THE COMMON TAGS ($CMTAG) AREA
395 001466 012706 001100 MOV #SCMTAG,R6 ::FIRST LOCATION TO BE CLEARED
396 001472 005026 CLR (R6)+ ::CLEAR MEMORY LOCATION
397 001474 022706 001140 CMP #SWR,R6 ::DONE?
398 001500 001374 BNE -6 ::LOOP BACK IF NO
399 001502 012706 001100 MOV #STACK,SP ::SETUP THE STACK POINTER
400 ::INITIALIZE A FEW VECTORS
401 001506 012737 057454 000020 MOV #SCOPE,@IOTVEC ::IOT VECTOR FOR SCOPE ROUTINE
402 001514 012737 000340 000022 MOV #340,@IOTVEC+2 ::LEVEL 7
403 001522 012737 057734 000030 MOV #ERROR,@EMTVEC ::EMT VECTOR FOR ERROR ROUTINE
404 001530 012737 000340 000032 MOV #340,@EMTVEC+2 ::LEVEL 7
405 001536 012737 062070 000034 MOV #STRAP,@TRAPVEC ::TRAP VECTOR FOR TRAP CALLS
406 001544 012737 000340 000036 MOV #340,@TRAPVEC+2 ::LEVEL 7
407 001552 012737 062156 000024 MOV #SPWRDN,@PWRVEC ::POWER FAILURE VECTOR
408 001560 012737 000340 000026 MOV #340,@PWRVEC+2 ::LEVEL 7
409 001566 016767 055466 055456 MOV SENDCT,$EOPCT ::SETUP END-OF-PROGRAM COUNTER
410 001574 005067 177502 CLR $TIMES ::INITIALIZE NUMBER OF ITERATIONS
411 001600 005067 177500 CLR $ESCAPE ::CLEAR THE ESCAPE ON ERROR ADDRESS
412 001604 112767 000001 177303 MOVB #1,$ERMAX ::ALLOW ONE ERROR PER TEST
413 ::INITIALIZE THE 'T-BIT' TRAP VECTOR, THEN LOAD LOCATION '$RTRN', IN
414 ::THE 'END-OF-PASS' ($EOP) ROUTINE, WITH A 'RTI' OR 'RTT'.
415 001612 012737 057424 000014 MOV #RTRN,@TBITVEC ::SET 'T' BIT VECTOR TO $RTRN
416 001620 012737 000340 000016 MOV #340,@TBITVEC+2 ::LEVEL 7
417 001626 012767 000002 055570 MOV #RTI,$RTRN ::SET $RTRN TO A RTI
418 001634 012737 001662 000010 MOV #65,$RESVEC ::TRY TO DO A RTT
419 001642 005046 CLR -(SP) ::DUMMY PS
420 001644 012746 001652 MOV #64,$-(SP) ::AND PC
421 001650 000006 RTT ::TRY THE RTT
422 001652 012767 000006 055544 64$: MOV #RTT,$RTRN ::RTT IS LEGAL--SET $RTRN TO A RTT
423 001660 000402 BR 66$
424 001662 062706 000010 65$: ADD #10,SP ::RTT ILLEGAL--CLEAN OFF THE STACK
425 001666 012737 000012 000010 66$: MOV #RESVEC+2,@RESVEC ::RESTORE TRAP CATCHER
426 001674 005067 055532 CLR $TBIT ::CLEAR 'T' BIT SWITCH
427 001700 012767 001700 177200 MOV #.,$LPADR ::INITIALIZE THE LOOP ADDRESS FOR SCOPE
428 001706 012767 001706 177174 MOV #.,$LPERR ::SETUP THE ERROR LOOP ADDRESS
429 ::SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
430 ::EQUAL TO A '-1', SETUP FOR A SOFTWARE SWITCH REGISTER.
431 001714 013746 000004 MOV @ERRVEC,-(SP) ::SAVE ERROR VECTOR
432 001720 012737 001754 000004 MOV #67,@ERRVEC ::SET UP ERROR VECTOR
433 001726 012767 177570 177204 MOV #DSWR,$SWR ::SETUP FOR A HARDWARE SWICH REGISTER
434 001734 012767 177570 177200 MOV #DDISP,$DISPLAY ::AND A HARDWARE DISPLAY REGISTER
435 001742 022777 177777 177170 CMP #-1,$SWR ::TRY TO REFERENCE HARDWARE SWR
436 001750 001012 BNE 69$ ::BRANCH IF NO TIMEOUT TRAP OCCURRED
437 ::AND THE HARDWARE SWR IS NOT = -1
438 001752 000403 BR 68$ ::BRANCH IF NO TIMEOUT
439 001754 012716 001762 67$: MOV #68,$(SP) ::SET UP FOR TRAP RETURN
440 001760 000002 RTI
441 001762 012767 000176 177150 68$: MOV #SWREG,$SWR ::POINT TO SOFTWARE SWR
442 001770 012767 000174 177144 MOV #DISPREG,$DISPLAY
443 001776 012637 000004 69$: MOV (SP)+,@ERRVEC ::RESTORE ERROR VECTOR
444
445 002002 005067 177316 CLR $PASS ::CLEAR PASS COUNT
446 002006 132767 000200 177323 BITB #APTSIZE,$ENVM ::TEST USER SIZE UNDER APT
447 002014 001403 BEQ 70$ ::YES,USE NON-APT SWITCH
    
```

```

448 002016 012767 001340 177114      MOV    #$$SWREG,SWR      ;;NO,USE APT SWITCH REGISTER
449 002024
450
451
452 002024 005227 177777      .SBTTL TYPE PROGRAM NAME
453 002030 001053              ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
454 002032 022737 057322 000042      INC    #-1              ;;FIRST TIME?
455 002040 001447              BNE    71$              ;;BRANCH IF NO
456 002042 104401 002110      CMP    #SENDAD,@#42    ;;ACT-11?
457
458 002046 005737 000042      BEQ    71$              ;;BRANCH IF YES
459 002052 001012              TYPE   ,72$            ;;TYPE ASCIZ STRING
460 002054 126727 177256 000001      .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
461 002062 001406              TST    @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
462 002064 026727 177050 000176      BNE    73$              ;;BRANCH IF YES
463 002072 001005              CMPB   $ENV,#1         ;;ARE WE RUNNING UNDER APT?
464 002074 104406              BEQ    73$              ;;BRANCH IF YES
465 002076 000403              CMP    SWR,#SWREG     ;;SOFTWARE SWITCH REG SELECTED?
466 002100 112767 000001 177026      BNE    74$              ;;BRANCH IF NO
467 002106
468 002106 000424              GTSWR                    ;;GET SOFT-SWR SETTINGS
469
470 002160
471
472 002160
473 002160 012706 001100      BR     74$              ;;SET AUTO-MODE INDICATOR
474 002164 012737 002220 000004      MOVB   #1,$AUTOB
475 002172 012737 000340 000006      BR     71$              ;;GET OVER THE ASCIZ
476 002200 012737 000002 164000      .:72$: .ASCIZ <CRLF>*CJKDCB, KEF11-A FP DIAGNOSTIC PART 1* <CRLF>
477 002206 012737 000006 000004      .:71$:
478 002214 005037 000006
479 002220
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496 002220 000004
497 002222 104414
498 002224 012700 177777      LOOP: MOV    #STACK,SP      ;SET UP STACK POINTER
499 002230 012737 002336 000244      MOV    #1$,@#4         ;SET UP FOR TIMEOUT IF NO MULTI-TESTER
500 002236 012737 002350 000010      MOV    #340,@#6
501 002244 012737 002402 000004      MOV    #2,@#164000    ;SET 'GO' ON MULTI-TESTER
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
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
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

```

*****
*TEST 1          LDFPS, STFPS AND DATA PATHS TEST
*
*THIS IS A TEST OF THE LDFPS (LOAD FLOATING POINT STATUS) AND STFPS
*(STORE FLOATING POINT STATUS) INSTRUCTIONS. A COUNT PATTERN IS GENERATED
*AND RUN THROUGH THE FLOATING POINT STATUS REGISTER.
*DMD AND SMO ARE USED.
*NOTE THAT A MASK MUST BE USED BECAUSE SOME OF THE FPS BITS CANNOT
*BE SET.
*****
TST1:  SCOPE
        LPERR                    ;SET UP THE LOOP ON ERROR ADDRESS.
        MOV    #-1,R0             ;INITIALIZE THE COUNT PATTERN.
        MOV    #AERR1,@#FPVECT   ;SET UP FOR UNABLE TO DECODE
        MOV    #AERR2,@#10       ;FPP INSTRUCTION TRAP TO 244 OR 10.
        MOV    #AERR3,@#ERRVECT  ;IF EITHER INSTRUCTION
        ;FAILS TO GO THROUGH THE
        ;CORRECT SRC OR DST MODE AN

```

```

504                                     ;ODD ADDRESS TRAP WILL OCCUR.
505 002252
506 002252 010004
507 002254 042704 030020
508 002260 170104
509
510 002262 012701 177777
511 002266 170201
512 002270 012737 062524 000244
513 002276 010004
514 002300 042704 030020
515 002304 012737 062556 000004
516 002312 012737 062574 000010
517 002320 020401
518
519 002322 001401
520 002324 104004
521
522 002326 012700 000001
523 002332 077031
524 002334 000443
525
526
527                                     ;UNABLE TO DECODE FPP INSTRUCTION. TRAPPED TO 244.
528 002336 011637 001236
529
530 002342 022626
531 002344 104004
532 002346 000436
533
534                                     ;UNABLE TO DECODE INSTRUCTION. TRAPPED TO 10.
535 002350 021627 002254
536 002354 001405
537 002356 021627 002270
538 002362 001402
539 002364 000137 062574
540
541
542 002370 011637 001236
543 002374 022626
544 002376 104004
545 002400 000421
546
547
548 002402 021627 002254
549 002406 001405
550 002410 021627 002270
551 002414 001407
552 002416 000137 062556
553
554
555 002422 011637 001236
556 002426 022626
557 002430 104004
558 002432 000404
559
    
```

```

A1:
A11: MOV R0,R4
      BIC #30020,R4
      LDFPS R4

A12: MOV #-1,R1
      STFPS R1
      MOV #FPSPUR,@#FPVECT
      MOV R0,R4
      BIC #30020,R4
      MOV #CPSPUR,@#ERRVECT
      MOV #CPTWO,@#10
      CMP R4,R1

A2:  MOV #1,R0
      SOB R0,A1
      BR  ADONE
    
```

```

;ODD ADDRESS TRAP WILL OCCUR.
;TEST INSTRUCTION.
;TEST INSTRUCTION.
;SET UP FOR UNEXPECTED TRAPS.
;MASK OFF UNSETTABLE BITS.
;COMPARE DATA EXPECTED WITH
;THE DATA READ.
;IF NOT EQUAL GO REPORT ERROR.
;NEXT PATTERN WILL BE ALL ZERO
;DECREMENT COUNT PATTERN
;UNTIL IT IS ZERO.
    
```

```

;UNABLE TO DECODE FPP INSTRUCTION. TRAPPED TO 244.
AERR1: MOV (SP),@#$TMP2
;SAVE PC OF TRAP.
    
```

```

1$:  CMP (SP)+,(SP)+
      ERROR 4
      BR  ADONE
    
```

```

;UNABLE TO DECODE INSTRUCTION. TRAPPED TO 10.
AERR2: CMP (SP),#A11+2
;DID TRAP OCCUR OF FPP INSTRUCTION?
      BEQ 1$
      CMP (SP),#A12+2
      BEQ 1$
      JMP @#CPTWO
    
```

```

;IF NOT FPP INSTRUCTION THEN
;REPORT SPURIOUS TRAP TO 10.
;OTHERWISE REPORT IR DECODE ERROR.
    
```

```

1$:  MOV (SP),@#$TMP2
      CMP (SP)+,(SP)+
2$:  ERROR 4
      BR  ADONE
    
```

```

;TRAP TO 4 HANDLER:
AERR3: CMP (SP),#A11+2
;DID THE TRAP OCCUR ON THE
      BEQ 1$
;LDFPS INSTRUCTION?
      CMP (SP),#A12+2
;OR THE STFPS INSTRUCTION?
      BEQ 2$
      JMP @#CPSPUR
;IF NEITHER THEN REPORT
;UNEXPECTED TRAP TO 4.
    
```

```

1$:  MOV (SP),@#$TMP2
      CMP (SP)+,(SP)+
15$: ERROR 4
      BR  ADONE
    
```

560 002434 011637 001236
 561 002440 022626
 562 002442 104004
 563
 564 002444
 565 002444 104413
 566
 567
 568
 569
 570
 571
 572
 573
 574
 575
 576
 577
 578 002446 000004
 579 002450 104414
 580 002452 012700 000017
 581
 582 002456
 583 002456 170100
 584
 585 002460
 586 002460 170000
 587
 588 002462 013703 177776
 589 002466 042703 177760
 590 002472 020003
 591 002474 001002
 592
 593 002476 077011
 594 002500 000422
 595
 596 002502
 597 002502 170201
 598 002504 012737 002460 001236
 599
 600 002512 020001
 601 002514 001006
 602
 603 002516 010337 001240
 604 002522 010037 001242
 605 002526 104001
 606 002530 000762
 607
 608 002532
 609 002532 010037 001240
 610 002536 010137 001242
 611 002542 104001
 612 002544 000754
 613
 614 002546
 615 002546 104413

```

2$:  MOV    (SP),@#$TMP2
     CMP    (SP)+,(SP)+
25$: ERROR  4

ADONE:  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?).

:*****
:*TEST 2          CFCC TEST
:*
:*THIS IS A TEST OF THE COPY CONDITION CODES INSTRUCTION, CFCC.
:*
:*****
TST2:  SCOPE
       LPERR
       MOV    #17,R0                        ;SET UP THE LOOP ON ERROR ADDRESS.
                                             ;R0 CONTAINS TO TEST PATTERN.

B1:    LDFPS  R0                            ;LOAD THE TEST PATTERN

B2:    CFCC                                ;COPY CONDITION CODES.
                                             ;SEE IF PATTERN TRANSFERED.
       MOV    @#PSW,R3
       BIC    #177760,R3
       CMP    R0,R3
       BNE   BERR

B3:    SOB   R0,B1
       BR    BDONE

BERR:  STFPS  R1                            ;WAS FPS MODIFIED BY CFCC?
       MOV    #B2,@#$TMP2

       CMP    R0,R1
       BNE   BERR1

1$:    MOV    R3,@#$TMP3
       MOV    R0,@#$TMP4
       ERROR  1
       BR    B3

BERR1: MOV    R0,@#$TMP3
       MOV    R1,@#$TMP4

1$:    ERROR  1
       BR    B3

BDONE: RSETUP                                ;GO INITIALIZE THE FPS AND STACK; AND
    
```



```

672 002700 012737 002706 001236      MOV      #C35,@#$TMP2
673 002706 170011      SETD                                ;SETD FD BIT.
674
675 002710 170201      STFPS   R1
676 002712 012702 147757      MOV     #147757,R2
677 002716 020102      CMP     R1,R2                    ;RESULT CORRECT?
678 002720 001403      BEQ    1$
679 002722 010137 001126      MOV     R1,@#$BDDAT              ;STORE BAD DATA
680 002726 104001      ERROR  1
681
682 002730      1$:
683 002730 104414      LPERR                                ;SET UP THE LOOP ON ERROR ADDRESS.
684 002732 005000      C4:   CLR     R0
685 002734 170100      LDFPS  R0                          ;CLEAR FPS.
686 002736 012737 002744 001236      MOV     #C45,@#$TMP2
687
688 002744 170011      C45:  SETD                                ;SET FD BIT.
689
690 002746 170201      STFPS   R1                          ;GET RESULT.
691 002750 012702 000200      MOV     #200,R2
692 002754 020102      CMP     R1,R2                    ;RESULT CORRECT?
693 002756 001403      BEQ    1$
694 002760 010137 001126      MOV     R1,@#$BDDAT              ;STORE BAD DATA
695 002764 104001      ERROR  1
696
697 002766      1$:
698 002766 104414      LPERR                                ;SET UP THE LOOP ON ERROR ADDRESS.
699 002770 012737 000204 001250      C5:   MOV     #204,@#$TMP7
700 002776 005000      CLR     R0
701
702 003000 170100      LDFPS  R0                          ;CLEAR FPS
703 003002 012737 003010 001236      MOV     #C55,@#$TMP2
704
705 003010 170002      C55:  SETI                                ;CLEAR FL BIT.
706
707 003012 170201      STFPS   R1                          ;GET RESULT.
708 003014 005002      CLR     R2
709 003016 020201      CMP     R2,R1                    ;RESULT CORRECT?
710 003020 001403      BEQ    1$
711 003022 010137 001126      MOV     R1,@#$BDDAT              ;STORE BAD DATA
712 003026 104001      ERROR  1
713
714 003030      1$:
715 003030 104414      LPERR                                ;SET UP THE LOOP ON ERROR ADDRESS.
716 003032 012700 147757      C6:   MOV     #147757,R0
717 003036 170100      LDFPS  R0                          ;PUT 147757 INTO FPS
718 003040 012737 003046 001236      MOV     #C65,@#$TMP2
719
720 003046 170002      C65:  SETI                                ;CLEAR FL BIT.
721
722 003050 170201      STFPS   R1                          ;GET THE RESULT.
723 003052 012702 147657      MOV     #147657,R2
724 003056 020102      CMP     R1,R2                    ;RESULT CORRECT?
725 003060 001403      BEQ    1$
726 003062 010137 001126      MOV     R1,@#$BDDAT              ;STORE BAD DATA
727 003066 104001      ERROR  1
    
```

```

728
729 003070          1$:
730 003070 104414          LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
731 003072 012737 000205 001250 C7:  MOV #205,@#$TMP7
732 003100 012700 147757          MOV #147757,R0
733 003104 170100          LDFPS R0          ;SET FPS TO 147757.
734 003106 012737 003114 001236  MOV #C75,@#$TMP2
735
736 003114 170012          C75:  SETL          ;SET FL BIT.
737
738 003116 170201          STFPS R1          ;GET THE RESULT.
739 003120 012702 147757          MOV #147757,R2
740 003124 020102          CMP R1,R2          ;RESULT CORRECT?
741 003126 001403          BEQ 1$
742 003130 010137 001126          MOV R1,@#$BDDAT ;STORE BAD DATA
743 003134 104001          ERROR 1
744
745 003136          1$:
746 003136 104414          LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
747 003140 005000          C8:  CLR R0
748 003142 170100          LDFPS R0          ;CLEAR FPS.
749 003144 012737 003152 001236  MOV #C85,@#$TMP2
750
751 003152 170012          C85:  SETL          ;SET FL BIT.
752
753 003154 170201          STFPS R1
754 003156 012702 000100          MOV #100,R2
755 003162 020102          CMP R1,R2          ;RESULT CORRECT.
756 003164 001403          BEQ 1$
757 003166 010137 001126          MOV R1,@#$BDDAT ;STORE BAD DATA
758 003172 104001          ERROR 1
759
760 003174          1$:
761 003174          CDONE:
762 003174 104413          RSETUP          ;GO INITIALIZE THE FPS AND STACK; AND
763                                     ;SEE IF THE USER HAS EXPRESSED
764                                     ;THE DESIRE TO CHANGE THE SOFTWARE
765                                     ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
766                                     ;THE USER TYPED CONTROL G?).
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
    
```

```

:*****
:*TEST 4          ILLEGAL FPP OP CODES AND STST TEST
:*
:*THIS IS A TEST OF THE FPP OPERATION CODES:
:*              170003
:*              170004
:*              :
:*              170010
:*              170013
:*              170014
:*              :
:*              170077
:*THESE ARE ILLEGAL INSTRUCTIONS AND (WITH INTERRUPTS ENABLED)
:*SHOULD CAUSE A TRAP TO 244.
:*ALSO TESTED HERE IS THE INSTRUCTION:
    
```

```

784                                     : *          STST  R1
785                                     : *WHICH SHOULD PUT THE FEC CODE 2 IN R1, AFTER ANY OF THE ABOVE
786                                     : *OP CODES IS EXECUTED.
787                                     : *
788                                     : *
789                                     : *****
789 003176 000004 TST4: SCOPE
790 003200 104414 LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
791 003202 012705 170003 MOV #170003,R5 ;INITIAL OP CODE.
792 003206 012737 003412 000004 MOV #DERR2,@#ERRVECT
793 003214 012737 003316 000244 MOV #DERR1,@#FPVECT
794
795 003222 005000 D1: CLR R0 ;CLEAR FPS.
796 003224 170100 LDFPS R0
797 003226 005002 CLR R2
798 003230 010537 003246 MOV R5,@#D2 ;SET UP THE ILLEGAL INSTRUCTION.
799 003234 010537 001244 MOV R5,@#$TMP5
800 003240 012737 003246 001236 MOV #D2,@#$TMP2
801 003246 000000 D2: .WORD 0
802 003250 170000 D3: CFCC
803 003252 005202 INC R2
804 003254 005202 D4: INC R2
805
806 003256 170201 STFPS R1 ;REPORT FAILURE. DID NOT TRAP.
807 003260 010137 001240 MOV R1,@#$TMP3
808 003264 104001 1$: ERROR 1
809
810 003266 022705 170010 D5: CMP #170010,R5 ;COMPUTE NEXT OP CODE
811 003272 001003 BNE D6
812 003274 012705 170013 MOV #170013,R5
813 003300 000750 BR D1
814
815 003302 022705 170077 D6: CMP #170077,R5
816 003306 001001 BNE D7
817 003310 000452 BR DDONE
818 003312 005205 D7: INC R5
819 003314 000742 BR D1
820
821 003316 022716 003250 DERR1: CMP #D3,(SP) ;DID TRAP OCCUR ON TEST INSTRUCTION?
822 003322 001402 BEQ 1$
823 003324 000137 062524 JMP @#FPSPUR
824
825 003330 022626 1$: CMP (SP)+,(SP)+
826 003332 170201 STFPS R1 ;GET THE FPS AND SEE IF IT IS
827 003334 022701 100000 CMP #100000,R1 ;SET CORRECTLY.
828 003340 001406 BEQ 3$
829
830 003342 012737 100000 001240 MOV #100000,@#$TMP3
831 003350 010137 001242 MOV R1,@#$TMP4
832 003354 104001 2$: ERROR 1
833
834 003356 012704 000001 3$: MOV #1,R4
835 003362 170304 D8: STST R4 ;GET THE FEC CODE. NOTE THAT
836 ;IF THE DESTINATION MODE IS
837 ;IMPROPERLY DECODED AN ODD
838 ;ADDRESS TRAP TO 4 SHOULD OCCUR.
839 003364 022704 000002 CMP #2,R4 ;WAS FEC CORRECT?
    
```


896 003514 010137 001240
 897 003520 012737 140000 001242
 898 003526 104001
 899 003530 000422
 900
 901 003532
 902 003532 010537 001240
 903 003536 010437 001242
 904 003542 104001
 905 003544 000414
 906
 907 003546 021627 003470
 908 003552 001402
 909 003554 000137 062524
 910
 911 003560
 912 003560 011637 001236
 913 003564 022626
 914 003566 170201
 915 003570 010137 001240
 916 003574 104001
 917
 918 003576
 919 003576 104413
 920
 921
 922
 923
 924
 925
 926
 927
 928
 929
 930
 931
 932
 933
 934
 935
 936
 937
 938
 939 003600 000004
 940
 941 003602
 942 003602 104414
 943 003604 012737 003654 001236
 944 003612 005000
 945 003614 170100
 946 003616 170011
 947 003620 012701 005372
 948 003624 012702 005436
 949 003630 012703 000010
 950
 951 003634 012221

```

MOV R1,@#STMP3
MOV #140000,@#STMP4
1$: ERROR 1
BR EDONE

EERR1: ;REPORT FEC NOT 2.
MOV R5,@#STMP3
MOV R4,@#STMP4
1$: ERROR 1
BR EDONE

EERR2: CMP (SP),#E4 ;DID THE ILLEGAL INSTRUCTION TRAP?
BEQ 1$
JMP @#FPSPUR

1$: MOV (SP),@#STMP2
CMP (SP)+,(SP)+
STFPS R1
MOV R1,@#STMP3
2$: ERROR 1

EDONE: 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?).
    
```

```

*****
*TEST 6 LDD AND STD, WITH SRC AND DST MODE 1, TEST
*
*THIS IS A TEST OF BOTH THE INSTRUCTION:
* LDD (R0),ACO
*AND THE INSTRUCTION:
* STD ACO,(R0)
*MOST OF THE FAILURES ARE ISOLATED TO THE SRC OR DST FLOWS. NOTE
*THAT THE INTEGRITY OF ACO HAS NOT BEEN ASSURED. THIS MEANS THAT
*IN SOME CASES IT WILL BE IMPOSSIBLE TO ISOLATE CERTAIN DATA PATTERN
*FAILURES TO EITHER THE FLOWS OR THIS ACCUMULATOR.
*****
    
```

```

TST6: SCOPE
F1: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
MOV #F3,@#STMP2
CLR R0
LDFPS R0
SETD
MOV #FDAT10,R1 ;SET UP THE LOAD DATA.
MOV #FXDAT0,R2
MOV #10,R3
F2: MOV (R2)+,(R1)+
    
```

952	003636	077302			SOB	R3,F2	
953							
954	003640	012700	005402		MOV	#FDAT14,R0	:SETUP R0 FOR THE LDD (R0),ACO.
955	003644	012737	005072	000004	MOV	#FERR20,@#ERRVECT	:IF THE SRC FLOWS FAIL THEN
956							:AN ODD ADDRESS MAY OCCUR.
957	003652	005003			CLR	R3	
958							
959	003654	172410		F3:	LDD	(R0),ACO	
960	003656	005203		F4:	INC	R3	
961	003660	005203			INC	R3	
962							
963	003662	020027	005402		CMP	R0,#FDAT14	:WAS R0 AFFECTED?
964	003666	001402			BEQ	F5	
965	003670	000137	004236		JMP	@#FERR1	
966							
967	003674	020327	000002	F5:	CMP	R3,#2	:SEE IF THE PC WAS ADVERSELY
968	003700	001402			BEQ	1\$:AFFECTED DURING THE INSTRUCTION.
969	003702	000137	004334		JMP	@#FERR2	
970							
971	003706	012701	005372	1\$:	MOV	#FDAT10,R1	:MAKE SURE THE SOURCE DATA WAS
972	003712	012702	005436		MOV	#FXDAT0,R2	:NOT AFFECTED.
973	003716	012703	000010		MOV	#10,R3	
974	003722	022122		2\$:	CMP	(R1)+,(R2)+	
975	003724	001402			BEQ	3\$	
976	003726	000137	004200		JMP	@#FERRO	
977	003732	077305		3\$:	SOB	R3,2\$	
978							
979	003734	170201			STFPS	R1	:MAKE SURE THE FPS IS CORRECT.
980	003736	022701	000200		CMP	#200,R1	
981	003742	001402			BEQ	F6	
982	003744	000137	005052		JMP	@#FERR11	
983							
984	003750			F6:	LPERR		:SET UP THE LOOP ON ERROR ADDRESS.
985	003750	104414			MOV	#F10,@#\$TMP2	
986	003752	012737	004014	001236			
987							
988	003760	012703	177777		MOV	#-1,R3	
989	003764	012704	000010		MOV	#10,R4	
990	003770	012705	005414		MOV	#FDAT00,R5	:SET UP THE OUTPUT DATA BUFFER.
991	003774	010325		F7:	MOV	R3,(R5)+	
992	003776	077402			SOB	R4,F7	
993							
994	004000	012700	005424		MOV	#FDAT04,R0	:SET UP R0 FOR DST MODE 1 REG 0.
995	004004	012737	005232	000004	MOV	#FERR25,@#ERRVECT	:IF THE DST FLOWS FAIL AN ODD
996							:ADDRESS COULD OCCUR.
997	004012	005003			CLR	R3	
998							
999	004014	174010		F10:	STD	ACO,(R0)	:TEST INSTRUCTION.
1000	004016	005203		F11:	INC	R3	
1001	004020	005203			INC	R3	
1002							
1003	004022	020027	005424		CMP	R0,#FDAT04	:WAS R0 MODIFIED?
1004	004026	001402			BEQ	F12	
1005	004030	000137	004374		JMP	@#FERR3	
1006							
1007	004034	020327	000002	F12:	CMP	R3,#2	:WAS THE PC AFFECTED CORRECTLY?

```

1008 004040 001402          BEQ    F135
1009 004042 000167 000320  JMP    FERR4
1010
1011 004046 012701 005414  F135:  MOV    #FDAT00,R1
1012 004052 012702 005436  MOV    #FXDAT0,R2
1013
1014 004056 022122          CMP    (R1)+,(R2)+          ;SEE IF THE DATA WAS OUTPUT
1015 004060 001402          BEQ    F13                  ;TO THE TARGET AREA CORRECTLY.
1016 004062 000137 004472  JMP    @#FERR5
1017
1018 004066 022122  F13:  CMP    (R1)+,(R2)+
1019 004070 001402          BEQ    F14
1020 004072 000137 004472  JMP    @#FERR5
1021
1022 004076 022122  F14:  CMP    (R1)+,(R2)+
1023 004100 001402          BEQ    F15
1024 004102 000137 004472  JMP    @#FERR5
1025
1026 004106 022122  F15:  CMP    (R1)+,(R2)+
1027 004110 001402          BEQ    F16
1028 004112 000137 004472  JMP    @#FERR5
1029
1030 004116 022122  F16:  CMP    (R1)+,(R2)+
1031 004120 001402          BEQ    F17
1032 004122 000137 005016  JMP    @#FERR10
1033
1034 004126 022122  F17:  CMP    (R1)+,(R2)+
1035 004130 001402          BEQ    F20
1036 004132 000137 004526  JMP    @#FERR6
1037
1038 004136 022122  F20:  CMP    (R1)+,(R2)+
1039 004140 001402          BEQ    F21
1040 004142 000167 000514  JMP    FERR7
1041
1042 004146 022122  F21:  CMP    (R1)+,(R2)+
1043 004150 001402          BEQ    F22
1044 004152 000137 005016  JMP    @#FERR10
1045
1046 004156 005001  F22:  CLR    R1
1047 004160 170201          STFPS R1                    ;MAKE SURE FPS IS CORRECT.
1048 004162 022701 000200  CMP    #200,R1
1049 004166 001402          BEQ    F23
1050 004170 000137 005052  JMP    @#FERR11
1051 004174 000137 005456  F23:  JMP    @#FDONE
1052
1053 004200          FERRO:
1054 004200 012737 005436 001240  MOV    #FXDAT0,@#STMP3
1055 004206 012737 005450 001242  MOV    #FXDAT0+12,@#STMP4
1056 004214 012737 005372 001244  MOV    #FDAT10,@#STMP5
1057 004222 012737 005404 001246  MOV    #FDAT10+12,@#STMP6
1058 004230 104001          1$:  ERROR 1
1059 004232 000137 005456  JMP    @#FDONE
1060
1061 004236 012737 005402 001242  FERR1: MOV    #FDAT14,@#STMP4
1062 004244 010037 001240          MOV    R0,@#STMP3
1063 004250 012737 000762 001244  MOV    #762,@#STMP5
    
```

1064	004256	012737	000321	001250	MOV	#321,@#STMP7		
1065								
1066	004264	022700	005372		CMP	#FDATIO,R0		;FSRC MODE 4?
1067	004270	001004			BNE	1\$		
1068	004272	012737	000324	001246	MOV	#324,@#STMP6		
1069	004300	000412			BR	4\$		
1070								
1071	004302	022700	005412		1\$: CMP	#FDATI4+10,R0		;FSRC MODE 2?
1072	004306	001004			BNE	2\$		
1073	004310	012737	000322	001246	MOV	#322,@#STMP6		
1074	004316	000403			BR	4\$		
1075								
1076	004320				2\$:			
1077	004320				3\$:			
1078	004320	104001			ERROR	1		;RO WAS MODIFIED, LDD
1079	004322	000137	005456		JMP	@#FDONE		
1080	004326				4\$:			
1081	004326				5\$:			
1082	004326	104001			ERROR	1		;RO WAS MODIFIED, LDD
1083	004330	000137	005456		JMP	@#FDONE		
1084								
1085								
1086	004334	012701	003656		FERR2:	MOV #F4,R1		;THE PC WAS INCORRECTLY AFFECTED
1087								;DURING THE INSTRUCTION.
1088	004340	010137	001242		FER2:	MOV R1,@#STMP4		
1089	004344	162701	000004		SUB	#4,R1		
1090	004350	006303			ASL	R3		
1091	004352	060301			ADD	R3,R1		
1092	004354	010137	001240		MOV	R1,@#STMP3		
1093	004360	104001			1\$: ERROR	1		
1094	004362	000137	005456		JMP	@#FDONE		
1095								
1096	004366	012701	004016		FERR4:	MOV #F11,R1		
1097	004372	000762			BR	FER2		
1098								
1099	004374	012737	005424	001242	FERR3:	MOV #FDATO4,@#STMP4		;FAILURE IN THE FDST FLOWS.
1100	004402	010037	001240		MOV	R0,@#STMP3		
1101	004406	012737	000527	001244	MOV	#527,@#STMP5		
1102	004414	012737	000641	001250	MOV	#641,@#STMP7		
1103								
1104	004422	022700	005414		CMP	#FDAT00,R0		;DST MODE 4?
1105	004426	001004			BNE	1\$		
1106	004430	012737	000644	001246	MOV	#644,@#STMP6		
1107	004436	000412			BR	4\$		
1108								
1109	004440	022700	005434		1\$: CMP	#FDATO4+10,R0		;DST MODE 2?
1110	004444	001004			BNE	2\$		
1111	004446	012737	000642	001246	MOV	#642,@#STMP6		
1112	004454	000403			BR	4\$		
1113								
1114	004456				2\$:			
1115	004456	104001			3\$:			
1116	004460	000137	005456		ERROR	1		;RO WAS MODIFIED, STD
1117					JMP	@#FDONE		
1118	004464				4\$:			
1119	004464	104001			5\$:			
					ERROR	1		;RO WAS MODIFIED, STD

```

1120
1121 004466 000137 005456          JMP      @#FDONE
1122
1123 004472          FERR5:          ;FAILURE OF STD.
1124 004472 010037 001240          MOV      R0,@#STMP3
1125 004476 012737 005414 001242          MOV      #FDAT00,@#STMP4
1126 004504 012737 005432 001244          MOV      #FDAT07,@#STMP5
1127 004512 012737 005436 001246          MOV      #FXDAT0,@#STMP6
1128 004520 104003          1$:          ERROR      3          ;OUTPUT BAD, STD
1129 004522 000137 005456          JMP      @#FDONE
1130
1131 004526 012701 005426          FERR6:          ;DID (BUT GR7) FAIL IN THE FDST
1132 004532 012702 177777          MOV      #FDAT05,R1          ;FLOWS?
1133 004536 012703 000003          MOV      #-1,R2
1134 004542 020221          1$:          MOV      #3,R3
1135 004544 001017          CMP      R2,(R1)+
1136 004546 077303          BNE      5$
1137          SOB      R3,1$
1138          ;REPORT FAILURE OF (BUT GR7) IN
1139 004550 010037 001240          MOV      R0,@#STMP3          ;THE FDST FLOWS.
1140 004554 012737 000412 001244          MOV      #412,@#STMP5
1141 004562 012737 000147 001246          MOV      #147,@#STMP6
1142 004570 012737 000145 001250          MOV      #145,@#STMP7
1143 004576 104001          2$:          ERROR      1
1144 004600 000137 005456          JMP      @#FDONE
1145
1146 004604 012701 005426          5$:          MOV      #FDAT05,R1          ;DID (BUT GR7) FAIL IN THE SRC FLOWS?
1147 004610 012703 000003          MOV      #3,R3
1148 004614 005721          6$:          TST      (R1)+
1149 004616 001402          BEQ      7$
1150 004620 000137 005016          JMP      @#FERR10
1151 004624 077305          7$:          SOB      R3,6$
1152
1153          ;REPORT FAILURE OF (BUT GR7) IN
1154 004626 010037 001240          MOV      R0,@#STMP3          ;THE FSRC FLOWS.
1155 004632 012737 000207 001244          MOV      #207,@#STMP5
1156 004640 012737 000176 001246          MOV      #176,@#STMP6
1157 004646 012737 000174 001250          MOV      #174,@#STMP7
1158
1159 004654 104001          10$:         ERROR      1
1160 004656 000137 005456          JMP      @#FDONE
1161
1162 004662 012701 005430          FERR7:          ;DID (BUT FD) FAIL IN THE FDST FLOWS?
1163 004666 012702 177777          MOV      #FDAT06,R1
1164 004672 012703 000002          MOV      #-1,R2
1165 004676 020221          1$:          MOV      #2,R3
1166 004700 001017          CMP      R2,(R1)+
1167 004702 077303          BNE      5$
1168          SOB      R3,1$
1169          ;REPORT FAILURE OF (BUT FD) IN THE
1170 004704 010037 001240          MOV      R0,@#STMP3          ;FDST FLOWS.
1171 004710 012737 000707 001244          MOV      #707,@#STMP5
1172 004716 012737 000244 001246          MOV      #244,@#STMP6
1173 004724 012737 000245 001250          MOV      #245,@#STMP7
1174 004732 104001          2$:          ERROR      1
1175 004734 000137 005456          JMP      @#FDONE
    
```

```

1176
1177 004740 012701 005430 5$: MOV #FDAT06,R1 ;DID (BUT FD) FAIL IN THE FSRC FLOWS?
1178 004744 012703 000002 MOV #2,R3
1179 004750 005721 6$: TST (R1)+
1180 004752 001402 BEQ 7$
1181 004754 000137 005016 JMP @#FERR10
1182 004760 077305 7$: SOB R3,6$
1183
1184 ;REPORT FAILURE OF (BUT FD) IN THE
1185 004762 010037 001240 MOV R0,@#STMP3 ;FSRC FLOWS.
1186 004766 012737 000441 001244 MOV #441,@#STMP5
1187 004774 012737 000076 001246 MOV #76,@#STMP6
1188 005002 012737 000077 001290 MOV #77,@#STMP7
1189 005010 104003 10$: ERROR 3 ;BAD DATA
1190 005012 000137 005456 JMP @#FDONE
1191
1192 005016 FERR10: ;REPORT DATA ERROR.
1193 005016 010037 001240 MOV R0,@#STMP3
1194 005022 012737 005424 001242 MOV #FDAT04,@#STMP4
1195 005030 012737 005432 001244 MOV #FDAT07,@#STMP5
1196 005036 012737 005446 001246 MOV #FXDAT4,@#STMP6
1197 005044 104001 1$: ERROR 1
1198 005046 000137 005456 JMP @#FDONE
1199
1200 005052 FERR11: ;REPORT BAD FPS.
1201 005052 010137 001240 MOV R1,@#STMP3
1202 005056 012737 000200 001242 MOV #200,@#STMP4
1203 005064 104001 1$: ERROR 1
1204 005066 000137 005456 JMP @#FDONE
1205
1206 005072 012737 062735 001264 FERR20: MOV #NULL,@#STMP15 ;THE EXECUTION OF THE LDD
1207 005100 005037 001252 CLR @#STMP10 ;CAUSED A TRAP TO 4, BECAUSE
1208 005104 011637 001236 MOV (SP),@#STMP2 ;A FSRC FLOW FAILURE RESULTED
1209 005110 012737 005402 001240 MOV #FDAT14,@#STMP3 ;IN AN ODD ADDRESS.
1210 005116 012737 000321 001250 MOV #321,@#STMP7
1211 005124 012737 000762 001244 MOV #762,@#STMP5
1212
1213 005132 021627 003660 CMP (SP),#F4+2 ;SEE IF FSRC MODE 6 OR 7 WAS
1214 005136 001424 BEQ FERR21 ;EXECUTED.
1215
1216 005140 020027 005400 CMP R0,#FDAT13 ;FSRC MODE 5?
1217 005144 001006 BNE 2$
1218
1219 ;REPORT FSRC FLOW FAILURE TO
1220 005146 012737 000325 001246 MOV #325,@#STMP6 ;MODE 5.
1221 005154 022626 CMP (SP)+,(SP)+
1222 005156 104001 1$: ERROR 1
1223 005160 000536 BR #FDONE
1224
1225 005162 020027 005404 2$: CMP R0,#FDAT15 ;FSRC MODE 3?
1226 005166 001402 BEQ 3$
1227 005170 000137 062556 JMP @#CPSPUR
1228
1229 005174 3$: ;REPORT FSRC FLOW FAILURE TO
1230 005174 012737 000323 001246 MOV #323,@#STMP6 ;MODE 3.
1231 005202 022626 CMP (SP)+,(SP)+
    
```

```

1232 005204 104001      4$:  ERROR 1
1233 005206 000523      BR      F DONE
1234
1235 005210 022626      FERR21: CMP  (SP)+,(SP)+      ;REPORT FSRC FLOW FAILURE TO
1236                                     ;MODE 6 OR MODE 7.
1237 005212 012737 000326 001246      MOV  #326,@#$TMP6
1238 005220 012737 000327 001252      MOV  #327,@#$TMP10
1239 005226 104001      1$:  ERROR 1
1240 005230 000512      BR      F DONE
1241
1242 005232 012737 062735 001264      FERR25: MOV  #NULL,@#$TMP15      ;THE EXECUTION OF THE STD INSTRUCTION
1243 005240 005037 001252      CLR  @#$TMP10      ;TRAPPED TO 4, BECAUSE A FAILURE
1244 005244 012737 005424 001240      MOV  #FDAT04,@#$TMP3      ;IN THE FDST FLOWS RESULTED
1245 005252 011637 001236      MOV  (SP),@#$TMP2      ;IN AN ODD ADDRESS.
1246 005256 012737 000527 001244      MOV  #527,@#$TMP5
1247 005264 012737 000641 001250      MOV  #641,@#$TMP7
1248
1249 005272 021627 004016      CMP  (SP),#F10+2      ;FLOW FAILURE TO FDST MODE 6 OR 7?
1250 005276 001424      BEQ  FERR26
1251
1252 005300 020027 005422      CMP  R0,#FDAT03      ;DID FDST FLOW FAIL TO MODE 5?
1253 005304 001006      BNE  2$
1254
1255                                     ;REPORT FLOW FAILURE TO FDST
1256 005306 012737 000645 001246      MOV  #645,@#$TMP6      ;MODE 5.
1257 005314 022626      CMP  (SP)+,(SP)+
1258 005316 104001      1$:  ERROR 1
1259 005320 000456      BR      F DONE
1260
1261 005322 020027 005426      2$:  CMP  R0,#FDAT05      ;DID FDST FLOW FAIL TO MODE 3?
1262 005326 001402      BEQ  3$
1263 005330 000137 062556      JMP  @#CPSPUR
1264
1265                                     ;REPORT FDST FLOW FAILED TO MODE 3.
1266 005334 012737 000643 001246      3$:  MOV  #643,@#$TMP6
1267 005342 022626      CMP  (SP)+,(SP)+
1268 005344 104001      4$:  ERROR 1
1269 005346 000443      BR      F DONE
1270
1271                                     ;REPORT FDST FLOW FAILURE TO MODE
1272 005350 012737 000646 001246      FERR26: MOV  #646,@#$TMP6
1273 005356 012737 000647 001252      MOV  #647,@#$TMP10
1274 005364 022626      CMP  (SP)+,(SP)+
1275 005366 104001      1$:  ERROR 1
1276 005370 000432      BR      F DONE
1277
1278 005372 177777      FDAT10: -1
1279 005374 177777      FDAT11: -1
1280 005376 177777      FDAT12: -1
1281 005400 177777      FDAT13: -1
1282 005402 177777      FDAT14: -1
1283 005404 177777      FDAT15: -1
1284 005406 177777      FDAT16: -1
1285 005410 177777      FDAT17: -1
1286 005412 177777      -1
1287 005414 177777      FDAT00: -1
    
```


1288	005416	177777	FDAT01:	-1
1289	005420	177777	FDAT02:	-1
1290	005422	177777	FDAT03:	-1
1291	005424	177777	FDAT04:	-1

1292	005426	177777	
1293	005430	177777	
1294	005432	177777	
1295	005434	177777	
1296	005436	177777	
1297	005440	177777	
1298	005442	177777	
1299	005444	177777	
1300	005446	052525	
1301	005450	031463	
1302	005452	007417	
1303	005454	000477	

FDAT05: -1
 FDAT06: -1
 FDAT07: -1
 -1
 FXDAT0: -1
 FXDAT1: -1
 FXDAT2: -1
 FXDAT3: -1
 FXDAT4: 052525
 FXDAT5: 031463
 FXDAT6: 007417
 FXDAT7: 000477

1304			
1305			
1306	005456		
1307	005456	104413	

FDONE: 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?).

1308
 1309
 1310
 1311
 1312
 1313
 1314
 1315
 1316
 1317
 1318
 1319

```

:*****
:*TEST 7          FSRC MODE 0 TEST
:*
:*THIS IS A TEST OF FSRC MODE ZERO USING THE LDD AND LDF INSTRUCTIONS.
:*
:*****
    
```

1320	005460	000004	
1321	005462	104414	

TST7: SCOPE LPERR

;SET UP THE LOOP ON ERROR ADDRESS.

1322			
1323	005464		
1324	005464	170011	
1325	005466	012700	006274
1326	005472	012701	006244
1327	005476	012702	000004
1328	005502	012120	
1329	005504	077202	

I1: SETD ;SET FD.
 MOV #IDATIO,R0
 MOV #IPATIO,R1
 MOV #4,R2
 I2: MOV (R1)+,(R0)+ ;SET UP THE INPUT DATA BUFFER.
 SOB R2,I2

1330			
1331	005506	012700	006274
1332	005512	172510	
1333			
1334	005514	012700	006254
1335	005520	172410	

MOV #IDATIO,R0 ;LOAD AC1
 LDD (R0),AC1
 MOV #IPAT20,R0 ;LOAD AC0
 LDD (R0),AC0

1336			
1337	005522	012701	000001
1338	005526	012737	006044 000004
1339	005534	012737	005542 001236

MOV #1,R1 ;IN CASE THE FSRC FLOWS FAIL
 MOV #IERRO,@#ERRVECT ;AN ODD ADDRESS TRAP TO 4 MAY OCCUR.
 MOV #I3,@#STMP2
 I3: LDD AC1,AC0 ;TEST INSTRUCTION.

1340	005542	172401	
1341	005544	000240	
1342	005546	000240	
1343			

I4: NOP
 I5: NOP

1344	005550	012700	006264
1345	005554	174010	
1346			
1347	005556	012700	006264

MOV #IDATIO,R0 ;GET ACO, THE RESULTS.
 STD ACO,(R0)
 MOV #IDATIO,R0 ;SEE IF DATA IS CORRECT.

1348	005562	012701	006274		MOV	#IDATIO,R1	
1349	005566	012702	000004		MOV	#4,R2	
1350	005572	022021		16:	CMP	(R0)+,(R1)+	
1351	005574	001424			BEQ	I105	
1352							
1353	005576	012700	006270		MOV	#IDATO2,R0	;SEE IF (BUT FD) FAILED.
1354	005602	012702	000002		MOV	#2,R2	
1355	005606	005720		17:	TST	(R0)+	
1356	005610	001413			BEQ	I10	
1357							
1358	005612	012700	006270		MOV	#IDATO2,R0	
1359	005616	012702	000002		MOV	#2,R2	
1360	005622	022720	177777	1\$:	CMP	#-1,(R0)+	
1361	005626	001402			BEQ	2\$	
1362	005630	000137	006126		JMP	@#IERR1	
1363	005634	077206		2\$:	SOB	R2,1\$	
1364	005636	000401			BR	I106	
1365	005640	077216		I10:	SOB	R2,I7	
1366	005642	000137	006146	I106:	JMP	@#IERR2	
1367							
1368	005646	077227		I105:	SOB	R2,I6	
1369							
1370							;NOW TEST THE LOAD INSTRUCTION WITH FSRC MODE ZERO AND FD CLEAR.
1371							
1372	005650			I11:			
1373	005650	104414			LPERR		;SET UP THE LOOP ON ERROR ADDRESS.
1374							
1375	005652	012700	006244	I12:	MOV	#IPAT10,R0	
1376	005656	012701	006274		MOV	#IDATIO,R1	
1377	005662	012702	000004		MOV	#4,R2	
1378	005666	012021		I13:	MOV	(R0)+,(R1)+	
1379	005670	077202			SOB	R2,I13	
1380							
1381	005672	012700	006274		MOV	#IDATIO,R0	;SET UP AC1
1382	005676	172510			LDD	(R0),AC1	
1383							
1384	005700	012700	006254		MOV	#IPAT20,R0	;SET UP AC0
1385	005704	172410			LDD	(R0),AC0	
1386							
1387	005706	012701	000001		MOV	#1,R1	
1388	005712	012737	005722 001236		MOV	#I14,@#STMP2	
1389	005720	170001			SETF		;CLEAR FD.
1390							
1391	005722	172401		I14:	LDF	AC1,AC0	;TEST INSTRUCTION.
1392	005724	000240		I15:	NOP		
1393	005726	000240		I16:	NOP		
1394							
1395	005730	170200			STFPS	R0	;SEE IF FPS IS STILL CLEAR.
1396	005732	022700	000004		CMP	#4,R0	
1397	005736	001402			BEQ	I17	
1398	005740	000137	006220		JMP	@#IERR3	
1399							
1400	005744			I17:			;RESET TO DOUBLE MODE.
1401	005744	170011			SETD		
1402							
1403	005746	012700	006264		MOV	#IDAT00,R0	

```

1404 005752 174010          STD      ACO,(R0)          ;GET ACO
1405
1406 005754 012737 177777 006300  MOV     #-1,@#IDAT12
1407 005762 012737 177777 006302  MOV     #-1,@#IDAT13
1408 005770 012700 006264          MOV     #IDAT00,R0
1409 005774 012701 006274          MOV     #IDAT10,R1
1410 006000 012702 000004          MOV     #4,R2
1411 006004 022021          120:   CMP     (R0)+,(R1)+      ;SEE IF ACO WAS CORRECT.
1412 006006 001414          BEQ     I23
1413
1414 006010 023737 006270 006250  CMP     @#IDAT02,@#IPAT12      ;DID (BUT FD) FAIL?
1415 006016 001402          BEQ     I22
1416 006020 000137 006126          121:   JMP     @#IERR1
1417 006024 023737 006272 006252  122:   CMP     @#IDAT03,@#IPAT13
1418 006032 001372          BNE     I21
1419 006034 000137 006174          JMP     @#IERR4
1420
1421 006040 077217          123:   SOB     R2,I20
1422
1423 006042 000520          BR      IDONE              ;NO ERRORS.
1424
1425          ;IF AN ODD ADDRESS TRAP OCCURS COME HERE TO ANALYZE THE FSRC FAILURE.
1426 006044 022716 005544          IERR0: CMP     #14,(SP)          ;MAKE SURE THE TRAP OCCURRED
1427 006050 001413          BEQ     1$                ;ON THE INSTRUCTION BEING TESTED.
1428 006052 022716 005546          CMP     #15,(SP)
1429 006056 001410          BEQ     1$
1430 006060 022716 005724          CMP     #115,(SP)
1431 006064 001405          BEQ     1$
1432 006066 022716 005726          CMP     #116,(SP)
1433 006072 001402          BEQ     1$
1434 006074 000137 062556          JMP     @#CPSPUR
1435
1436 006100 011637 001236          1$:   MOV     (SP),@#$TMP2          ;REPORT FAILURE.
1437 006104 012737 000627 001240  MOV     #627,@#$TMP3
1438 006112 012737 000320 001242  MOV     #320,@#$TMP4
1439 006120 022626          CMP     (SP)+,(SP)+
1440 006122 104001          2$:   ERROR  1
1441 006124 000467          BR      IDONE
1442
1443          ;REPORT DATA ERROR.
1444 006126          IERR1:
1445 006126 012737 006274 001242  MOV     #IDAT10,@#$TMP4
1446 006134 012737 006264 001244  MOV     #IDAT00,@#$TMP5
1447 006142 104004          1$:   ERROR  4
1448 006144 000457          BR      IDONE
1449
1450          ;REPORT FAILURE OF (BUT FD)
1451 006146 012737 000153 001244  IERR2: MOV     #153,@#$TMP5
1452 006154 012737 000434 001246  MOV     #434,@#$TMP6
1453 006162 012737 000435 001250  MOV     #435,@#$TMP7
1454 006170          IERR25:
1455 006170 104001          1$:   ERROR  1
1456 006172 000444          BR      IDONE
1457 006174 012737 000153 001244  IERR4: MOV     #153,@#$TMP5
1458 006202 012737 000435 001246  MOV     #435,@#$TMP6
1459 006210 012737 000434 001250  MOV     #434,@#$TMP7
    
```

```

1460 006216 000764          BR      IERR25
1461
1462          ;REPORT INCORRECT FPS AFTER LOAD INSTRUCTION.
1463          IERR3:
1464 006220          MOV      #I14,@#$TMP2
1465 006226 012737 005722 001236          MOV      R0,@#$TMP3
1466 006232 012737 000004 001242          MOV      #4,@#$TMP4
1467 006240 104001          1$:      ERROR  1
1468 006242 000420          BR      IDONE
1469
1470
1471 006244 000000          IPAT10: 0
1472 006246 170360          IPAT11: 170360
1473 006250 016161          IPAT12: 016161
1474 006252 052525          IPAT13: 052525
1475
1476 006254 177777          IPAT20: -1
1477 006256 177777          IPAT21: -1
1478 006260 177777          IPAT22: -1
1479 006262 177777          IPAT23: -1
1480
1481 006264 000000          IDAT00: 0
1482 006266 000000          IDAT01: 0
1483 006270 000000          IDAT02: 0
1484 006272 000000          IDAT03: 0
1485
1486 006274 000000          IDAT10: 0
1487 006276 000000          IDAT11: 0
1488 006300 000000          IDAT12: 0
1489 006302 000000          IDAT13: 0
1490
1491 006304          IDONE:
1492 006304 104413          RSETUP          ;GO INITIALIZE THE FPS AND STACK; AND
1493                                     ;SEE IF THE USER HAS EXPRESSED
1494                                     ;THE DESIRE TO CHANGE THE SOFTWARE
1495                                     ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
1496                                     ;THE USER TYPED CONTROL G?).
1497
1498
1499          ;*****
1500          ;*TEST 10          FDST MODE 0 TEST
1501          ;*
1502          ;*THIS IS A TEST OF THE STORE INSTRUCTIONS, STD AND STF, WITH FDST MODE 0.
1503          ;*
1504          ;*****
1505 006306 000004          TST10: SCOPE
1506 006310          T1:
1507 006310 104414          LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
1508 006312 170011          SETD          ;SET FD
1509 006314 012700 007036          MOV      #TPAT10,R0
1510 006320 012701 007066          MOV      #TDAT10,R1
1511 006324 012702 000004          MOV      #4,R2
1512 006330 012021          T2:      MOV      (R0)+,(R1)+          ;SET UP THE INPUT DATA BUFFER.
1513 006332 077202          SOB      R2,T2
1514
1515 006334 012700 007066          MOV      #TDAT10,R0          ;LOAD ACO
    
```

```

1516 006340 172410          LDD      (R0),AC0
1517
1518 006342 012700 007046    MOV      #TPAT20,R0          ;LOAD AC1
1519 006346 172510          LDD      (R0),AC1
1520
1521 006350 012701 000001    MOV      #1,R1              ;IF THE (BUT FDST) FORK FAILS
1522 006354 012737 006644 000004  MOV      #TERR0,@#ERRVECT   ;AN ODD ADDRESS TRAP COULD RESULT.
1523 006362 012737 006370 001236  MOV      #T3,@#$TMP2
1524 006370 174001          STD      AC0,AC1
1525 006372 000240          T3:     NOP
1526 006374 000240          T4:     NOP
1527
1528 006376 012700 007056    MOV      #TDAT00,R0
1529 006402 174110          STD      AC1,(R0)          ;GET THE DATA.
1530
1531 006404 012703 007056    MOV      #TDAT00,R3        ;SEE IF THE DATA IS CORRECT.
1532 006410 012704 007066    MOV      #TDAT10,R4
1533 006414 012705 000004    MOV      #4,R5
1534 006420 022324          T6:     CMP      (R3)+,(R4)+
1535 006422 001413          BEQ      T105
1536
1537 006424 012703 007062    MOV      #TDAT02,R3        ;DID (BUT FD) FAIL?
1538 006430 012705 000002    MOV      #2,R5
1539 006434 005723          T7:     TST      (R3)+
1540 006436 001402          BEQ      T10
1541 006440 000137 006726    JMP      @#TERR1
1542 006444 077505          T10:    SOB      R5,T7
1543 006446 000137 006746    JMP      @#TERR2
1544
1545 006452 077516          T105:   SOB      R5,T6
1546
1547          ;NOW TEST THE STF AC0,AC1 INSTRUCTION.
1548
1549 006454          T11:
1550 006454 104414          LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
1551
1552 006456 012700 007036    T12:    MOV      #TPAT10,R0        ;SET UP THE INPUT DATA BUFFER.
1553 006462 012701 007066    MOV      #TDAT10,R1
1554 006466 012702 000004    MOV      #4,R2
1555 006472 012021          T13:    MOV      (R0)+,(R1)+
1556 006474 077202          SOB      R2,T13
1557
1558 006476 012700 007066    MOV      #TDAT10,R0        ;SET UP AC0
1559 006502 172410          LDD      (R0),AC0
1560
1561 006504 012700 007046    MOV      #TPAT20,R0        ;SET UP AC1
1562 006510 172510          LDD      (R0),AC1
1563
1564 006512 012701 000001    MOV      #1,R1
1565 006516 012737 006526 001236  MOV      #T14,@#$TMP2
1566 006524 170001          SETF          ;CLEAR FD
1567 006526 174001          T14:    STF      AC0,AC1
1568 006530 000240          T15:    NOP
1569 006532 000240          T16:    NOP
1570
1571 006534 005000          CLR      R0
    
```

```

1572 006536 170200          STFPS  R0          ;SEE IF FPS IS CLEAR.
1573 006540 022700 000010  CMP    #10,R0
1574 006544 001401          BEQ    T17
1575 006546 000521          BR     TERR3
1576
1577 006550          T17:  SETD          ;SET FD.
1578 006550 170011
1579
1580 006552 012700 007056  MOV    #TDAT00,R0
1581 006556 174110          STD    AC1,(R0)    ;PICK UP AC1.
1582
1583
1584 006560 012737 177777 007072  MOV    #-1,@#TDAT12
1585 006566 012737 177777 007074  MOV    #-1,@#TDAT13
1586 006574 012703 007056          MOV    #TDAT00,R3
1587 006600 012704 007066          MOV    #TDAT10,R4
1588 006604 012705 000004          MOV    #4,R5
1589 006610 022324          T20:  CMP    (R3)+,(R4)+ ;WAS THE DATA TRANSFERRED CORRECTLY?
1590 006612 001412          BEQ    T23
1591
1592 006614 023737 007062 007042  CMP    @#TDAT02,@#TPAT12 ;DID (BUT FD) FAIL.
1593 006622 001401          BEQ    T22
1594 006624 000440          BR     TERR1
1595 006626 023737 007064 007044  T21:  CMP    @#TDAT03,@#TPAT13
1596 006634 001373          T22:  BNE    T21
1597 006636 000456          BR     TERR4
1598
1599 006640 077515          T23:  SOB    R5,T20
1600 006642 000515          BR     TDONE
1601
1602
1603          ;TRAP HERE THROUGH VECTOR 4 IF AN ODD ADDRESS OCCURS.
1604 006644 022716 006372  TERR0: CMP    #T4,(SP)    ;MAKE SURE THE TRAP WAS ON
1605 006650 001413          BEQ    1$          ;AN INSTRUCTION BEING TESTED.
1606 006652 022716 006374          CMP    #T5,(SP)
1607 006656 001410          BEQ    1$
1608 006660 022716 006530          CMP    #T15,(SP)
1609 006664 001405          BEQ    1$
1610 006666 022716 006532          CMP    #T16,(SP)
1611 006672 001402          BEQ    1$
1612 006674 000137 062556          JMP    @#CPSPUR
1613
1614 006700 011637 001236          1$:  MOV    (SP),@#STMP2
1615 006704 022626          CMP    (SP)+,(SP)+
1616 006706 012737 000527 001240  MOV    #527,@#STMP3
1617 006714 012737 000640 001242  MOV    #640,@#STMP4
1618 006722 104001          2$:  ERROR 1
1619 006724 000464          BR     TDONE
1620
1621          ;REPORT DATA FAILURE.
1622 006726          TERR1:
1623 006726 012737 007066 001242  MOV    #TDAT10,@#STMP4
1624 006734 012737 007056 001244  MOV    #TDAT00,@#STMP5
1625 006742 104001          1$:  ERROR 1
1626 006744 000454          BR     TDONE
1627
    
```

```

1628
1629 006746 012737 000160 001246 :REPORT FAILURE OF (BUT FD).
1630 006754 012737 000161 001250 TERR2: MOV #160,@#STMP6
1631 006762 012737 000640 001244 TERR25: MOV #161,@#STMP7
1632 006770 104012 1$: ERROR #640,@#STMP5
1633 006772 000441 BR TDONE
1634 006774 012737 000161 001246 TERR4: MOV #161,@#STMP6
1635 007002 012737 000160 001250 MOV #160,@#STMP7
1636 007010 000764 BR TERR25
1637
1638 :REPORT INCORRECT FPS AFTER STORE INSTRUCTION.
1639 007012 TERR3:
1640 007012 012737 006530 001236 MOV #T15,@#STMP2
1641 007020 010037 001240 MOV R0,@#STMP3
1642 007024 012737 000010 001242 MOV #10,@#STMP4
1643 007032 104001 1$: ERROR 1
1644 007034 000420 BR TDONE
1645
1646 007036 000000 TPAT10: 0
1647 007040 170360 TPAT11: 170360
1648 007042 016161 TPAT12: 016161
1649 007044 052525 TPAT13: 052525
1650
1651 007046 177777 TPAT20: -1
1652 007050 177777 TPAT21: -1
1653 007052 177777 TPAT22: -1
1654 007054 177777 TPAT23: -1
1655
1656 007056 000000 TDATA0: 0
1657 007060 000000 TDATA1: 0
1658 007062 000000 TDATA2: 0
1659 007064 000000 TDATA3: 0
1660
1661 007066 000000 TDATA10: 0
1662 007070 000000 TDATA11: 0
1663 007072 000000 TDATA12: 0
1664 007074 000000 TDATA13: 0
1665
1666 007076 TDONE:
1667 007076 104413 RSETUP
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
    ;GO INITIALIZE THE FPS AND STACK; AND
    ;SEE IF THE USER HAS EXPRESSED
    ;THE DESIRE TO CHANGE THE SOFTWARE
    ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
    ;THE USER TYPED CONTROL G?).

:*****
:*TEST 11 ACCUMULATORS DATA PATTERNS TEST
:*
:*THIS IS A TEST OF THE FLOATING POINT PROCESSOR ACCUMULATORS.
:*EACH ACCUMULATOR IS TESTED IN TWO WAYS:
:* 1 TEST PATTERN GENERATED BY FLOATING A ONE ACROSS
:* A FIELD OF ZEROES.
:* 2 TEST PATTERN GENERATED BY FLOATING A ZERO ACROSS
:* A FIELD OF ONES.
    
```



```

1684 ;*EACH OF ACCUMULATORS ACO THROUGH AC5 IS TESTED.
1685 ;*****
1686 007100 000004 TST11: SCOPE
1687 007102 170011 SETD ;SET FD.
1688 ;TEST ACCUMULATOR 0 WITH FLOATING ONE
1689 007104 012737 007134 001236 MOV #G1,@#STMP2
1690 007112 012700 011002 MOV #GPAT00,R0
1691 007116 012701 011042 MOV #GDAT00,R1
1692 007122 104414 LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
1693 007124 004737 010650 JSR PC,@#GSETUP ;LOAD TEST PATTERN.
1694 007130 012703 000102 MOV #102,R3
1695 007134 G1:
1696 007134 172410 LDD (R0),ACO
1697 007136 174000 STD ACO,ACO
1698 007140 172400 LDD ACO,ACO ;STORE THE TEST PATTERN.
1699 007142 174011 STD ACO,(R1)
1700 007144 004737 010746 JSR PC,@#GCMP ;COMPARE THE DATA READ WITH
1701 ;THAT WHICH WAS WRITTEN.
1702 007150 005737 010776 TST @#GFLAG1
1703 007154 001004 BNE G2
1704 007156 005137 010776 COM @#GFLAG1
1705 007162 000261 SEC
1706 007164 000401 BR G3
1707 007166 000241 G2:
1708 007170 006160 000006 G3: CLC ;GENERATE THE NEXT TEST PATTERN.
1709 007174 006160 000004 ROL 6(R0)
1710 007200 006160 000002 ROL 4(R0)
1711 007204 006110 ROL 2(R0)
1712 007206 004737 010726 ROL (R0)
1713 JSR PC,@#GRESET ;RESET DEFAULT PATTERN IN OUTPUT
1714 007212 077330 SOB R3,G1 ;BUFFER.
1715
1716 ;TEST ACCUMULATOR 0 WITH FLOATING ZERO
1717 007214 012737 007244 001236 MOV #G4,@#STMP2
1718 007222 012700 011012 MOV #GPAT10,R0
1719 007226 012701 011042 MOV #GDAT00,R1
1720 007232 104414 LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
1721 007234 004737 010650 JSR PC,@#GSETUP ;LOAD TEST PATTERN.
1722 007240 012703 000102 MOV #102,R3
1723 007244 G4:
1724 007244 172410 LDD (R0),ACO
1725 007246 174000 STD ACO,ACO
1726 007250 172400 LDD ACO,ACO ;STORE THE TEST PATTERN.
1727 007252 174011 STD ACO,(R1)
1728 007254 004737 010746 JSR PC,@#GCMP ;COMPARE THE DATA READ WITH
1729 ;THAT WHICH WAS WRITTEN.
1730 007260 005737 010776 TST @#GFLAG1
1731 007264 001004 BNE G5
1732 007266 005137 010776 COM @#GFLAG1
1733 007272 000241 CLC
1734 007274 000401 BR G6
1735 007276 000261 G5: SEC
1736 007300 006160 000006 G6: ROL 6(R0) ;GENERATE THE NEXT TEST PATTERN.
1737 007304 006160 000004 ROL 4(R0)
1738 007310 006160 000002 ROL 2(R0)
1739 007314 006110 ROL (R0)
    
```

```

1740 007316 004737 010726      JSR    PC,@#GRESET      ;RESET DEFAULT PATTERN IN OUTPUT
1741                               ;BUFFER.
1742 007322 077330      SOB    R3,G4
1743
1744                               ;TEST ACCUMULATOR 1 WITH FLOATING ONE
1745 007324 012737 007354 001236  MOV    #G7,@#STMP2
1746 007332 012700 011002      MOV    #GPAT00,R0
1747 007336 012701 011042      MOV    #GDAT00,R1
1748 007342 104414      LPERR                               ;SET UP THE LOOP ON ERROR ADDRESS.
1749 007344 004737 010650      JSR    PC,@#GSETUP      ;LOAD TEST PATTERN.
1750 007350 012703 000102      MOV    #102,R3
1751 007354
1752 007354 172410      G7:   LDD    (R0),ACO
1753 007356 174001      STD    ACO,AC1
1754 007360 172401      LDD    AC1,ACO          ;STORE THE TEST PATTERN.
1755 007362 174011      STD    ACO,(R1)
1756 007364 004737 010746      JSR    PC,@#GCMP        ;COMPARE THE DATA READ WITH
1757                               ;THAT WHICH WAS WRITTEN.
1758 007370 005737 010776      TST    @#GFLAG1
1759 007374 001004      BNE    G10
1760 007376 005137 010776      COM    @#GFLAG1
1761 007402 000261      SEC
1762 007404 000401      BR     G11
1763 007406 000241      G10:  CLC
1764 007410 006160 000006      G11:  ROL    6(R0)        ;GENERATE THE NEXT TEST PATTERN.
1765 007414 006160 000004      ROL    4(R0)
1766 007420 006160 000002      ROL    2(R0)
1767 007424 006110      ROL    (R0)
1768 007426 004737 010726      JSR    PC,@#GRESET      ;RESET DEFAULT PATTERN IN OUTPUT
1769                               ;BUFFER.
1770 007432 077330      SOB    R3,G7
1771
1772
1773                               ;TEST ACCUMULATOR 1 WITH FLOATING ZERO
1774 007434 012737 007464 001236  MOV    #G12,@#STMP2
1775 007442 012700 011012      MOV    #GPAT10,R0
1776 007446 012701 011042      MOV    #GDAT00,R1
1777 007452 104414      LPERR                               ;SET UP THE LOOP ON ERROR ADDRESS.
1778 007454 004737 010650      JSR    PC,@#GSETUP      ;LOAD TEST PATTERN.
1779 007460 012703 000102      MOV    #102,R3
1780 007464
1781 007464 172410      G12:  LDD    (R0),ACO
1782 007466 174001      STD    ACO,AC1
1783 007470 172401      LDD    AC1,ACO          ;STORE THE TEST PATTERN.
1784 007472 174011      STD    ACO,(R1)
1785 007474 004737 010746      JSR    PC,@#GCMP        ;COMPARE THE DATA READ WITH
1786                               ;THAT WHICH WAS WRITTEN.
1787 007500 005737 010776      TST    @#GFLAG1
1788 007504 001004      BNE    G13
1789 007506 005137 010776      COM    @#GFLAG1
1790 007512 000241      CLC
1791 007514 000401      BR     G14
1792 007516 000261      G13:  SEC
1793 007520 006160 000006      G14:  ROL    6(R0)        ;GENERATE THE NEXT TEST PATTERN.
1794 007524 006160 000004      ROL    4(R0)
1795 007530 006160 000002      ROL    2(R0)
    
```

```

1796 007534 006110          ROL    (R0)
1797 007536 004737 010726  JSR    PC,@#GRESET          ;RESET DEFAULT PATTERN IN OUTPUT
1798                                     ;BUFFER.
1799 007542 077330          SOB    R3,G12
1800
1801                                     ;TEST ACCUMULATOR 2 WITH FLOATING ONE
1802 007544 012737 007574 001236  MOV    #G15,@#STMP2
1803 007552 012700 011002      MOV    #GPAT00,R0
1804 007556 012701 011042      MOV    #GDAT00,R1
1805 007562 104414          LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
1806 007564 004737 010650  JSR    PC,@#GSETUP          ;LOAD TEST PATTERN.
1807 007570 012703 000102      MOV    #102,R3
1808 007574          G15:
1809 007574 172410          LDD    (R0),AC0
1810 007576 174002          STD    AC0,AC2
1811 007600 172402          LDD    AC2,AC0          ;STORE THE TEST PATTERN.
1812 007602 174011          STD    AC0,(R1)
1813 007604 004737 010746  JSR    PC,@#GCMP          ;COMPARE THE DATA READ WITH
1814                                     ;THAT WHICH WAS WRITTEN.
1815 007610 005737 010776      TST    @#GFLAG1
1816 007614 001004          BNE    G16
1817 007616 005137 010776      COM    @#GFLAG1
1818 007622 000261          SEC
1819 007624 000401          BR     G17
1820 007626 000241          G16:  CLC
1821 007630 006160 000006  G17:  ROL    6(R0)          ;GENERATE THE NEXT TEST PATTERN.
1822 007634 006160 000004      ROL    4(R0)
1823 007640 006160 000002      ROL    2(R0)
1824 007644 006110          ROL    (R0)
1825 007646 004737 010726  JSR    PC,@#GRESET          ;RESET DEFAULT PATTERN IN OUTPUT
1826                                     ;BUFFER.
1827 007652 077330          SOB    R3,G15
1828
1829                                     ;TEST ACCUMULATOR 2 WITH FLOATING ZERO
1830 007654 012737 007704 001236  MOV    #G20,@#STMP2
1831 007662 012700 011012      MOV    #GPAT10,R0
1832 007666 012701 011042      MOV    #GDAT00,R1
1833 007672 104414          LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
1834 007674 004737 010650  JSR    PC,@#GSETUP          ;LOAD TEST PATTERN.
1835 007700 012703 000102      MOV    #102,R3
1836 007704          G20:
1837 007704 172410          LDD    (R0),AC0
1838 007706 174002          STD    AC0,AC2
1839 007710 172402          LDD    AC2,AC0          ;STORE THE TEST PATTERN.
1840 007712 174011          STD    AC0,(R1)
1841 007714 004737 010746  JSR    PC,@#GCMP          ;COMPARE THE DATA READ WITH
1842                                     ;THAT WHICH WAS WRITTEN.
1843 007720 005737 010776      TST    @#GFLAG1
1844 007724 001004          BNE    G21
1845 007726 005137 010776      COM    @#GFLAG1
1846 007732 000241          CLC
1847 007734 000401          BR     G22
1848 007736 000261          G21:  SEC
1849 007740 006160 000006  G22:  ROL    6(R0)          ;GENERATE THE NEXT TEST PATTERN.
1850 007744 006160 000004      ROL    4(R0)
1851 007750 006160 000002      ROL    2(R0)
    
```

```

1852 007754 006110          ROL    (R0)
1853 007756 004737 010726  JSR    PC,@#GRESET          ;RESET DEFAULT PATTERN IN OUTPUT
1854                               ;BUFFER.
1855 007762 077330          SOB    R3,G20
1856
1857                               ;TEST ACCUMULATOR 3 WITH FLOATING ONE
1858 007764 012737 010014 001236  MOV    #G23,@#STMP2
1859 007772 012700 011002      MOV    #GPAT00,R0
1860 007776 012701 011042      MOV    #GDAT00,R1
1861 010002 104414          LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
1862 010004 004737 010650  JSR    PC,@#GSETUP          ;LOAD TEST PATTERN.
1863 010010 012703 000102      MOV    #102,R3
1864
1865                               G23:
1866 010014 172410          LDD    (R0),AC0
1867 010016 174003          STD    AC0,AC3
1868 010020 172403          LDD    AC3,AC0
1869 010022 174011          STD    AC0,(R1)
1870 010024 004737 010746  JSR    PC,@#GCMP          ;COMPARE THE DATA READ WITH
1871                               ;THAT WHICH WAS WRITTEN.
1872 010030 005737 010776      TST    @#GFLAG1
1873 010034 001004          BNE    G24
1874 010036 005137 010776      COM    @#GFLAG1
1875 010042 000261          SEC
1876 010044 000401          BR     G25
1877 010046 000241          G24:  CLC
1878 010050 006160 000006  G25:  ROL    6(R0)
1879 010054 006160 000004      ROL    4(R0)
1880 010060 006160 000002      ROL    2(R0)
1881 010064 006110          ROL    (R0)
1882 010066 004737 010726  JSR    PC,@#GRESET          ;RESET DEFAULT PATTERN IN OUTPUT
1883                               ;BUFFER.
1884                               SOB    R3,G23
1885
1886                               ;TEST ACCUMULATOR 3 WITH FLOATING ZERO
1887 010074 012737 010124 001236  MOV    #G26,@#STMP2
1888 010102 012700 011012      MOV    #GPAT10,R0
1889 010106 012701 011042      MOV    #GDAT00,R1
1890 010112 104414          LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
1891 010114 004737 010650  JSR    PC,@#GSETUP          ;LOAD TEST PATTERN.
1892 010120 012703 000102      MOV    #102,R3
1893
1894                               G26:
1895 010124 172410          LDD    (R0),AC0
1896 010126 174003          STD    AC0,AC3
1897 010130 172403          LDD    AC3,AC0
1898 010132 174011          STD    AC0,(R1)
1899 010134 004737 010746  JSR    PC,@#GCMP          ;COMPARE THE DATA READ WITH
1900                               ;THAT WHICH WAS WRITTEN.
1901 010140 005737 010776      TST    @#GFLAG1
1902 010144 001004          BNE    G27
1903 010146 005137 010776      COM    @#GFLAG1
1904 010152 000241          CLC
1905 010154 000401          BR     G30
1906 010156 000261          G27:  SEC
1907 010160 006160 000006  G30:  ROL    6(R0)
1908 010164 006160 000004      ROL    4(R0)
1909 010170 006160 000002      ROL    2(R0)

```

```

1908 010174 006110          ROL    (R0)
1909 010176 004737 010726  JSR    PC,@#GRESET          ;RESET DEFAULT PATTERN IN OUTPUT
1910                               ;BUFFER.
1911 010202 077330          SOB    R3,G26
1912
1913                               ;TEST ACCUMULATOR 4 WITH FLOATING ONE
1914 010204 012737 010234 001236  MOV    #G31,@#STMP2
1915 010212 012700 011002      MOV    #GPAT00,R0
1916 010216 012701 011042      MOV    #GDAT00,R1
1917 010222 104414          LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
1918 010224 004737 010650  JSR    PC,@#GSETUP          ;LOAD TEST PATTERN.
1919 010230 012703 000102      MOV    #102,R3
1920 010234
1921 010234 172410          G31:   LDD    (R0),AC0
1922 010236 174004          STD    AC0,AC4
1923 010240 172404          LDD    AC4,AC0          ;STORE THE TEST PATTERN.
1924 010242 174011          STD    AC0,(R1)
1925 010244 004737 010746  JSR    PC,@#GCMP          ;COMPARE THE DATA READ WITH
1926                               ;THAT WHICH WAS WRITTEN.
1927 010250 005737 010776          TST    @#GFLAG1
1928 010254 001004          BNE    G32
1929 010256 005137 010776          COM    @#GFLAG1
1930 010262 000261          SEC
1931 010264 000401          BR     G33
1932 010266 000241          G32:   CLC
1933 010270 006160 000006  G33:   ROL    6(R0)          ;GENERATE THE NEXT TEST PATTERN.
1934 010274 006160 000004      ROL    4(R0)
1935 010300 006160 000002      ROL    2(R0)
1936 010304 006110          ROL    (R0)
1937 010306 004737 010726  JSR    PC,@#GRESET          ;RESET DEFAULT PATTERN IN OUTPUT
1938                               ;BUFFER.
1939 010312 077330          SOB    R3,G31
1940
1941                               ;TEST ACCUMULATOR 4 WITH FLOATING ZERO
1942 010314 012737 010344 001236  MOV    #G34,@#STMP2
1943 010322 012700 011012      MOV    #GPAT10,R0
1944 010326 012701 011042      MOV    #GDAT00,R1
1945 010332 104414          LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
1946 010334 004737 010650  JSR    PC,@#GSETUP          ;LOAD TEST PATTERN.
1947 010340 012703 000102      MOV    #102,R3
1948 010344
1949 010344 172410          G34:   LDD    (R0),AC0
1950 010346 174004          STD    AC0,AC4
1951 010350 172404          LDD    AC4,AC0          ;STORE THE TEST PATTERN.
1952 010352 174011          STD    AC0,(R1)
1953 010354 004737 010746  JSR    PC,@#GCMP          ;COMPARE THE DATA READ WITH
1954                               ;THAT WHICH WAS WRITTEN.
1955 010360 005737 010776          TST    @#GFLAG1
1956 010364 001004          BNE    G35
1957 010366 005137 010776          COM    @#GFLAG1
1958 010372 000241          CLC
1959 010374 000401          BR     G36
1960 010376 000261          G35:   SEC
1961 010400 006160 000006  G36:   ROL    6(R0)          ;GENERATE THE NEXT TEST PATTERN.
1962 010404 006160 000004      ROL    4(R0)
1963 010410 006160 000002      ROL    2(R0)
    
```

```

1964 010414 006110          ROL    (R0)
1965 010416 004737 010726 JSR    PC,@#GRESET          ;RESET DEFAULT PATTERN IN OUTPUT
1966                                     ;BUFFER.
1967 010422 077330          SOB    R3,G34
1968
1969                                     ;TEST ACCUMULATOR 5 WITH FLOATING ONE
1970 010424 012737 010454 001236 MOV    #G37,@#STMP2
1971 010432 012700 011002 MOV    #GPAT00,R0
1972 010436 012701 011042 MOV    #GDAT00,R1
1973 010442 104414          LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
1974 010444 004737 010650 JSR    PC,@#GSETUP          ;LOAD TEST PATTERN.
1975 010450 012703 000102 MOV    #102,R3
1976 010454          G37: LDD    (R0),AC0
1977 010454 172410          STD    AC0,AC5
1978 010456 174005          LDD    AC5,AC0          ;STORE THE TEST PATTERN.
1979 010460 172405          STD    AC0,(R1)
1980 010462 174011          JSR    PC,@#GCMP          ;COMPARE THE DATA READ WITH
1981 010464 004737 010746          ;THAT WHICH WAS WRITTEN.
1982
1983 010470 005737 010776          TST    @#GFLAG1
1984 010474 001004          BNE    G40
1985 010476 005137 010776          COM    @#GFLAG1
1986 010502 000261          SEC
1987 010504 000401          BR     G41
1988 010506 000241          G40: CLC
1989 010510 006160 000006          G41: ROL    6(R0)          ;GENERATE THE NEXT TEST PATTERN.
1990 010514 006160 000004          ROL    4(R0)
1991 010520 006160 000002          ROL    2(R0)
1992 010524 006110          ROL    (R0)
1993 010526 004737 010726 JSR    PC,@#GRESET          ;RESET DEFAULT PATTERN IN OUTPUT
1994                                     ;BUFFER.
1995 010532 077330          SOB    R3,G37
1996
1997                                     ;TEST ACCUMULATOR 5 WITH FLOATING ZERO
1998 010534 012737 010564 001236 MOV    #G42,@#STMP2
1999 010542 012700 011012 MOV    #GPAT10,R0
2000 010546 012701 011042 MOV    #GDAT00,R1
2001 010552 104414          LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
2002 010554 004737 010650 JSR    PC,@#GSETUP          ;LOAD TEST PATTERN.
2003 010560 012703 000102 MOV    #102,R3
2004 010564          G42: LDD    (R0),AC0
2005 010564 172410          STD    AC0,AC5
2006 010566 174005          LDD    AC5,AC0          ;STORE THE TEST PATTERN.
2007 010570 172405          STD    AC0,(R1)
2008 010572 174011          JSR    PC,@#GCMP          ;COMPARE THE DATA READ WITH
2009 010574 004737 010746          ;THAT WHICH WAS WRITTEN.
2010
2011 010600 005737 010776          TST    @#GFLAG1
2012 010604 001004          BNE    G43
2013 010606 005137 010776          COM    @#GFLAG1
2014 010612 000241          CLC
2015 010614 000401          BR     G44
2016 010616 000261          G43: SEC
2017 010620 006160 000006          G44: ROL    6(R0)          ;GENERATE THE NEXT TEST PATTERN.
2018 010624 006160 000004          ROL    4(R0)
2019 010630 006160 000002          ROL    2(R0)
    
```

```

2020 010634 006110          ROL    (R0)
2021 010636 004737 010726 JSR    PC,@#GRESET      ;RESET DEFAULT PATTERN IN OUTPUT
2022                                     ;BUFFER.
2023 010642 077330          SOB    R3,G42
2024
2025
2026 010644 000137 011052          JMP    @#GDONE
2027
2028                                     ;USE THIS ROUTINE TO INITIALIZE ALL THE DATA BUFFERS.
2029 010650 012705 010776 GSETUP: MOV    #GFLAG1,R5
2030 010654 012704 000026          MOV    #26,R4
2031 010660 005025          1$:    CLR    (R5)+
2032 010662 077402          SOB    R4,1$
2033
2034 010664 012705 011012          MOV    #GPAT10,R5
2035 010670 012704 000010          MOV    #10,R4
2036 010674 005125          2$:    COM    (R5)+
2037 010676 077402          SOB    R4,2$
2038
2039 010700 020067 000076          GS1:   CMP    R0,GPAT00
2040 010704 001401          BEQ    3$
2041 010706 000207          RTS    PC
2042
2043 010710 012705 011042          3$:    MOV    #GDAT00,R5
2044 010714 012704 000004          MOV    #4,R4
2045 010720 005125          4$:    COM    (R5)+
2046 010722 077402          SOB    R4,4$
2047 010724 000207          RTS    PC
2048
2049 010726 012705 011042          GRESET: MOV    #GDAT00,R5
2050 010732 012704 000004          MOV    #4,R4
2051 010736 005025          1$:    CLR    (R5)+
2052 010740 077402          SOB    R4,1$
2053 010742 000137 010700          JMP    @#GS1
2054
2055                                     ;SEE IF THE DATA WRITTEN MATCHES THE DATA READ.
2056 010746 011637 001244          GCMP:  MOV    (SP),@#$TMP5      ;STORE ERROR PC
2057 010752 012705 011042          MOV    #GDAT00,R5
2058 010756 012704 000004          MOV    #4,R4
2059 010762 010002          MOV    R0,R2
2060 010764 022225          1$:    CMP    (R2)+,(R5)+
2061 010766 001401          BEQ    2$
2062 010770 104003          ERROR  3
2063 010772 077404          2$:    SOB    R4,1$      ;PROBABLY BAD MMU, OTHERWISE IT IS THE FP CHIP
2064 010774 000207          RTS    PC
2065
2066
2067
2068
2069 010776 000000          GFLAG1: 0
2070 011000 000000          GFLAG2: 0
2071
2072 011002 000000          GPAT00: 0
2073 011004 000000          GPAT01: 0
2074 011006 000000          GPAT02: 0
2075 011010 000000          GPAT03: 0

```

2076
 2077 011012 177777
 2078 011014 177777
 2079 011016 177777
 2080 011020 177777
 2081
 2082 011022 177777
 2083 011024 177777
 2084 011026 177777
 2085 011030 177777
 2086
 2087 011032 000000
 2088 011034 000000
 2089 011036 000000
 2090 011040 000000
 2091
 2092 011042 000000
 2093 011044 000000
 2094 011046 000000
 2095 011050 000000
 2096
 2097
 2098 011052
 2099 011052 104413
 2100
 2101
 2102
 2103
 2104
 2105
 2106
 2107
 2108
 2109
 2110
 2111
 2112
 2113 011054 000004
 2114 011056 104414
 2115
 2116 011060 005037 011604
 2117 011064 012700 011606
 2118 011070 012701 011726
 2119 011074 012703 000024
 2120 011100 012120
 2121 011102 077302
 2122
 2123 011104 004767 000422
 2124
 2125 011110 170011
 2126
 2127 011112 012700 011606
 2128 011116 172410
 2129 011120 174001
 2130
 2131 011122 012700 011616

GPAT10: -1
 GPAT11: -1
 GPAT12: -1
 GPAT13: -1
 GAND0: -1
 GAND1: -1
 GAND2: -1
 GAND3: -1
 GOR0: 0
 GOR1: 0
 GOR2: 0
 GOR3: 0
 GDAT00: 0
 GDAT01: 0
 GDAT02: 0
 GDAT03: 0

GDONE: 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?).

 *TEST 12 FPP ACCUMULATORS DUAL ADDRESS TEST
 *
 *THIS TEST PERFORMS A DUAL ADDRESSING TEST ON THE FLOATING ACCUMULATORS.
 *NOTE THAT ACCUMULATOR ZERO IS USED TO ACCESS ALL THE OTHERS.
 *

TST12: SCOPE LPERR ;SET UP THE LOOP ON ERROR ADDRESS.

H1: CLR @#HFLAG ;INITIALIZE THE LOAD BUFFER DATA.
 MOV #HA1W,RO
 MOV #HDAT1,R1
 MOV #24,R3

H2: MOV (R1)+,(R0)+
 SOB R3,H2

JSR PC,HCLR ;CLEAR THE OUTPUT DATA BUFFER.

H3: SETD ;LOAD ACCUMULATOR 1
 MOV #HA1W,RO
 LDD (RO),ACO
 STD ACO,AC1
 ;LOAD ACCUMULATOR 2
 MOV #HA2W,RO


```

2132 011126 172410          LDD      (R0),AC0
2133 011130 174002          STD      AC0,AC2
2134          :LOAD ACCUMULATOR 3
2135 011132 012700 011626    MOV      #HA3W,R0
2136 011136 172410          LDD      (R0),AC0
2137 011140 174003          STD      AC0,AC3
2138          :LOAD ACCUMULATOR 4
2139 011142 012700 011636    MOV      #HA4W,R0
2140 011146 172410          LDD      (R0),AC0
2141 011150 174004          STD      AC0,AC4
2142          :LOAD ACCUMULATOR 5
2143 011152 012700 011646    MOV      #HA5W,R0
2144 011156 172410          LDD      (R0),AC0
2145 011160 174005          STD      AC0,AC5
2146
2147 011162 004737 011416    H4:     JSR      PC,@#HSTD          ;GO READ ALL ACCUMULATORS BACK.
2148
2149 011166 004737 011474          JSR      PC,@#HCMP          ;SEE IF DATA IS CORRECT.
2150
2151          :COMPLIMENT EACH WORD OF THE DATA STORED IN ACCUMULATOR 1,
2152          :RELOAD THAT ACCUMULATOR, READ ALL THE ACCUMULATORS BACK AND CHECK
2153          :THE DATA.
2154 011172 012700 011606    MOV      #HA1W,R0
2155 011176 012702 000004    MOV      #4,R2
2156 011202 010001          MOV      R0,R1
2157 011204 005121          H5:     COM      (R1)+
2158 011206 172410          LDD      (R0),AC0
2159 011210 174001          STD      AC0,AC1
2160 011212 004737 011416    JSR      PC,@#HSTD          ;READ ALL THE ACCUMULATORS BACK.
2161 011216 004737 011474    JSR      PC,@#HCMP          ;CHECK THE DATA.
2162 011222 077210          SOB      R2,H5
2163
2164          :COMPLIMENT EACH WORD OF THE DATA STORED IN ACCUMULATOR 2,
2165          :RELOAD THAT ACCUMULATOR, READ ALL THE ACCUMULATORS BACK AND CHECK
2166          :THE DATA.
2167 011224 012700 011616    MOV      #HA2W,R0
2168 011230 012702 000004    MOV      #4,R2
2169 011234 010001          MOV      R0,R1
2170 011236 005121          H6:     COM      (R1)+
2171 011240 172410          LDD      (R0),AC0
2172 011242 174002          STD      AC0,AC2
2173 011244 004737 011416    JSR      PC,@#HSTD          ;READ ALL THE ACCUMULATORS BACK.
2174 011250 004737 011474    JSR      PC,@#HCMP          ;CHECK THE DATA.
2175 011254 077210          SOB      R2,H6
2176
2177          :COMPLIMENT EACH WORD OF THE DATA STORED IN ACCUMULATOR 3,
2178          :RELOAD THAT ACCUMULATOR, READ ALL THE ACCUMULATORS BACK AND CHECK
2179          :THE DATA.
2180 011256 012700 011626    MOV      #HA3W,R0
2181 011262 012702 000004    MOV      #4,R2
2182 011266 010001          MOV      R0,R1
2183 011270 005121          H7:     COM      (R1)+
2184 011272 172410          LDD      (R0),AC0
2185 011274 174003          STD      AC0,AC3
2186 011276 004737 011416    JSR      PC,@#HSTD          ;READ ALL THE ACCUMULATORS BACK.
2187 011302 004737 011474    JSR      PC,@#HCMP          ;CHECK THE DATA.
    
```

2188 011306 077210
 2189
 2190
 2191
 2192
 2193 011310 012700 011636
 2194 011314 012702 000004
 2195 011320 010001
 2196 011322 005121
 2197 011324 172410
 2198 011326 174004
 2199 011330 004737 011416
 2200 011334 004737 011474
 2201 011340 077210
 2202
 2203
 2204
 2205
 2206 011342 012700 011646
 2207 011346 012702 000004
 2208 011352 010001
 2209 011354 005121
 2210 011356 172410
 2211 011360 174005
 2212 011362 004737 011416
 2213 011366 004737 011474
 2214 011372 077210
 2215
 2216
 2217 011374 005737 011604
 2218 011400 001402
 2219 011402 000137 011776
 2220
 2221 011406 005137 011604
 2222 011412 000137 011110
 2223
 2224
 2225 011416 004737 011532
 2226
 2227 011422 012704 011656
 2228 011426 172401
 2229 011430 174014
 2230
 2231 011432 012704 011666
 2232 011436 172402
 2233 011440 174014
 2234
 2235 011442 012704 011676
 2236 011446 172403
 2237 011450 174014
 2238
 2239 011452 012704 011706
 2240 011456 172404
 2241 011460 174014
 2242
 2243 011462 012704 011716

```

SOB R2,H7

:COMPLIMENT EACH WORD OF THE DATA STORED IN ACCUMULATOR 4,
:RELOAD THAT ACCUMULATOR, READ ALL THE ACCUMULATORS BACK AND CHECK
:THE DATA.
MOV #HA4W,R0
MOV #4,R2
MOV R0,R1
H10: COM (R1)+
LDD (R0),AC0
STD AC0,AC4
JSR PC,@#HSTD ;READ ALL THE ACCUMULATORS BACK.
JSR PC,@#HCMP ;CHECK THE DATA.
SOB R2,H10

:COMPLIMENT EACH WORD OF THE DATA STORED IN ACCUMULATOR 5,
:RELOAD THAT ACCUMULATOR, READ ALL THE ACCUMULATORS BACK AND CHECK
:THE DATA.
MOV #HA5W,R0
MOV #4,R2
MOV R0,R1
H11: COM (R1)+
LDD (R0),AC0
STD AC0,AC5
JSR PC,@#HSTD ;READ ALL THE ACCUMULATORS BACK.
JSR PC,@#HCMP ;CHECK THE DATA.
SOB R2,H11

TST @#HFLAG
BEQ H12
JMP @#HDONE

H12: COM @#HFLAG
JMP @#H3

:STORE ALL ACCUMULATORS IN THE OUTPUT BUFFERS.
HSTD: JSR PC,@#HCLR ;CLEAR ALL OUTPUT BUFFERS.
:STORE ACCUMULATOR 1
MOV #HA1R,R4
LDD AC1,AC0
STD AC0,(R4)
:STORE ACCUMULATOR 2
MOV #HA2R,R4
LDD AC2,AC0
STD AC0,(R4)
:STORE ACCUMULATOR 3
MOV #HA3R,R4
LDD AC3,AC0
STD AC0,(R4)
:STORE ACCUMULATOR 4
MOV #HA4R,R4
LDD AC4,AC0
STD AC0,(R4)
:STORE ACCUMULATOR 5
MOV #HA5R,R4

```

```

2244 011466 172405          LDD    AC5,AC0
2245 011470 174014          STD    AC0,(R4)
2246 011472 000207          RTS    PC
2247
2248          ;COMPARE DATA LOADED WITH DATA READ.
2249 011474 012637 011602    HCMP:  MOV    (SP)+,@#HADR          ;SAVE RETURN ADDRESS.
2250 011500 012703 011606          MOV    #HA1W,R3
2251 011504 012704 011656          MOV    #HA1R,R4
2252 011510 012705 000024          MOV    #24,R5
2253 011514 022324          HCMP1: CMP    (R3)+,(R4)+
2254 011516 001402          BEQ    HCMP2
2255 011520 000137 011550          JMP    @#HERROR
2256 011524 077505          HCMP2: SOB   R5,HCMP1
2257 011526 000177 000050          JMP    @#HADR
2258
2259          ;CLEAR THE DATA OUTPUT BUFFER.
2260 011532 012704 011656    HCLR:  MOV    #HA1R,R4
2261 011536 012705 000024          MOV    #24,R5
2262 011542 005024          HCLR1: CLR   (R4)+
2263 011544 077502          SOB   R5,HCLR1
2264 011546 000207          RTS    PC
2265
2266          ;REPORT ERROR.
2267 011550          HERROR:
2268 011550 012703 011606          MOV    #HA1W,R3
2269 011554 012704 001236          MOV    #STMP2,R4
2270 011560 012705 000012          MOV    #12,R5
2271 011564 010324          1$:    MOV    R3,(R4)+
2272 011566 062703 000010          ADD   #10,R3
2273 011572 077504          SOB   R5,1$
2274 011574 104001          2$:    ERROR 1
2275 011576 000137 011776          JMP    @#HDONE
2276
2277
2278 011602 000000          HADR:  0
2279 011604 000000          HFLAG: 0
2280
2281 011606 000000 000000 000000 HA1W:  .WORD 0,0,0,0
2282 011614 000000          HA2W:  .WORD 0,0,0,0
2283 011616 000000 000000 000000 HA3W:  .WORD 0,0,0,0
2284 011624 000000          HA4W:  .WORD 0,0,0,0
2285 011626 000000 000000 000000 HA5W:  .WORD 0,0,0,0
2286 011634 000000          HA1R:  .WORD 0,0,0,0
2287 011636 000000 000000 000000 HA2R:  .WORD 0,0,0,0
2288 011644 000000          HA3R:  .WORD 0,0,0,0
2289 011646 000000 000000 000000 HA4R:  .WORD 0,0,0,0
2290 011654 000000          HA5R:  .WORD 0,0,0,0
2291
2292 011656 000000 000000 000000 HA1R:  .WORD 0,0,0,0
2293 011664 000000          HA2R:  .WORD 0,0,0,0
2294 011666 000000 000000 000000 HA3R:  .WORD 0,0,0,0
2295 011674 000000          HA4R:  .WORD 0,0,0,0
2296 011676 000000 000000 000000 HA5R:  .WORD 0,0,0,0
2297 011704 000000
2298 011706 000000 000000 000000
2299 011714 000000
    
```

```

2300 011716 000000 000000 000000 HA5R: .WORD 0,0,0,0
2301 011724 000000
2302
2303 011726 073567 073567 073567 HDAT1: .WORD 73567,73567,73567,73567
2304 011734 073567
2305 011736 063146 063146 063146 HDAT2: .WORD 63146,63146,63146,63146
2306 011744 063146
2307 011746 010421 010421 010421 HDAT3: .WORD 10421,10421,10421,10421
2308 011754 010421
2309 011756 031463 031463 031463 HDAT4: .WORD 31463,31463,31463,31463
2310 011764 031463
2311 011766 042104 042104 042104 HDAT5: .WORD 42104,42104,42104,42104
2312 011774 042104
2313

```

```

2314 011776 HDONE:
2315 011776 104413 RSETUP ;GO INITIALIZE THE FPS AND STACK; AND
2316 ;SEE IF THE USER HAS EXPRESSED
2317 ;THE DESIRE TO CHANGE THE SOFTWARE
2318 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
2319 ;THE USER TYPED CONTROL G?).
2320
2321
2322

```

```

*****
: *TEST 13 FSRC MODE 0 WITH ILLEGAL ACCUMULATOR TEST
: *
: *THIS IS A TEST OF FSRC MODE 0 WITH ACCUMULATORS 6 AND 7. USE OF
: *EITHER OF THESE NON-EXISTENT ACCUMULATORS SHOULD RESULT IN A TRAP TO 244
: *WITH FEC=2 (ILLEGAL OPCODE TRAP)
: *
*****

```

```

2323
2324
2325
2326
2327
2328
2329
2330 012000 000004
2331 012002
2332 012002 104414
2333 012004 170011
2334 012006 012700 012512
2335 012012 172410
2336
2337 012014 012737 012212 000244
2338
2339
2340 012022 012700 000001
2341
2342 012026 012737 012422 000004
2343 012034 005003
2344
2345 012036 172407 S2:
2346 012040 170000 S3:
2347 012042 005203 INC R3
2348 012044 005203 S4: INC R3
2349
2350 012046 012701 012522
2351 012052 174011
2352
2353 012054 012701 012522
2354 012060 012702 012512
2355 012064 012703 000004

```

```

TST13: SCOPE
S1: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
SETD ;SET FD
MOV #SPAT10,R0 ;LOAD ACO
LDD (R0),AC0
MOV #SERRO,@#FPVECT ;USE OF THE NON-EXISTENT AC-
;CUMULATOR SHOULD RESULT IN
;A TRAP TO 244.
MOV #1,R0 ;A FAILURE IN THE FSRC FLOWS
;WILL RESULT IN AN ODD ADDRESS
;TRAP TO 4.
CLR R3
LDD AC7,AC0
CFCC
INC R3
INC R3
MOV #SDAT00,R1 ;NO TRAP OCCURRED!!
STD ACO,(R1) ;SEE IF ACO WAS MODIFIED.
MOV #SDAT00,R1
MOV #SPAT10,R2
MOV #4,R3

```

2356	012070	022122			S5:	CMP	(R1)+,(R2)+	
2357	012072	001402				BEQ	S6	
2358	012074	000137	012352			JMP	@#SERR2	;BAD DATA
2359	012100	077305			S6:	SOB	R3,S5	
2360								
2361	012102	000137	012376			JMP	@#SERR3	;MISSING TRAP
2362								
2363								
2364	012106							
2365	012106	104414			S7:	LPERR		;SET UP THE LOOP ON ERROR ADDRESS.
2366	012110	170011				SETD		
2367								
2368	012112	012700	012512			MOV	#SPAT10,R0	;LOAD ACO
2369	012116	172410				LDD	(R0),AC0	
2370								
2371	012120	012737	012270	000244		MOV	#SERR4,@#FPVECT	
2372	012126	012700	000001			MOV	#1,R0	
2373	012132	012737	012454	000004		MOV	#SERR5,@#ERRVECT	
2374	012140	005003				CLR	R3	
2375								
2376	012142	172406			S8:	LDD	AC6,AC0	
2377	012144	170000			S9:	CFCC		
2378	012146	005203				INC	R3	
2379	012150	005203			S10:	INC	R3	
2380								
2381	012152	012701	012522			MOV	#SDAT00,R1	
2382	012156	174011				STD	AC0,(R1)	;NO TRAP! GET ACO.
2383								
2384	012160	012701	012522			MOV	#SDAT00,R1	;WAS ACO MODIFIED.
2385	012164	012702	012512			MOV	#SPAT10,R2	
2386	012170	012703	000004			MOV	#4,R3	
2387	012174	022122			S11:	CMP	(R1)+,(R2)+	
2388	012176	001402				BEQ	S12	
2389	012200	000137	012364			JMP	@#SERR6	
2390	012204	077305			S12:	SOB	R3,S11	
2391	012206	000137	012410			JMP	@#SERR7	
2392								
2393								
2394	012212	021627	012040					
2395	012216	001402			SERR0:	CMP	(SP),#S3	;PC OF TRAP CORRECT?
2396	012220	000137	062524			BEQ	1\$	
2397						JMP	@#FPSPUR	
2398	012224	012737	012106	012506	1\$:	MOV	#S7,@#SADR	
2399								
2400	012232	011637	001236		SERR10:	MOV	(SP),@#SAMP2	
2401	012236	022626				CMP	(SP)+,(SP)+	
2402	012240	005004				CLR	R4	
2403	012242	170204				STFPS	R4	;IS FPS CORRECT?
2404	012244	022704	100200			CMP	#100200,R4	
2405	012250	001020				BNE	SERR15	
2406								
2407	012252	005004				CLR	R4	
2408	012254	170304				STST	R4	;IS FEC CORRECT?
2409	012256	022704	000002			CMP	#2,R4	
2410	012262	001023				BNE	SERR20	
2411	012264	000177	000216			JMP	@SADR	

```

2412
2413 012270 021627 012144 SERR4: CMP (SP),#S9
2414 012274 001402 BEQ 1$
2415 012276 000137 062524 JMP @#FPSPUR
2416 012302 012737 012532 012506 1$: MOV #SDONE,@#SADR
2417 012310 000750 BR SERR10
2418
2419
2420 012312 012737 100200 001242 :REPORT FPS FAILURE:
SERR15: MOV #100200,@#STMP4
2421 012320 010437 001240 MOV R4,@#STMP3
2422 012324 104001 1$: ERROR 1
2423 012326 000177 000154 JMP @SADR
2424
2425
2426 012332 012737 000002 001242 :REPORT FEC BAD:
SERR20: MOV #2,@#STMP4
2427 012340 010437 001240 MOV R4,@#STMP3
2428 012344 104001 1$: ERROR 1
2429 012346 000177 000134 JMP @SADR
2430
2431
2432 :ACO WAS MODIFIED. (BUT FSRC) FORK FAILED.
2433 012352 012737 012036 001236 SERR2: MOV #S2,@#STMP2
2434 012360 104001 1$: ERROR 1
2435 012362 000463 BR SDONE
2436 012364 012737 012142 001236 SERR6: MOV #S8,@#STMP2
2437 012372 104001 1$: ERROR 1
2438 012374 000456 BR SDONE
2439
2440 012376 012737 012036 001236 SERR3: MOV #S2,@#STMP2
2441 012404 104001 1$: ERROR 1
2442 012406 000451 BR SDONE
2443 012410 012737 012142 001236 SERR7: MOV #S8,@#STMP2
2444 012416 104001 1$: ERROR 1
2445 012420 000444 BR SDONE
2446
2447 :FAILURE OF (BUT FSRC) CAUSED AN ODD ADDRESS TRAP TO 4.
2448 012422 021627 012040 SERR1: CMP (SP),#S3 ;DID TRAP OCCUR ON TESTED INSTRUCTION?
2449 012426 001405 BEQ 1$
2450 012430 021627 012044 CMP (SP),#S4
2451 012434 001402 BEQ 1$
2452 012436 000137 062556 JMP @#CPSPUR
2453
2454 012442 011637 001236 1$: MOV (SP),@#STMP2
2455 012446 022626 CMP (SP)+,(SP)+
2456 012450 104001 2$: ERROR 1
2457 012452 000427 BR SDONE
2458
2459 012454 021627 012142 SERR5: CMP (SP),#S8 ;DID TRAP OCCUR ON TEST INSTRUCTION?
2460 012460 001405 BEQ 1$
2461 012462 021627 012144 CMP (SP),#S9
2462 012466 001402 BEQ 1$
2463 012470 000137 062556 JMP @#CPSPUR
2464
2465 012474 011637 001236 1$: MOV (SP),@#STMP2
2466 012500 022626 CMP (SP)+,(SP)+
2467 012502 104001 2$: ERROR 1
    
```

```

2468 012504 000412          BR      SDONE
2469
2470 012506 000000          SADR:   0
2471 012510 177777          -1
2472 012512 010421          SPAT10: 10421
2473 012514 021042          SPAT11: 21042
2474 012516 031463          SPAT12: 31463
2475 012520 042104          SPAT13: 42104
2476
2477 012522 000000          SDAT00: 0
2478 012524 000000          SDAT01: 0
2479 012526 000000          SDAT02: 0
2480 012530 000000          SDAT03: 0
2481
2482 012532
2483 012532 104413          SDONE:  RSETUP
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496 012534 000004
2497 012536 104414
2498
2499 012540
2500 012540 170011
2501
2502 012542 012700 013016          MOV     #JDAT0,R0
2503 012546 172410          LDD     (R0),ACO          ;LOAD ACO=ALL 1
2504
2505 012550 012700 012776          MOV     #JDAT10,R0
2506 012554 005003          CLR     R3
2507 012556 012737 012646 000004          MOV     #JERR0,@#ERRVECT
2508
2509 012564 172420          J2:    LDD     (R0)+,ACO          ;TEST INSTRUCTION
2510 012566 005203          J3:    INC     R3
2511 012570 005203          J4:    INC     R3
2512
2513 012572 012701 013006          MOV     #JDAT00,R1
2514 012576 174011          STD     ACO,(R1)          ;PICK UP RESULTS
2515
2516 012600 020027 012766          CMP     R0,#JBUFO
2517 012604 001001          BNE    1$
2518 012606 000442          BR     JERR1
2519
2520 012610 012702 012776          1$:   MOV     #JDAT10,R2          ;IS DATA CORRECT?
2521 012614 012703 013006          MOV     #JDAT00,R3
2522 012620 012704 000004          MOV     #4,R4
2523 012624 022223          J5:   CMP     (R2)+,(R3)+
    
```

```

;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).
    
```

```

*****
;TEST 14          FSRC MODE 2 TEST
;
; THIS IS A TEST OF FSRC MODE 2, AUTO
; INCREMENT MODE.
;
*****
    
```

```

TST14:  SCOPE
        LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
    
```

```

J1:    SETD          ;SET DOUBLE MODE
    
```

```

        MOV     #JDAT0,R0
        LDD     (R0),ACO          ;LOAD ACO=ALL 1
    
```

```

        MOV     #JDAT10,R0
        CLR     R3
        MOV     #JERR0,@#ERRVECT
    
```

```

J2:    LDD     (R0)+,ACO          ;TEST INSTRUCTION
J3:    INC     R3
J4:    INC     R3
    
```

```

        MOV     #JDAT00,R1
        STD     ACO,(R1)          ;PICK UP RESULTS
    
```

```

        CMP     R0,#JBUFO
        BNE    1$
        BR     JERR1
    
```

```

1$:   MOV     #JDAT10,R2          ;IS DATA CORRECT?
    
```

```

        MOV     #JDAT00,R3
        MOV     #4,R4
        J5:   CMP     (R2)+,(R3)+
    
```

```

2524 012626 001401          BEQ      J6
2525 012630 000443          BR       JERR2
2526 012632 077404          SOB      R4,J5
2527
2528 012634 022700 013006    CMP      #JDAT10+10,R0    ;WAS R0 INCREM.
2529 012640 001401          BEQ      J7              ;BY 10 (OCTAL)
2530 012642 000424          BR       JERR1
2531
2532 012644 000470          J7:      BR       JDONE
2533
2534          ;IF A TRAP THROUGH 4 OCCURS COME HERE
2535
2536 012646 021627 012566    JERR0:   CMP      (SP),#J3    ;SEE IF THE TRAP
2537 012652 001405          BEQ      J10            ;OCCURRED ON THE
2538 012654 021627 012570    CMP      (SP),#J4    ;TESTED INSTRUCTION
2539 012660 001402          BEQ      J10
2540 012662 000137 062556    JMP      @#CPSPUR
2541
2542 012666 012737 000762 001240 J10:     MOV      #762,@#STMP3    ;REPORT FSRC FLOW
2543 012674 012737 000322 001242    MOV      #322,@#STMP4    ;FAILURE
2544 012702 011637 001236    MOV      (SP),@#STMP2
2545 012706 022626          CMP      (SP)+,(SP)+
2546 012710 104001          1$:     ERROR    1
2547 012712 000445          BR       JDONE
2548
2549 012714          JERR1:   ;REPORT, R0 NOT
2550 012714 012737 012564 001236    MOV      #J2,@#STMP2    ;CORRECTLY AFFECTED
2551 012722 010037 001240          MOV      R0,@#STMP3
2552 012726 012737 013006 001242    MOV      #JDAT10+10,@#STMP4
2553 012734 104001          1$:     ERROR    1
2554 012736 000433          BR       JDONE
2555
2556          ;REPORT DATA FAILURE
2557
2558 012740          JERR2:   MOV      #J2,@#STMP2
2559 012740 012737 012564 001236    MOV      #JDAT10,@#STMP3
2560 012746 012737 012776 001240    MOV      #JDAT00,@#STMP4
2561 012754 012737 013006 001242          1$:     ERROR    1
2562 012762 104001          BR       JDONE
2563 012764 000420
2564
2565 012766 010421          JBUF0:   .WORD    010421
2566 012770 021042          JBUF1:   .WORD    021042
2567 012772 042104          JBUF2:   .WORD    042104
2568 012774 031463          JBUF3:   .WORD    031463
2569
2570 012776 052525          JDAT10:  .WORD    052525
2571 013000 114631          JDAT11:  .WORD    114631
2572 013002 063146          JDAT12:  .WORD    063146
2573 013004 073567          JDAT13:  .WORD    073567
2574
2575 013006 000000          JDAT00:  .WORD    0
2576 013010 000000          JDAT01:  .WORD    0
2577 013012 000000          JDAT02:  .WORD    0
2578 013014 000000          JDAT03:  .WORD    0
2579
    
```



```

2580 013016 177777          JDAT0:          -1
2581 013020 177777          JDAT1:          -1
2582 013022 177777          JDAT2:          -1
2583 013024 177777          JDAT3:          -1
2584
2585
2586 013026                JDONE:
2587 013026 104413          RSETUP          ;GO INITIALIZE THE FPS AND STACK; AND
2588                                     ;SEE IF THE USER HAS EXPRESSED
2589                                     ;THE DESIRE TO CHANGE THE SOFTWARE
2590                                     ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
2591                                     ;THE USER TYPED CONTROL G?).
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601 013030 000004          :*****
2602 013032 104414          :*TEST 15          FSRC MODE 4 TEST
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
    013030 000004          :* THIS IS A TEST OF FSRC MODE 4, AUTO
    013032 104414          :* DECREMENT MODE.
    013033 000004          :*****
    013034 170011          TST15: SCOPE
    013036 012700 013310   LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
    013038 172410          K1:
    013040 012700 013270   SETD          ;SET DOUBLE MODE
    013042 172410          MOV          #KPATO,R0
    013044 012700 013270   LDD          (R0),ACO          ;LOAD A DEFAULT
    013046 005003          CLR          R3          ;PATTERN INTO ACO
    013048 012737 013142 000004   MOV          #KERRO,@#ERRVECT
    013050 172440          K2: LDD          -(R0),ACO          ;TEST INSTRUCTION
    013052 005203          K3: INC          R3
    013054 005203          K4: INC          R3
    013056 012701 013300   MOV          #KDAT00,R1
    013058 174011          STD          ACO,(R1)          ;PICK UP THE RESULT
    013060 020027 013300   CMP          R0,#KBUFO+10      ;WAS AN AUTO
    013062 001001          BNE          1$          ;INCREMENT EXECUTED
    013064 000441          BR          KERR1
    013066 012702 013260   1$: MOV          #KDAT10,R2          ;IS DATA CORRECT?
    013068 012703 013300   MOV          #KDAT00,R3
    013070 012704 000004   MOV          #4,R4
    013072 022223          K5: CMP          (R2)+,(R3)+
    013074 001401          BEQ          K6
    013076 000442          BR          KERR2
    013078 077404          K6: SOB          R4,K5
    013080 022700 013260   CMP          #KBUFO-10,R0      ;WAS R0 DECREMENTED
    013082 001401          BEQ          K7          ;PROPERLY?
    013084 000423          BR          KERR1
    
```

```

2636
2637 013140 000467          K7:      BR      KDONE
2638
2639          ;TRAP TO HERE ON AN ODD ADDRESS ERROR
2640
2641 013142 021627 013062    KERR0:  CMP      (SP),#K3      ;SEE IF THE ERROR
2642 013146 001405          BEQ      K10          ;OCCURRED AT THE
2643 013150 021627 013064    CMP      (SP),#K4      ;INSTRUCTION TESTED.
2644 013154 001402          BEQ      K10
2645 013156 000137 062556    JMP      @#CPSPUR
2646
2647 013162 012737 000762 001240 K10:    MOV      #762,@#STMP3 ;REPORT FAILURE IN
2648 013170 012737 000324 001242 MOV      #324,@#STMP4 ;FSRC FLOWS
2649 013176 011637 001236    MOV      (SP),@#STMP2
2650 013202 104001          1$:     ERROR    1
2651 013204 000445          BR      KDONE
2652
2653 013206          KERR1:          ;REPORT, RO
2654 013206 012737 013060 001236 MOV      #K2,@#STMP2 ;INCORRECTLY AFFECTED.
2655 013214 010037 001240    MOV      RO,@#STMP3
2656 013220 012737 013260 001242 MOV      #KDATIO,@#STMP4
2657 013226 104001          1$:     ERROR    1
2658 013230 000433          BR      KDONE
2659
2660          ;REPORT DATA FAILURE
2661
2662 013232          KERR2:
2663 013232 012737 013060 001236 MOV      #K2,@#STMP2
2664 013240 012737 013260 001240 MOV      #KDATIO,@#STMP3
2665 013246 012737 013300 001242 MOV      #KDATOO,@#STMP4
2666 013254 104001          1$:     ERROR    1
2667 013256 000420          BR      KDONE
2668
2669 013260 052525    KDATIO: .WORD    052525
2670 013262 114631    KDATI1:      114631
2671 013264 063140    KDATI2:      063140
2672 013266 073567    KDATI3:      073567
2673
2674 013270 010421    KBUF0:      010421
2675 013272 031463    KBUF1:      031463
2676 013274 042104    KBUF2:      042104
2677 013276 021042    KBUF3:      021042
2678
2679 013300 000000    KDAT00:      0
2680 013302 000000    KDAT01:      0
2681 013304 000000    KDAT02:      0
2682 013306 000000    KDAT03:      0
2683
2684 013310 177777    KPAT0:      -1
2685 013312 177777    KPAT1:      -1
2686 013314 177777    KPAT2:      -1
2687 013316 177777    DPAT3:      -1
2688
2689 013320          KDONE:
2690 013320 104413          RSETUP      ;GO INITIALIZE THE FPS AND STACK; AND
2691          ;SEE IF THE USER HAS EXPRESSED
    
```

2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747

013322 000004
013324 104414
013326 170011
013330 012700 013576
013334 172410
013336 012700 013620
013342 012701 013606
013346 012702 000004
013352 012120
013354 077202
013356 012700 013620
013362 005003
013364 170001
013366 172420
013370 005203
013372
013372 170011
013374 012701 013632
013400 174011
013402 020027 013624
013406 001401
013410 000421
013412 012737 177777 013624
013420 012737 177777 013626
013426 012702 013620
013432 012703 013632
013436 012704 000004
013442 022223
013444 001401
013446 000427
013450 077404

```

;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

:*****
:*TEST 16      FSRC MODE 2, WITH FD=0, TEST
:*
:* THIS IS A TEST OF FSRC MODE 2 WITH
:* FD=0. (AUTO INCREMENT)
:*
:*****
TST16: SCOPE
        LPERR                                ;SET UP THE LOOP ON ERROR ADDRESS.

L1:     SETD                                ;SET DOUBLE MODE

        MOV      #LPAT10,R0
        LDD      (R0),AC0                    ;LOAD AC0

        MOV      #LDAT10,R0                  ;SET UP THE INPUT
        MOV      #LPAT20,R1                  ;DATA
        MOV      #4,R2

1$:     MOV      (R1)+,(R0)+
        SOB      R2,1$

        MOV      #LDAT10,R0
        CLR      R3
        SETF                                ;CLEAR FD.

L2:     LDF      (R0)+,AC0
L3:     INC      R3

L4:     SETD                                ;SET FD

        MOV      #LDAT00,R1
        STD      AC0,(R1)                    ;PICK UP RESULTS

        CMP      R0,#LDAT12                  ;WAS R0 INCREMENTED
        BEQ      1$                          ;CORRECTLY BY 4
        BR      LERR1

1$:     MOV      #-1,@#LDAT12
        MOV      #-1,@#LDAT13
        MOV      #LDAT10,R2
        MOV      #LDAT00,R3
        MOV      #4,R4

L5:     CMP      (R2)+,(R3)+
        BEQ      L6
        BR      LERR2
L6:     SOB      R4,L5
    
```

```

2748 013452 000473 BR LDONE
2749
2750 013454 LERR1: MOV #L2,@#STMP2 ;REPORT FAILURE
2751 013454 012737 013366 001236 MOV R0,@#STMP3 ;RO NOT INCREMENTED
2752 013462 010037 001240 MOV #LDAT12,@#STMP4 ;BY 4
2753 013466 012737 013624 001242 1$: ERROR 1
2754 013474 104001 BR LDONE
2755 013476 000461
2756
2757 013500 LERR3: MOV #L2,@#STMP2 ;REPORT DATA FAILURE.
2758 013500 012737 013366 001236 MOV #LDATIO,@#STMP3
2759 013506 012737 013620 001240 MOV #LDAT00,@#STMP4
2760 013514 012737 013632 001242 1$: ERROR 1
2761 013522 104001 BR LDONE
2762 013524 000446
2763
2764 013526 012702 013606 LERR2: MOV #LPAT20,R2 ;DID (BUT FD)
2765 013532 012703 013632 MOV #LDAT00,R3 ;FAIL.
2766 013536 012704 000004 MOV #4,R4
2767 013542 022223 1$: CMP (R2)+,(R3)+
2768 013544 001355 BNE LERR3
2769 013546 077403 SOB R4,1$
2770 013550 012737 013366 001236 MOV #L2,@#STMP2
2771 013556 012737 013620 001240 MOV #LDATIO,@#STMP3
2772 013564 012737 013634 001242 MOV #LDAT01,@#STMP4
    
```

2773 013572 104001
2774 013574 000422
2775
2776 013576 177777
2777 013600 177777
2778 013602 177777
2779 013604 177777
2780
2781 013606 052525
2782 013610 114631
2783 013612 063142
2784 013614 073567
2785 013616 000001
2786 013620 000000
2787 013622 000000
2788 013624 000000
2789 013626 000000
2790 013630 000001
2791 013632 000000
2792 013634 000000
2793 013636 000000
2794 013640 000000
2795
2796 013642
2797 013642 104413
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812 013644 000004
2813
2814 013646
2815 013646 170011
2816
2817 013650 012700 014142
2818 013654 172410
2819
2820 013656 005004
2821 013660 012737 014102 000004
2822
2823 013666 172427 000000
2824 013670 013670
2825 013670 005204
2826 013672 005204
2827 013674 005204
2828 013676 005204

2\$: ERROR 1
BR LDONE
LPAT10: .WORD -1
LPAT11: -1
LPAT12: -1
LPAT13: -1
LPAT20: 052525
LPAT21: 114631
LPAT22: 063142
LPAT23: 073567
.WORD 000001
LDAT10: 0
LDAT11: 0
LDAT12: 0
LDAT13: 0
.WORD 00001
LDAT00: 0
LDAT01: 0
LDAT02: 0
LDAT03: 0

LDONE: 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?).

:TEST 17 FSRC MODE 2 WITH GR7, IMMEDIATE MODE, TEST
:
:* THIS IS A TEST OF FSRC MODE 2
:* USING GR7 (THE PC). THIS IS IMMEDIATE
:* MODE.
:
:*****

TST17: SCOPE

M1: SETD

MOV #MPAT10,R0
LDD (R0),AC0 ;LOAD BACKGROUND
;PATTERN INTO AC0.

CLR R4
MOV #MERR3,@#ERRVECT

M15: LDD #0,AC0

:TEST INSTRUCTION
:EFFECTIVELY: 05204 IS PUT IN THE FIRST
:16 BIT WORD, OR THE 'EXP-FRACTION' WORD.

M2: INC R4
M3: INC R4
M4: INC R4

:NOTE THAT
:005204=INC R4


```

2885 014120 012737 013672 001242      MOV    #M2,@#STMP4      ;BAD CONSTANT
2886 014126 012737 013666 001236      MOV    #M15,@#STMP2    ;GENERATED
2887 014134 022626                CMP    (SP)+,(SP)+
2888 014136 104001                2$:   ERROR    1
2889 014140 000414                BR     MDONE
2890
2891 014142 177777                MPAT10:   -1
2892 014144 177777                MPAT11:   -1
2893 014146 177777                MPAT12:   -1
2894 014150 177777                MPAT13:   -1
2895
2896 014152 005204                MPAT20:   5204
2897 014154 005204                MPAT21:   5204
2898 014156 005204                MPAT22:   5204
2899 014160 005204                MPAT23:   5204
2900
2901 014162 000000                MDAT00:   0
2902 014164 000000                MDAT01:   0
2903 014166 000000                MDAT02:   0
2904 014170 000000                MDAT03:   0
2905
2906 014172                MDONE:
2907 014172 104413                RSETUP    ;GO INITIALIZE THE FPS AND STACK; AND
2908                                     ;SEE IF THE USER HAS EXPRESSED
2909                                     ;THE DESIRE TO CHANGE THE SOFTWARE
2910                                     ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
2911                                     ;THE USER TYPED CONTROL G?).
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921 014174 000004                TST20:   SCOPE
2922
2923
2924 014176 170011                N1:      SETD                ;SET FD MODE
2925
2926 014200 012700 014660                MOV    #NPAT10,R0
2927 014204 172410                LDD    (R0),AC0          ;LOAD AC0 WITH A DEFAULT
2928                                     ;PATTERN
2929 014206 012700 014646                MOV    #NPAT20,R0
2930 014212 005003                CLR    R3
2931 014214 012737 014370 000004                MOV    #NERRO,@#ERRVECT ;IF A FAILURE OCCURS
2932                                     ;IN THE FSRC FLOWS AN
2933                                     ;ODD TRAP TO 4 COULD OCCUR
2934                                     ;TEST INSTRUCTION.
2934 014222 172430                N2:      LDD    @(R0)+,AC0
2935 014224 005203                N3:      INC    R3
2936 014226 005203                N4:      INC    R3
2937
2938 014230 012701 014626                MOV    #NDAT00,R1
2939 014234 174011                STD    AC0,(R1)          ;GET THE DATA
2940

```

```

2941 014236 020027 014650      CMP      R0,#NPAT20+2      ;WAS R0 INCREMENTED
2942 014242 001437              BEQ      N12                ;BY 2?
2943
2944 014244 020027 014656      N5:     CMP      R0,#NPAT20+10 ;FSRC MODE 2?
2945 014250 001001              BNE     N6
2946 014252 000506              BR      NERR1
2947
2948 014254 020027 014636      N6:     CMP      R0,#NPAT20-10 ;FSRC MODE 4?
2949 014260 001001              BNE     N7
2950 014262 000520              BR      NERR2
2951
2952 014264 020027 014646      N7:     CMP      R0,#NPAT20
2953 014270 001023              BNE     N11
2954
2955 014272 012702 014626      MOV     #NDAT00,R2        ;FSRC MODE 0?
2956 014276 012703 000004      MOV     #4,R3
2957 014302 022227 177777      N8:     CMP     (R2)+,#-1
2958 014306 001002              BNE     N9
2959 014310 077304              SOB    R3,N8
2960 014312 000510              BR      NERR3
2961
2962 014314 012702 014626      N9:     MOV     #NDAT00,R2        ;FSRC MODE 1
2963 014320 012703 014646      MOV     #NPAT20,R3
2964 014324 012704 000004      MOV     #4,R4
2965 014330 022223              N10:    CMP     (R2)+,(R3)+
2966 014332 001002              BNE     N11
2967 014334 077403              SOB    R4,N10
2968 014336 000502              BR      NERR4
2969
2970 014340 000505      N11:    BR      NERR5
2971
2972 014342 012702 014626      N12:    MOV     #NDAT00,R2        ;DATA CORRECT?
2973 014346 012703 014670      MOV     #NDAT10,R3
2974 014352 012704 000004      MOV     #4,R4
2975 014356 022223      N13:    CMP     (R2)+,(R3)+
2976 014360 001002              BNE     N14
2977 014362 077403              SOB    R4,N13
2978 014364 000545              BR      NDONE
2979
2980 014366 000504      N14:    BR      NERR6
2981
2982      ;IF AN ODD ADDRESS TRAP OCCURS COME HERE
2983      ;TO SEE IF THE FAILURE WAS IN THE FSRC
2984      ;FLOWS
2985
2986 014370 022716 014226      NERR0:  CMP     #N4,(SP)        ;FSRC MODE 6 OR 7?
2987 014374 001412              BEQ     NERR10
2988 014376 022716 014224      CMP     #N3,(SP)
2989 014402 001402              BEQ     1$
2990 014404 000137 062556      JMP     @#CPSPUR
2991 014410 020027 014644      1$:    CMP     R0,#NPAT20-2      ;FSRC MODE 5?
2992 014414 001407              BEQ     NERR11
2993 014416 000137 062556      JMP     @#CPSPUR
2994
2995 014422      NERR10:
2996 014422 011637 001236      MOV     (SP),@#STMP2      ;WENT TO FSRC
                                ;MODE 6 OR 7.
    
```



```

2997 014426 022626
2998 014430 104001
2999 014432 000522
3000
3001 014434 011637 001236 NERR11: MOV (SP),@#STMP2 ;WENT TO FSRC
3002 014440 022626 CMP (SP)+,(SP)+ ;MODE 5.
3003 014442 012737 000627 001244 MOV #627,@#STMP5
3004 014450 012737 000323 001250 MOV #323,@#STMP7
3005 014456 012737 000325 001246 MOV #325,@#STMP6
3006 014464 104001 1$: ERROR 1
3007 014466 000504 BR NDONE
3008 014470 012737 000322 001246 NERR1: MOV #322,@#STMP6 ;FSRC MODE 2.
3009 014476 012737 000627 001244 NERR20: MOV #627,@#STMP5
3010 014504 012737 000323 001250 MOV #323,@#STMP7
3011 014512 012737 014222 001236 MOV #N2,@#STMP2
3012 014520 104001 1$: ERROR 1
3013 014522 000466 BR NDONE
3014 014524 012737 000324 001246 NERR2: MOV #324,@#STMP6 ;FSRC MODE 4
3015 014532 000761 BR NERR20
3016 014534 012737 000320 001246 NERR3: MOV #320,@#STMP6 ;FSRC MODE 0
3017 014542 000755 BR NERR20
3018 014544 012737 000321 001246 NERR4: MOV #321,@#STMP6 ;FSRC MODE 1
3019 014552 000751 BR NERR20
3020
3021 014554 010037 001240 NERR5: MOV R0,@#STMP3 ;R0 NOT
3022 014560 012737 014650 001242 MOV #NPAT20+2,@#STMP4 ;INCREMENTED
3023 014566 012737 014222 001236 MOV #N2,@#STMP2 ;PROPERLY.
3024 014574 104001 1$: ERROR 1
3025 014576 000440 BR NDONE
3026
3027 014600 NERR6: MOV #N2,@#STMP2 ;DATA FAILURE.
3028 014600 012737 014222 001236 MOV #NDAT00,@#STMP3
3029 014606 012737 014626 001240 MOV #NDAT10,@#STMP4
3030 014614 012737 014670 001242 1$: ERROR 1
3031 014622 104001 BR NDONE
3032 014624 000425
3033
3034 014626 000000 NDAT00: .WORD 0
3035 014630 000000 NDAT01: 0
3036 014632 000000 NDAT02: 0
3037 014634 000000 NDAT03: 0
3038
3039 014636 052525 052525 052525 .WORD 52525,52525,52525,52525
3040 014644 052525
3041 014646 014670 NPAT20: .WORD NDAT10
3042 014650 070707 NPAT21: 070707
3043 014652 070707 NPAT22: 070707
3044 014654 070707 NPAT23: 070707
3045 014656 000001 .WORD 1
3046 014660 177777 NPAT10: .WORD -1
3047 014662 177777 NPAT11: -1
3048 014664 177777 NPAT12: -1
3049 014666 177777 NPAT13: -1
3050
3051 014670 010421 NDAT10: .WORD 010421
3052 014672 021042 NDAT11: 021042
    
```

3053 014674 031463
 3054 014676 042104
 3055
 3056 014700
 3057 014700 104413
 3058
 3059
 3060
 3061
 3062
 3063
 3064
 3065
 3066
 3067
 3068
 3069
 3070
 3071
 3072 014702 000004
 3073
 3074 014704
 3075 014704 170011
 3076
 3077 014706 012700 015364
 3078 014712 172410
 3079
 3080 014714 012700 015352
 3081 014720 005003
 3082 014722 012737 015074 000004
 3083
 3084
 3085
 3086 014730 172450
 3087 014732 005203
 3088 014734 005203
 3089
 3090 014736 012701 015332
 3091 014742 174011
 3092
 3093 014744 020027 015350
 3094 014750 001436
 3095
 3096 014752 020027 015362
 3097 014756 001001
 3098 014760 000505
 3099
 3100 014762 020027 015342
 3101 014766 001001
 3102 014770 000517
 3103
 3104 014772 020027 015352
 3105
 3106 014776 012702 015334
 3107 015002 012703 000004
 3108 015006 022227 177777

NDAT12: 031463
 NDAT13: 042104

NDONE: 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?).

 ;*TEST 21 FSRC MODE 5 TEST
 ;*
 ;* THIS IS A TEST OF FSRC MODE 5, AUTO DECREMENT
 ;* DEFERRED.
 ;*

 ;TST21: SCOPE

01: SETD ;SET FD MODE
 MOV #OPAT10,R0
 LDD (R0),AC0 ;LOAD ACO WITH A
 ;DEFAULT PATTERN.
 MOV #OPAT21,R0
 CLR R3
 MOV #OERR0,@#ERRVEC ;IF A FAILURE
 ;OCCURS IN THE FSRC
 ;FLOWS AN ODD ADDR.
 ;TRAP TO 4 MAY OCCUR.
 ;TEST INSTRUCTION
 02: LDD @-(R0),AC0
 03: INC R3
 04: INC R3
 MOV #ODAT00,R1
 STD ACO,(R1) ;GET THE DATA
 CMP R0,#OPAT20 ;WAS R0 DECREMENTED
 BEQ 012 ;BY 2?
 05: CMP R0,#OPAT21+10 ;FSRC MODE 2
 BNE 06
 BR OERR1
 06: CMP R0,#OPAT21-10 ;FSRC MODE 4?
 BNE 07
 BR OERR2
 07: CMP R0,#OPAT21
 MOV #ODAT01,R2 ;FSRC MODE 0?
 MOV #4,R3
 08: CMP (R2)+,#-1

```

3109 015012 001002      BNE      09
3110 015014 077304      SOB      R3,08
3111 015016 000510      BR       OERR3
3112
3113 015020 012702 015332      09:     MOV      #ODAT00,R2      ;FSRC MODE 1?
3114 015024 012703 015352      MOV      #OPAT21,R3
3115 015030 012704 000004      MOV      #4,R4
3116 015034 022223      010:    CMP      (R2)+,(R3)+
3117 015036 001002      BNE      011
3118 015040 077403      SOB      R4,010
3119 015042 000502      BR       OERR4
3120
3121 015044 000505      011:    BR       OERR5
3122
3123 015046 012702 015332      012:    MOV      #ODAT00,R2      ;DATA CORRECT?
3124 015052 012703 015374      MOV      #ODAT10,R3
3125 015056 012704 000004      MOV      #4,R4
3126 015062 022223      013:    CMP      (R2)+,(R3)+
3127 015064 001002      BNE      014
3128 015066 077403      SOB      R4,013
3129 015070 000545      BR       ODONE
3130
3131 015072 000504      014:    BR       OERR6
3132
3133      ;IF AN ODD ADDRESS TRAP OCCURS COME
3134      ;HERE TO SEE IF THE FAILURE WAS IN THE
3135      ;FSRC FLOWS:
3136
3137 015074 022716 014734      OERR0:  CMP      #04,(SP)      ;FSRC MODE 6 OR 7?
3138 015100 001412      BEQ      OERR10
3139 015102 022716 014732      CMP      #03,(SP)
3140 015106 001402      BEQ      1$
3141 015110 000137 062556      JMP      @#CPSPUR
3142 015114 020027 015354      1$:     CMP      R0,#OPAT21+2      ;FSRC MODE 3?
3143 015120 001425      BEQ      OERR1
3144 015122 000137 062556      JMP      @#CPSPUR
3145
3146 015126      OERR10: MOV      (SP),@#STMP2      ;WENT TO FSRC
3147 015126 011637 001236      CMP      (SP)+,(SP)+      ;MODE 6 OR 7
3148 015132 022626      1$:     ERROR   1
3149 015134 104001      BR       ODONE
3150 015136 000522
3151
3152 015140 011637 001240      OERR11: MOV      (SP),@#STMP3      ;WENT TO FSRC MODE
3153 015144 022626      CMP      (SP)+,(SP)+      ;3
3154 015146 012737 000627 001244      MOV      #627,@#STMP5
3155 015154 012737 000325 001250      MOV      #325,@#STMP7
3156 015162 012737 000323 001246      MOV      #323,@#STMP6
3157 015170 104001      1$:     ERROR   1
3158 015172 000504      BR       ODONE
3159
3160 015174 012737 000322 001246      OERR1:  MOV      #322,@#STMP6      ;FSRC MODE2
3161 015202 012737 000627 001242      OERR20: MOV      #627,@#STMP4
3162 015210 012737 000325 001250      MOV      #325,@#STMP7
3163 015216 012737 014730 001236      MOV      #02,@#STMP2
3164 015224 104001      1$:     ERROR   1
    
```

```

3165 015226 000466 BR ODONE
3166 015230 012737 000324 001246 OERR2: MOV #324,@#STMP6 ;FSRC MODE 4
3167 015236 000761 BR OERR20
3168 015240 012737 000320 001246 OERR3: MOV #320,@#STMP6 ;FSRC MODE 0
3169 015246 000755 BR OERR20
3170 015250 012737 000321 001246 OERR4: MOV #321,@#STMP6 ;FSRC MODE 1
3171 015256 000751 BR OERR20
3172
3173
3174 015260 010037 001240 OERR5: MOV R0,@#STMP3 ;R0 NOT DECREMENTED
3175 015264 012737 015350 001242 MOV #OPAT20,@#STMP4 ;PROPERLY
3176 015272 012737 014734 001236 MOV #04,@#STMP2
3177 015300 104001 1$: ERROR 1
3178 015302 000440 BR ODONE
3179
3180 015304 OERR6: ;DATA FAILURE
3181 015304 012737 014730 001236 MOV #02,@#STMP2
3182 015312 012737 015332 001240 MOV #ODAT00,@#STMP3
3183 015320 012737 015374 001242 MOV #ODATIO,@#STMP4
3184 015326 104001 1$: ERROR 1
3185 015330 000425 BR ODONE
3186
3187 015332 000000 ODAT00: .WORD 0
3188 015334 000000 ODAT01: 0
3189 015336 000000 ODAT02: 0
3190 015340 000000 ODAT03: 0
3191
3192 015342 052525 052525 052525 .WORD 52525,52525,52525
3193 015350 015374 OPAT20: .WORD ODATIO
3194 015352 070707 OPAT21: 070707
3195 015354 070707 OPAT22: 070707
3196 015356 070707 OPAT23: 070707
3197 015360 070707 OPAT24: 070707
3198 015362 000001 .WORD 1
3199 015364 177777 OPAT10: .WORD -1
3200 015366 177777 OPAT11: -1
3201 015370 177777 OPAT12: -1
3202 015372 177777 OPAT13: -1
3203
3204 015374 073567 ODATIO: .WORD 73567
3205 015376 004210 ODATI1: 004210
3206 015400 114631 ODATI2: 114631
3207 015402 125252 ODATI3: 125252
3208
3209 015404 ODONE:
3210 015404 104413 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?).
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
:*****
:*TEST 22 FSRC MODE 6 TEST
:*
:* THIS IS A TEST OF FSRC MODE 6, INDEX MODE
    
```

```

3221          :*
3222          :*****
3223 015406 000004  TST22: SCOPE
3224
3225          P1:
3226 015410 170011          SETD          ;SET FD MODE
3227
3228 015412 012700 016030  MOV    #PPAT10,R0
3229 015416 172410          LDD    (R0),AC0      ;LOAD A DEFAULT PATTERN
3230          ;INTO AC0
3231 015420 012737 015526 000004  MOV    #PERR0,@#ERRVECT ;IF THE (BUT FSRC) FORQ
3232          ;FAILS AN ODD ADDRESS TRAP
3233 015426 012700 015577          MOV    #PDAT10-241,R0 ;COULD OCCUR.
3234
3235 015432 172460 000241  P2:    LDD    241(R0),AC0
3236          P3=P2+2
3237
3238 015436 012701 016050  P4:    MOV    #PDAT00,R1
3239 015442 174011          STD    AC0,(R1)      ;GET THE DATA
3240 015444 012703 000004          MOV    #4,R3
3241 015450 012702 016040          MOV    #PDAT10,R2
3242 015454 012701 016050          MOV    #PDAT00,R1
3243 015460 022221          P5:    CMP    (R2)+,(R1)+ ;CHECK THE DATA
3244 015462 001007          BNE   P6
3245 015464 077303          SOB   R3,P5
3246 015466 022700 015577          CMP    #PDAT10-241,R0 ;RO CORRECT?
3247 015472 001401          BEQ   1$
3248 015474 000512          BR    PERR21
3249 015476 000137 016060  1$:    JMP    @#PDONE
3250
3251 015502 012701 016050  P6:    MOV    #PDAT00,R1
3252 015506 012703 000004          MOV    #4,R3
3253 015512 022721 177777          P7:    CMP    #-1,(R1)+ ;WAS IT FSRC MODE 0?
3254 015516 001401          BEQ   P8
3255 015520 000512          BR    PERR1
3256 015522 077305          P8:    SOB   R3,P7
3257 015524 000523          BR    PERR2
3258          ;TRAP TO HERE ON AN ODD ADDRESS
3259 015526 021627 015434  PERR0: CMP    (SP),#P3
3260 015532 001411          BEQ   PERR11
3261 015534 021627 015436          CMP    (SP),#P4      ;WAS IT FSRC MODE 7?
3262 015540 001402          BEQ   PERR10
3263 015542 000137 062556          JMP    @#CPSPUR
3264
3265 015546 012737 000327 001246  PERR10: MOV    #327,@#STMP6
3266 015554 000443          BR    PERR17
3267 015556 022700 015577          PERR11: CMP    #PDAT10-241,R0 ;WAS IT FSRC MODE 1
3268 015562 001004          BNE   PERR12
3269 015564 012737 000321 001246          MOV    #321,@#STMP6
3270 015572 000434          BR    PERR17
3271 015574 022700 015607          PERR12: CMP    #PDAT10-241+10,R0 ;WAS IT FSRC MODE 2
3272 015600 001004          BNE   PERR13
3273 015602 012737 000322 001246          MOV    #322,@#STMP6
3274 015610 000425          BR    PERR17
3275 015612 022700 015601          PERR13: CMP    #PDAT10-241+2,R0 ;WAS IT FSRC MODE 3
3276 015616 001004          BNE   PERR14
    
```

```

3277 015620 012737 000323 001246      MOV      #323,@#STMP6
3278 015626 000416                      BR      PERR17
3279 015630 022700 015567      PERR14: CMP      #PDAT10-241-10,R0      ;WAS IT FSRC MODE 4
3280 015634 001004                      BNE     PERR15
3281 015636 012737 000324 001246      MOV      #324,@#STMP6
3282 015644 000407                      BR      PERR17
3283 015646 022700 015575      PERR15: CMP      #PDAT10-241-2,R0      ;WAS IT FSRC MODE 5
3284 015652 001401                      BEQ     PERR16
3285 015654 000416                      BR      PERR20
3286 015656 012737 000325 001246      PERR16: MOV      #325,@#STMP6
3287
3288 015664 012737 000627 001244      PERR17: MOV      #627,@#STMP5      ;REPORT FSRC
3289 015672 012737 000326 001250      MOV      #326,@#STMP7      ;FLOWS FAILURE.
3290 015700 011637 001236      MOV      (SP),@#STMP2
3291 015704 022626                      CMP     (SP)+,(SP)+
3292 015706 104001      1$:      ERROR    1
3293 015710 000463                      BR      PDONE
3294
3295 015712 011637 001236      PERR20: MOV      (SP),@#STMP2      ;REPORT R0 AFFECTED
3296 015716 022626                      CMP     (SP)+,(SP)+
3297 015720 000403                      BR      PERR22
3298 015722 012737 015432 001236      PERR21: MOV      #P2,@#STMP2
3299 015730                      PERR22:
3300 015730 010037 001240      MOV      R0,@#STMP3
3301 015734 012737 015577 001242      MOV      #PDAT10-241,@#STMP4
3302 015742 104001      1$:      ERROR    1
3303 015744 000445                      BR      PDONE
3304
3305 015746                      PERR1:      ;DATA FAILURE.
3306 015746 012737 015432 001236      MOV      #P2,@#STMP2
3307 015754 012737 016040 001240      MOV      #PDAT10,@#STMP3
3308 015762 012737 016050 001242      MOV      #PDAT00,@#STMP4
3309 015770 104001      1$:      ERROR    1
3310 015772 000432                      BR      PDONE
3311
3312 015774                      PERR2:      ;FSRC FAILURE TO
3313 015774 012737 015432 001236      MOV      #P2,@#STMP2      ;MODE 0
3314 016002 012737 000627 001244      MOV      #627,@#STMP5
3315 016010 012737 000326 001250      MOV      #326,@#STMP7
3316 016016 012737 000320 001246      MOV      #320,@#STMP6
3317 016024 104001      1$:      ERROR    1
3318 016026 000414                      BR      PDONE
3319
3320 016030 177777      PPAT10: .WORD  -1
3321 016032 177777      PPAT11:         -1
3322 016034 177777      PPAT12:         -1
3323 016036 177777      PPAT13:         -1
3324
3325 016040 010421      PDAT10: .WORD  010421
3326 016042 031463      PDAT11:         031463
3327 016044 052525      PDAT12:         052525
3328 016046 073567      PDAT13:         073567
3329
3330 016050 000000      PDAT00: .WORD   0
3331 016052 000000      PDAT01:         0
3332 016054 000000      PDAT02:         0
    
```

3333 016056 000000
 3334
 3335 016060
 3336 016060 104413
 3337
 3338
 3339
 3340
 3341
 3342
 3343
 3344
 3345
 3346
 3347
 3348
 3349
 3350 016062 000004
 3351
 3352 016064
 3353 016064 170011
 3354
 3355 016066 012700 016520
 3356 016072 172410
 3357
 3358 016074 012737 016226 000004
 3359
 3360
 3361
 3362 016102 012700 016267
 3363
 3364 016106 172470 000241
 3365 016110
 3366
 3367 016112 012701 016540
 3368 016116 174011
 3369
 3370 016120 012703 000004
 3371 016124 012704 016540
 3372 016130 012705 016550
 3373 016134 022425
 3374 016136 001007
 3375 016140 077303
 3376
 3377 016142 022700 016267
 3378 016146 001401
 3379 016150 000514
 3380 016152 000137 016560
 3381
 3382 016156 012701 016540
 3383 016162 012703 000004
 3384 016166 022721 177777
 3385 016172 001002
 3386 016174 077304
 3387 016176 000513
 3388

PDAT03: 0

PDONE: 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?).

 :*TEST 23 FSRC MODE 7 TEST
 :*
 :* THIS IS A TEST OF FSRC MODE 7, INDEX
 :* DEFERRED MODE.
 :*

 TST23: SCOPE

Q1:

SETD

MOV #QPAT10,R0
 LDD (R0),AC0

:LOAD A DEFAULT
 :PATTERN INTO AC0
 :IF THE (BUT FSRC)
 :FORK FAILS AN
 :ODD ADR TRAP COULD
 :OCCUR

MOV #QERR0,@#ERRVECT

MOV #QPAT20-241,R0

Q2:
 Q3=Q2+2

LDD @241(R0),AC0

Q4:

MOV #QDAT00,R1
 STD AC0,(R1)

:GET THE DATA

Q5:

MOV #4,R3
 MOV #QDAT00,R4
 MOV #QDAT10,R5
 CMP (R4)+,(R5)+
 BNE Q6
 SOB R3,Q5

:CHECK THE DATA

CMP #QPAT20-241,R0 :CHECK R0.

BEQ 1\$
 BR QERR21
 JMP @#QDONE

1\$:

Q6:

MOV #QDAT00,R1
 MOV #4,R3

Q7:

CMP #-1,(R1)+
 BNE Q8
 SOB R3,Q7
 BR QERR2

:WAS IT FSRC MODE 0?

```

3389 016200 012701 016530      Q8:   MOV   #QPAT20,R1
3390 016204 012702 016540      MOV   #QDAT00,R2
3391 016210 012703 000004      MOV   #4,R3
3392 016214 022122                Q9:   CMP   (R1)+,(R2)+      ;WAS IT FSRC 6
3393 016216 001401                BEQ   Q10                ;OR DATA FAILURE
3394 016220 000524                BR    QERR1
3395 016222 077304                Q10:  SOB   R3,Q9
3396 016224 000504                BR    QERR3
3397
3398                ;TRAP TO HERE ON AN ODD ADR FAILURE
3399
3400 016226 021627 015434      QERR0: CMP   (SP),#P3
3401 016232 000137 062556      JMP   @#CPSPUR
3402
3403 016236 022700 016267      QERR11: CMP  #QPAT20-241,R0      ;WAS IT FSRC
3404 016242 001004                BNE  QERR12              ;MODE 1?
3405 016244 012737 000321 001246  MOV  #321,@#STMP6
3406 016252 000434                BR   QERR17
3407 016254 022700 016277      QERR12: CMP  #QPAT20-241+10,R0    ;WAS IT FSRC
3408 016260 001004                BNE  QERR13              ;MODE 2?
3409 016262 012737 000322 001246  MOV  #322,@#STMP6
3410 016270 000425                BR   QERR17
3411 016272 022700 016271      QERR13: CMP  #QPAT20-241+2,R0    ;WAS IT FSRC
3412 016276 001004                BNE  QERR14              ;MODE 3?
3413 016300 012737 000323 001246  MOV  #323,@#STMP6
3414 016306 000416                BR   QERR17
3415 016310 022700 016257      QERR14: CMP  #QPAT20-241-10,R0   ;WAS IT FSRC
3416 016314 001004                BNE  QERR15              ;MODE 4
3417 016316 012737 000324 001246  MOV  #324,@#STMP6
3418 016324 000407                BR   QERR17
3419
3420 016326 022700 016265      QERR15: CMP  #QPAT20-241-2,R0   ;WAS IT FSRC
3421 016332 001401                BEQ  QERR16              ;MODE 5
3422 016334 000416                BR   QERR20
3423
3424 016336 012737 000325 001246  QERR16: MOV  #325,@#STMP6
3425
3426 016344 012737 000627 001244  QERR17: MOV  #627,@#STMP5      ;REPORT FSRC FAILURE
3427 016352 012737 000327 001250  MOV  #327,@#STMP7
3428 016360 011637 001236      MOV  (SP),@#STMP2
3429 016364 022626                CMP  (SP)+,(SP)+
3430 016366 104001                1$:  ERROR 1
3431 016370 000473                BR   QDONE
3432
3433 016372 011637 001236      QERR20: MOV  (SP),@#STMP2      ;REPORT R0 AFFECTED.
3434 016376 022626                CMP  (SP)+,(SP)+
3435 016400 000403                BR   QERR22
3436 016402 012737 016106 001236  QERR21: MOV  #Q2,@#STMP2
3437 016410                QERR22:
3438 016410 010037 001240                MOV  R0,@#STMP3
3439 016414 012737 016267 001242  MOV  #QPAT20-241,@#STMP4
3440 016422 104001                1$:  ERROR 1
3441 016424 000455                BR   QDONE
3442
3443 016426 012737 000320 001246  QERR2: MOV  #320,@#STMP6      ;WENT TO FSRC
3444 016434 000403                BR   QERR4              ;MODE 0
    
```



```

3445 016436 012737 000326 001246 QERR3: MOV #326,@#STMP6 ;WENT TO FSRC
3446 ;MODE 6
3447 016444 012737 000627 001244 QERR4: MOV #627,@#STMP5
3448 016452 012737 000327 001250 MOV #327,@#STMP7
3449 016460 012737 016106 001236 MOV #Q2,@#STMP2
3450 016466 104001 1$: ERROR 1
3451 016470 000433 BR QDONE
3452
3453 016472 QERR1: ;DATA FAILURE
3454 016472 012737 016106 001236 MOV #Q2,@#STMP2
3455 016500 012737 016550 001240 MOV #QDAT10,@#STMP3
3456 016506 012737 016540 001242 MOV #QDAT00,@#STMP4
3457 016514 104001 1$: ERROR 1
3458 016516 000420 BR QDONE
3459
3460 016520 177777 QPAT10: .WORD -1
3461 016522 177777 QPAT11: -1
3462 016524 177777 QPAT12: -1
3463 016526 177777 QPAT13: -1
3464
3465 016530 016550 QPAT20: .WORD QDAT10
3466 016532 052525 QPAT21: 52525
3467 016534 052525 QPAT22: 52525
3468 016536 052525 QPAT23: 52525
3469
3470 016540 000000 QDAT00: .WORD 0
3471 016542 000000 QDAT01: 0
3472 016544 000000 QDAT02: 0
3473 016546 000000 QDAT03: 0
3474
3475 016550 073567 QDAT10: .WORD 073567
3476 016552 052525 QDAT11: .WORD 052525
3477 016554 031463 QDAT12: .WORD 031463
3478 016556 010421 QDAT13: .WORD 010421
3479
3480 016560 QDONE:
3481 016560 104413 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?).

```

```

*****
;*TEST 24 (BUT EZBT Y8),(BUT ENBT) AND (BUT FIUV) TEST
;*
;* LOAD INSTRUCTION FLOWS.
;* EACH OF THE PATTERNS:
;* 0
;* +NUM
;* -NUM
;* -0
;* IS LOADED TWICE, ONCE WITH AC>0 THEN
;* WITH AC=0. AFTER EACH LOAD THE FPS IS
;* CHECK TO INSURE THAT CONTROL WAS PASSED
;* THROUGH WITH THE FORKS PROPERLY.
;*

```

3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500

```

3501
3502 016562 000004
3503 016564 005037 017662
3504 016570 012700 017612
3505 016574 012701 000004
3506 016600 012720 177777
3507 016604 077103
3508
3509 016606 012737 000033 017664
3510 016614 012737 000023 017666
3511 016622 012737 017342 000244
3512 016630
3513 016630 104414
3514 016632 012700 000200
3515 016636 170100
3516 016640 012700 017612
3517 016644 172410
3518 016646 013737 017664 017670
3519 016654 012737 000001 017672
3520 016662 012737 000254 017674
3521
3522 016670 012700 017622
3523 016674 172410
3524 016676 010037 001252
3525 016702 012737 016674 001236
3526
3527 016710 012704 000204
3528 016714 170205
3529
3530 016716 020405
3531 016720 001402
3532 016722 000137 017366
3533
3534 016726
3535 016726 104414
3536 016730 012700 000200
3537 016734 170100
3538
3539 016736 012700 017612
3540 016742 172410
3541 016744 013737 017666 017670
3542 016752 012737 000003 017672
3543 016760 012737 000054 017674
3544
3545 016766 012700 017632
3546
3547 016772 172410
3548 016774 010037 001252
3549 017000 012737 016772 001236
3550 017006 012704 000200
3551 017012 170205
3552
3553 017014 020405
3554 017016 001402
3555 017020 000137 017452
3556 017024
    
```

```

*****
TST24: SCOPE
CLR @#UFLAG
MOV #UPAT00,R0 ;SET UP AC#0 DATA.
MOV #4,R1
U0: MOV #-1,(R0)+
SOB R1,U0
MOV #033,@#UTMP1
MOV #023,@#UTMP2
MOV #UERR0,@#FPVECT ;IN CASE (BUT FIUV FAILS)
U1: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
MOV #200,R0
LDFPS R0
MOV #UPAT00,R0 ;LOAD ACO
LDD (R0),AC0
MOV @#UTMP1,@#UROM1
MOV #001,@#UROM2
MOV #254,@#UROM3
U2: MOV #UPAT10,R0 ;LOAD 0 INTO ACO
LDD (R0),AC0
MOV R0,@#STMP10
MOV #U2,@#STMP2
MOV #204,R4 ;SEE IF FPS IS CORRECT
STFPS R5
CMP R4,R5
BEQ U3
JMP @#UERR1
U3: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
MOV #200,R0
LDFPS R0
MOV #UPAT00,R0 ;LOAD ACO
LDD (R0),AC0
MOV @#UTMP2,@#UROM1
MOV #003,@#UROM2
MOV #054,@#UROM3
U4: MOV #UPAT20,R0 ;LOAD A POSITIVE NUMBER
;INTO ACO
LDD (R0),AC0
MOV R0,@#STMP10
MOV #U4,@#STMP2
MOV #200,R4 ;FPS CORRECT?
STFPS R5
CMP R4,R5
BEQ U5
JMP @#UERR2
U5:
    
```

```

3557 017024 104414          LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
3558 017026 012700 000200  MOV      #200,R0
3559 017032 170100          LDFPS     R0
3560 017034 012700 017612  MOV      #UPAT00,R0 ;LOAD ACO
3561 017040 172410          LDD      (R0),AC0
3562 017042 013737 017666 017670  MOV      @#UTMP2,@#UROM1
3563 017050 012737 000403 017672  MOV      #403,@#UROM2
3564 017056 012737 000056 017674  MOV      #056,@#UROM3
3565 017064 012700 017642  MOV      #UPAT30,R0 ;LOAD A NEGATIVE
3566                                ;NUMBER INTO ALO
3567 017070 172410          U6:  LDD      (R0),AC0
3568 017072 010037 001252  MOV      R0,@#STMP10
3569 017076 012737 017070 001236  MOV      #U6,@#STMP2
3570 017104 012704 000210  MOV      #210,R4 ;FPS CORRECT
3571 017110 170205          STFPS    R5
3572 017112 020405          CMP      R4,R5
3573 017114 001402          BEQ     U7
3574 017116 000137 017452  JMP      @#UERR2
3575 017122                                U7:
3576 017122 104414          LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
3577 017124 012700 000200  MOV      #200,R0
3578 017130 170100          LDFPS     R0
3579 017132 012700 017612  MOV      #UPAT00,R0 ;LOAD ACO
3580 017136 172410          LDD      (R0),AC0
3581 017140 013737 017664 017670  MOV      @#UTMP1,@#UROM1
3582 017146 012737 000401 017672  MOV      #401,@#UROM2
3583 017154 012737 000256 017674  MOV      #256,@#UROM3
3584 017162 012700 017652  MOV      #UPAT40,R0 ;LOAD -0 INTO ACO
3585 017166 172410          U10:  LDD      (R0),AC0
3586 017170 000240          U11:  NOP
3587 017172 010037 001252  MOV      R0,@#STMP10 ;TRAP FROM HERE IF
3588 017176 012737 017166 001236  MOV      #U10,@#STMP2 ;(BUT FIUV) FAULTS!
3589 017204 012704 000214  MOV      #214,R4 ;SEE IF FPS IS CORRECT.
3590 017210 170205          STFPS    R5
3591 017212 020405          CMP      R4,R5
3592 017214 001402          BEQ     U12
3593 017216 000137 017366  JMP      @#UERR1
3594 017222 005737 017662  U12:  TST      @#UFLAG ;SEE IF ALL THE PATTERNS
3595 017226 001021          BNE     U14 ;HAVE BEEN TEST WITH
3596                                ;BOTH AC NOT EQUAL TO 0 AND AC=0
3597 017230 012700 017612  MOV      #UPAT00,R0 ;IF NOT GO BACK AND
3598 017234 012701 000004  MOV      #4,R1 ;CHECK THEM WITH AC=0
3599 017240 005020          U13:  CLR      (R0)+
3600 017242 077102          SOB     R1,U13
3601 017244 012737 177777 017662  MOV      #-1,@#UFLAG
3602 017252 012737 000233 017664  MOV      #233,@#UTMP1
3603 017260 012737 000223 017666  MOV      #223,@#UTMP2
3604 017266 000137 016630  JMP      @#U1
3605 017272                                U14:
3606 017272 104414          LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
3607                                ;NOW SEE IF A TRAP CAN BE FORCED BY SETTING FIUV AND LOADING -0
3608 017274 012737 017536 000244  MOV      #UERR3,@#FPVECT
3609 017302 012700 004200  MOV      #4200,R0 ;SET FD AND FIUV
3610 017306 170100          LDFPS     R0
3611 017310 012700 017612  MOV      #UPAT00,R0 ;SET UP ACO
3612 017314 172410          LDD      (R0),AC0
    
```

```

3613 017316 012700 017652
3614 017322 172410
3615 017324 170000
3616 017326 000240
3617 017330 012737 017322 001236
3618
3619 017336 104001
3620 017340 000556
3621
3622
3623 017342 021627 017170
3624 017346 001402
3625 017350 000137 062524
3626 017354 011637 001236
3627 017360 022626
3628 017362 104001
3629 017364 000544
3630
3631
3632
3633 017366 032705 000004
3634 017372 001432
3635 017374 012737 000443 001244
3636 017402 013703 017674
3637 017406 010337 001250
3638 017412 032703 000200
3639 017416 001403
3640 017420 042703 000200
3641 017424 000402
3642 017426 052703 000200
3643 017432 010337 001246
3644 017436 010537 001240
3645 017442 010437 001242
3646 017446 104001
3647 017450 000512
3648 017452 032705 000004
3649 017456 001746
3650 017460 013737 017670 001244
3651 017466 013703 017672
3652 017472 010337 001250
3653 017476 032703 000400
3654 017502 001403
3655 017504 042703 000400
3656 017510 000402
3657 017512 052703 000400
3658 017516 010337 001246
3659 017522 010537 001240
3660 017526 010437 001242
3661 017532 104001
3662 017534 000460
3663
3664
3665 017536 021627 017324
3666 017542 001402
3667 017544 000137 062524
3668 017550 022626

MOV #UPAT40,R0 ;LOAD -0
LDD (R0),AC0 ;SHOULD TRAP TO 244
CFCC
NOP
MOV #U15,@#STMP2 ;REPORT ERROR.
;DIDN'T TRAP
; (BUT FIUV) FAILED.
1$: ERROR 1
BR UDONE

;TRAPPED TO 244. DID (BUT FIUV) FAIL?
UERR0: CMP (SP),#U11
BEQ 1$
JMP @#FPSPUR
1$: MOV (SP),@#STMP2
CMP (SP)+,(SP)+
2$: ERROR 1
BR UDONE

;COME HERE TO ANALYZE FPS ERRORS
UERR1: BIT #4,R5
BEQ UERR20
UERR10: MOV #443,@#STMP5
MOV @#UROM3,R3
MOV R3,@#STMP7
BIT #200,R3
BEQ 1$
BIC #200,R3
BR 2$
1$: BIS #200,R3
2$: MOV R3,@#STMP6
UERR11: MOV R5,@#STMP3
MOV R4,@#STMP4
1$: ERROR 1
BR UDONE
UERR2: BIT #4,R5
BEQ UERR10
UERR20: MOV @#UROM1,@#STMP5
MOV @#UROM2,R3
MOV R3,@#STMP7
BIT #400,R3
BEQ 1$
BIC #400,R3
BR 2$
1$: BIS #400,R3
2$: MOV R3,@#STMP6
UERR21: MOV R5,@#STMP3
MOV R4,@#STMP4
1$: ERROR 1
BR UDONE

; INTERRUPT HERE WHEN FIUV SET AND ATTEMPTED TO LOAD-0
UERR3: CMP (SP),#U16
BEQ 1$
JMP @#FPSPUR
1$: CMP (SP)+,(SP)+
    
```

```

3669 017552 005000 CLR R0
3670 017554 170300 STST R0 ;GET FEC.
3671 017556 022700 000014 CMP #14,R0 ;CORRECT
3672 017562 001001 BNE UERR4
3673 017564 000444 BR UDONE
3674 017566 012737 017322 001236 UERR4: MOV #U15,@#STMP2
3675 017574 012737 000012 001242 MOV #12,@#STMP4
3676 017602 010037 001240 MOV R0,@#STMP3
3677 017606 104001 1$: ERROR 1
3678 017610 000432 BR UDONE
3679 017612 000000 UPAT00: .WORD 0
3680 017614 000000 UPAT01: 0
3681 017616 000000 UPAT02: 0
3682 017620 000000 UPAT03: 0
3683
3684 017622 000000 UPAT10: .WORD 0 ;0
3685 017624 000000 UPAT11: 0
3686 017626 000000 UPAT12: 0
3687 017630 000000 UPAT13: 0
3688
3689 017632 010421 UPAT20: .WORD 010421 ;POS NUM
3690 017634 114631 UPAT21: 114631
3691 017636 125252 UPAT22: 125252
3692 017640 177777 UPAT23: 177777
3693
3694 017642 114631 UPAT30: 114631 ;NEG NUM
3695 017644 135673 UPAT31: 135673
3696 017646 146314 UPAT32: 146314
3697 017650 167356 UPAT33: 167356
3698
3699 017652 100000 UPAT40: 100000 ;NEG ZERO
3700 017654 000000 UPAT41: 0
3701 017656 000000 UPAT42: 0
3702 017660 000000 UPAT43: 0
3703
3704 017662 000000 UFLAG: .WORD 0
3705 017664 000000 UTMP1: 0
3706 017666 000000 UTMP2: 0
3707 017670 000000 UROM1: 0
3708 017672 000000 UROM2: 0
3709 017674 000000 UROM3: 0
3710 017676 UDONE:
3711
3712
3713 ;*****
3714 ;*TEST 25 ADDF,ADDD,SUBF AND SUBD WITH FSRC=AC=0 TEST
3715
3716 ;*
3717 ;* THIS IS A TEST OF ADD AND SUB WITH FSRC=AC=0
3718 ;*
3719 ;*****
3720
3721 017676 000004 TST25: SCOPE
3722 017700 W1:
3723 017700 104414 LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
3724 017702 012700 000200 MOV #200,R0

```

3725	017706	170100			LDFPS	R0	:SET DOUBLE MODE
3726	017710	012700	020364		MOV	#WPAT00,R0	:LOAD AC0=:
3727	017714	172410			LDD	(R0),AC0	
3728	017716	012737	017730	001236	MOV	#W2,@#STMP2	:SAVE FP INSTRUCTION
3729	017724	012700	020364		MOV	#WPAT00,R0	
3730	017730	172010		W2:	ADDD	(R0),AC0	:TEST INSTRUCTION. ADD ITSELF
3731	017732	170205			STFPS	R5	:GET FPS
3732	017734	170011			SETD		:SET DOUBLE MODE
3733	017736	012700	020364		MOV	#WPAT00,R0	
3734	017742	174010			STD	AC0,(R0)	:GET THE RESULT
3735	017744	012701	020364		MOV	#WPAT00,R1	
3736	017750	012702	000004		MOV	#4,R2	
3737	017754	022021		W3:	CMP	(R0)+,(R1)+	:IS RESULT CORRECT
3738	017756	001403			BEQ	W4	:NO
3739							
3740	017760	104002		1\$:	ERROR	2	
3741	017762	000137	020404		JMP	@#WDONE	
3742	017766	077206		W4:	SOB	R2,W3	
3743	017770	022705	000204		CMP	#204,R5	:IS FPS CORRECT
3744	017774	001410			BEQ	W5	:NO
3745							
3746	017776	012737	000204	001242	MOV	#204,@#STMP4	
3747	020004	010537	001240		MOV	R5,@#STMP3	
3748	020010	104001		1\$:	ERROR	1	
3749	020012	000137	020404		JMP	@#WDONE	
3750	020016			W5:			
3751	020016	104414			LPERR		:SET UP THE LOOP ON ERROR ADDRESS.
3752	020020	012700	000200		MOV	#200,R0	
3753	020024	170100			LDFPS	R0	:SET DOUBLE MODE
3754	020026	012700	020364		MOV	#WPAT00,R0	:LOAD AC0=0
3755	020032	172410			LDD	(R0),AC0	
3756	020034	012737	020052	001236	MOV	#W6,@#STMP2	
3757	020042	005000			CLR	R0	
3758	020044	170100			LDFPS	R0	:GO TO FLOATING MODE
3759	020046	012700	020364		MOV	#WPAT00,R0	
3760	020052	172010		W6:	ADDF	(R0),AC0	:TEST INSTRUCTION
3761	020054	170205			STFPS	R5	:GET FPS
3762	020056	170011			SETD		:RESET TO DOUBLE MODE
3763	020060	012700	020364		MOV	#WPAT00,R0	
3764	020064	174010			STD	AC0,(R0)	:GET THE RESULT
3765	020066	012701	020364		MOV	#WPAT00,R1	
3766	020072	012702	000004		MOV	#4,R2	
3767	020076	022021		W7:	CMP	(R0)+,(R1)+	:WAS THE RESULT
3768	020100	001402			BEQ	W10	:NO. REPORT FAILURE.
3769	020102	104002		1\$:	ERROR	2	
3770	020104	000537			BR	WDONE	
3771	020106	077205		W10:	SOB	R2,W7	
3772	020110	022705	000004		CMP	#4,R5	:WAS FPS CORRECT
3773	020114	001407			BEQ	W11	:INCORRECT FPS.
3774							
3775	020116	012737	000004	001242	MOV	#4,@#STMP4	
3776	020124	010537	001240		MOV	R5,@#STMP3	
3777	020130	104002		1\$:	ERROR	2	
3778	020132	000524			BR	WDONE	
3779	020134			W11:			
3780	020134	104414			LPERR		:SET UP THE LOOP ON ERROR ADDRESS.

3781	020136	012700	000200		MOV	#200,R0	
3782	020142	170100			LDFPS	R0	;SET DOUBLE MODE
3783	020144	012700	020364		MOV	#WPAT00,R0	;LOAD AC0=0
3784	020150	172410			LDD	(R0),AC0	
3785	020152	012737	020164	001236	MOV	#W12,@#STMP2	
3786	020160	012700	020364		MOV	#WPAT00,R0	
3787	020164	173010		W12:	SUBD	(R0),AC0	;TEST INSTRUCTION
3788	020166	170205			STFPS	R5	;GET FPS
3789	020170	170011			SETD		;SET DOUBLE MODE
3790	020172	012700	020364		MOV	#WPAT00,R0	
3791	020176	174010			STD	AC0,(R0)	;GET THE RESULT
3792	020200	012701	020364		MOV	#WPAT00,R1	
3793	020204	012702	000004		MOV	#4,R2	
3794	020210	022021		W13:	CMP	(R0)+,(R1)+	;IS RESULT CORRECT?
3795	020212	001402			BEQ	W14	;NO.
3796							
3797	020214	104002		1\$:	ERROR	2	
3798	020216	000472			BR	WDONE	
3799	020220	077205		W14:	SOB	R2,W13	
3800	020222	022705	000204		CMP	#204,R5	;IS FPS CORRECT?
3801	020226	001407			BEQ	W15	;NO.
3802							
3803	020230	012737	000204	001242	MOV	#204,@#STMP4	
3804	020236	010537	001240		MOV	R5,@#STMP3	
3805	020242	104001		1\$:	ERROR	1	
3806	020244	000457			BR	WDONE	
3807	020246			W15:			
3808	020246	104414			LPERR		;SET UP THE LOOP ON ERROR ADDRESS.
3809	020250	012700	000200		MOV	#200,R0	
3810	020254	170100			LDFPS	R0	;SET DOUBLE MODE
3811	020256	012700	020364		MOV	#WPAT00,R0	;LOAD AC0=0
3812	020262	172410			LDD	(R0),AC0	
3813	020264	012737	020302	001236	MOV	#W16,@#STMP2	
3814	020272	005000			CLR	R0	
3815	020274	170100			LDFPS	R0	;ENTER FLOATING MODE.
3816	020276	012700	020364		MOV	#WPAT00,R0	
3817	020302	173010		W16:	SUBF	(R0),AC0	;TEST INSTRUCTION.
3818	020304	170205			STFPS	R5	;GET FPS

```

3819 020306 170011          SETD          ;RESET TO DOUBLE MODE
3820 020310 012700 020364  MOV #WPAT00,R0 ;GET THE RESULT.
3821 020314 174010          STD AC0,(R0)
3822 020316 012701 020364  MOV #WPAT00,R1
3823 020322 012702 000004  MOV #4,R2
3824 020326 022021          W17:  CMP (R0)+,(R1)+ ;IS RESULT CORRECT?
3825 020330 001402          BEQ W20          ;NO.
3826
3827 020332 104002          1$:  ERROR 2
3828 020334 000423          BR WDONE
3829 020336 077205          W20:  SOB R2,W17
3830 020340 022705 000004  CMP #4,R5      ;IS FPS CORRECT?
3831 020344 001417          BEQ WDONE
3832
3833 020346 012737 000004 001242  MOV #4,@#STMP4 ;NO
3834 020354 010537 001240  MOV R5,@#STMP3
3835 020360 104001          1$:  ERROR 1
3836 020362 000410          BR WDONE
3837
3838 020364 000000          WPAT00: .WORD 0
3839 020366 000000          WPAT01:      0
3840 020370 000000          WPAT02:      0
3841 020372 000000          WPAT03:      0
3842
3843 020374 000000          WDAPO0: .WORD 0
3844 020376 000000          WDAT01:      0
3845 020400 000000          WDAT02:      0
3846 020402 000000          WDAT03:      0
3847
3848 020404          WDONE:
3849 020404 104413          RSETUP      ;GO INITIALIZE THE FPS AND STACK; AND
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3860
3861
3862
3863 020406 000004          ;*****
3864 020410          ;*TEST 26      ADDD AND SUB WITH FSRC=0
3865 020410 104414          ;*
3866 020412 012700 000200  ;* THIS IS A TEST OF ADDD AND SUB WITH FSRC=0.
3867 020416 170100          ;*
3868 020420 012700 021150  ;*****
3869 020424 010037 021136  TST26: SCOPE
3870 020430 172410          X1:
3871 020432 012737 020444 001236  LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
3872 020440 012700 021160  MOV #200,R0    ;SET DOUBLE MODE
3873 020444 172010          LDFPS R0      ;SET AC0 TO POSITIVE
3874 020446 170205          MOV #XPAT00,R0 ;NUMBER #0
3875
3876
3877
3878
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3999
4000
    
```



```

3875 020450 170011          SETD
3876 020452 012700 021140  MOV    #XDAT00,R0      ;GET RESULT.
3877 020456 174010          STD    ACO,(R0)
3878 020460 012701 021150  MOV    #XPAT00,R1
3879 020464 012702 000004  MOV    #4,R2
3880 020470 022021          X3:   CMP    (R0)+,(R1)+   ;IS RESULT CORRECT?
3881 020472 001401          BEQ    X4
3882 020474 000553          BR     XERR1
3883 020476 077204          X4:   SOB   R2,X3
3884 020500 012704 000200  MOV    #200,R4
3885 020504 020405          CMP    R4,R5          ;IS FPS CORRECT?
3886 020506 001402          BEQ    X5
3887 020510 000137 021106  JMP    @XERR2
3888 020514          X5:   LPERR
3889 020514 104414          ;SET UP THE LOOP ON ERROR ADDRESS.
3890 020516 012700 000200  MOV    #200,R0
3891 020522 170100          LDFPS R0              ;SET DOUBLE MODE
3892 020524 012700 021170  MOV    #XPAT20,R0     ;SET ACO TO
3893 020530 010037 021136  MOV    R0,@XTMP       ;NEGATIVE NUMBER
3894 020534 172410          LDD    (R0),ACO
3895 020536 012737 020550 001236  MOV    #X6,@STMP2
3896 020544 012700 021160  MOV    #XPAT10,R0
3897 020550 172010          X6:   ADDD (R0),ACO
3898 020552 170205          STFPS R5              ;FSRC=0
3899 020554 170011          SETD                    ;TEST INSTRUCTION
3900 020556 012700 021140  MOV    #XDAT00,R0     ;GET RESULT
3901 020562 174010          STD    ACO,(R0)
3902 020564 012701 021170  MOV    #XPAT20,R1
3903 020570 012702 000004  MOV    #4,R2
3904 020574 022021          X7:   CMP    (R0)+,(R1)+   ;IS RESULT CORRECT?
3905 020576 001401          BEQ    X10
3906 020600 000511          BR     XERR1
3907 020602 077204          X10:  SOB   R2,X7
3908 020604 012704 000210  MOV    #210,R4
3909 020610 020405          CMP    R4,R5          ;IS FPS CORRECT?
3910 020612 001401          BEQ    X11
3911 020614 000534          BR     XERR2
3912 020616          X11:  LPERR
3913 020616 104414          ;SET UP THE LOOP ON ERROR ADDRESS.
3914 020620 012700 000200  MOV    #200,R0
3915 020624 170100          LDFPS R0              ;SET DOUBLE MODE
3916 020626 012700 021150  MOV    #XPAT00,R0     ;SET ACO TO NON-ZERO
3917 020632 010037 021136  MOV    R0,@XTMP       ;POSITIVE NUMBER
3918 020636 172410          LDD    (R0),ACO
3919 020640 012737 020652 001236  MOV    #X12,@STMP2
3920 020646 012700 021160  MOV    #XPAT10,R0
3921 020652 173010          X12:  SUBD (R0),ACO
3922 020654 170205          STFPS R5              ;FSRC=0
3923 020656 170011          SETD                    ;TEST INSTRUCTION
3924 020660 012700 021140  MOV    #XDAT00,R0     ;GET RESULT
3925 020664 174010          STD    ACO,(R0)
3926 020666 012701 021150  MOV    #XPAT00,R1
3927 020672 012702 000004  MOV    #4,R2
3928 020676 022021          X13:  CMP    (R0)+,(R1)+   ;IS RESULT CORRECT?
3929 020700 001401          BEQ    X14
3930 020702 000463          BR     XERR3
    
```

```

3931 020704 077204          X14: SOB      R2,X13
3932 020706 012704 000200   MOV      #200,R4      ;IS FPS CORRECT?
3933 020712 020405          CMP      R4,R5
3934 020714 001401          BEQ      X15
3935 020716 000501          BR       XERR4
3936 020720          X15: LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
3937 020720 104414          MOV      #200,R0
3938 020722 012700 000200   LDFPS   R0            ;SET DOUBLE MODE
3939 020726 170100          MOV      #XPAT20,R0  ;SET ACO=A NEGATIVE
3940 020730 012700 021170   MOV      R0,@#XTMP   ;NUMBER
3941 020734 010037 021136   LDD      (R0),ACO
3942 020740 172410          MOV      #X16,@#STMP2
3943 020742 012737 020754 001236 MOV      #XPAT10,R0  ;FSRC=0
3944 020750 012700 021160   MOV      (R0),ACO    ;TEST INSTRUCTION.
3945 020754 173010          SUBD     R5
3946 020756 170205          STFPS   R5
3947 020760 170011          SETD
3948 020762 012700 021140   MOV      #XDAT00,R0  ;GET RESULT
3949 020766 174010          STD     ACO,(R0)
3950 020770 012701 021170   MOV      #XPAT20,R1
3951 020774 012702 000004   MOV      #4,R2
3952 021000 022021          X17: CMP      (R0)+,(R1)+ ;IS RESULT CORRECT?
3953 021002 001401          BEQ     X20
3954 021004 000422          BR      XERR3
3955 021006 077204          X20: SOB      R2,X17
3956 021010 012704 000210   MOV      #210,R4      ;IS FPS CORRECT?
3957 021014 020405          CMP      R4,R5
3958 021016 001401          BEQ     X21
3959 021020 000440          BR      XERR4
3960 021022 000466          X21: BR       XDONE
3961
3962          ;REPORT DATA ERRORS
3963
3964 021024 012737 021160 001240 XERR1: MOV      #XPAT10,@#STMP3
3965 021032 013737 021136 001242   MOV      @#XTMP,@#STMP4
3966 021040 012737 021140 001244   MOV      #XDAT00,@#STMP5
3967 021046 104001          1$:  ERROR    1
3968 021050 000453          BR      XDONE
3969 021052 012737 021160 001240 XERR3: MOV      #XPAT10,@#STMP3
3970 021060 013737 021136 001242   MOV      @#XTMP,@#STMP4
3971 021066 012737 021140 001244   MOV      #XDAT00,@#STMP5
3972 021074 013737 021136 001246   MOV      @#XTMP,@#STMP6
3973 021102 104002          1$:  ERROR    2
3974 021104 000435          BR      XDONE
3975
3976          ;REPORT FPS ERRORS
3977
3978 021106          XERR2:
3979 021106 010537 001240   MOV      R5,@#STMP3
3980 021112 010437 001242   MOV      R4,@#STMP4
3981 021116 104001          1$:  ERROR    1
3982 021120 000427          BR      XDONE
3983 021122          XERR4:
3984 021122 010537 001240   MOV      R5,@#STMP3
3985 021126 010437 001242   MOV      R4,@#STMP4
3986 021132 104001          1$:  ERROR    1
    
```

```

3987 021134 000421
3988 021136 000000
3989 021140 000000
3990 021142 000000
3991 021144 000000
3992 021146 000000
3993
3994 021150 010421
3995 021152 021042
3996 021154 031463
3997 021156 042104
3998
3999 021160 000000
4000 021162 000000
4001 021164 000000
4002 021166 000000
4003 021170 104210
4004 021172 114631
4005 021174 125252
4006 021176 135673
4007
4008 021200
4009 021200 104413
4010
4011
4012
4013
4014
4015
4016
4017
4018
4019
4020
4021
4022
4023 021202 000004
4024 021204 005037 021534
4025 021210 012737 021554 021536
4026 021216 012737 021564 021540
4027 021224 012737 000210 021542
4028 021232
4029 021232 104414
4030 021234 012700 000200
4031 021240 170100
4032 021242 012700 021574
4033 021246 172410
4034 021250 013700 021536
4035 021254 173010
4036 021256 170205
4037 021260 170011
4038 021262 012700 021544
4039 021266 174010
4040 021270 012702 000004
4041 021274 013701 021540
4042 021300 022021

BR XDONE
XTMP: .WORD 0
XDAT00: .WORD 0
XDAT01: 0
XDAT02: 0
XDAT03: 0
XPAT00: .WORD 010421
XPAT01: 021042
XPAT02: 031463
XPAT03: 042104
XPAT10: .WORD 0
XAPT11: 0
XPAT12: 0
XPAT13: 0
XPAT20: .WORD 104210
XPAT21: 114631
XPAT22: 125252
XPAT23: 135673
XDONE: 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?).

:*****
:*TEST 27 SUBD WITH AC=0 TEST
:*
:* THIS IS A TEST OF SUBD WITH AC=0. BOTH POSITIVE
:* AND NEGATIVE FSRC'S ARE TRIED.
:*
:*****
TST27: SCOPE
CLR @#YFLAG
MOV #YPAT00,@#YTMP1 :P
MOV #YPAT10,@#YTMP2 :N
MOV #210,@#YTMP3
Y1: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
MOV #200,R0
LDFPS R0 ;SET DOUBLE MODE
MOV #YPAT20,R0 ;SET ACO=0
LDD (R0),AC0
MOV @#YTMP1,R0
Y2: SUBD (R0),AC0 ;TEST INSTRUCTION
STFPS R5
SETD
MOV #YDAT00,R0 ;GET RESULT
STD ACO,(R0)
MOV #4,R2
MOV @#YTMP2,R1 ;CHECK RESULT.
Y3: CMP (R0)+,(R1)+
    
```

4043	021302	001026			BNE	Y6	
4044	021304	077203			SOB	R2,Y3	
4045	021306	023705	021542		CMP	@#YTMP3,R5	;FPS CORRECT?
4046	021312	001401			BEQ	Y4	
4047	021314	000475			BR	YERR3	
4048	021316	005737	021534	Y4:	TST	@#YFLAG	;FINISHED TEST?
4049	021322	001015			BNE	Y5	
4050	021324	012737	177777	021534	MOV	#-1,@#YFLAG	
4051	021332	012737	021564	021536	MOV	#YPAT10,@#YTMP1	
4052	021340	012737	021554	021540	MOV	#YPAT00,@#YTMP2	
4053	021346	012737	000200	021542	MOV	#200,@#YTMP3	
4054	021354	000726			BR	Y1	
4055	021356	000512			BR	YDONE	
4056	021360	012702	000004	Y5:	MOV	#4,R2	
4057	021364	012700	021536	Y6:	MOV	#YTMP1,R0	;DID XOR OF SIGN BIT
4058	021370	012701	021544		MOV	#YDAT00,R1	;FAIL?
4059	021374	022021		Y7:	CMP	(R0)+,(R1)+	
4060	021376	001002			BNE	YERR1	
4061	021400	077203			SOB	R2,Y7	
4062	021402	000421			BR	YERR2	
4063	021404			YERR1:			;DATA FAILURE
4064	021404	012737	021254	001236	MOV	#Y2,@#STMP2	
4065	021412	013737	021536	001240	MOV	@#YTMP1,@#STMP3	
4066	021420	012737	021574	001242	MOV	#YPAT20,@#STMP4	
4067	021426	012737	021544	001244	MOV	#YDAT00,@#STMP5	
4068	021434	013737	021540	001246	MOV	@#YTMP2,@#STMP6	
4069	021442	104001		1\$:	ERROR	1	
4070	021444	000457			BR	YDONE	
4071	021446			YERR2:			;XOR OF SIGN BIT
4072	021446	012737	021254	001236	MOV	#Y2,@#STMP2	;FAILED
4073	021454	013737	021536	001240	MOV	@#YTMP1,@#STMP3	
4074	021462	012737	021574	001242	MOV	#YPAT20,@#STMP4	
4075	021470	012737	021544	001244	MOV	#YDAT00,@#STMP5	
4076	021476	013737	021540	001246	MOV	@#YTMP2,@#STMP6	
4077	021504	104002		1\$:	ERROR	2	
4078	021506	000436			BR	YDONE	
4079	021510			YERR3:			;FPS WRONG.
4080	021510	012737	021254	001236	MOV	#Y2,@#STMP2	
4081	021516	010537	001240		MOV	R5,@#STMP3	
4082	021522	013737	021542	001242	MOV	@#YTMP3,@#STMP4	
4083	021530	104002		1\$:	ERROR	2	
4084	021532	000424			BR	YDONE	
4085							
4086	021534	000000		YFLAG:	.WORD	0	
4087	021536	000000		YTMP1:		0	
4088	021540	000000		YTMP2:		0	
4089	021542	000000		YTMP3:		0	
4090							
4091	021544	000000		YDAT00:	.WORD	0	
4092	021546	000000		YDAT01:		0	
4093	021550	000000		YDAT02:		0	
4094	021552	000000		YDAT03:		0	
4095							
4096	021554	063146		YPAT00:		063146	
4097	021556	052525		YPAT01:		052525	
4098	021560	042104		YPAT02:		042104	

```

4099 021562 167356      YPAT03:      167356
4100
4101 021564 163146      YPAT10:      163146
4102 021566 052525      YPAT11:      052525
4103 021570 042104      YPAT12:      042104
4104 021572 167356      YPAT13:      167356
4105
4106 021574 000000      YPAT20:      0
4107 021576 000000      YPAT21:      0
4108 021600 000000      YPAT22:      0
4109 021602 000000      YPAT23:      0
4110
4111 021604
4112 021604 104413      YDONE:      RSETUP
4113
4114
4115
4116
4117
4118
4119
4120
4121
4122
4123
4124
4125
4126
4127 021606 000004
4128 021610 005067 000224
4129 021614 012737 022056 022042
4130 021622 012737 000200 022044
4131 021630
4132 021630 104414
4133 021632 012700 000200
4134 021636 170100
4135 021640 012700 022076
4136 021644 172410
4137 021646 013700 022042
4138 021652 172010
4139 021654 170205
4140 021656 170011
4141 021660 012700 022046
4142 021664 174010
4143 021666 012702 000004
4144 021672 013701 022042
4145 021676 022021
4146 021700 001401
4147 021702 000423
4148 021704 077204
4149 021706 023705 022044
4150 021712 001401
4151 021714 000437
4152 021716 005737 022040
4153 021722 001012
4154 021724 012737 177777 022040

      YPAT03:      167356
      YPAT10:      163146
      YPAT11:      052525
      YPAT12:      042104
      YPAT13:      167356
      YPAT20:      0
      YPAT21:      0
      YPAT22:      0
      YPAT23:      0
      YDONE:      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?).
      ;*****
      ;*TEST 30      ADDD WITH AC=0 TEST
      ;*****
      ;@
      ;@ THIS IS A TEST OF ADDD WITH AC=0. BOTH
      ;* POSITIVE AND NEGATIVE FSRC'S ARE TRIED.
      ;*
      ;*****
      TST30:  SCOPE
      CLR      ZFLAG
      MOV      #ZPAT00,@#ZTMP1 ;P
      MOV      #200,@#ZTMP2
      Z1:
      LPERR
      MOV      #200,R0
      LDFPS   R0
      MOV      #ZPAT20,R0
      LDD      (R0),AC0
      MOV      @#ZTMP1,R0
      ADDD    (R0),AC0
      STFPS   R5
      SETD
      MOV      #ZDAT00,R0
      STD     AC0,(R0)
      MOV      #4,R2
      MOV      @#ZTMP1,R1
      CMP     (R0)+,(R1)+
      BEQ     Z4
      BR      ZERR1
      SOB     R2,Z3
      CMP     @#ZTMP2,R5
      BEQ     Z5
      BR      ZERR2
      TST     @#ZFLAG
      BNE     Z6
      MOV     #-1,@#ZFLAG
      ;SET UP THE LOOP ON ERROR ADDRESS.
      ;SET DOUBLE MODE
      ;SET AC0=0
      ;TEST INSTRUCTION
      ;GET RESULT
      ;RESULT CORRECT?
      ;FPS CORRECT?
      ;FINISHED TEST?
    
```

```

4155 021732 012737 022066 022042      MOV      #ZPAT10,@#ZTMP1
4156 021740 012737 000210 022044      MOV      #210,@#ZTMP2
4157 021746 000730                BR       Z1
4158 021750 000456                Z6:     BR       ZDONE
4159 021752                ZERR1:                ;DATA FAILURE
4160 021752 012737 021652 001236      MOV      #Z2,@#STMP2
4161 021760 013737 022042 001240      MOV      @#ZTMP1,@#STMP3
4162 021766 012737 022076 001242      MOV      #ZPAT20,@#STMP4
4163 021774 012737 022046 001244      MOV      #ZDAT00,@#STMP5
4164 022002 013737 022042 001246      MOV      @#ZTMP1,@#STMP6
4165 022010 104002                1$:     ERROR    2
4166 022012 000435                BR       ZDONE
4167 022014                ZERR2:
4168 022014 012737 021652 001236      MOV      #Z2,@#STMP2
4169 022022 010537 001240      MOV      R5,@#STMP3
4170 022026 013737 022044 001242      MOV      @#ZTMP2,@#STMP4
4171 022034 104002                1$:     ERROR    2
4172 022036 000423                BR       ZDONE
4173
4174 022040 000000                ZFLAG:  .WORD    0
4175 022042 000000                ZTMP1:
4176 022044 000000                ZTMP2:
4177
4178 022046 000000                ZDAT00: .WORD    0
4179 022050 000000                ZDAT01:
4180 022052 000000                ZDAT02:
4181 022054 000000                ZDAT03:
4182
4183 022056 031463                ZPAT00:      031463
4184 022060 010421                ZPAT01:      010421
4185 022062 146314                ZPAT02:      146314
4186 022064 156735                ZPAT03:      156735
4187
4188 022066 156735                ZPAT10:      156735
4189 022070 167356                ZPAT11:      167356
4190 022072 135673                ZPAT12:      135673
4191 022074 146314                ZPAT13:      146314
4192
4193 022076 000000                ZPAT20:      0
4194 022100 000000                ZPAT21:      0
4195 022102 000000                ZPAT22:      0
4196 022104 000000                ZPAT23:      0
4197
4198 022106                ZDONE:
4199 022106 104413                RSETUP
4200
4201
4202
4203
4204
4205
4206
4207
4208
4209
4210

```

```

;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF *THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

```

```

;*****
;*TEST 31      ADDF AND ADDD WITH E(AC)=E(FSRC) TEST AND (BUT FT) TEST
;*
;* THIS IS A TEST OF THE ADD INSTRUCTION WITH THE
;* OPERANDS HAVING EQUAL EXPONENTS.

```

```

4211      ;* THE ROUND/TRUNK FLOWS IS ALSO TESTED.
4212      ;*
4213      ;*****
4214 022110 000004      TST31: SCOPE
4215 022112                AA1:
4216 022112 104414      LPERR                ;SET UP THE LOOP ON ERROR ADDRESS.
4217 022114 012700 003240      MOV #3240,R0
4218 022120 170100      LDFPS R0                ;SET FIU FIV FD AND FT
4219 022122 012737 022472 000244      MOV #AAERRO,@#FPVECT ;IN CASE THE OVER/UNDER
4220 022130 012700 023050      MOV #AAPATO,R0        ;FLOWS IN TRAP WILL
4221                ;OCCUR
4222 022134 172410      LDD (R0),AC0          ;SET UP AC0
4223 022136 012737 022150 001236      MOV #AA2,@#STMP2     ;OPERAND
4224 022144 012700 023060      MOV #AAPAT1,R0
4225 022150 172010      AA2: ADDD (R0),AC0    ;TEST INSTRUCTION
4226                ;SHOULD TRUNCATE
4227 022152 012700 023040      AA3: MOV #AADATO,R0
4228 022156 174010      STD AC0,(R0)         ;GET THE RESULT
4229 022160 012701 023070      MOV #AAPAT2,R1
4230 022164 012702 000004      MOV #4,R2
4231 022170 022021      AA4: CMP (R0)+,(R1)+ ;CORRECT?
4232
4233                BEQ AA7
4234 022174 012700 023100      MOV #AAPAT3,R0      ;DID (BUT FT) FAIL
4235 022200 012701 023040      MOV #AADATO,R1
4236 022204 012702 000004      MOV #4,R2
4237 022210 022021      AA5: CMP (R0)+,(R1)+
4238 022212 001401      BEQ AA6
4239 022214 000561      BR AAERR1 ;DATA ERROR
4240 022216 077204      AA6: SOB R2,AA5
4241 022220 000137 022614      JMP @#AAERR2        ;(BUT FT) ERROR
4242 022224 077217      AA7: SOB R2,AA4
4243
4244      ;NOW TEST DOUBLE FLOATING ROUND MODE.
4245      ;A 1 SHOULD BE ADDED TO THE LSB ON ROUND MODE.
4246
4247 022226                AA10:
4248 022226 104414      LPERR                ;SET UP THE LOOP ON ERROR ADDRESS.
4249 022230 012700 003200      MOV #3200,R0        ;SET FD FIV FIV. FT=0
4250 022234 170100      LDFPS R0
4251 022236 012700 023050      MOV #AAPATO,R0
4252 022242 172410      LDD (R0),AC0        ;SET UP AC0 OPERAND
4253 022244 012737 022256 001236      MOV #AA11,@#STMP2
4254 022252 012700 023060      MOV #AAPAT1,R0
4255 022256 172010      AA11: ADDD (R0),AC0 ;TEST INSTRUCTION
4256                ;SHOULD ROUND
4257 022260 012700 023040      AA12: MOV #AADATO,R0
4258 022264 174010      STD AC0,(R0)        ;GET THE RESULT
4259 022266 012701 023100      MOV #AAPAT3,R1
4260 022272 012702 000004      MOV #4,R2
4261 022276 022021      AA13: CMP (R0)+,(R1)+ ;CORRECT?
4262 022300 001425      BEQ AA20
4263 022302 012700 023070      MOV #AAPAT2,R0      ;DID (BUT FT) FAIL?
4264 022306 012701 023040      MOV #AADATO,R1
4265 022312 012702 000004      MOV #4,R2
4266 022316 022021      AA14: CMP (R0)+,(R1)+

```

```

4267 022320 001413      BEQ      AA17
4268 022322 012700 023110  MOV      #AAPAT4,R0      ;WAS THE FLOATING
4269 022326 012701 023040  MOV      #AADATO,R1      ;CONSTANT USED
4270 022332 012702 000004  MOV      #4,R2           ;INSTEAD OF THE
4271 022336 022021      AA15:    CMP      (R0)+,(R1)+     ;DOUBLE CONSTANT
4272 022340 001401      BEQ      AA16           ;IN THE ROUND
4273 022342 000542      BR      AAERR3 ; FLOWS?
4274 022344 077204      AA16:    SOB      R2,AA15     ;DATA ERROR
4275 022346 000544      BR      AAERR4 ; CONSTANT ERROR
4276 022350 077216      AA17:    SOB      R2,AA14
4277 022352 000560      BR      AAERR5 ; (BUT FT) ERROR
4278 022354 077230      AA20:    SOB      R2,AA13
4279
4280      ;NOW TEST ADDF WITH FT=0, ROUND MODE
4281
4282      AA21:
4283 022356 104414      LPERR      ;SET UP THE LOOP ON ERROR ADDRESS.
4284 022360 012700 003200  MOV      #3200,R0      ;FIV=1, FIV=1, FT=0
4285 022364 170100      LDFPS     R0
4286 022366 012700 023050  MOV      #AAPATO,R0     ;LOAD ACO OPERAND
4287 022372 172410      LDD      (R0),ACO
4288 022374 170001      SETF
4289 022376 012737 022410 001236  MOV      #AA22,@#STMP2  ;ENTER FLOATING MODE
4290 022404 012700 023120  MOV      #AAPAT5,R0
4291 022410 172010      AA22:    ADDF     (R0),ACO     ;TEST INSTRUCTION
4292      ;SHOULD ROUND
4293 022412
4294 022412 170011      AA23:    SETD
4295      ;RESET TO DOUBLE
4296 022414 012700 023040  MOV      #AADATO,R0     ;MODE
4297 022420 174010      STD      ACO,(R0)      ;GET THE RESULT
4298 022422 012701 023130  MOV      #AAPAT6,R1     ;CORRECT?
4299 022426 012702 000002  MOV      #2,R2
4300 022432 022021      AA24:    CMP      (R0)+,(R1)+
4301 022434 001413      BEQ      AA27
4302 022436 012700 023070  MOV      #AAPAT2,R0     ;WAS THE DOUBLE
4303 022442 012701 023040  MOV      #AADATO,R1     ;CONSTANT USED INSTEAD
4304 022446 012702 000002  MOV      #2,R2           ;OF THE FLOATING
4305 022452 022011      AA25:    CMP      (R0)+,(R1)   ;CONSTANT IN THE
4306 022454 001401      BEQ      AA26           ;ROUND FLOWS?
4307 022456 000534      BR      AAERR6 ; DATA ERROR
4308 022460 077204      AA26:    SOB      R2,AA25
4309 022462 000550      BR      AAERR7 ; CONSTANT ERROR
4310 022464 077216      AA27:    SOB      R2,AA24
4311 022466 000137 023140  JMP      @#AADONE
4312
4313      ;COME HERE IF A TRAP OCCURS TO 244.
4314
4315 022472 013700 001236  AAERRO:  MOV      @#STMP2,R0 ;SEE IF THE TRAP WAS
4316 022476 005720      TST      (R0)+ ;AT A TEST INSTRUCTION
4317 022500 020016      CMP      R0,(SP)
4318 022502 001402      BEQ      1$
4319 022504 000137 062524  10$:    JMP      @#FPSPUR
4320 022510
4321 022510 170300      1$:
4322 022512 020027 000010  STST     R0 ;GET FEC
      CMP      R0,#10
    
```



```

4323 022516 001405          BEQ      20$          ;OVERFLOW
4324 022520 020027 000012  CMP      R0,#12
4325 022524 001410          BEQ      30$          ;UNDERFLOW
4326 022526 000766          BR       10$
4327 022530 022532          20$
4328 022532 011637 001236 20$:    MOV     (SP),@#STMP2 ;REPORT OVERFLOW ERROR
4329 022536 022626          CMP     (SP)+,(SP)+
4330 022540 104002          21$:    ERROR  2
4331 022542 000137 023140 25$:    JMP     @#AADONE
4332 022546 011637 001236 30$:    MOV     (SP),@#STMP2 ;REPORT UNDERFLOW
4333 022552 022626          CMP     (SP)+,(SP)+ ;ERROR
4334 022554 104002          31$:    ERROR  2
4335 022556 000771          BR     25$

```

```

4336 022556 000771          ;ADD RESULT INCORRECT
4337 022556 000771          AAERR1: MOV    #AAPAT2,@#STMP6
4338 022560 012737 023070 001246 AAERR10: MOV   #AAPAT0,@#STMP4
4339 022566 012737 023050 001242 AAERR10: MOV   #AAPAT1,@#STMP3
4340 022574 012737 023060 001240 AAERR10: MOV   #AADATO,@#STMP5
4341 022602 012737 023040 001244 1$:    ERROR  2
4342 022610 104002          BR     AADONE
4343 022612 000552          AAERR2: MOV    #AAPAT2,@#STMP6 ;(BUT FT) FAILED.
4344 022614 012737 023070 001246 AAERR2: MOV    #AAPAT0,@#STMP4
4345 022622 012737 023050 001242 AAERR2: MOV    #AAPAT1,@#STMP3
4346 022630 012737 023060 001240 AAERR2: MOV    #AADATO,@#STMP5
4347 022636 012737 023040 001244 1$:    ERROR  2
4348 022644 104002          BR     AADONE
4349 022646 000534          AAERR3: MOV    #AAPAT3,@#STMP6 ;DATA ERROR.
4350 022650 012737 023100 001246 AAERR3: BR     AAERR10
4351 022656 000743          AAERR4: MOV    #AAPAT3,@#STMP6 ;BAD CONSTANT
4352 022660 012737 023100 001246 AAERR4: MOV    #AAPAT0,@#STMP4
4353 022666 012737 023050 001242 AAERR4: MOV    #AAPAT1,@#STMP3
4354 022674 012737 023060 001240 AAERR4: MOV    #AADATO,@#STMP5
4355 022702 012737 023040 001244 1$:    ERROR  2
4356 022710 104002          BR     AADONE
4357 022712 000512          AAERR5: MOV    #AAPAT3,@#STMP6 ;(BUT FT) FAILED.
4358 022714 012737 023100 001246 AAERR5: MOV    #AAPAT0,@#STMP4
4359 022722 012737 023050 001242 AAERR5: MOV    #AAPAT1,@#STMP3
4360 022730 012737 023060 001240 AAERR5: MOV    #AADATO,@#STMP5
4361 022736 012737 023040 001244 1$:    ERROR  2
4362 022744 104002          BR     AADONE
4363 022746 000474          AAERR6: MOV    #AAPAT5,@#STMP3 ;FD=0 AND
4364 022750 012737 023120 001240 AAERR6: MOV    #AAPAT0,@#STMP4 ;DATA ERROR
4365 022756 012737 023050 001242 AAERR6: MOV    #AADATO,@#STMP5
4366 022764 012737 023040 001244 AAERR6: MOV    #AAPAT6,@#STMP6
4367 022772 012737 023130 001246 1$:    ERROR  2
4368 023000 104002          BR     AADONE
4369 023002 000456          AAERR7: MOV    #AAPAT5,@#STMP3 ;CONSTANT ERROR
4370 023004 012737 023120 001240 AAERR7: MOV    #AAPAT0,@#STMP4
4371 023012 012737 023050 001242 AAERR7: MOV    #AADATO,@#STMP5
4372 023020 012737 023040 001244 AAERR7: MOV    #AAPAT6,@#STMP6
4373 023026 012737 023130 001246 1$:    ERROR  2
4374 023034 104002          BR     AADONE
4375 023036 000440          AADATO: 0
4376 023040 000000          AADATO: 0
4377 023042 000000          AADATO: 0
4378 023044 000000          AADATO: 0

```

4379	023046	000000			
4380	023050	000200	AAPAT0:	200	
4381	023052	000000		0	
4382	023054	000000		0	
4383	023056	000000		0	
4384	023060	000200	AAPAT1:	200	
4385	023062	000000		0	
4386	023064	000000		0	
4387	023066	000001		1	
4388	023070	000400	AAPAT2:	400	
4389	023072	000000		0	
4390	023074	000000		0	
4391	023076	000000		0	
4392	023100	000400	AAPAT3:	400	
4393	023102	000000		0	
4394	023104	000000		0	
4395	023106	000001		1	
4396	023110	000400	AAPAT4:	400	
4397	023112	000000		0	
4398	023114	100000		100000	
4399	023116	000000		0	
4400	023120	000200	AAPAT5:	200	
4401	023122	000001		1	
4402	023124	000000		0	
4403	023126	000000		0	
4404	023130	000400	AAPAT6:	400	
4405	023132	000001		1	
4406	023134	000000		0	
4407	023136	000000		0	
4408	023140		AADONE:		
4409	023140	104413	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?).

::*****
 :*TEST 32 ADDF AND ADDD WITH E(AC) LESS THAN E(FSRC) TEST
 :*
 :*THIS IS A TEST OF THE ADDD AND ADDF
 :*INSTRUCTIONS AND THE ALIGN AC ALGORITHM
 :*FLOWS. THE CONSTANT (25 FOR FLOATING, 57 FOR
 :*DOUBLE) USED IS CHECKED. THEN SIMPLE
 :*AND WORST CASE ALIGNMENT SITUATIONS ARE
 :*TRIED. NOTE E(AC) IS LESS THEN E(FSRC)
 :*
 :*****

4424					
4425	023142	000004			
4426					
4427	023144				
4428	023144	104414			
4429	023146	012704	003200		
4430	023152	170104			
4431	023154	012737	023174	001236	
4432	023162	012700	024576		
4433	023166	172410			
4434	023170	012700	024616		

TST32: SCOPE
 :EXPONENT DIFFERENCE=57=71 (OCT) FD=1
 CC1:
 LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
 MOV #3200,R4 ;SET FIV,FIV, AND FD
 LDFPS R4
 MOV #CC2,@#STMP2
 MOV #CCP0,R0 ;SET ACO OPERAND
 LDD (R0),ACO ;ACO
 MOV #CCP2,R0

4435	023174	172010			CC2:	ADD (R0),ACO	:TEST INSTRUCTION
4436	023176	170205				STFPS R5	:GET FPS
4437	023200	012700	024566			MOV #CCDATO,R0	:GET THE RESULT
4438	023204	174010				STD ACO,(R0)	
4439	023206	012701	024616			MOV #CCP2,R1	:IS IT CORRECT
4440	023212	012702	000004			MOV #4,R2	
4441	023216	022021			CC3:	CMP (R0)+,(R1)+	
4442	023220	001415				BEQ CC6	
4443	023222	012700	024566			MOV #CCDATO,R0	:DID A BAD
4444	023226	012701	024576			MOV #CCP0,R1	:CONSTANT (NOT 57)
4445	023232	012702	000004			MOV #4,R2	:GET GENERATED
4446	023236	022021			CC4:	CMP (R0)+,(R1)+	:FOR THE ALIGNMENT
4447	023240	001402				BEQ CC5	:FLOWS?
4448	023242	000137	024164			JMP @#CCER1	:DATA ERROR.D
4449	023246	077205			CC5:	SOB R2,CC4	
4450	023250	000137	024222			JMP @#CCER2	:BAD CONSTANT.D
4451	023254	077220			CC6:	SOB R2,CC3	
4452	023256	020405				CMP R4,R5	:FPS CORRECT?
4453	023260	001402				BEQ CC7	
4454	023262	000137	024130			JMP @#CCERO	:BAD FPS.
4455						:EXPONENT DIFFERENCE=56=70 (OCT) FD=1	
4456	023266				CC7:	LPERR	:SET UP THE LOOP ON ERROR ADDRESS.
4457	023266	104414				MOV #3200,R4	:SET FIV,FIV, AND FD
4458	023270	012704	003200			LDFPS R4	
4459	023274	170104				MOV #CC8,@#STMP2	
4460	023276	012737	023316	001236		MOV #CCP0,R0	:SET ACO OPERAND
4461	023304	012700	024576			LDD (R0),ACO	
4462	023310	172410				MOV #CCP1,R0	:FSRC
4463	023312	012700	024606		CC8:	ADD (R0),ACO	:TEST INSTRUCTION
4464	023316	172010				STFPS R5	:GET FPS
4465	023320	170205				MOV #CCDATO,R0	:GET THE RESULT
4466	023322	012700	024566			STD ACO,(R0)	
4467	023326	174010				MOV #CCP7,R1	:IS IT CORRECT
4468	023330	012701	024666			MOV #4,R2	
4469	023334	012702	000004		CC9:	CMP (R0)+,(R1)+	
4470	023340	022021				BEQ CC12	
4471	023342	001415				MOV #CCDATO,R0	:DID A BAD
4472	023344	012700	024566			MOV #CCP1,R1	:CONSTANT (NOT 57)
4473	023350	012701	024606			MOV #4,R2	:GET GENERATED
4474	023354	012702	000004		CC10:	CMP (R0)+,(R1)+	:FOR THE ALIGNMENT
4475	023360	022021				BEQ CC11	:FLOWS?
4476	023362	001402				JMP @#CCER3	:DATA ERROR.D
4477	023364	000137	024260		CC11:	SOB R2,CC10	
4478	023370	077205				JMP @#CCER4	:BAD CONSTANT.D
4479	023372	000137	024276		CC12:	SOB R2,CC9	
4480	023376	077220				CMP R4,R5	:FPS CORRECT?
4481	023400	020405				BEQ CC13	
4482	023402	001402				JMP @#CCERO	:BAD FPS.
4483	023404	000137	024130			:EXPONENT DIFFERENCE=25=31 (OCT) FD=0	
4484					CC13:	LPERR	:SET UP THE LOOP ON ERROR ADDRESS.
4485	023410					MOV #CC14,@#STMP2	
4486	023410	104414				MOV #CCP0,R0	:SET UP ACO OPERAND.
4487	023412	012737	023440	001236		LDD (R0),ACO	
4488	023420	012700	024576			MOV #3000,R4	:SET FIV,FIV. CLEAR FD.
4489	023424	172410					
4490	023426	012704	003000				

4491	023432	170104			LDFPS	R4	
4492	023434	012700	024656		MOV	#CCP6,R0	:FSRC
4493	023440	172010		CC14:	ADDF	(R0),ACO	:TEST INSTRUCTION
4494	023442	170205			STFPS	R5	
4495	023444	170011			SETD		:REENTER DOUBLE MOVE
4496	023446	012700	024566		MOV	#CCDATO,R0	:GET THE RESULT
4497	023452	174010			STD	ACO,(R0)	
4498	023454	012701	024656		MOV	#CCP6,R1	:IS THE RESULT CORRECT?
4499	023460	012702	000002		MOV	#2,R2	
4500	023464	022021		CC15:	CMP	(R0)+,(R1)+	
4501	023466	001415			BEQ	CC18	
4502	023470	012700	024566		MOV	#CCDATO,R0	:WAS A BAD CONSTANT
4503	023474	012701	024626		MOV	#CCP3,R1	:USED (NOT 25) IN
4504	023500	012702	000002		MOV	#2,R2	:THE ALIGN FLOWS?
4505	023504	022021		CC16:	CMP	(R0)+,(R1)+	
4506	023506	001402			BEQ	CC17	
4507	023510	000137	024334		JMP	@#CCER5	:DATA ERROR F
4508	023514	077205		CC17:	SOB	R2,CC16	
4509	023516	000137	024370		JMP	@#CCER6	:BAD CONSTANT F
4510	023522	077220		CC18:	SOB	R2,CC15	
4511	023524	020405			CMP	R4,R5	
4512	023526	001402			BEQ	CC19	
4513	023530	000137	024146		JMP	@#CCER90	:BAD FPS.
4514							:EXPONENT DIFFERENCE=24=30 (OCT) FD=0
4515	023534			CC19:	LPERR		:SET UP THE LOOP ON ERROR ADDRESS.
4516	023534	104414			MOV	#CC20,@#STMP2	
4517	023536	012737	023564	001236	MOV	#CCP3,R0	:SET UP ACO OPERAND.
4518	023544	012700	024626		LDD	(R0),ACO	
4519	023550	172410			MOV	#3000,R4	:SET FIV,FIV. CLEAR FD.
4520	023552	012704	003000		LDFPS	R4	
4521	023556	170104			MOV	#CCP5,R0	:FSRC
4522	023560	012700	024646		ADDF	(R0),ACO	:TEST INSTRUCTION
4523	023564	172010		CC20:	STFPS	R5	
4524	023566	170205			SETD		:REENTER DOUBLE MOVE
4525	023570	170011			MOV	#CCDATO,R0	:GET THE RESULT
4526	023572	012700	024566		STD	ACO,(R0)	
4527	023576	174010			MOV	#CCP10,R1	:IS THE RESLT CORRECT?
4528	023600	012701	024676		MOV	#2,R2	
4529	023604	012702	000002		MOV	#2,R2	
4530	023610	022021		CC21:	CMP	(R0)+,(R1)+	
4531	023612	001415			BEQ	CC24	
4532	023614	012700	024566		MOV	#CCDATO,R0	:WAS A BAD CONSTANT
4533	023620	012701	024646		MOV	#CCP5,R1	:USED (NOT 25) IN
4534	023624	012702	000002		MOV	#2,R2	:THE ALIGN FLOWS?
4535	023630	022021		CC22:	CMP	(R0)+,(R1)+	
4536	023632	001402			BEQ	CC23	
4537	023634	000137	024424		JMP	@#CCER7	:DATA ERROR F
4538	023640	077205		CC23:	SOB	R2,CC22	
4539	023642	000137	024442		JMP	@#CCER8	:BAD CONSTANT F
4540	023646	077220		CC24:	SOB	R2,CC21	
4541	023650	020405			CMP	R4,R5	
4542	023652	001402			BEQ	CC25	
4543	023654	000137	024146		JMP	@#CCER90	:BAD FPS.
4544							:EXPONENT DIFFERENCE=1 FD=1
4545	023660			CC25:	LPERR		:SET UP THE LOOP ON ERROR ADDRESS.
4546	023660	104414					

4547	023662	012704	003200		MOV	#3200,R4	;SET FIV,FIV, AND FD
4548	023666	170104			LDFPS	R4	
4549	023670	012737	023710	001236	MOV	#CC26,@#\$TMP2	
4550	023676	012700	024576		MOV	#CCP0,R0	;SET ACO OPERAND
4551	023702	172410			LDD	(R0),ACO	
4552	023704	012700	024626		MOV	#CCP3,R0	;FSRC
4553	023710	172010		CC26:	ADDD	(R0),ACO	;TEST INSTRUCTION
4554	023712	170205			STFPS	R5	;GET FPS
4555	023714	012700	024566		MOV	#CCDAT0,R0	;GET THE RESULT
4556	023720	174010			STD	ACO,(R0)	
4557	023722	012701	024706		MOV	#CCP11,R1	;IS IT CORRECT
4558	023726	012702	000004		MOV	#4,R2	
4559	023732	022021		CC27:	CMP	(R0)+,(R1)+	
4560	023734	001415			BEQ	CC30	
4561	023736	012700	024566		MOV	#CCDAT0,R0	;DID A BAD
4562	023742	012701	024626		MOV	#CCP3,R1	;CONSTANT (NOT 57)
4563	023746	012702	000004		MOV	#4,R2	;GET GENERATED
4564	023752	022021		CC28:	CMP	(R0)+,(R1)+	;FOR THE ALIGNMENT
4565	023754	001402			BEQ	CC29	;FLOWS?
4566	023756	000137	024476		JMP	@#CCER10	;DATA ERROR.D
4567	023762	077205		CC29:	SOB	R2,CC28	
4568	023764	000137	024514		JMP	@#CCER11	;BAD CONSTANT.D
4569	023770	077220		CC30:	SOB	R2,CC27	
4570	023772	020405			CMP	R4,R5	;FPS CORRECT?
4571	023774	001402			BEQ	CC31	
4572	023776	000137	024130		JMP	@#CCERO	;BAD FPS.
4573							;EXPONENT DIFFERENCE=100=144 (OCT) FD=1
4574	024002			CC31:	LPERR		;SET UP THE LOOP ON ERROR ADDRESS.
4575	024002	104414			MOV	#3200,R4	;SET FIV,FIV, AND FD
4576	024004	012704	003200		LDFPS	R4	
4577	024010	170104		001236	MOV	#CC32,@#\$TMP2	
4578	024012	012737	024032		MOV	#CCP0,R0	;SET ACO OPERAND
4579	024020	012700	024576		LDD	(R0),ACO	
4580	024024	172410			MOV	#CCP4,R0	;FSRC
4581	024026	012700	024636		ADDD	(R0),ACO	;TEST INSTRUCTION
4582	024032	172010		CC32:	STFPS	R5	;GET FPS
4583	024034	170205			MOV	#CCDAT0,R0	;GET THE RESULT
4584	024036	012700	024566		STD	ACO,(R0)	
4585	024042	174010			MOV	#CCP4,R1	;IS IT CORRECT
4586	024044	012701	024636		MOV	#4,R2	
4587	024050	012702	000004		CC33:	CMP	(R0)+,(R1)+
4588	024054	022021			BEQ	CC36	
4589	024056	001415			MOV	#CCDAT0,R0	;DID A BAD
4590	024060	012700	024566		MOV	#CCP4,R1	;CONSTANT (NOT 57)
4591	024064	012701	024636		MOV	#4,R2	;GET GENERATED
4592	024070	012702	000004		CC34:	CMP	(R0)+,(R1)+
4593	024074	022021			BEQ	CC35	;FLOWS?
4594	024076	001402			JMP	@#CCER12	;DATA ERROR.D
4595	024100	000137	024532		CC35:	SOB	R2,CC34
4596	024104	077205			JMP	@#CCER13	;BAD CONSTANT.D
4597	024106	000137	024550		CC36:	SOB	R2,CC33
4598	024112	077220			CMP	R4,R5	;FPS CORRECT?
4599	024114	020405			BEQ	CC37	
4600	024116	001402			JMP	@#CCERO	;BAD FPS.
4601	024120	000137	024130		CC37:	JMP	@#CCDONE
4602	024124	000137	024726				

```

4603 024130 010437 001242 CCER0: MOV R4,@#STMP4 ;FPS ERROR D
4604 024134 010537 001240 MOV R5,@#STMP3
4605 024140 104002 1$: ERROR 2
4606 024142 000137 024726 JMP @#CCDONE
4607 024146 010437 001242 CCER90: MOV R4,@#STMP4 ;FPS ERROR F
4608 024152 010537 001240 MOV R5,@#STMP3
4609 024156 104002 1$: ERROR 2
4610 024160 000137 024726 JMP @#CCDONE
4611 024164 012737 024616 001240 CCER1: MOV #CCP2,@#STMP3 ;DATA ERROR D
4612 024172 012737 024616 001246 MOV #CCP2,@#STMP6
4613 024200 012737 024576 001242 CCER50: MOV #CCP0,@#STMP4
4614 024206 012737 024566 001244 MOV #CCDATO,@#STMP5
4615 024214 104002 1$: ERROR 2
4616 024216 000137 024726 JMP @#CCDONE
4617 024222 012737 024616 001240 CCER2: MOV #CCP2,@#STMP3 ;CONSTANT BAD D(B)
4618 024230 012737 024616 001246 MOV #CCP2,@#STMP6
4619 024236 012737 024576 001242 CCER22: MOV #CCP0,@#STMP4
4620 024244 012737 024566 001244 MOV #CCDATO,@#STMP5
4621 024252 104002 1$: ERROR 2
4622 024254 000137 024726 JMP @#CCDONE
4623 024260 012737 024606 001240 CCER3: MOV #CCP1,@#STMP3
4624 024266 012737 024666 001246 MOV #CCP7,@#STMP6
4625 024274 000741 BR CCER50
4626 024276 012737 024606 001240 CCER4: MOV #CCP1,@#STMP3 ;CONSTANT BAD D(G)
4627 024304 012737 024666 001246 MOV #CCP7,@#STMP6
4628 024312 012737 024576 001242 CCER44: MOV #CCP0,@#STMP4
4629 024320 012737 024566 001244 MOV #CCDATO,@#STMP5
4630 024326 104002 1$: ERROR 2
4631 024330 000137 024726 JMP @#CCDONE
4632 024334 012737 024656 001240 CCER5: MOV #CCP6,@#STMP3 ;DATA ERROR F
4633 024342 012737 024656 001246 MOV #CCP6,@#STMP6
4634 024350 012737 024576 001242 CCER55: MOV #CCP0,@#STMP4
4635 024356 012737 024566 001244 MOV #CCDATO,@#STMP5
4636 024364 104002 1$: ERROR 2
4637 024366 000557 BR CCDONE
4638 024370 012737 024656 001240 CCER6: MOV #CCP6,@#STMP3 ;CONSTANT BAD F(B)
4639 024376 012737 024656 001246 MOV #CCP6,@#STMP6
4640 024404 012737 024576 001242 MOV #CCP0,@#STMP4
4641 024412 012737 024566 001244 MOV #CCDATO,@#STMP5
4642 024420 104002 1$: ERROR 2
4643 024422 000541 BR CCDONE
4644 024424 012737 024646 001240 CCER7: MOV #CCP5,@#STMP3 ;DATA ERROR F
4645 024432 012737 024676 001246 MOV #CCP10,@#STMP6
4646 024440 000743 BR CCER55
4647 024442 012737 024646 001240 CCER8: MOV #CCP5,@#STMP3 ;CONSTANT BAD F(G)
4648 024450 012737 024676 001246 MOV #CCP10,@#STMP6
4649 024456 012737 024566 001244 MOV #CCDATO,@#STMP5
4650 024464 012737 024576 001242 MOV #CCP0,@#STMP4
4651 024472 104002 1$: ERROR 2
4652 024474 000514 BR CCDONE
4653 024476 012737 024626 001240 CCER10: MOV #CCP3,@#STMP3 ;DATA ERROR D
4654 024504 012737 024706 001246 MOV #CCP11,@#STMP6
4655 024512 000632 BR CCER50
4656 024514 012737 024626 001240 CCER11: MOV #CCP3,@#STMP3 ;CONSTANT BAD D(G)
4657 024522 012737 024706 001246 MOV #CCP11,@#STMP6
4658 024530 000670 BR CCER44
    
```

```

4659 024532 012737 024636 001240 CCER12: MOV #CCP4,@#STMP3 ;DATA ERROR D
4660 024540 012737 024636 001246 MOV #CCP4,@#STMP6
4661 024546 000614 BR CCER50
4662 024550 012737 024636 001240 CCER13: MOV #CCP4,@#STMP3 ;CONSTANT BAD D(B)
4663 024556 012737 024636 001246 MOV #CCP4,@#STMP6
4664 024564 000624 BR CCER22
4665 024566 000000 CCDATO: 0
4666 024570 000000 0
4667 024572 000000 0
4668 024574 000000 0
4669 024576 000200 CCP0: 200 ;E(AC)=1
4670 024600 000000 0
4671 024602 000000 0
4672 024604 000000 0
4673 024606 016200 CCP1: 16200 ;E(FSRC)=E(AC)+56=57
4674 024610 000000 0 ; =71(OCT)
4675 024612 000000 0
4676 024614 000000 0
4677 024616 016400 CCP2: 16400 ;E(FSRC)=E(AC)+57=58
4678 024620 000000 0 ; =72(OCT)
4679 024622 000000 0
4680 024624 000000 0
4681 024626 000400 CCP3: 400 ;E(FSRC)=E(AC)+1=2
4682 024630 000000 0
4683 024632 000000 0
4684 024634 000000 0
4685 024636 031200 CCP4: 31200 ;E(FSRC)=E(AC)+100=101=145(OCT)
4686 024640 000000 0
4687 024642 000000 0
4688 024644 000000 0
4689 024646 006200 CCP5: 6200 ;E(FSRC)=E(AC)+24=25=31(OCT)
4690 024650 000000 0
4691 024652 000000 0
4692 024654 000000 0
4693 024656 006400 CCP6: 6400 ;E(FSRC)=E(AC)+25=26=32(OCT)
4694 024660 000000 0
4695 024662 000000 0
4696 024664 000000 0
4697 024666 016200 CCP7: 16200 ;CCP1 RES
4698 024670 000000 0
4699 024672 000000 0
4700 024674 000001 1
4701 024676 006200 CCP10: 6200 ;CCP5 RES
4702 024700 000001 1
4703 024702 000000 0
4704 024704 000000 0
4705 024706 000500 CCP11: 500 ;CCP3 RES
4706 024710 000000 0
4707 024712 000000 0
4708 024714 000000 0
4709 024716 000200 CCP12: 200 ;BAD CONSTANT
4710 024720 000000 0 ;RES CCP2,CCP4
4711 024722 000000 0
4712 024724 000000 0
4713
4714 024726 CCDONE:
    
```

```

4715 024726 104413
4716
4717
4718
4719
4720
4721
4722
4723
4724
4725
4726
4727
4728
4729
4730
4731
4732 024730 000004
4733
4734 024732
4735 024732 104414
4736 024734 012704 003200
4737 024740 170104
4738 024742 012737 025570 000244
4739 024750 012737 024770 001236
4740
4741 024756 012700 026132
4742 024762 172410
4743 024764 012700 026122
4744 024770 172010
4745 024772 170205
4746 024774 012700 026102
4747 025000 174010
4748 025002 012701 026132
4749 025006 012702 000004
4750 025012 022021
4751 025014 001402
4752 025016 000137 025630
4753 025022 077205
4754
4755 025024 020405
4756 025026 001402
4757 025030 000137 025570
4758
4759 025034
4760 025034 104414
4761 025036 012704 003200
4762 025042 170104
4763 025044 012737 025064 001236
4764 025052 012700 026152
4765 025056 172410
4766 025060 012700 026122
4767 025064 172010
4768 025066 170205
4769 025070 012700 026102
4770 025074 174010
    
```

```

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?).

:*****
:*TEST 33 ADDF AND ADDD WITH E(AC) GREATER THAN E(FSRC) TEST
:*
:*THIS IS A TEST OF THE ADDD AND ADDF
:*INSTRUCTIONS AND THE ALIGN FSRC ALGORITHM
:*FLOWS. FIRST THE CONSTANT USED IS CHECKED.
:*THEN SIMPLE AND WORST CASE ALIGNMENT
:*SITUATIONS ARE TRIED. NOTE E(AC)
:*IS GREATER THAN E(FSRC).
:*
:*****
TST33: SCOPE
;EXPONENT DIFFERENCE=57=71 (OCT) FD=1
BB1:
LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
MOV #3200,R4 ;SET FIV FIV, AND FD
LDFPS R4
MOV #BBERO,@#FPVECT ;SET UP FOR ERROR
MOV #BB2,@#STMP2 ;IN CASE THE OVER\
;UNDER FLOWS FAIL.
;SET ACO OPERAND.
MOV #BBPAT2,R0
LDD (R0),AC0
MOV #BBPAT1,R0 ;FSRC
BB2: ADDD (R0),AC0 ;TEST INSTRUCTION
STFPS R5
BB3: MOV #BBDATO,R0 ;GET THE RESULT
STD ACO,(R0)
MOV #BBPAT2,R1 ;RESULT CORRECT?
MOV #4,R2
BB4: CMP (R0)+,(R1)+
BEQ BB5
JMP @#BBER1 ;DATA ERROR D
BB5: SOB R2,BB4 ;WAS FPS CORRECT?
CMP R4,R5
BEQ BB6
JMP @#BBERO ;FPS ERROR
;EXPONENT DIFFERENCE=56=70 (OCT) FD=1
BB6:
LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
MOV #3200,R4 ;SET FIV,FIV, AND FD
LDFPS R4
MOV #BB7,@#STMP2
MOV #BBPAT4,R0 ;SET ACO OPERAND
LDD (R0),AC0
MOV #BBPAT1,R0 ;FSRC
BB7: ADDD (R0),AC0 ;TEST INSTRUCTION
STFPS R5 ;GET FPS
MOV #BBDATO,R0 ;GET THE RESULT
STD ACO,(R0)
    
```



```

4771 025076 012701 026212      MOV      #BBP10,R1      ;IS IT CORRECT
4772 025102 012702 000004      MOV      #4,R2
4773 025106 022021      BB10:   CMP      (R0)+,(R1)+
4774 025110 001415      BEQ      BB13
4775 025112 012700 026102      MOV      #BBDAT0,R0    ;DID A BAD
4776 025116 012701 026152      MOV      #BBPAT4,R1    ;CONSTANT (NOT 57)
4777 025122 012702 000004      MOV      #4,R2          ;GET GENERATED
4778 025126 022021      BB11:   CMP      (R0)+,(R1)+ ;FOR THE ALIGNMENT
4779 025130 001402      BEQ      BB12          ;FLOWS?
4780 025132 000137 025666      JMP      @#BBER2       ;DATA ERROR.D
4781 025136 077205      BB12:   SOB      R2, BB11
4782 025140 000137 025704      JMP      @#BBER3       ;BAD CONSTANT.D
4783 025144 077220      BB13:   SOB      R2, BB10
4784 025146 020405      CMP      R4,R5          ;FPS CORRECT?
4785 025150 001402      BEQ      BB14
4786 025152 000137 025570      JMP      @#BBER0       ;BAD FPS.
4787                                ;EXPONENT DIFFERENCE=25=31 (OCT) FD=0
4788 025156      BB14:   LPERR
4789 025156 104414      MOV      #BB15,@#STMP2 ;SET UP THE LOOP ON ERROR ADDRESS.
4790 025160 012737 025206 001236      MOV      #BBPAT0,R0    ;SET UP ACO OPERAND
4791 025166 012700 026112      LDD      (R0),AC0
4792 025172 172410      LDD      (R0),AC0
4793 025174 012704 003000      MOV      #3000,R4      ;SET FIV AND FIV
4794                                ;CLEAR FD
4795 025200 170104      LDFPS   R4
4796 025202 012700 026122      MOV      #BBPAT1,R0    ;FSRC
4797 025206 172010      BB15:   ADDF   (R0),AC0    ;TEST INSTRUCTION
4798 025210 170205      STFPS   R5
4799 025212 170011      SETD
4800 025214 012700 026102      MOV      #BBDAT0,R0    ;REENTERED DOUBLE MODE.
4801 025220 174010      STD     ACO,(R0)       ;GET THE RESULT
4802 025222 012701 026112      MOV      #BBPAT0,R1    ;IS THE RESULT
4803 025226 012702 000002      MOV      #2,R2         ;CORRECT?
4804 025232 022021      BB16:   CMP      (R0)+,(R1)+
4805 025234 001402      BEQ      BB17
4806 025236 000137 025740      JMP      @#BBER4       ;DATA ERROR F
4807 025242 077205      BB17:   SOB      R2, BB16
4808 025244 020405      CMP      R4,R5          ;IS FPS CORRECT?
4809 025246 001402      BEQ      BB20
4810 025250 000137 025610      JMP      @#BBER10      ;FPS ERROR.
4811                                ;EXPONENT DIFFERENCE=24=30 (OCT)
4812 025254      BB20:   LPERR
4813 025254 104414      MOV      #BB21,@#STMP2 ;SET UP THE LOOP ON ERROR ADDRESS.
4814 025256 012737 025304 001236      MOV      #BBPAT3,R0    ;SET UP ACO OPERAND.
4815 025264 012700 026142      LDD      (R0),AC0
4816 025270 172410      LDD      (R0),AC0
4817 025272 012704 003000      MOV      #3000,R4      ;SET FIU,FIV. CLEAR FD.
4818 025276 170104      LDFPS   R4
4819 025300 012700 026122      MOV      #BBPAT1,R0    ;FSRC
4820 025304 172010      BB21:   ADDF   (R0),AC0    ;TEST INSTRUCTION
4821 025306 170205      STFPS   R5
4822 025310 170011      SETD
4823 025312 012700 026102      MOV      #BBDAT0,R0    ;REENTER DOUBLE MODE
4824 025316 174010      STD     ACO,(R0)       ;GET THE RESULT
4825 025320 012701 026202      MOV      #BBP7,R1      ;IS THE RESULT CORRECT?
4826 025324 012702 000002      MOV      #2,R2
    
```

4827	025330	022021			BB22:	CMP	(R0)+,(R1)+	
4828	025332	001415				BEQ	BB25	
4829	025334	012700	026102			MOV	#BBDATO,R0	:WAS A BAD CONSTANT
4830	025340	012701	026142			MOV	#BBPAT3,R1	:USED (NOT 25) IN
4831	025344	012702	000002			MOV	#2,R2	:THE ALLIGN FLOWS?
4832	025350	022021			BB23:	CMP	(R0)+,(R1)+	
4833	025352	001402				BEQ	BB24	
4834	025354	000137	025774			JMP	@#BBER5	:DATA ERROR F
4835	025360	077205			BB24:	SOB	R2,BB23	
4836	025362	000137	026012			JMP	@#BBER6	:BAD CONSTANT F
4837	025366	077220			BB25:	SOB	R2,BB22	
4838	025370	020405				CMP	R4,R5	
4839	025372	001402				BEQ	BB26	
4840	025374	000137	025610			JMP	@#BBER10	:BAD FPS.
4841								:EXPONENT DIFFERENCE=1
4842	025400				BB26:	LPERR		:SET UP THE LOOP ON ERROR ADDRESS.
4843	025400	104414				MOV	#BB27,@#\$TMP2	
4844	025402	012737	025430	001236		MOV	#3200,R4	
4845	025410	012704	003200			LDFPS	R4	:SET UP ACO OPERAND
4846	025414	170104				MOV	#BBPAT5,R0	
4847	025416	012700	026162			LDD	(R0),AC0	
4848	025422	172410				MOV	#BBPAT1,R0	:FSRC
4849	025424	012700	026122		BB27:	ADDD	(R0),AC0	:TEST INSTRUCTION
4850	025430	172010				STFPS	R5	
4851	025432	170205				MOV	#BBDATO,R0	:GET THE RESULT.
4852	025434	012700	026102			STD	AC0,(R0)	
4853	025440	174010				MOV	#BBP11,R1	:IS IT CORRECT?
4854	025442	012701	026222			MOV	#4,R2	
4855	025446	012702	000004		BB30:	CMP	(R0)+,(R1)+	
4856	025452	022021				BEQ	BB31	
4857	025454	001402				JMP	@#BBER7	:DATA ERROR D
4858	025456	000137	026046		BB31:	SOB	R2,BB30	
4859	025462	077205				CMP	R4,R5	:IS FPS CORRECT
4860	025464	020405				BEQ	BB32	
4861	025466	001402				JMP	@#BBERO	
4862	025470	000137	025570					:EXPONENT DIFFERENCE=100=144 (OCT)
4863					BB32:	LPERR		:SET UP THE LOOP ON ERROR ADDRESS.
4864	025474					MOV	#BB33,@#\$TMP2	
4865	025474	104414				MOV	#3200,R4	
4866	025476	012737	025524	001236		LDFPS	R4	:SET FIV,FIV AND FD
4867	025504	012704	003200			MOV	#BBPAT6,R0	:SET UP ACO OPERAND.
4868	025510	170104				LDD	(R0),AC0	
4869	025512	012700	026172			MOV	#BBPAT1,R0	:FSRC
4870	025516	172410			BB33:	ADDD	(R0),AC0	:TEST INSTRUCTION
4871	025520	012700	026122			STFPS	R5	
4872	025524	172010				MOV	#BBDATO,R0	:GET THE RESULT
4873	025526	170205						
4874	025530	012700	026102					

```

4875 025534 174010          STD      ACO,(R0)
4876 025536 012701 026172    MOV      #BBPAT6,R1      ;IS IT CORRECT
4877 025542 012702 000004    MOV      #4,R2
4878 025546 022021          BB34:   CMP      (R0)+,(R1)+
4879 025550 001402          BEQ      BB35
4880 025552 000137 026064    JMP      @#BBER8        ;DATA ERROR D
4881 025556 077205          BB35:   SOB      R2,BB34
4882 025560 020405          CMP      R4,R5          ;IS FPS CORRECT
4883 025562 001002          BNE      BBER0
4884 025564 000167 000442    JMP      BBDONE
4885 025570 010437 001242    BBER0:  MOV      R4,@#STMP4    ;FPS ERROR D
4886 025574 010537 001240    MOV      R5,@#STMP3
4887 025600 104001          1$:     ERROR    1
4888 025602 104413          RSETUP
4889
4890
4891
4892
4893 025604 000137 026232          JMP      @#BBDONE
4894 025610 010437 001242    BBER10: MOV      R4,@#STMP4    ;FPS ERROR F
4895 025614 010537 001240    MOV      R5,@#STMP3
4896 025620 104002          1$:     ERROR    2
4897 025622 104413          RSETUP
4898
4899
4900
4901
4902 025624 000137 026232          JMP      @#BBDONE
4903 025630 012737 026132 001242    BBER1:  MOV      #BBPAT2,@#STMP4 ;DATA ERROR D
4904 025636 012737 026132 001246    MOV      #BBPAT2,@#STMP6
4905 025644 012737 026122 001240    BBER11: MOV      #BBPAT1,@#STMP3
4906 025652 012737 026102 001244    MOV      #BBDATO,@#STMP5
4907 025660 104002          1$:     ERROR    2
4908 025662 000137 026232          JMP      @#BBDONE
4909 025666 012737 026152 001242    BBER2:  MOV      #BBPAT4,@#STMP4
4910 025674 012737 026212 001246    MOV      #BBP10,@#STMP6
4911 025702 000760          BR      BBER11
4912 025704 012737 026152 001242    BBER3:  MOV      #BBPAT4,@#STMP4 ;BAD CONSTANT D
4913 025712 012737 026212 001246    MOV      #BBP10,@#STMP6
4914 025720 012737 026122 001240    MOV      #BBPAT1,@#STMP3
4915 025726 012737 026102 001244    MOV      #BBDATO,@#STMP5
4916 025734 104002          1$:     ERROR    2
4917 025736 000535          BR      BBDONE
4918 025740 012737 026112 001242    BBER4:  MOV      #BBPAT0,@#STMP4 ;DATA ERROR F
4919 025746 012737 026112 001246    MOV      #BBPAT0,@#STMP6
4920 025754 012737 026122 001240    BBER40: MOV      #BBPAT1,@#STMP3
4921 025762 012737 026102 001244    MOV      #BBDATO,@#STMP5
4922 025770 104002          1$:     ERROR    2
4923 025772 000517          BR      BBDONE
4924 025774 012737 026142 001242    BBER5:  MOV      #BBPAT3,@#STMP4
4925 026002 012737 026202 001246    MOV      #BBP7,@#STMP6
4926 026010 000761          BR      BBER40
4927 026012 012737 026142 001242    BBER6:  MOV      #BBPAT3,@#STMP4 ;CONSTANT ERROR F
4928 026020 012737 026202 001246    MOV      #BBP7,@#STMP6
4929 026026 012737 026122 001240    MOV      #BBPAT1,@#STMP3
4930 026034 012737 026102 001244    MOV      #BBDATO,@#STMP5

```

```

4931 026042 104002 1$: ERROR 2
4932 026044 000472 BR BBDONE
4933 026046 012737 026162 001242 BBER7: MOV #BBPAT5,@#STMP4
4934 026054 012737 026122 001246 MOV #BBPAT11,@#STMP6
4935 026062 000670 BR BBER11
4936 026064 012737 026172 001242 BBER8: MOV #BBPAT6,@#STMP4
4937 026072 012737 026172 001246 MOV #BBPAT6,@#STMP6
4938 026100 000661 BR BBER11
4939 026102 000000 BBDATO: 0
4940 026104 000000 0
4941 026106 000000 0
4942 026110 000000 0
4943 026112 006400 BBPAT0: 6400 ;F(AC)=E(FSRC)+25=26
4944 026114 000000 0 ; =32(OCT)
4945 026116 000000 0
4946 026120 000000 0
4947 026122 000200 BBPAT1: 200 ;E(FSRC)=1
4948 026124 000000 0
4949 026126 000000 0
4950 026130 000000 0
4951 026132 016400 BBPAT2: 16400 ;E(AC)=E(FSRC)+57=58
4952 026134 000000 0 ; =72(OCT)
4953 026136 000000 0
4954 026140 000000 0
4955 026142 006200 BBPAT3: 6200 ;E(AC)=E(FSRC)+24=25
4956 026144 000000 0 ; =31(OCT)
4957 026146 000000 0
4958 026150 000000 0
4959 026152 016200 BBPAT4: 16200 ;E(AC)=E(FSRC)+56=57
4960 026154 000000 0 ; =71(OCT)
4961 026156 000000 0
4962 026160 000000 0
4963 026162 000400 BBPAT5: 400 ;E(AC)=E(FSRC)+1=2
4964 026164 000000 0
4965 026166 000000 0
4966 026170 000000 0
4967 026172 031200 BBPAT6: 31200 ;E(AC)=E(FSRC)+100=101
4968 026174 000000 0 ; =145(OCT)
4969 026176 000000 0
4970 026200 000000 0
4971 026202 006200 BBP7: 6200 ;BBPAT3 RES
4972 026204 000001 1
4973 026206 000000 0
4974 026210 000000 0
4975 026212 016200 BBP10: 16200 ;BBPAT4 RES
4976 026214 000000 0
4977 026216 000000 0
4978 026220 000001 1
4979 026222 000500 BBP11: 500 ;BBPAT5 RES
4980 026224 000000 0
4981 026226 000000 0
4982 026230 000000 0
4983 026232 BBDONE:
4984 026232 104413 RSETUP ;GO INITIALIZE THE FPS AND STACK; AND
4985 ;SEE IF THE USER HAS EXPRESSED
4986 ;THE DESIRE TO CHANGE THE SOFTWARE
    
```

4987
 4988
 4989
 4990
 4991
 4992
 4993
 4994
 4995
 4996
 4997 026234 000004
 4998
 4999 026236
 5000 026236 104414
 5001 026240 012704 003200
 5002 026244 170104
 5003 026246 012737 026266 001236
 5004 026254 012700 030116
 5005 026260 172410
 5006 026262 012700 030116
 5007 026266 172010
 5008 026270 170205
 5009 026272 012700 030076
 5010 026276 174010
 5011 026300 012701 030216
 5012 026304 012702 000004
 5013 026310 022021
 5014 026312 001415
 5015 026314 012700 030076
 5016 026320 012701 030146
 5017 026324 012702 000004
 5018 026330 022021
 5019 026332 001402
 5020 026334 000137 027326
 5021 026340 077205
 5022 026342 000137 027364
 5023 026346 077220
 5024 026350 052704 000010
 5025 026354 020405
 5026 026356 001402
 5027 026360 000137 027310
 5028
 5029 026364
 5030 026364 104414
 5031 026366 012704 003200
 5032 026372 170104
 5033 026374 012737 026414 001236
 5034 026402 012700 030126
 5035 026406 172410
 5036 026410 012700 030116
 5037 026414 172010
 5038 026416 170205
 5039 026420 012700 030076
 5040 026424 174010
 5041 026426 012701 030106
 5042 026432 012702 000004

```

;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).
:*****
:*TEST 34      ADDD WITH NEGATIVE OPRANDS TEST
:*
:*THIS IS A TEST OF THE ADDD INSTRUCTION
:*WITH NEGATIVE OPRANDS.  EVERY COMBINATION OF
:*OPERAND SIGNS IS TRIED.
:*
:*****
TST34: SCOPE
:BOTH OPRANDS NEGATIVE
DD1:
      LPERR                ;SET UP THE LOOP ON ERROR ADDRESS.
      MOV      #3200,R4    ;SET FIO, FIV, AND FD
      LDFPS   R4
      MOV      #DD2,@#STMP2
      MOV      #DDP1,R0    ;SET ACO OPERAND
      LDD     (R0),ACO
      MOV      #DDP1,R0    ;ESRC
      ADDD    (R0),ACO    ;TEST INSTRUCTION
      STFPS   R5          ;GET FPS
                        ;GET THE RESULT
      MOV      #DDDATO,R0
      STD     ACO,(R0)
      MOV      #DDP9,R1    ;IS IT CORRECT
      DD2:    MOV      #4,R2
      DD3:    CMP     (R0)+,(R1)+
      BEQ     DD6
      MOV      #DDDATO,R0    ;DID A ADD-SUB
      MOV      #DDP4,R1    ;FLOW A FAILURE
      DD4:    MOV      #4,R2
      CMP     (R0)+,(R1)+
      BEQ     DD5          ;216,442,500
      JMP     @DDDER1      ;DATA ERROR,D
      DD5:    SOB     R2,DD4
      JMP     @DDDER2      ;FLOW FAILURE,D
      DD6:    SOB     R2,DD3
      BIS     #10,R4
      CMP     R4,R5
      BEQ     DD7
      JMP     @DDDERO      ;BAD,FPS
;AC POS FSRC NEG      AC=-FSRC
DD7:
      LPERR                ;SET UP THE LOOP ON ERROR ADDRESS.
      MOV      #3200,R4    ;SET FIO, FIV, AND FD
      LDFPS   R4
      MOV      #DD8,@#STMP2
      MOV      #DDP2,R0    ;SET ACO OPERAND
      LDD     (R0),ACO
      MOV      #DDP1,R0    ;FSPC
      ADDD    (R0),ACO    ;TEST INSTRUCTION
      STFPS   R5          ;GET FPS
                        ;GET THE RESULT
      MOV      #DDDATO,R0
      STD     ACO,(R0)
      MOV      #DDP0,R1    ;IS IT CORRECT
      MOV      #4,R2
    
```

5043	026436	022021		DD10:	CMP	(R0)+,(R1)+	
5044	026440	001402			BEQ	DD11	
5045	026442	000137	027422		JMP	@WDDER3	;FLOW FAILURE
5046	026446	077205		DD11:	SOB	R2,DD10	
5047	026450	052704	000004		BIS	#4,R4	
5048	026454	020405			CMP	R4,R5	;FPS CORRECT?
5049	026456	001402			BEQ	DD12	
5050	026460	000137	027310		JMP	@WDDERO	;BAD FPS
5051							AC=-FSRC
5052	026464			DD12:	LPERR		;SET UP THE LOOP ON ERROR ADDRESS.
5053	026464	104414			MOV	#3200,R4	;SET FIU, FIV, AND FD
5054	026466	012704	003200		LDFPS	R4	
5055	026472	170104			MOV	#DD13,@#STMP2	
5056	026474	012737	026514	001236	MOV	#DDP1,R0	;SET ACO OPERAND
5057	026502	012700	030116		LDD	(R0),ACO	
5058	026506	172410			MOV	#DDP2,R0	;FSRC
5059	026510	012700	030126		ADD	(R0),ACO	;TEST INSTRUCTION
5060	026514	172010		DD13:	STFPS	R5	;GET FPS
5061	026516	170205			MOV	#DDDATO,R0	;GET THE RESULT
5062	026520	012700	030076		STD	ACO,(R0)	
5063	026524	174010			MOV	#DDP0,R1	;IS IT CORRECT
5064	026526	012701	030106		MOV	#4,R2	
5065	026532	012702	000004		CMP	(R0)+,(R1)+	
5066	026536	022021		DD14:	BEQ	DD15	
5067	026540	001402			JMP	@WDDER4	;FLOW FAILURE 216,440,121
5068	026542	000137	027460		SOB	R2,DD14	
5069	026546	077205		DD15:	BIS	#4,R4	
5070	026550	052704	000004		CMP	R4,R5	;EPS CORRECT?
5071	026554	020405			BEQ	DD16	
5072	026556	001402			JMP	@WDDERO	;BAD FPS
5073	026560	000137	027310			FSRC NEG	/AC/ > /FSRC/
5074				DD16:	LPERR		;SET UP THE LOOP ON ERROR ADDRESS.
5075	026564				MOV	#3200,R4	;SET FIV, FIV AND FD
5076	026564	104414			LDFPS	R4	
5077	026566	012704	003200		MOV	#DD17,@#STMP2	
5078	026572	170104			MOV	#DDP3,R0	;SET ACO OPERAND
5079	026574	012737	026614	001236	LDD	(R0),ACO	
5080	026602	012700	030136		MOV	#DDP6,R0	;ESPC
5081	026606	172410			ADD	(R0),ACO	;TEST INSTRUCTION
5082	026610	012700	030166		STFPS	R5	;GET FPS
5083	026614	172010		DD17:	MOV	#DDDATO,R0	;GET THE RESULT
5084	026616	170205			STD	ACO,(R0)	
5085	026620	012700	030076		MOV	#DDP7,R1	;IS IT CORRECT
5086	026624	174010			MOV	#4,R2	
5087	026626	012701	030176		CMP	(R0)+,(R1)+	
5088	026632	012702	000004	DD18:	BEQ	DD21	
5089	026636	022021			MOV	#DDDATO,R0	;FLOWS FAILURE
5090	026640	001415			MOV	#DDP8,R1	;216,440,101
5091	026642	012700	030076		MOV	#4,R2	;GET GENERATED
5092	026646	012701	030206		CMP	(R0)+,(R1)+	
5093	026652	012702	000004	DD19:	BEQ	DD20	
5094	026656	022021			JMP	@WDDER5	;DATA ERROR.
5095	026660	001402		DD20:	SOB	R2,DD19	
5096	026662	000137	027516		JMP	@WDDER6	
5097	026666	077205					
5098	026670	000137	027554				

```

5099 026674 077220 DD21: SOB R2,DD18
5100 026676 020405 CMP R4,R5 ;EPS CORRECT?
5101 026700 001402 BEQ DD22
5102 026702 000137 027310 JMP @#DDERO ;BAD FPS
5103 ;AC NEG FSRC POS /FSRC/ > /AC/
5104 026706 DD22: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
5105 026706 104414 MOV #3200,R4 ;SET FIO,FIV, AND FD
5106 026710 012704 003200 LDFPS R4
5107 026714 170104 MOV #DD23,@#STMP2
5108 026716 012737 026736 001236 MOV #DDP6,R0 ;SET ACO OPERAND
5109 026724 012700 030166 LDD (R0),ACO
5110 026730 172410 MOV #DDP5,R0 ;FSPC
5111 026732 012700 030136 DD23: ADDD (R0),ACO ;TEST INSTRUCTION
5112 026736 172010 STFPS R5 ;GET FPS
5113 026740 170205 MOV #DDDATO,R0 ;GET THE RESULT
5114 026742 012700 030076 STD ACO,(R0)
5115 026746 174010 MOV #DDP7,R1 ;IS IT CORRECT?
5116 026750 012701 030176 MOV #4,R2
5117 026754 012702 000004 DD24: CMP (R0)+,(R1)+
5118 026760 022021 BEQ DD27
5119 026762 001415 MOV #DDDATO,R0 ;FLO,S FAILURE
5120 026764 012700 030076 MOV #DDP8,R1 ;CONSTANT (NOT 57)
5121 026770 012701 030206 MOV #4,R2 ;216,042,101
5122 026774 012702 000004 DD25: CMP (R0),(R1)
5123 027000 021011 BEQ DD26
5124 027002 001402 DD26: SOB R2,DD25 ;DATA ERROR.
5125 027004 000137 027612 JMP @#DDER7
5126 027010 077205 DD27: SOB R2,DD24
5127 027012 000137 027650 CMP R4,R5 ;FPS CORRECT?
5128 027016 077220 BEQ DD30
5129 027020 020405 DD30: @#DDERO ;BAD FPS
5130 027022 001402 JMP FSRC NEG /AC/</FSRC/
5131 027024 000137 027310 ;ACO POS
5132 DD30: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
5133 027030 MOV #3200,R4 ;SET FIO,FIV,AND FD
5134 027030 104414 LDFPS R4
5135 027032 012704 003200 MOV #DD31,@#STMP2
5136 027036 170104 MOV #DDP4,R0 ;SET ACO OPERAND
5137 027040 012737 027060 001236 LDD (R0),ACO
5138 027046 012700 030146 MOV #DDP5,R0 ;FSPC
5139 027052 172410 DD31: ADDD (R0),ACO ;TEST INSTRUCTION
5140 027054 012700 030150 STFPS R5 ;GET FPS
5141 027060 172010 MOV #DDDATO,R0 ;GET THE RESULT
5142 027062 170205 STD ACO,(R0)
5143 027064 012700 030076 MOV #DDP8,R1 ;IS IT CORRECT
5144 027070 174010 MOV #4,R2
5145 027072 012701 030206 DD32: CMP (R0)+,(R1)+
5146 027076 012702 000004 BEQ DD35 ;ADD-SUB
5147 027102 022021 MOV #DDDATO,R0 ;FLOWAS FAILURE
5148 027104 001415 MOV #DDP7,R1 ;CON 216 N440 NOT 141
5149 027106 012700 030076 MOV #4,R2 ;GET GENERATED
5150 027112 012701 030176 DD33: CMP (R0)+,(R1)+ ;FOR THE ALLIGNMENT
5151 027116 012702 000004 BEQ DD34 ;FLOWS?
5152 027122 022021 JMP @#DDER9 ;DATA ERROR, D
5153 027124 001402
5154 027126 000137 027706
    
```

```

5155 027132 077205 DD34: SOB R2,DD33
5156 027134 000137 027744 JMP @WDDER10
5157 027140 077220 DD35: SOB R2,DD32
5158 027142 052704 000010 BIS #10,R4
5159 027146 020405 CMP R4,R5 ;FPS CORRECT?
5160 027150 001402 BEQ #DD36 ;BAD FPS
5161 027152 000137 027310 JMP @WDDERO
5162 :ACO NEG FSRC POS /FSRC/</AC/
5163 DD36: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
5164 027156 104414 MOV #3200,R4 ;SET FIO, FIV, AND FD
5165 027160 012704 003200 LDFPS R4
5166 027164 170104 LDFPS R4
5167 027166 012737 027206 001236 MOV #DD37,@#STMP2
5168 027174 012700 030156 MOV #DDP5,R0 ;SET ACO OPERAND
5169 027200 172410 LDD (R0),ACO
5170 027202 012700 030146 MOV #DDP4,R0 ;FSPC
5171 027206 172010 DD37: ADDD (R0),ACO ;TEST INSTRUCTION
5172 027210 170205 STFPS R5 ;GET FPS
5173 027212 012700 030076 MOV #DDDATO,R0 ;GET THE RESULT
5174 027216 174010 STD ACO,(R0)
5175
5176 027220 012701 030206 MOV #DDP8,R1 ;IS IT CORRECT
5177 027224 012702 000004 DD38: MOV #4,R2
5178 027230 022021 CMP (R0)+,(R1)+
5179 027232 001415 BEQ DD41
5180 027234 012700 030076 MOV #DDDATO,R0 ;ADD SUB
5181 027240 012701 030176 MOV #DDP7,R1 ;FLOWS FAILURES
5182 027244 012702 000004 DD39: MOV #4,R2 ;GET 216,042,141
5183 027250 022021 CMP (R0)+,(R1)+ ;FOR THE ALLIGNMENT
5184 027252 001402 BEQ DD40 ;FLOWS?
5185 027254 000137 030002 DD40: SOB R2,DD39 ;DATA ERROR. D
5186 027260 077205 DD40: SOB R2,DD39 ;BAD CONSTANT.D
5187 027262 000137 030040 DD41: SOB R2,DD38
5188 027266 077220 DD41: SOB R2,DD38
5189 027270 052704 000010 BIS #10,R4 ;FPS CORRECT?
5190 027274 020405 CMP R4,R5 ;BAD FPS
5191 027276 001402 BEQ DD42 ;FPS ERROR
5192 027300 000137 027310 DD42: JMP @WDDERO
5193 027304 000137 030226 DDERO: MOV #DDDDONE
5194 027310 010437 001242 MOV R4,@#STMP4
5195 027314 010537 001240 MOV R5,@#STMP3
5196 027320 104002 1$: ERROR 2
5197 027322 000137 030226 DD42: JMP @WDDDDONE
5198 027326
5199 027326 012737 030116 001240 DDER1: MOV #DDP1,@#STMP3
5200 027334 012737 030116 001242 MOV #DDP1,@#STMP4
5201 027342 012737 030076 001244 MOV #DDDATO,@#STMP5
5202 027350 012737 030116 001246 MOV #DDP1,@#STMP6
5203 027356 104002 1$: ERROR 2
5204 027360 000137 030226 DDER2: JMP @WDDDDONE
5205 027364
5206 027364 012737 030116 001240 MOV #DDP1,@#STMP3
5207 027372 012737 030116 001242 MOV #DDP1,@#STMP4
5208 027400 012737 030076 001244 MOV #DDDATO,@#STMP5
5209 027406 012737 030216 001246 MOV #DDP9,@#STMP6
5210 027414 104002 1$: ERROR 2
    
```


5211	027416	000137	030226		JMP	@#DDDDONE
5212	027422				DDER3:	
5213	027422	012737	030116	001240	MOV	#DDP1,@#STMP3
5214	027430	012737	030126	001242	MOV	#DDP2,@#STMP4
5215	027436	012737	030076	001244	MOV	#DDDATO,@#STMP5
5216	027444	012737	030106	001246	MOV	#DDPO,@#STMP6
5217	027452	104002			1\$:	ERROR
5218	027454	000137	030226		JMP	@#DDDDONE
5219	027460				DDER4:	
5220	027460	012737	030126	001240	MOV	#DDP2,@#STMP3
5221	027466	012737	030116	001242	MOV	#DDP1,@#STMP4
5222	027474	012737	030076	001244	MOV	#DDDATO,@#STMP5
5223	027502	012737	030106	001246	MOV	#DDPO,@#STMP6
5224	027510	104002			1\$:	ERROR
5225	027512	000137	030226		JMP	@#DDDDONE
5226	027516				DDER5:	
5227	027516	012737	030166	001240	MOV	#DDP6,@#STMP3
5228	027524	012737	030136	001242	MOV	#DDP3,@#STMP4
5229	027532	012737	030076	001244	MOV	#DDDATO,@#STMP5
5230	027540	012737	030176	001246	MOV	#DDP7,@#STMP6
5231	027546	104002			1\$:	ERROR
5232	027550	000137	030226		JMP	@#DDDDONE
5233	027554				DDER6:	
5234	027554	012737	030166	001240	MOV	#DDP6,@#STMP3
5235	027562	012737	030136	001242	MOV	#DDP3,@#STMP4
5236	027570	012737	030076	001244	MOV	#DDDATO,@#STMP5
5237	027576	012737	030176	001246	MOV	#DDP7,@#STMP6
5238	027604	104002			1\$:	ERROR
5239	027606	000137	030226		JMP	@#DDDDONE
5240	027612				DDER7:	
5241	027612	012737	030136	001240	MOV	#DDP3,@#STMP3
5242	027620	012737	030166	001242	MOV	#DDP6,@#STMP4
5243	027626	012737	030076	001244	MOV	#DDDATO,@#STMP5
5244	027634	012737	030176	001246	MOV	#DDP7,@#STMP6
5245	027642	104002			1\$:	ERROR
5246	027644	000137	030226		JMP	@#DDDDONE
5247	027650				DDER8:	
5248	027650	012737	030136	001240	MOV	#DDP3,@#STMP3
5249	027656	012737	030166	001242	MOV	#DDP6,@#STMP4
5250	027664	012737	030076	001244	MOV	#DDDATO,@#STMP5
5251	027672	012737	030176	001246	MOV	#DDP7,@#STMP6
5252	027700	104002			1\$:	ERROR
5253	027702	000137	030226		JMP	@#DDDDONE
5254	027706				DDER9:	
5255	027706	012737	030156	001240	MOV	#DDP5,@#STMP3
5256	027714	012737	030146	001242	MOV	#DDP4,@#STMP4
5257	027722	012737	030076	001244	MOV	#DDDATO,@#STMP5
5258	027730	012737	030206	001246	MOV	#DDP8,@#STMP6
5259	027736	104002			1\$:	ERROR
5260	027740	000137	030226		JMP	@#DDDDONE
5261	027744				DDER10:	
5262	027744	012737	030156	001240	MOV	#DDP5,@#STMP3
5263	027752	012737	030146	001242	MOV	#DDP4,@#STMP4
5264	027760	012737	030076	001244	MOV	#DDDATO,@#STMP5
5265	027766	012737	030206	001246	MOV	#DDP8,@#STMP6
5266	027774	104002			1\$:	ERROR

```

5267 027776 000137 030226          JMP      @#DDDDONE
5268 030002          DDERR11: MOV     #DDP4,@#STMP3
5269 030002 012737 030146 001240      MOV     #DDP5,@#STMP4
5270 030010 012737 030156 001242      MOV     #DDDATO,@#STMP5
5271 030016 012737 030076 001244      MOV     #DDP8,@#STMP6
5272 030024 012737 030206 001246      1$:    ERROR 2
5273 030032 104002          JMP     @#DDDDONE
5274 030034 000137 030226          DDERR12: MOV     #DDP4,@#STMP3
5275 030040          MOV     #DDP5,@#STMP4
5276 030040 012737 030146 001240      MOV     #DDDATO,@#STMP5
5277 030046 012737 030156 001242      MOV     #DDP8,@#STMP6
5278 030054 012737 030076 001244      1$:    ERROR 2
5279 030062 012737 030206 001246      JMP     @#DDDDONE
5280 030070 104002          DDDATO: 0
5281 030072 000137 030226          0
5282 030076 000000          0
5283 030100 000000          0
5284 030102 000000          0
5285 030104 000000          DDP0:   0
5286 030106 000000          0
5287 030110 000000          0
5288 030112 000000          0
5289 030114 000000          DDP1:  100200      :-DDP2
5290 030116 100200          0
5291 030120 000000          0
5292 030122 000000          0
5293 030124 000000          DDP2:   200        :-DDP1
5294 030126 000200          0
5295 030130 000000          0
5296 030132 000000          DDP3:  1100       :EXP=4
5297 030134 000000          0        :FRAC=...110...
5298 030136 001100          0
5299 030140 000000          0
5300 030142 000000          DDP4:   600       :EXP=3
5301 030144 000000          0        :FRAC=...100...
5302 030146 000600          0
5303 030150 000000          0
5304 030152 000000          DDP5:  101100     :-DDP3
5305 030154 000000          0
5306 030156 101100          0
5307 030160 000000          0
5308 030162 000000          DDP6:  100600     :-DDP4
5309 030164 000000          0
5310 030166 100600          0
5311 030170 000000          0
5312 030172 000000          DDP7:  1000       :DDP3+DDP6
5313 030174 000000          0
5314 030176 001000          0
5315 030200 000000          DDP8:  101000     :DDP5+DDP4
5316 030202 000000          0
5317 030204 000000          0
5318 030206 101000          DDP9:  100400     :DDP1+DDP1
5319 030210 000000          0
5320 030212 000000          0
5321 030214 000000          0
5322 030216 100400          0
    
```

```

5323 030220 000000      0
5324 030222 000000      0
5325 030224 000000      0
5326 030226 000005
5327
5328
5329
5330
5331
5332
5333
5334
5335 030230 000004
5336
5337 030232
5338 030232 104414
5339 030234 012704 003200
5340 030240 170104
5341 030242 012737 030262 001236
5342 030250 012700 030742
5343 030254 172410
5344 030256 012700 030742
5345 030262 173010
5346 030264 170205
5347 030266 012700 030720
5348 030272 174010
5349 030274 012701 030730
5350 030300 012702 000004
5351 030304 022021
5352 030306 001415
5353 030310 012700 030720
5354
5355 030314 012701 030752
5356 030320 012702 000004
5357 030324 022021
5358 030326 001402
5359 030330 000137 030530
5360 030334 077205
5361 030336 000137 030566
5362 030342 077220
5363 030344 052704 000004
5364 030350 020405
5365 030352 001402
5366 030354 000137 030512
5367
5368 030360
5369 030360 104414
5370 030362 012704 003200
5371 030366 170104
5372 030370 012737 030410 001236
5373 030376 012700 030762
5374 030402 172410
5375 030404 012700 030762
5376 030410 173010
5377 030412 170205
5378 030414 012700 030720

DDDONE:          RESET
:*****
:*TEST 35      SUBD TEST
:*
:* THIS IS A TEST OF THE SUBD INSTRUCTION.
:* BOTH A POSITIVE AND A NEGATIVE NUMBER
:* IS SUBTRACTED FROM IT SELF
:*
:*****
TST35:  SCOPE
        USE POSITIVE OPERANDS
EE1:
        LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
        MOV           #3200,R4          ;SET FIU, FIV, AND FD
        LDFPS        R4
        MOV           #EE2,@#$TMP2
        MOV           #EEP1,R0          ;SET ACO OPERAND
        LDD          (R0),ACO
        MOV           #EEP1,R0          ;FSPC
        SUBD         (R0),ACO          ;TEST INSTRUCTION
        STFPS        R5                ;GET FPS
        MOV           #EEDATO,R0       ;GET THE RESULT
        STD          ACO,(R0)
        MOV           #EEO,R1          ;IS IT CORRECT?
        MOV           #4,R2
        EE3:         CMP          (R0)+,(R1)+
        BEQ          EE6
        MOV           #EEDATO,R0       ;DID A BAD
        MOV           #EEP2,R1          ;CONSTANT (NOT 57)
        MOV           #4,R2            ;GET GENERATED
        EE4:         CMP          (R0)+,(R1)+ ;FOR THE ALLIGNMENT
        BEQ          EE5                ;FLOWS?
        JMP          @#LEER1           ;DATA ERROR.D
        EE5:         SOB          R2,EE4 ;BAD CONSTANT.D
        EE6:         SOB          R2,EE3
        BIS          #4,R4
        CMP          R4,R5              ;FPS CORRECT?
        BEQ          EE7
        JMP          @#EEERO           ;BAD FPS
        EE7:         ;USE NEGATIVE OPERANDS
        LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
        MOV           #3200,R4          ;SET FIO, FIV, AND FD
        LDFPS        R4
        MOV           #EE8,@#$TMP2
        MOV           #EEP3,R0          ;SET ACO OPERAND
        LDD          (R0),ACO
        MOV           #EEP3,R0          ;FSPC
        EE8:         SUBD         (R0),ACO ;TEST INSTRUCTION
        STFPS        R5                ;GET FPS
        MOV           #EEDATO,R0       ;GET THE RESULT
    
```

5379	030420	174010			STD	AC0,(R0)	
5380	030422	012701	030730		MOV	#EEP0,R1	;IS IT CORRECT?
5381	030426	012702	000004		MOV	#4,R2	
5382	030432	022021		EE9:	CMP	(R0)+,(R1)+	
5383	030434	001415			BEQ	EE12	
5384	030436	012700	030720		MOV	#EEDATO,R0	;DID A BAD
5385	030442	012701	030772		MOV	#EEP4,R1	;CONSTANT (NOT 57)
5386	030446	012702	000004		MOV	#4,R2	;GET GENERATED
5387	030452	022021		EE10:	CMP	(R0)+,(R1)+	;FOR THE ALLIGNMENT
5388	030454	001402			BEQ	EE11	;FLOWS?
5389	030456	000137	030624		JMP	@#EEER3	;DATA ERROR.D
5390	030462	077205		EE11:	SOB	R2,EE10	
5391	030464	000137	030662		JMP	@#EEER4	;BAD CONSTANT.D
5392	030470	077220		EE12:	SOB	R2,EE9	
5393	030472	052704	000004		BIS	#4,R4	
5394	030476	020405			CMP	R4,R5	;FPS CORRECT?
5395	030500	001402			BEQ	EE13	
5396	030502	000137	030512		JMP	@#EEERO	;BAD FPS.
5397	030506	000137	031002	EE13:	JMP	@#EEDONE	
5398	030512	010437	001242	EEERO:	MOV	R4,@#STMP4	;BAD FPS
5399	030516	010537	001240		MOV	R5,@#STMP3	
5400	030522	104002		1\$:	ERROR	2	
5401	030524	000137	031002		JMP	@#EEDONE	
5402	030530			EEER1:			
5403	030530	012737	030742	001240	MOV	#EEP1,@#STMP3	
5404	030536	012737	030742	001242	MOV	#EEP1,@#STMP4	
5405	030544	012737	030720	001244	MOV	#EEDATO,@#STMP5	
5406	030552	012737	030730	001246	MOV	#EEP0,@#STMP6	
5407	030560	104002		1\$:	ERROR	2	
5408	030562	000137	031002		JMP	@#EEDONE	
5409	030566			EEER2:			
5410	030566	012737	030742	001240	MOV	#EEP1,@#STMP3	
5411	030574	012737	030742	001242	MOV	#EEP1,@#STMP4	
5412	030602	012737	030720	001244	MOV	#EEDATO,@#STMP5	
5413	030610	012737	030730	001246	MOV	#EEP0,@#STMP6	
5414	030616	104002		1\$:	ERROR	2	
5415	030620	000137	031002		JMP	@#EEDONE	
5416	030624			EEER3:			
5417	030624	012737	030762	001240	MOV	#EEP3,@#STMP3	
5418	030632	012737	030762	001242	MOV	#EEP3,@#STMP4	
5419	030640	012737	030720	001244	MOV	#EEDATO,@#STMP5	
5420	030646	012737	030730	001246	MOV	#EEP0,@#STMP6	
5421	030654	104002		1\$:	ERROR	2	
5422	030656	000137	031002		JMP	@#EEDONE	
5423	030662			EEER4:			
5424	030662	012737	030762	001240	MOV	#EEP3,@#STMP3	
5425	030670	012737	030762	001242	MOV	#EEP3,@#STMP4	
5426	030676	012737	030720	001244	MOV	#EEDATO,@#STMP5	
5427	030704	012737	030730	001246	MOV	#EEP0,@#STMP6	
5428	030712	104002		1\$:	ERROR	2	
5429	030714	000137	031002		JMP	@#EEDONE	
5430	030720	000000		EEDATO:	0		
5431	030722	000000			0		
5432	030724	000000			0		
5433	030726	000000			0		
5434	030730	000000		EEPO:	0		

5435 030732 000000
 5436 030734 000000
 5437 030736 000000
 5438 030740 000000
 5439 030742 000200
 5440 030744 000000
 5441 030746 000000
 5442 030750 000000
 5443 030752 000400
 5444 030754 000000
 5445 030756 000000
 5446 030760 000000
 5447 030762 100200
 5448 030764 000000
 5449 030766 000000
 5450 030770 000000
 5451 030772 100400
 5452 030774 000000
 5453 030776 000000
 5454 031000 000000
 5455 031002
 5456 031002 104413
 5457
 5458
 5459
 5460
 5461
 5462
 5463
 5464
 5465
 5466
 5467
 5468
 5469
 5470
 5471 031004 000004
 5472
 5473 031006
 5474 031006 104414
 5475 031010 012704 003200
 5476 031014 170104
 5477 031016 012737 031036 001236
 5478 031024 012700 031330
 5479 031030 172410
 5480 031032 012700 031340
 5481 031036 172010
 5482 031040 170205
 5483 031042 012700 031300
 5484 031046 174010
 5485 031050 012701 031350
 5486 031054 012702 000004
 5487 031060 022021
 5488 031062 001401
 5489 031064 000466
 5490 031066 077204

00000 0
 00000 0
 00000 0
 EEP1: 200
 00000 0
 00000 0
 00000 0
 EEP2: 400
 00000 0
 00000 0
 00000 0
 EEP3: 100200
 00000 0
 00000 0
 00000 0
 EEP4: 100400
 00000 0
 00000 0
 00000 0
 EEDONE: 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?).

 ;*TEST 36 NORMALIZE ALGORITHM TEST
 ;*

;* THIS IS A TEST OF THE NORMALIZE
 ;* FLOW ALGORITHM. TWO PATTERNS ARE USED,
 ;* FIRST THE MINIMUM SITUATION REQUIRING ONE
 ;* LEFT SHIFT AND THEN THE MAXIMUM SITUATION
 ;* REQUIRING 56 SHIFTS.
 ;*

TST36: SCOPE
 ;USE DATA PATTERNS THAT REQUIRE ONLY ONE LEFT SHIFT TO NORMALIZE
 FF1:

LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
 MOV #3200,R4 ;SET FIO, FIV, AND FD
 LDFPS R4
 MOV #FF2,@#STMP2
 LDD (R0),ACO ;SET ACO OPERAND
 MOV #FFP2,R0
 MOV (R0),ACO ;FSPC
 ADDD (R0),ACO ;TEST INSTRUCTION
 STFPS R5 ;GET FPS
 MOV #FFDATO,R0 ;GET THE RESULT
 STD ACO,(R0)
 MOV #FFP4,R1 ;IS IT CORRECT
 MOV #4,R2
 FF2: CMP (R0)+,(R1)+
 BEQ FF4
 BR FFER2 ;BAD DATA
 FF3: SOB R2,FF3
 FF4:

```

5491 031070 020405          CMP      R4,R5          ;FPS CORRECT?
5492 031072 001401          BEQ      FF5
5493 031074 000435          BR       FFER0         ;BAD FPS
5494
5495          ;USE DATA PATTERNS WHICH REQUIRE 56 LEFT SHIFTS TO NORMALIZE
5496 031076          ;THE RESULT
5497 031076 104414          FF5:      LPERR
5498 031100 012704 003200          MOV      #3200,R4      ;SET UP THE LOOP ON ERROR ADDRESS.
5499 031104 170104          LDFPS   R4             ;SET FIU, FIV, AND FD
5500 031106 012737 031126 001236          MOV      #FF6,@#STMP2
5501 031114 012700 031310          MOV      #FFP0,R0     ;SET ACO OPERAND
5502 031120 172410          LDD     (R0),ACO
5503 031122 012700 031320          MOV      #FFP1,R0     ;FSRC
5504 031126 172010          FF6:      ADDD    (R0),ACO ;TEST INSTRUCTION
5505 031130 170205          STFPS  R5             ;GET FPS
5506 031132 012700 031300          MOV      #FFDATO,R0  ;GET THE RESULT
5507 031136 174010          STD     ACO,(R0)
5508 031140 012701 031350          MOV      #FFP4,R1     ;IS IT CORRECT
5509 031144 012702 000004          MOV      #4,R2
5510 031150 022021          FF7:      CMP      (R0)+,(R1)+
5511 031152 001401          BEQ      FF10
5512 031154 000413          BR       FFER1         ;BATA
5513 031156 077204          FF10:     SOB     R2,FF7
5514 031160 020405          CMP      R4,R5          ;FPS CORRECT?
5515 031162 001401          BEQ      FF11
5516 031164 000401          BR       FFER0         ;BAD FPS
5517 031166 000474          FF11:     BR       FFDONE
5518
5519 031170 010537 001240          FFER0:    MOV      R5,@#STMP3
5520 031174 010437 001242          MOV      R4,@#STMP4
5521 031200 104002          1$:      ERROR   2
5522 031202 000466          BR       FFDONE
5523
5524 031204          FFER1:
5525 031204 012737 031320 001240          MOV      #FFP1,@#STMP3
5526 031212 012737 031310 001242          MOV      #FFP0,@#STMP4
5527 031220 012737 031300 001244          MOV      #FFDATO,@#STMP5
5528 031226 012737 031350 001246          MOV      #FFP4,@#STMP6
5529 031234 104001          1$:      ERROR   1
5530 031236 000137 031360          JMP      @#FFDONE
5531
5532 031242          FFER2:
5533 031242 012737 031340 001240          MOV      #FFP3,@#STMP3
5534 031250 012737 031330 001242          MOV      #FFP2,@#STMP4
5535 031256 012737 031300 001244          MOV      #FFDATO,@#STMP5
5536 031264 012737 031350 001246          MOV      #FFP4,@#STMP6
5537 031272 104001          1$:      ERROR   1
5538 031274 000137 031360          JMP      @#FFDONE
5539
5540          FFDATO: 0
5541 031300 000000          0
5542 031302 000000          0
5543 031304 000000          0
5544 031306 000000          0
5545
5546 031310 016000          FFPO:   16000
    
```

5547 031312 000000
 5548 031314 000000
 5549 031316 000001
 5550 031320 116000
 5551 031322 000000
 5552 031324 000000
 5553 031326 000000
 5554 031330 000500
 5555 031332 000000
 5556 031334 000000
 5557 031336 000000
 5558 031340 100400
 5559 031342 000000
 5560 031344 000000
 5561 031346 000000
 5562 031350 000200
 5563 031352 000000
 5564 031354 000000
 5565 031356 000000

0
 0
 1
 FFP1: 116000
 0
 0
 0
 FFP2: 500
 0
 0
 0
 FFP3: 100400
 0
 0
 0
 FFP4: 200
 0
 0
 0

;FFP4=FFP0+FFP1
 ; =FFP3+FFP4

5566
 5567 031360
 5568
 5569
 5570 031360
 5571
 5572
 5573
 5574
 5575
 5576
 5577
 5578
 5579
 5580
 5581
 5582
 5583
 5584
 5585
 5586
 5587
 5588
 5589
 5590
 5591

FFDONE:
 TST37:

 :FLOATING POINT SECOND PART

 :*TEST 37 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
 :*
 :*****

5592 031360 000004
 5593
 5594
 5595
 5596 031362
 5597 031362 104414 003200
 5598 031364 012704 003200
 5599 031370 170104
 5600 031372 012737 031412 001236
 5601 031400 012700 033142
 5602 031404 172410

TST37: 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,@#STMP2
 MOV #HHP0,R0 ;SET ACO OPERAND
 LDD (R0),ACO

```

5603 031406 012700 033152
5604 031412 172010
5605 031414 170205
5606 031416 012700 033132
5607 031422 174010
5608 031424 012701 033162
5609 031430 012702 000004
5610 031434 022021
5611 031436 001415
5612 031440 012700 033132
5613 031444 012701 033172
5614 031450 012702 000004
5615 031454 022021
5616 031456 001402
5617 031460 000137 032152
5618 031464 077205
5619 031466 000137 032220
5620 031472 077220
5621 031474 020405
5622 031476 001402
5623 031500 000137 032266
5624
5625
5626
5627
5628
5629 031504
5630 031504 104414
5631 031506 012704 043200
5632
5633 031512 170104
5634 031514 012737 031542 001236
5635 031522 012737 033062 000244
5636 031530 012700 033212
5637 031534 172410
5638 031536 012700 033222
5639 031542 172010
5640 031544 170205
5641 031546 012700 033132
5642 031552 174010
5643 031554 012701 033202
5644 031560 012702 000004
5645 031564 022021
5646 031566 001402
5647 031570 000137 032334
5648 031574 077205
5649 031576 052704 100004
5650 031602 020405
5651 031604 001402
5652 031606 000137 032402
5653
5654
5655 031612
5656 031612 104414
5657
5658 031614 012704 043200

HH2:  MOV #HHP1,R0           ;FSPC
      ADDD (R0),ACO         ;TEST INSTRUCTION
      STFPS R5              ;GET FPS
      MOV #HHDATA,R0       ;GET THE RESULT
      STD ACO,(R0)
      MOV #HHP2,R1         ;IS IT CORRECT
      MOV #4,R2
      CMP (R0)+,(R1)+
      BEQ HH6
      MOV #HHDATA,R0       ;DID FLOW GO
      MOV #HHP3,R1         ;FROM STATE 663
      MOV #4,R2            ;TO 313 INSTEAD
      CMP (R0)+,(R1)+     ;OF TO 353
      BEQ HH5
      JMP @#HHERO
HH3:  MOV #HHDATA,R0
      MOV #HHP3,R1
      MOV #4,R2
      CMP (R0)+,(R1)+
      BEQ HH5
      JMP @#HHERO
HH4:  SOB R2,HH4
      JMP @#HHER1
HH5:  SOB R2,HH3
      CMP R4,R5             ;FPS CORRECT?
      BEQ HH7
      JMP @#HHERO
      ;THIS IS A TEST OF THE ABILITY
      ;OF NORMALIZE TO PRODUCE A ZERO EXP. AND
      ;OF THE R\T ALGORITHM TO PROPERLY SET THE FPS
HH7:  LPERR                 ;SET UP THE LOOP ON ERROR ADDRESS.
      MOV #043200,R4       ;SET FIU,FIV,AND FD
                               ;FID
      LDFPS R4
      MOV #HH8,@#STMP2     ;IN CASE UNDERFLOW
      MOV #HHTRAP,@#FPVECT ;TRAP OCCURS
      MOV #HHP5,R0         ;SET ACO OPERAND
      LDD (R0),ACO
      MOV #HHP6,R0         ;FSPC
      ADDD (R0),ACO         ;TEST INSTRUCTION
      STFPS R5              ;GET FPS
      MOV #HHDATA,R0       ;GET THE RESULT
      STD ACO,(R0)
      MOV #HHP4,R1         ;IS IT CORRECT
      MOV #4,R2
      CMP (R0)+,(R1)+
      BEQ HH10
      JMP @#HHER2
HH8:  SOB R2,HH8
      BIS #100004,R4       ;FPS CORRECT?
      CMP R4,R5
      BEQ HH11
      JMP @#HHER3
      ;THIS IS A TEST OF THE R\T ALGORITHM'S
      ;ABILITY TO SET BOTH N AND Z ON A - 0 RESULT.
HH9:  LPERR                 ;SET UP THE LOOP ON ERROR ADDRESS.
      MOV #043200,R4       ;SET FIV, FIV, AND FD
    
```



```

5659 031620 170104 LDFPS R4
5660 031622 012737 031642 001236 MOV #HH12,@#$TMP2
5661 031630 012700 033242 MOV #HHP8,R0 ;SET ACO OPERAND
5662 031634 172410 LDD (R0),ACO
5663 031636 012700 033252 MOV #HHP9,R0 ;FSPC
5664 031642 172010 HH12: ADDD (R0),ACO ;TEST INSTRUCTION
5665 031644 170205 STFPS R5 ;GET FPS
5666 031646 012700 033132 MOV #HHDATO,R0 ;GET THE RESULT
5667 031652 174010 STD ACO,(R0)
5668 031654 012701 033232 MOV #HHP7,R1 ;IS IT CORRECT
5669 031660 012702 000004 HH13: MOV #4,R2
5670 031664 022021 CMP (R0)+,(R1)+
5671 031666 001415 BEQ HH16
5672 031670 012700 033132 MOV #HHDATO,R0
5673 031674 012701 033202 MOV #HHP4,R1
5674 031700 012702 000004 HH14: MOV #4,R2
5675 031704 022021 CMP (R0)+,(R1)+
5676 031706 001402 BEQ HH15
5677 031710 000137 032450 HH15: JMP @#HHER4
5678 031714 077205 SOB R2,HH14
5679 031716 000137 032516 HH16: JMP @#HHER5
5680 031722 077220 SOB R2,HH13
5681 031724 052704 100014 BIS #100014,R4 ;FPS CORRECT?
5682 031730 020405 CMP R4,R5
5683 031732 001402 BEQ HH17
5684 031734 000137 032564 JMP @#HHER6
5685 ;TEST THAT CC ARE CLEARED BY R\T
5686 031740 HH17: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
5687 031740 104414 MOV #00200,R4 ;SET FIV, FIV, AND FD
5688 031742 012704 000200 LDFPS R4
5689 031746 170104 MOV #HH18,@#$TMP2
5690 031750 012737 031776 001236 MOV #FPSPUR,@#FPVECT
5691 031756 012737 062524 000244 MOV #HHP8,R0 ;SET ACO OPERAND
5692 031764 012700 033242 LDD (R0),ACO
5693 031770 172410 MOV #HHP8,R0 ;FSPC
5694 031772 012700 033242 HH18: ADDD (R0),ACO ;TEST INSTRUCTION
5695 031776 172010 STFPS R5 ;GET FPS
5696 032000 170205 MOV #HHDATO,R0 ;GET THE RESULT
5697 032002 012700 033132 STD ACO,(R0)
5698 032006 174010 MOV #HHP10,R1 ;IS IT CORRECT
5699 032010 012701 033262 HH19: MOV #4,R2
5700 032014 012702 000004 CMP (R0)+,(R1)+
5701 032020 022021 BEQ HH20
5702 032022 001402 JMP @#HHER7
5703 032024 000137 032632 HH20: SOB R2,HH19
5704 032030 077205 BIS #00000,R4 ;FPS CORRECT?
5705 032032 052704 000000 CMP R4,R5
5706 032036 020405 BEQ HH21
5707 032040 001402 JMP @#HHER8
5708 032042 000137 032700 ;TEST THAT N IS SET BY R\T
5709 HH21: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
5710 032046 104414 MOV #3200,R4 ;SET FIV, FIV, AND FD
5711 032046 012704 003200 LDFPS R4
5712 032050 170104 MOV #HH22,@#$TMP2
5713 032054 012737 032076 001236

```

```

5715 032064 012700 033212          MOV      #HHP5,R0          ;SET ACO OPERAND
5716 032070 172410          LDD      (R0),ACO
5717 032072 012700 033212          MOV      #HHP5,R0          ;FSPC
5718 032076 172010          HH22:   ADDD     (R0),ACO    ;TEST INSTRUCTION
5719 032100 170205          STFPS    R5                ;GET FPS
5720 032102 012700 033132          MOV      #HHDATA,R0       ;GET THE RESULT
5721 032106 174010          STD      ACO,(R0)
5722 032110 012701 033272          MOV      #HHP11,R1        ;IS IT CORRECT
5723 032114 012702 000004          MOV      #4,R2
5724 032120 022021          HH23:   CMP      (R0)+,(R1)+
5725 032122 001402          BEQ      HH24
5726 032124 000137 032746          JMP      @HHER9
5727 032130 077205          HH24:   SOB      R2,HH23
5728 032132 052704 000010          BIS      #10,R4
5729 032136 020405          CMP      R4,R5            ;FPS CORRECT?
5730 032140 001402          BEQ      HH25
5731 032142 000137 033014          JMP      @HHER10
5732 032146 000137 033302          HH25:   JMP      @HHDONE
5733 032152
5734 032152 010537 001252          HHER0:  MOV      R5,@$TMP10
5735 032156 010437 001254          MOV      R4,@$TMP11
5736 032162 012737 033152 001240          MOV      #HHP1,@$TMP3
5737 032170 012737 033142 001242          MOV      #HHP0,@$TMP4
5738 032176 012737 033132 001244          MOV      #HHDATA,@$TMP5
5739 032204 012737 033162 001246          MOV      #HHP2,@$TMP6
5740 032212 104002          1$:     ERROR    2
5741 032214 000137 033302          JMP      @HHDONE
5742 032220
5743 032220 010537 001252          HHER1:  MOV      R5,@$TMP10
5744 032224 010437 001254          MOV      R4,@$TMP11
5745 032230 012737 033152 001240          MOV      #HHP1,@$TMP3
5746 032236 012737 033142 001242          MOV      #HHP0,@$TMP4
5747 032244 012737 033132 001244          MOV      #HHDATA,@$TMP5
5748 032252 012737 033162 001246          MOV      #HHP2,@$TMP6
5749 032260 104002          1$:     ERROR    2
5750 032262 000137 033302          JMP      @HHDONE
5751 032266
5752 032266 010537 001252          HHER00: MOV      R5,@$TMP10
5753 032272 010437 001254          MOV      R4,@$TMP11
5754 032276 012737 033152 001240          MOV      #HHP1,@$TMP3
5755 032304 012737 033142 001242          MOV      #HHP0,@$TMP4
5756 032312 012737 033132 001244          MOV      #HHDATA,@$TMP5
5757 032320 012737 033162 001246          MOV      #HHP2,@$TMP6
5758 032326 104002          1$:     ERROR    2
5759 032330 000137 033302          JMP      @HHDONE
5760 032334
5761 032334 010537 001252          HHER2:  MOV      R5,@$TMP10
5762 032340 010437 001254          MOV      R4,@$TMP11
5763 032344 012737 033222 001240          MOV      #HHP6,@$TMP3
5764 032352 012737 033212 001242          MOV      #HHP5,@$TMP4
5765 032360 012737 033132 001244          MOV      #HHDATA,@$TMP5
5766 032366 012737 033202 001246          MOV      #HHP4,@$TMP6
5767 032374 104002          1$:     ERROR    2
5768 032376 000137 033302          JMP      @HHDONE
5769 032402
5770 032402 010537 001252          HHER3:  MOV      R5,@$TMP10
    
```


5827	032764	012737	033212	001242	MOV	#HHP5,@#STMP4	
5828	032772	012737	033132	001244	MOV	#HHDATA,@#STMP5	
5829	033000	012737	033272	001246	MOV	#HHP11,@#STMP6	
5830	033006	104002			1\$:	ERROR	2
5831	033010	000137	033302		JMP	@#HHDONE	
5832	033014				HHER10:		
5833	033014	010537	001252		MOV	R5,@#STMP10	
5834	033020	010437	001254		MOV	R4,@#STMP11	
5835	033024	012737	033212	001240	MOV	#HHP5,@#STMP3	
5836	033032	012737	033212	001242	MOV	#HHP5,@#STMP4	
5837	033040	012737	033132	001244	MOV	#HHDATA,@#STMP5	
5838	033046	012737	033272	001246	MOV	#HHP11,@#STMP6	
5839	033054	104002			1\$:	ERROR	2
5840	033056	000137	033302		JMP	@#HHDONE	
5841	033062	013703	001236		HHTRAP:	MOV	@#STMP2,R3
5842	033066	062703	000002		ADD	#2,R3	:WAS THE TRAP TO 244
5843	033072	020316			CMP	R3,(SP)	:ON THE INSTRUCTION
5844	033074	001402			BEQ	1\$:BEING TESTED?
5845	033076	000137	062524		JMP	@#FPSPUR	
5846	033102	011637	001236		1\$:	MOV	(SP),@#STMP2
5847							:FAILURE OF FPS INTERRUPT
5848	033106	022626			CMP	(SP)+,(SP)+	:DISABLE BIT (FID=1)
5849	033110	170201			STFPS	R1	:TO INHIBIT TRAP.
5850	033112	010137	001240		MOV	R1,@#STMP3	
5851	033116	170301			STST	R1	
5852	033120	010137	001242		MOV	R1,@#STMP4	
5853	033124	104002			2\$:	ERROR	2
5854	033126	000137	033302		JMP	@#HHDONE	
5855	033132	000000			HHDATA:	0	
5856	033134	000000				0	
5857	033136	000000				0	
5858	033140	000000				0	
5859	033142	000452			HHP0:	452	
5860	033144	125252				125252	
5861	033146	125252				125252	
5862	033150	125253			HHP1:	125253	
5863	033152	000252				252	
5864	033154	125252				125252	
5865	033156	125252				125252	
5866	033160	125252				125252	
5867	033162	000600			HHP2:	600	:HHP0 + HHP1 WITH
5868	033164	000000				0	:PROPER NORMALIZATION
5869	033166	000000				0	
5870	033170	000000				0	
5871	033172	000400			HHP3:	400	:HHP0 + HHP1 WITH
5872	033174	000000				0	:BAD NORMALIZATION
5873	033176	000000				0	
5874	033200	000000				0	
5875	033202	000000			HHP4:	0	
5876	033204	000000				0	
5877	033206	000000				0	
5878	033210	000000				0	
5879	033212	100200			HHP5:	100200	
5880	033214	000000				0	
5881	033216	000000				0	
5882	033220	000000				0	

5883 033222 000300
 5884 033224 000000
 5885 033226 000000
 5886 033230 000000
 5887 033232 100000
 5888 033234 000000
 5889 033236 000000
 5890 033240 000000
 5891 033242 000200
 5892 033244 000000
 5893 033246 000000
 5894 033250 000000
 5895 033252 100300
 5896 033254 000000
 5897 033256 000000
 5898 033260 000000
 5899 033262 000400
 5900 033264 000000
 5901 033266 000000
 5902 033270 000000
 5903 033272 100400
 5904 033274 000000
 5905 033276 000000
 5906 033300 000000
 5907 033302
 5908 033302 104413

HHP6: 300
 0
 0
 0
 HHP7: 100000
 0
 0
 0
 HHP8: 200
 0
 0
 0
 HHP9: 100300
 0
 0
 0
 HHP10: 400
 0
 0
 0
 HHP11: 100400
 0
 0
 0
 HHDONE:
 RSETUP

;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?).

 ;*TEST 40 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).
 ;*
 ;*
 ;*
 ;*****
 TST40: SCOPE

5929 033304 000004
 5930
 5931
 5932 033306
 5933 033306 104414
 5934 033310 012704 000200
 5935 033314 170104
 5936 033316 012737 033344 001236
 5937 033324 012737 034376 000244
 5938 033332 012700 036150

;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 #GGERO,@#FPVECT
 MOV #GGP5,R0 ;SET ACO OPERAND

5939	033336	172410			LDD	(R0),ACO	
5940	033340	012700	036150		MOV	#GGP5,R0	:FSRC
5941	033344	172010		GG2:	ADDD	(R0),ACO	:TEST INSTRUCTION
5942	033346	170205			STFPS	R5	:GET FPS
5943	033350	012700	036100		MOV	#GGDATO,R0	:GET THE RESULT
5944	033354	174010			STD	ACO,(R0)	
5945	033356	012701	036160		MOV	#GGP6,R1	:IS IT CORRECT
5946	033362	012702	000004		MCV	#4,R2	
5947	033366	022021		GG3:	CMP	(R0)+,(R1)+	
5948	033370	001402			BEQ	GG4	
5949	033372	000137	034474		JMP	@#GGER1	
5950	033376	077205		GG4:	SOB	R2,GG3	
5951	033400	052704	000006		BIS	#6,R4	:FPS CORRECT?
5952	033404	020405			CMP	R4,R5	
5953	033406	001402			BEQ	GG5	
5954	033410	000137	034542		JMP	@#GGER2	
5955							:TEST OVERFLOW WITH TRAPS ENABLED
5956							:FIV = 1
5957	033414			GG5:	LPERR		:SET UP THE LOOP ON ERROR ADDRESS.
5958	033414	104414			MOV	#1200,R4	:CLEAR FIU, SET FIV, AND FD
5959	033416	012704	001200		LDFPS	R4	
5960	033422	170104			MOV	#GG6,@#STMP2	
5961	033424	012737	033452	001236	MOV	#GG7,@#FPVECT	
5962	033432	012737	033470	000244	MOV	#GGP5,R0	:SET ACO OPERAND
5963	033440	012700	036150		LDD	(R0),ACO	
5964	033444	172410			MOV	#GGP5,R0	:FSPC
5965	033446	012700	036150		ADDD	(R0),ACO	:TEST INSTRUCTION
5966	033452	172010		GG6:	CFCC		:NO OVERFLOW TRAP OCCURED
5967	033454	170000			MOV	#GGDATO,R0	
5968	033456	012700	036100		STD	ACO,(R0)	
5969	033462	174010			JMP	@#GGER3	
5970	033464	000137	034610		GG7:	MOV	@#STMP2,R3
5971	033470	013703	001236		ADD	#2,R3	
5972	033474	062703	000002		CMP	R3,(SP)	:CHECK STACK DATA
5973	033500	020316			BEQ	1\$	
5974	033502	001402			JMP	@#FPSPUR	
5975	033504	000137	062524		1\$:	MOV	(SP),@#STMP2
5976	033510	011637	001236		CMP	(SP)+,(SP)+	
5977	033514	022626			STFPS	R5	
5978	033516	170205			MOV	#GGDATO,R0	:GET THE RESULT
5979	033520	012700	036100		STD	ACO,(R0)	
5980	033524	174010			MOV	#GGP6,R1	:IS IT CORRECT
5981	033526	012701	036160		MOV	#4,R2	
5982	033532	012702	000004		GG8:	CMP	(R0)+,(R1)+
5983	033536	022021			BEQ	GG9	
5984	033540	001402			JMP	@#GGER4	
5985	033542	000137	034656		GG9:	SOB	R2,GG8
5986	033546	077205			BIS	#100006,R4	:EXACT ZERO RESULTED IF OVERFLOW
5987	033550	052704	100006		CMP	R4,R5	:FPS CORRECT?, CHECK FER, FZ, FV
5988	033554	020405			BEQ	1\$	
5989	033556	001402			JMP	@#GGER6	
5990	033560	000137	034726		1\$:	MOV	#10,R4
5991	033564	012704	000010		:CHECK	FEC	
5992					STST	R5	
5993	033570	170305			CMP	R4,R5	
5994	033572	020405					

```

5995 033574 001402          BEQ      GG10
5996 033576 000137 034660  JMP      @#GGER5
5997                               :CHECK UNDER FLOW CONDITION WITH
5998                               :TRAPS DISABLED (FIU = 0)
5999                               GG10:
6000 033602 104414          LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
6001 033604 012704 000200  MOV      #0200,R4 ;SET FIU, FIV, AND FD
6002 033610 170104          LDFPS      R4
6003 033612 012737 033640 001236  MOV      #GG11,@#STMP2
6004 033620 012737 034774 000244  MOV      #GGER7,@#FPVECT
6005 033626 012700 036120          MOV      #GGP2,R0 ;SET ACO OPERAND
6006 033632 172410          LDD      (R0),ACO ;FSRC
6007 033634 012700 036130          MOV      #GGP3,R0
6008 033640 172010          GG11:  ADDD   (R0),ACO ;TEST INSTRUCTION
6009 033642 170205          STFPS     R5 ;GET FPS
6010 033644 012700 036100          MOV      #GGDATO,R0 ;GET THE RESULT
6011 033650 174010          STD      ACO,(R0)
6012 033652 012701 036160          MOV      #GGP6,R1 ;IS IT CORRECT
6013 033656 012702 000004          MOV      #4,R2
6014 033662 022021          GG12:  CMP      (R0)+,(R1)+
6015 033664 001402          BEQ      GG13
6016 033666 000137 035072          JMP      @#GGER8
6017 033672 077205          GG13:  SOB      R2,GG12
6018 033674 052704 000004          BIS      #4,R4 ;FPS CORRECT?
6019 033700 020405          CMP      R4,R5
6020 033702 001402          BEQ      GG14
6021 033704 000137 035140          JMP      @#GGER9
6022                               :CHECK UNDERFLOW CONDITION WITH
6023                               :TRAP ENABLED (FIU = 1)
6024                               GG14:
6024 033710 104414          LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
6025 033710 012704 002200  MOV      #2200,R4 ;SET FIU, FIV, AND FD
6026 033712 170104          LDFPS      R4
6027 033716 012737 033746 001236  MOV      #GG15,@#STMP2
6028 033720 012737 033764 000244  MOV      #GG16,@#FPVECT
6029 033726 012700 036120          MOV      #GGP2,R0 ;SET ACO OPERAND
6030 033734 172410          LDD      (R0),ACO ;FSPC
6031 033740 012700 036130          MOV      #GGP3,R0
6032 033742 172010          GG15:  ADDD   (R0),ACO ;TEST INSTRUCTION
6033 033746 170000          CFCC
6034 033750 012700 036100          MOV      #GGDATO,R0
6035 033752 174010          STD      ACO,(R0)
6036 033756 000137 035206          JMP      @#GGER10
6037 033760 013703 001236          GG16:  MOV      @#STMP2,R3
6038 033764 062703 000002          ADD      #2,R3
6039 033770 021603          CMP      (SP),R3
6040 033774 001402          BEQ      1$
6041 034000 000137 062524          JMP      @#FPSPUR
6042 034004 011637 001236          1$:  MOV      (SP),@#STMP2
6043 034010 022626          CMP      (SP)+,(SP)+
6044 034012 170205          STFPS     R5 ;GET FPS
6045 034014 012700 036100          MOV      #GGDATO,R0 ;GET THE RESULT
6046 034020 174010          STD      ACO,(R0)
6047 034022 012701 036170          MOV      #GGP7,R1 ;IS IT CORRECT
6048 034026 012702 000004          MOV      #4,R2
6049 034032 022021          GG17:  CMP      (R0)+,(R1)+
    
```

```

6051 034034 001402          BEQ      GG18
6052 034036 000137 035254    JMP      @#GGER11
6053 034042 077205          SOB      R2,GG17
6054 034044 052704 100000    BIS      #100000,R4
6055 034050 020405          CMP      R4,R5          ;FPS CORRECT?
6056 034052 001402          BEQ      2$
6057 034054 000137 035322    JMP      @#GGER12
6058 034060
6059 034060 012704 000012    2$:
1$:      MOV      #12,R4
:CHECK   FEC
6060
6061 034064 170305          STST    R5
6062 034066 020405          CMP      R4,R5
6063 034070 001402          BEQ      GG19
6064 034072 000177 001272    JMP      @#GGER13
6065
6066          :CHECK UNDERFLOW CONDITION WITH TRAPS
6067          :DISABLED (FIU = 0)
6067 034076
6068 034076 104414          LPERR
6069 034100 012704 000200    MOV      #0200,R4          ;SET UP THE LOOP ON ERROR ADDRESS.
6070 034104 170104          LDFPS   R4                ;SET FIU, FIV, AND FD
6071 034106 012737 034134 001236    MOV      #GG20,@#STMP2
6072 034114 012737 035436 000244    MOV      #GGER14,@#FPVECT
6073 034122 012700 036120          MOV      #GGP2,R0          ;SET ACO OPERAND
6074 034126 172410          LDD     (R0),ACO
6075 034130 012700 036200          MOV      #GGP8,R0          ;FSPC
6076 034134 172010          GG20:  ADDD    (R0),ACO      ;TEST INSTRUCTION
6077 034136 170205          STFPS   R5                ;GET FPS
6078 034140 012700 036100          MOV      #GGDATO,R0        ;GET THE RESULT
6079 034144 174010          STD     ACO,(R0)
6080 034146 012701 036160          MOV      #GGP6,R1          ;IS IT CORRECT
6081 034152 012702 000004          MOV      #4,R2
6082 034156 022021          GG21:  CMP     (R0)+,(R1)+
6083 034160 001402          BEQ     GG22
6084 034162 000137 035534          JMP     @#GGER15
6085 034166 077205          GG22:  SOB     R2,GG21
6086 034170 052704 000004          BIS     #4,R4          ;FPS CORRECT?
6087 034174 020405          CMP     R4,R5
6088 034176 001402          BEQ     GG23
6089 034200 000137 035602          JMP     @#GGER16
6090          :CHECK UNDERFLOW CONDITION WITH TRAP
6091          :ENABLED (FIU = 1)
6092 034204
6093 034204 104414          LPERR
6094 034206 012704 002200    MOV      #2200,R4          ;SET UP THE LOOP ON ERROR ADDRESS.
6095 034212 170104          LDFPS   R4                ;SET FIU, FIV, AND FD
6096 034214 012737 034242 001236    MOV      #GG24,@#STMP2
6097 034222 012737 034260 000244    MOV      #GG25,@#FPVECT
6098 034230 012700 036120          MOV      #GGP2,R0          ;SET ACO OPERAND
6099 034234 172410          LDD     (R0),ACO
6100 034236 012700 036200          MOV      #GGP8,R0          ;FSRC
6101 034242 172010          GG24:  ADDD    (R0),ACO      ;TEST INSTRUCTION
6102 034244 170000          CFCC
6103 034246 012700 036100          MOV      #GGDATO,R0
6104 034252 174010          STD     ACO,(R0)
6105 034254 000137 035650          JMP     @#GGER17
6106 034260 013700 001236          GG25:  MOV     @#STMP2,R0
    
```


6107	034264	062700	000002		ADD	#2,R0	
6108	034270	020016			CMP	R0,(SP)	
6109	034272	001402			BEQ	1\$	
6110	034274	000137	062524		JMP	@#FPSPUR	
6111	034300	011637	001236	1\$:	MOV	(SP),@#\$TMP2	
6112	034304	022626			CMP	(SP)+,(SP)+	
6113	034306	170205			STFPS	R5	:GET FPS
6114	034310	012700	036100		MOV	#GGDATO,R0	:GET THE RESULT
6115	034314	174010			STD	AC0,(R0)	
6116	034316	012701	036210		MOV	#GGP9,R1	:IS IT CORRECT
6117	034322	012702	000004		MOV	#4,R2	
6118	034326	022021		GG26:	CMP	(R0)+,(R1)+	
6119	034330	001402			BEQ	GG27	
6120	034332	000137	035716		JMP	@#GGER18	
6121	034336	077205		GG27:	SOB	R2,GG26	
6122	034340	052704	100004		BIS	#100004,R4	
6123	034344	020405			CMP	R4,R5	:FPS CORRECT?
6124	034346	001402			BEQ	1\$	
6125	034350	000167	001456		JMP	GGER20	
6126	034354	012704	000012	1\$:	MOV	#12,R4	
6127						:CHECK FEC	

6128	034360	170305			STST	R5
6129	034362	020405			CMP	R4,R5
6130	034364	001402			BEQ	GG28
6131	034366	000167	001372		JMP	GGER19
6132						
6133	034372	000137	036220		GG28:	JMP @#GGDONE
6134						
6135	034376	013701	001236		GGER0:	MOV @#STMP2,R1
6136	034402	062701	000002			ADD #2,R1
6137	034406	020116				CMP R1,(SP)
6138	034410	001402				BEQ 10\$
6139	034412	000137	062524		5\$:	JMP @#FPSPUR
6140	034416				10\$:	
6141	034416	170301				STST R1
6142	034420	020127	000010			CMP R1,#10
6143	034424	001372				BNE 5\$
6144	034426	022626				CMP (SP)+,(SP)+
6145	034430	012700	036100			MOV #GGDATO,R0
6146	034434	174010				STD ACO,(R0)
6147	034436	012737	036150	001240		MOV #GGP5,@#STMP3
6148	034444	012737	036150	001242		MOV #GGP5,@#STMP4
6149	034452	012737	036100	001244		MOV #GGDATO,@#STMP5
6150	034460	012737	036160	001246		MOV #GGP6,@#STMP6
6151	034466	104001			1\$:	ERROR 1
6152	034470	000137	036220			JMP @#GGDONE
6153						
6154	034474				GGER1:	
6155	034474	010537	001252			MOV R5,@#STMP10
6156	034500	010437	001254			MOV R4,@#STMP11
6157	034504	012737	036150	001240		MOV #GGP5,@#STMP3
6158	034512	012737	036150	001242		MOV #GGP5,@#STMP4
6159	034520	012737	036100	001244		MOV #GGDATO,@#STMP5
6160	034526	012737	036160	001246		MOV #GGP6,@#STMP6
6161	034534	104002			1\$:	ERROR 2
6162	034536	000137	036220			JMP @#GGDONE
6163						
6164	034542				GGER2:	
6165	034542	010537	001252			MOV R5,@#STMP10
6166	034546	010437	001254			MOV R4,@#STMP11
6167	034552	012737	036150	001240		MOV #GGP5,@#STMP3
6168	034560	012737	036150	001242		MOV #GGP5,@#STMP4
6169	034566	012737	036100	001244		MOV #GGDATO,@#STMP5
6170	034574	012737	036160	001246		MOV #GGP6,@#STMP6
6171	034602	104002			1\$:	ERROR 2
6172	034604	000137	036220			JMP @#GGDONE
6173						
6174	034610				GGER3:	
6175	034610	010537	001252			MOV R5,@#STMP10
6176	034614	010437	001254			MOV R4,@#STMP11
6177	034620	012737	036150	001240		MOV #GGP5,@#STMP3
6178	034626	012737	036150	001242		MOV #GGP5,@#STMP4
6179	034634	012737	036100	001244		MOV #GGDATO,@#STMP5
6180	034642	012737	036160	001246		MOV #GGP6,@#STMP6
6181	034650	104002			1\$:	ERROR 2
6182	034652	000137	036220			JMP @#GGDONE
6183						

6184	034656	000706			GGER4:	BR	GGER1
6185							
6186	034660				GGER5:		
6187	034660	010537	001252			MOV	R5,@#STMP10
6188	034664	010437	001254			MOV	R4,@#STMP11
6189	034670	012737	036150	001240		MOV	#GGP5,@#STMP3
6190	034676	012737	036150	001242		MOV	#GGP5,@#STMP4
6191	034704	012737	036100	001244		MOV	#GGDATO,@#STMP5
6192	034712	012737	036160	001246		MOV	#GGP6,@#STMP6
6193	034720	104002			1\$:	ERROR	2
6194	034722	000137	036220			JMP	@#GGDONE
6195							
6196	034726				GGER6:		
6197	034726	010537	001252			MOV	R5,@#STMP10
6198	034732	010437	001254			MOV	R4,@#STMP11
6199	034736	012737	036150	001240		MOV	#GGP5,@#STMP3
6200	034744	012737	036150	001242		MOV	#GGP5,@#STMP4
6201	034752	012737	036100	001244		MOV	#GGDATO,@#STMP5
6202	034760	012737	036160	001246		MOV	#GGP6,@#STMP6
6203	034766	104002			1\$:	ERROR	2
6204	034770	000137	036220			JMP	@#GGDONE
6205							
6206	034774	013701	001236		GGER7:	MOV	@#STMP2,R1
6207	035000	062701	000002			ADD	#2,R1
6208	035004	020116				CMP	R1,(SP)
6209	035006	001402				BEQ	10\$
6210	035010	000137	062524		5\$:	JMP	@#FPSPUR
6211	035014				10\$:		
6212	035014	170301				STST	R1
6213	035016	020127	000012			CMP	R1,#12
6214	035022	001372				BNE	5\$
6215	035024	022626				CMP	(SP)+,(SP)+
6216	035026	012700	036100			MOV	#GGDATO,R0
6217	035032	174010				STD	AC0,(R0)
6218	035034	012737	036130	001240		MOV	#GGP3,@#STMP3
6219	035042	012737	036120	001242		MOV	#GGP2,@#STMP4
6220	035050	012737	036100	001244		MOV	#GGDATO,@#STMP5
6221	035056	012737	036160	001246		MOV	#GGP6,@#STMP6
6222	035064	104002			1\$:	ERROR	2
6223	035066	000137	036220			JMP	@#GGDONE
6224							
6225	035072				GGER8:		
6226	035072	010537	001252			MOV	R5,@#STMP10
6227	035076	010437	001254			MOV	R4,@#STMP11
6228	035102	012737	036130	001240		MOV	#GGP3,@#STMP3
6229	035110	012737	036120	001242		MOV	#GGP2,@#STMP4
6230	035116	012737	036100	001244		MOV	#GGDATO,@#STMP5
6231	035124	012737	036160	001246		MOV	#GGP6,@#STMP6
6232	035132	104002			1\$:	ERROR	2
6233	035134	000137	036220			JMP	@#GGDONE
6234							
6235	035140				GGER9:		
6236	035140	010537	001252			MOV	R5,@#STMP10
6237	035144	010437	001254			MOV	R4,@#STMP11
6238	035150	012737	036130	001240		MOV	#GGP3,@#STMP3
6239	035156	012737	036120	001242		MOV	#GGP2,@#STMP4

6240	035164	012737	036100	001244		MOV	#GGDATO,@#STMP5
6241	035172	012737	036160	001246		MOV	#GGP6,@#STMP6
6242	035200	104002			1\$:	ERROR	2
6243	035202	000137	036220			JMP	@#GGDONE
6244							
6245	035206				GGER10:		
6246	035206	010537	001252			MOV	R5,@#STMP10
6247	035212	010437	001254			MOV	R4,@#STMP11
6248	035216	012737	036130	001240		MOV	#GGP3,@#STMP3
6249	035224	012737	036120	001242		MOV	#GGP2,@#STMP4
6250	035232	012737	036100	001244		MOV	#GGDATO,@#STMP5
6251	035240	012737	036170	001246		MOV	#GGP7,@#STMP6
6252	035246	104002			1\$:	ERROR	2
6253	035250	000137	036220			JMP	@#GGDONE
6254							
6255	035254				GGER11:		
6256	035254	010537	001252			MOV	R5,@#STMP10
6257	035260	010437	001254			MOV	R4,@#STMP11
6258	035264	012737	036130	001240		MOV	#GGP3,@#STMP3
6259	035272	012737	036120	001242		MOV	#GGP2,@#STMP4
6260	035300	012737	036100	001244		MOV	#GGDATO,@#STMP5
6261	035306	012737	036170	001246		MOV	#GGP7,@#STMP6
6262	035314	104002			1\$:	ERROR	2
6263	035316	000137	036220			JMP	@#GGDONE
6264							
6265	035322				GGER12:		
6266	035322	010537	001252			MOV	R5,@#STMP10
6267	035326	010437	001254			MOV	R4,@#STMP11
6268	035332	012737	036130	001240		MOV	#GGP3,@#STMP3
6269	035340	012737	036120	001242		MOV	#GGP2,@#STMP4
6270	035346	012737	036100	001244		MOV	#GGDATO,@#STMP5
6271	035354	012737	036170	001246		MOV	#GGP7,@#STMP6
6272	035362	104002			1\$:	ERROR	2
6273	035364	000137	036220			JMP	@#GGDONE
6274							
6275	035370				GGER13:		
6276	035370	010537	001252			MOV	R5,@#STMP10
6277	035374	010437	001254			MOV	R4,@#STMP11
6278	035400	012737	036130	001240		MOV	#GGP3,@#STMP3
6279	035406	012737	036120	001242		MOV	#GGP2,@#STMP4
6280	035414	012737	036100	001244		MOV	#GGDATO,@#STMP5
6281	035422	012737	036170	001246		MOV	#GGP7,@#STMP6
6282	035430	104002			1\$:	ERROR	2
6283	035432	000137	036220			JMP	@#GGDONE
6284							
6285	035436	013701	001236		GGER14:	MOV	@#STMP2,R1
6286	035442	062701	000002			ADD	#2,R1
6287	035446	020116				CMP	R1,(SP)
6288	035450	001402				BEQ	10\$
6289	035452	000137	062524		5\$:	JMP	@#FPSPUR
6290	035456				10\$:		
6291	035456	170301				STST	R1
6292	035460	020127	000012			CMP	R1,#12
6293	035464	001372				BNE	5\$
6294	035466	022626				CMP	(SP)+,(SP)+
6295	035470	012700	036100			MOV	#GGDATO,R0

6296	035474	174010			STD	ACO,(R0)
6297						
6298	035476	012737	036110	001240	MOV	#GGP1,@\$TMP3
6299	035504	012737	036130	001242	MOV	#GGP3,@\$TMP4
6300	035512	012737	036100	001244	MOV	#GGDATO,@\$TMP5
6301	035520	012737	036160	001246	MOV	#GGP6,@\$TMP6
6302	035526	104002			1\$:	ERROR
6303	035530	000137	036220		JMP	@#GGDONE
6304						
6305	035534				GGER15:	
6306	035534	010537	001252		MOV	R5,@\$TMP10
6307	035540	010437	001254		MOV	R4,@\$TMP11
6308	035544	012737	036120	001240	MOV	#GGP2,@\$TMP3
6309	035552	012737	036200	001242	MOV	#GGP8,@\$TMP4
6310	035560	012737	036100	001244	MOV	#GGDATO,@\$TMP5
6311	035566	012737	036160	001246	MOV	#GGP6,@\$TMP6
6312	035574	104002			1\$:	ERROR
6313	035576	000137	036220		JMP	@#GGDONE
6314						
6315	035602				GGER16:	
6316	035602	010537	001252		MOV	R5,@\$TMP10
6317	035606	010437	001254		MOV	R4,@\$TMP11
6318	035612	012737	036120	001240	MOV	#GGP2,@\$TMP3
6319	035620	012737	036200	001242	MOV	#GGP8,@\$TMP4
6320	035626	012737	036100	001244	MOV	#GGDATO,@\$TMP5
6321	035634	012737	036160	001246	MOV	#GGP6,@\$TMP6
6322	035642	104002			1\$:	ERROR
6323	035644	000137	036220		JMP	@#GGDONE
6324						
6325	035650				GGER17:	
6326	035650	010537	001252		MOV	R5,@\$TMP10
6327	035654	010437	001254		MOV	R4,@\$TMP11
6328	035660	012737	036120	001240	MOV	#GGP2,@\$TMP3
6329	035666	012737	036200	001242	MOV	#GGP8,@\$TMP4
6330	035674	012737	036100	001244	MOV	#GGDATO,@\$TMP5
6331	035702	012737	036210	001246	MOV	#GGP9,@\$TMP6
6332	035710	104002			1\$:	ERROR
6333	035712	000137	036220		JMP	@#GGDONE
6334						
6335	035716				GGER18:	
6336	035716	010537	001252		MOV	R5,@\$TMP10
6337	035722	010437	001254		MOV	R4,@\$TMP11
6338	035726	012737	036120	001240	MOV	#GGP2,@\$TMP3
6339	035734	012737	036200	001242	MOV	#GGP8,@\$TMP4
6340	035742	012737	036100	001244	MOV	#GGDATO,@\$TMP5
6341	035750	012737	036210	001246	MOV	#GGP9,@\$TMP6
6342	035756	104002			1\$:	ERROR
6343	035760	000137	036220		JMP	@#GGDONE
6344						
6345	035764				GGER19:	
6346	035764	010537	001252		MOV	R5,@\$TMP10
6347	035770	010437	001254		MOV	R4,@\$TMP11
6348	035774	012737	036120	001240	MOV	#GGP2,@\$TMP3
6349	036002	012737	036200	001242	MOV	#GGP8,@\$TMP4
6350	036010	012737	036100	001244	MOV	#GGDATO,@\$TMP5
6351	036016	012737	036210	001246	MOV	#GGP9,@\$TMP6

```

6352 036024 104002
6353 036026 000137 036220
6354
6355 036032
6356 036032 010537 001252
6357 036036 010437 001254
6358 036042 012737 036120 001240
6359 036050 012737 036200 001242
6360 036056 012737 036100 001244
6361 036064 012737 036210 001246
6362 036072 104002
6363 036074 000137 036220
6364
6365 036100 000000
6366 036102 000000
6367 036104 000000
6368 036106 000000
6369
6370 036110 000300
6371 036112 000000
6372 036114 000000
6373 036116 000000
6374 036120 100200
6375 036122 000000
6376 036124 000000
6377 036126 000000
6378 036130 000200
6379 036132 000000
6380 036134 000000
6381 036136 000001
6382 036140 010200
6383 036142 000000
6384 036144 000000
6385 036146 000000
6386 036150 077600
6387 036152 000000
6388 036154 000000
6389 036156 000000
6390 036160 000000
6391 036162 000000
6392 036164 000000
6393 036166 000000
6394
6395 036170 062400
6396 036172 000000
6397 036174 000000
6398 036176 000000
6399 036200 000340
6400 036202 000000
6401 036204 000000
6402 036206 000000
6403 036210 000100
6404 036212 000000
6405 036214 000000
6406 036216 000000
6407 036220
    
```

```

1$: ERROR 2
    JMP @#GGDONE

GGER20:
    MOV R5,@#STMP10
    MOV R4,@#STMP11
    MOV #GGP2,@#STMP3
    MOV #GGP8,@#STMP4
    MOV #GGDATO,@#STMP5
    MOV #GGP9,@#STMP6
1$: ERROR 2
    JMP @#GGDONE

GGDATO: 0
        0
        0
        0

GGP1: 300
      0
      0
      0

GGP2: 100200
      0
      0
      0

GGP3: 200
      0
      0
      1

GGP4: 10200
      0
      0
      0

GGP5: 77600
      0
      0
      0

GGP6: 0
      0
      0
      0

GGP7: 62400
      0
      0
      0

GGP8: 340
      0
      0
      0

GGP9: 100
      0
      0
      0

GGDONE:
    
```

:OVER FLOW = GGP5 + GGP5

```

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

6408 036220 104413
 6409
 6410
 6411
 6412
 6413
 6414
 6415
 6416
 6417
 6418
 6419
 6420
 6421 036222 000004
 6422
 6423 036224
 6424 036224 104414
 6425 036226 012704 000200
 6426 036232 170104
 6427 036234 012700 037640
 6428 036240 172410
 6429 036242 012700 037650
 6430 036246 012737 036254 001236
 6431 036254 177420
 6432 036256 020027 037654
 6433 036262 001402
 6434 036264 000137 037216
 6435 036270
 6436 036270 170205
 6437 036272 012700 037630
 6438 036276 174010
 6439 036300 012701 037720
 6440 036304 012702 000004
 6441 036310 022120
 6442 036312 001415
 6443 036314 012701 037650
 6444 036320 012700 037630
 6445 036324 012702 000004
 6446 036330 022120
 6447 036332 001402
 6448 036334 000137 037256
 6449 036340 077205
 6450 036342 000137 037306
 6451 036346 077220
 6452 036350 012704 000200
 6453 036354 020405
 6454 036356 001402
 6455 036360 000137 037354
 6456
 6457 036364
 6458 036364 104414
 6459 036366 012704 000200
 6460 036372 170104
 6461
 6462 036374 012700 037640
 6463 036400 172410

```

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?).

*****
*TEST 41          LDCFD AND LDCDF TEST
*
*THIS IS A TEST OF LDCFD AND LDCDF.
*
*****
TST41: SCOPE
;TEST FOR CORRECT AUTO INCREMENT CONSTANT.
HX1:
    LPERR                                ;SET UP THE LOOP ON ERROR ADDRESS.
    MOV #200,R4                          ;SET LONG INTEGER MODE
    LDFPS R4
    MOV #HXP1,R0
    LDD (R0),ACO
    MOV #HXP2,R0
    MOV #HX2,@#STMP2
    LDCFD (R0)+,ACO
    CMP R0,#HXP2+4                       ;IS R0 CORRECT
    BEQ HX3
    JMP @#HXER1
HX3:
    STFPS R5                             ;GET FPS
    MOV #HXDAT0,R0
    STD ACO,(R0)                         ;GET ACO
    MOV #HXP7,R1
    MOV #4,R2
    CMP (R1)+,(R0)+                      ;SEE IF RESULT IS
    BEQ HX7                               ;CORRECT
    MOV #HXP2,R1                          ;DID FD GET
    MOV #HXDAT0,R0                       ;COMPLIMENTED?
    MOV #4,R2
    CMP (R1)+,(R0)+
    BEQ HX6
    JMP @#HXER2
HX6:
    SOB R2,HX5
    JMP @#HXER3
HX7:
    SOB R2,HX4
    MOV #200,R4
    CMP R4,R5
    BEQ HX8
    JMP @#HXER8
;NOW
HX8:
    LPERR                                ;SET UP THE LOOP ON ERROR ADDRESS.
    MOV #200,R4
    LDFPS R4
    MOV #HXP1,R0
    LDD (R0),ACO
    
```

```

6464
6465 036402 012700 037650      MOV    #HXP2,R0
6466 036406 012737 036416 001236  MOV    #HX9,@#STMP2
6467
6468 036414 170001                SETF
6469
6470 036416 177420      HX9:   LDCFD  (R0)+,ACO      ;TEST INSTRUCTION
6471
6472 036420 020027 037660      CMP    R0,#HXP2+10      ;WAS A GOOD
6473 036424 001402                BEQ    HX10              ;CONSTANT USED
6474 036426 000137 037236      JMP    @#HXER5          ;TO INCREMENT R0?
6475
6476 036432      HX10:  STFPS  R5
6477 036432 170205                MOV    #HXDATO,R0
6478 036434 012700 037630      SETD
6479 036440 170011                STD    ACO,(R0)         ;GET RESULT
6480 036442 174010                MOV    #HXP8,R1
6481 036444 012701 037730      MOV    #4,R2
6482 036450 012702 000004      HX11:  CMP    (R1)+,(R0)+  ;IS IT CORRECT?
6483 036454 022120                BEQ    HX14
6484 036456 001415
6485
6486 036460 012701 037720      MOV    #HXP7,R1
6487 036464 012700 037630      MOV    #HXDATO,R0
6488 036470 012702 000004      MOV    #4,R2
6489 036474 022110      HX12:  CMP    (R1)+,(R0)   ;DID FD FAIL TO GET
6490 036476 001402                BEQ    HX13             ;COMPLIMENTED?
6491 036500 000137 037372      JMP    @#HXER6
6492 036504 077205      HX13:  SOB    R2,HX12
6493 036506 000137 037422      JMP    @#HXER7
6494
6495 036512 077220      HX14:  SOB    R2,HX11
6496
6497 036514 012704 000000      MOV    #0,R4           ;FPS CORRECT?
6498 036520 020405                CMP    R4,R5
6499 036522 001402                BEQ    HX15
6500 036524 000137 037354      JMP    @#HXER8
6501
6502      ;TEST GR7 IMMEDIATE MODE CONSTANT
6503
6504 036530      HX15:  LPERR
6505 036530 104414                ;SET UP THE LOOP ON ERROR ADDRESS.
6506
6507 036532 012704 000200      MOV    #200,R4
6508 036536 170104                LDFPS R4                ;SET FD
6509 036540 012737 036556 001236  MOV    #HX16,@#STMP2
6510 036546 012737 037452 000004  MOV    #HXER9,@#ERRVECT
6511 036554 005001                CLR    R1
6512 036556 177427 043243      HX16:  LDCFD  #5201,ACO
6513 036562 005201      HX165: INC    R1
6514 036564 005201                INC    R1
6515 036566 005201                INC    R1
6516 036570 012737 062556 000004  MOV    #CPSPUR,@#ERRVECT
6517 036576 020127 000003      CMP    R1,#3           ;SEE IF PC WAS
6518 036602 001402                BEQ    HX17             ;CORRECT
6519 036604 000137 037506      JMP    @#HXER10
    
```



```

6520
6521 036610
6522 036610 104414
6523
6524 036612 012704 000200
6525 036616 170104
6526 036620 012737 036646 001236
6527 036626 012700 037710
6528 036632 172410
6529 036634 012737 062556 000004
6530 036642 012700 037650
6531 036646 177410
6532
6533 036650 012700 037630
6534 036654 174010
6535 036656 012701 037720
6536 036662 012702 000004
6537 036666 022021
6538 036670 001402
6539 036672 000137 037256
6540 036676 077205
6541
6542
6543 036700
6544 036700 104414
6545 036702 012704 000200
6546 036706 170104
6547 036710 012737 036730 001236
6548 036716 012700 037710
6549 036722 172410
6550 036724 012700 037670
6551 036730 177410
6552
6553 036732 012700 037630
6554 036736 174010
6555
6556 036740 012701 037700
6557 036744 012702 000004
6558 036750 022120
6559 036752 001415
6560
6561 036754 012701 037720
6562 036760 012700 037630
6563 036764 012702 000004
6564 036770 022120
6565 036772 001402
6566 036774 000137 037540
6567 037000 077205
6568 037002 000137 037560
6569
6570
6571 037006 077220
6572
6573
6574
6575 037010
    
```

```

HX17: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
MOV #200,R4
LDFPS R4
MOV #HX18,@#$TMP2
MOV #HXP6,R0
LDD (R0),ACO
MOV #CPSPUR,@#ERRVECT
MOV #HXP2,R0
HX18: LDCFD (R0),ACO
MOV #HXDATO,R0
STD ACO,(R0) ;GET RESULT.
MOV #HXP7,R1
MOV #4,R2
HX19: CMP (R0)+,(R1)+ ;IS RESULT CORRECT?
BEQ HX20
JMP @#HXER2
HX20: SOB R2,HX19
;TEST LDCFD WITH NEGATIVE OPERAND
HX21: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
MOV #200,R4
LDFPS R4
MOV #HX22,@#$TMP2
MOV #HXP6,R0
LDD (R0),ACO
MOV #HXP4,R0
HX22: LDCFD (R0),ACO
MOV #HXDATO,R0
STD ACO,(R0) ;GET RESULT
MOV #HXP5,R1
MOV #4,R2
HX23: CMP (R1)+,(R0)+
BEQ HX26
MOV #HXP7,R1
MOV #HXDATO,R0
MOV #4,R2
HX24: CMP (R1)+,(R0)+ ;WAS SIGN INCORRECT
BEQ HX25
JMP @#HXER11
HX25: SOB R2,HX24
JMP @#HXER12
HX26: SOB R2,HX23
;TEST LDCFD 0
HX27:
    
```

```

6576 037010 104414          LPERR
6577 037012 012704 000200  MOV      #200,R4
6578 037016 170104          LDFPS   R4
6579
6580 037020 012700 037640  MOV      #HXP1,R0
6581 037024 172410          LDD     (R0),ACO
6582 037026 172010          ADDD   (R0),ACO
6583
6584 037030 012737 037042 001236  MOV      #HX28,@#STMP2
6585 037036 012700 037640  MOV      #HXP1,R0
6586 037042 177410          LDCFD  (R0),ACO
6587
6588 037044 170205          STFPS  R5
6589
6590 037046 012700 037630  MOV      #HXDATO,R0
6591 037052 174010          STD     ACO,(R0)
6592
6593 037054 012701 037640  MOV      #HXP1,R1
6594 037060 012702 000004  MOV      #4,R2
6595 037064 022120          HX29:  CMP   (R1)+,(R0)+
6596 037066 001402          BEQ   HX30
6597 037070 000137 037610  HX30:  JMP   @#HXER13
6598 037074 077205          SOB   R2,HX29
6599
6600 037076 012704 000204  MOV      #204,R4
6601 037102 020405          CMP   R4,R5
6602 037104 001402          BEQ   HX31
6603 037106 000137 037336  JMP     @#HXER4
6604
6605          ;TEST  LDCFD  0
6606
6607 037112          HX31:
6608 037112 104414          LPERR
6609 037114 012704 000200  MOV      #200,R4
6610 037120 170104          LDFPS   R4
6611
6612 037122 012700 037710  MOV      #HXP6,R0
6613 037126 172410          LDD     (R0),ACO
6614
6615 037130 012737 037142 001236  MOV      #HX32,@#STMP2
6616 037136 012700 037640  MOV      #HXP1,R0
6617 037142 177410          HX32:  LDCFD  (R0),ACO
6618
6619 037144 170205          STFPS  R5
6620
6621 037146 012700 037630  MOV      #HXDATO,R0
6622 037152 174010          STD     ACO,(R0)
6623
6624 037154 012701 037640  MOV      #HXP1,R1
6625 037160 012702 000004  MOV      #4,R2
6626 037164 022120          HX33:  CMP   (R1)+,(R0)+
6627 037166 001402          BEQ   HX34
6628 037170 000137 037610  HX34:  JMP   @#HXER13
6629 037174 077205          SOB   R2,HX33
6630
6631 037176 012704 000204  MOV      #204,R4
    
```

;SET UP THE LOOP ON ERROR ADDRESS.

;GET RESULT

;IS IT 0?

;FPS CORRECT

;SET UP THE LOOP ON ERROR ADDRESS.

;GET RESULT

;IS IT ZERO?

;FPS CORRECT?

6632	037202	020405				CMP	R4,R5	
6633	037204	001402				BEQ	HX35	
6634	037206	000137	037336			JMP	@HXER4	
6635	037212	000137	037740		HX35:	JMP	@HXDONE	
6636								
6637								
6638								
6639	037216	012737	037654	001242	HXER1:	MOV	#HXP2+4,@\$TMP4	
6640	037224	010037	001240			MOV	R0,@\$TMP3	
6641	037230	104002			1\$:	ERROR	2	
6642	037232	000137	037740			JMP	@HXDONE	
6643								
6644	037236	012737	037660	001242	HXER5:	MOV	#HXP2+10,@\$TMP4	
6645	037244	010037	001240			MOV	R0,@\$TMP3	
6646	037250	104002			1\$:	ERROR	2	
6647	037252	000137	037740			JMP	@HXDONE	
6648								
6649	037256	012737	037650	001244	HXER2:	MOV	#HXP2,@\$TMP5	
6650	037264	012737	037720	001250		MOV	#HXP7,@\$TMP7	
6651	037272	012737	037630	001246	HXER22:	MOV	#HXDATO,@\$TMP6	
6652	037300	104002			1\$:	ERROR	2	
6653	037302	000137	037740			JMP	@HXDONE	
6654								
6655	037306	012737	037650	001244	HXER3:	MOV	#HXP2,@\$TMP5	
6656	037314	012737	037720	001250		MOV	#HXP7,@\$TMP7	
6657	037322	012737	037630	001246	HXER33:	MOV	#HXDATO,@\$TMP6	
6658	037330	104002			1\$:	ERROR	2	
6659	037332	000137	037740			JMP	@HXDONE	
6660								
6661	037336	010537	001240		HXER4:	MOV	R5,@\$TMP3	
6662	037342	010437	001242			MOV	R4,@\$TMP4	
6663	037346	104002			1\$:	ERROR	2	
6664	037350	000137	037740			JMP	@HXDONE	
6665								
6666	037354	010537	001240		HXER8:	MOV	R5,@\$TMP3	
6667	037360	010437	001242			MOV	R4,@\$TMP4	
6668	037364	104002			1\$:	ERROR	2	
6669	037366	000137	037740			JMP	@HXDONE	
6670	037372	012737	037650	001244	HXER6:	MOV	#HXP2,@\$TMP5	
6671	037400	012737	037730	001250		MOV	#HXP8,@\$TMP7	
6672	037406	012737	037630	001246	HXER66:	MOV	#HXDATO,@\$TMP6	
6673	037414	104002			1\$:	ERROR	2	
6674	037416	000137	037740			JMP	@HXDONE	
6675								
6676	037422	012737	037650	001244	HXER7:	MOV	#HXP2,@\$TMP5	
6677	037430	012737	037730	001250		MOV	#HXP8,@\$TMP7	
6678								
6679	037436	012737	037630	001246		MOV	#HXDATO,@\$TMP6	
6680	037444	104002			1\$:	ERROR	2	
6681	037446	000137	037740			JMP	@HXDONE	
6682								
6683	037452	032716	000001		HXER9:	BIT	#1,(SP)	:SEE IF IT
6684	037456	001005				BNE	1\$:AN ODD ADDRESS
6685	037460	022716	036562			CMP	#HX165,(SP)	
6686	037464	001402				BEQ	1\$	
6687	037466	000137	062556			JMP	@CPSPUR	

6688						
6689	037472	011637	001236	1\$:	MOV	(SP),@#\$TMP2
6690	037476	022626			CMP	(SP)+,(SP)+
6691	037500	104002		2\$:	ERROR	2
6692	037502	000137	037740		JMP	@#HXDONE
6693						
6694	037506	162701	000003	HXER10:	SUB	#3,R1
6695	037512	006301			ASL	R1
6696	037514	012702	036562		MOV	#HX165,R2
6697	037520	010237	001242		MOV	R2,@#\$TMP4
6698	037524	160102			SUB	R1,R2
6699	037526	010237	001240		MOV	R2,@#\$TMP3
6700	037532	104002		1\$:	ERROR	2
6701	037534	000137	037740		JMP	@#HXDONE
6702						
6703	037540	012737	037670	001244	HXER11:	MOV #HXP4,@#\$TMP5
6704	037546	012737	037700	001250		MOV #HXP5,@#\$TMP7
6705	037554	000137	037272			JMP @#HXER22
6706	037560	012737	037670	001244	HXER12:	MOV #HXP4,@#\$TMP5
6707	037566	012737	037700	001250		MOV #HXP5,@#\$TMP7
6708	037574	012737	037630	001246		MOV #HXDATO,@#\$TMP6
6709	037602	104002		1\$:	ERROR	2
6710	037604	000137	037740		JMP	@#HXDONE
6711						
6712	037610	012737	037640	001244	HXER13:	MOV #HXP1,@#\$TMP5
6713	037616	012737	037640	001250		MOV #HXP1,@#\$TMP7
6714	037624	000137	037272			JMP @#HXER22
6715						
6716	037630	000000		HXDATO:	0	
6717	037632	000000			0	
6718	037634	000000			0	
6719	037636	000000			0	
6720						
6721	037640	000000		HXP1:	0	
6722	037642	000000			0	
6723	037644	000000			0	
6724	037646	000000			0	
6725						
6726	037650	000577		HXP2:	577	
6727	037652	177776			177776	
6728	037654	177777			177777	
6729	037656	177776			177776	
6730	037660	005201		HXP3:	5201	
6731	037662	000000			0	
6732	037664	000000			0	
6733	037666	000000			0	
6734	037670	100577		HXP4:	100577	
6735	037672	177776			177776	
6736	037674	177777			177777	
6737	037676	177776			177776	
6738	037700	100577		HXP5:	100577	
6739	037702	177776			177776	
6740	037704	000000			0	
6741	037706	000000			0	
6742	037710	000252		HXP6:	252	
6743	037712	125252			125252	

6744 037714 125252
 6745 037716 125252
 6746
 6747 037720 000577
 6748 037722 177776
 6749 037724 000000
 6750 037726 000000
 6751 037730 000577
 6752 037732 177777
 6753 037734 000000
 6754 037736 000000
 6755
 6756 037740
 6757 037740 104413
 6758
 6759
 6760
 6761
 6762
 6763
 6764
 6765
 6766
 6767
 6768
 6769
 6770
 6771
 6772
 6773
 6774 037742 000004
 6775
 6776
 6777 037744
 6778 037744 104414
 6779 037746 004737 040556
 6780 037752 000000 000000 000000
 6781 037760 000000
 6782 037762 000000 000000 000000
 6783 037770 000000
 6784 037772 000200
 6785 037774 000204
 6786 037776 000200
 6787 040000 104002
 6788
 6789
 6790
 6791 040002
 6792 040002 104414
 6793 040004 004737 040556
 6794 040010 000000 000000 000000
 6795 040016 000000
 6796 040020 025252
 6797 040022 052525
 6798 040024 125252
 6799 040026 052525

HXP7: 577
 177776
 0
 0
 HXP8: 577
 177777
 0
 0
 HXDONE: 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?).

```

:*****
:*TEST 42      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
:*
:*****
TST42: SCOPE

;TEST THE CMPD INSTRUCTION WITH (FSRC=AC=0)
AAA1:
    LPERR                                ;SET UP THE LOOP ON ERROR ADDRESS.
    JSR      PC,@#CMPSUB
    .WORD   0,0,0,0                      ;AC0
    1$:
    .WORD   0,0,0,0                      ;FSRC
    2$:
    3$:      200                          ;FPS BEFORE EXECUTION
            204                          ;FPS AFTER EXECUTION
            200                          ;ERROR FPS
    4$:      ERROR 2                      ;FPS ERROR

;TEST CMPD WITH (AC=0) AND FSRC POSITIVE.
AAA2:
    LPERR                                ;SET UP THE LOOP ON ERROR ADDRESS.
    JSR      PC,@#CMPSUB
    .WORD   0,0,0,0                      ;AC
    1$:
    2$:      25252                        ;FSRC
            52525
            125252
            52525
    
```

```

6800 040030 000200      3$:      200      ;FPS BEFORE EXECUTION
6801 040032 000200      200      ;FPS AFTER EXECUTION
6802 040034 000210      210      ;ERROR FPS
6803 040036 104002      4$:      ERROR 2
6804
6805      ;TEST CMPD WITH (AC=0) AND FSRC NEGATIVE
6806 040040      AAA3:
6807 040040 104414      LPERR      ;SET UP THE LOOP ON ERROR ADDRESS.
6808 040042 004737 040556      JSR      PC,@#CMPSUB
6809 040046 000000 000000 000000 1$:      .WORD 0,0,0,0      ;AC
6810 040054 000000
6811 040056 125252      2$:      125252      ;FSRC
6812 040060 125252
6813 040062 052525      52525
6814 040064 125252      125252
6815 040066 000200      3$:      200      ;FPS BEFORE EXECUTION
6816 040070 000210      210      ;FPS AFTER EXECUTION
6817 040072 000200      200      ;ERROR FPS
6818 040074 104002      4$:      ERROR 2
6819
6820      ;TEST CMPD WITH (FSRC=0) AND AC POSITIVE
6821 040076      AAA4:
6822 040076 104414      LPERR      ;SET UP THE LOOP ON ERROR ADDRESS.
6823 040100 004737 040556      JSR      PC,@#CMPSUB
6824 040104 025252      1$:      25252      ;AC
6825 040106 052525      52525
6826 040110 125252      125252
6827 040112 052525      52525
6828 040114 000000 000000 000000 2$:      .WORD 0,0,0,0      ;FSRC
6829 040122 000000
6830 040124 000200      3$:      200      ;FPS BEFORE EXECUTION
6831 040126 000210      210      ;FPS AFTER EXECUTION
6832 040130 000200      200      ;ERROR FPS
6833 040132 104002      4$:      ERROR 2
6834
6835      ;TEST CMPD WITH (FSRC=0) AND AC NEGATIVE
6836      AAA5:
6837 040134
6838 040134 104414      LPERR      ;SET UP THE LOOP ON ERROR ADDRESS.
6839 040136 004737 040556      JSR      PC,@#CMPSUB
6840 040142 125252      1$:      125252      ;AC
6841 040144 125252
6842 040146 052525      52525
6843 040150 125252      125252
6844 040152 000000 000000 000000 2$:      .WORD 0,0,0,0      ;FSRC
6845 040160 000000
6846 040162 000200      3$:      200      ;FPS BEFORE EXECUTION
6847 040164 000200      200      ;FPS AFTER EXECUTION
6848 040166 000210      210      ;ERROR FPS
6849 040170 104002      4$:      ERROR 2
6850
6851      ;TEST CMPD WITH AC POSITIVE AND FSRC NEGATIVE
6852 040172      AAA6:
6853 040172 104414      LPERR      ;SET UP THE LOOP ON ERROR ADDRESS.
6854 040174 004737 040556      JSR      PC,@#CMPSUB
6855 040200 052525      1$:      52525      ;AC
    
```

```

6856 040202 125252 125252
6857 040204 052525 52525
6858 040206 125252 125252
6859 040210 125252 2$: 125252 ;;FSRC
6860 040212 052525 52525
6861 040214 125252 125252
6862 040216 052525 52525
6863 040220 000200 3$: 200 ;FPS BEFORE EXECUTION
6864 040222 000210 210 ;FPS AFTER EXECUTION
6865 040224 000200 200 ;ERROR FPS
6866 040226 104002 4$: ERROR 2
6867
6868
6869 ;TEST CMPD WITH AC NEGATIVE AND FSRC POSITIVE
6870 040230 AAA7:
6871 040230 104414 040556 LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
6872 040232 004737 040556 JSR PC,@#CMPSUB
6873 040236 125252 1$: 125252 ;AC
6874 040240 052525 52525
6875 040242 125252 125252
6876 040244 052525 52525
6877 040246 052525 2$: 52525 ;FSRC
6878 040250 125252 125252
6879 040252 052525 52525
6880 040254 125252 125252
6881 040256 000200 3$: 200 ;FPS BEFORE EXECUTION
6882 040260 000200 200 ;FPS AFTER EXECUTION
6883 040262 000210 210 ;ERROR FPS
6884 040264 104002 4$: ERROR 2
6885
6886 ;TEST CMPD WITH AC POSITIVE AND FSRC POSITIVE
6887 ;AND EAC LESS THAN EFSRC.
6888 040266 AAA8:
6889 040266 104414 040556 LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
6890 040270 004737 040556 JSR PC,@#CMPSUB
6891 040274 012345 1$: 12345 ;AC
6892 040276 067654 67654
6893 040300 032101 32101
6894 040302 023456 23456
6895 040304 023456 2$: 23456 ;FSRC
6896 040306 076543 76543
6897 040310 021012 21012
6898 040312 034567 34567
6899 040314 000200 3$: 200 ;FPS BEFORE EXECUTION
6900 040316 000200 200 ;FPS AFTER EXECUTION
6901 040320 000210 210 ;ERROR FPS
6902 040322 104002 4$: ERROR 2
6903
6904
6905 ;TEST CMPD WITH AC POSITIVE, FSRC POSITIVE AND EAC GREATER THAN EFSRC
6906 040324 AAA9:
6907 040324 104414 040556 LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
6908 040326 004737 040556 JSR PC,@#CMPSUB
6909 040332 045676 1$: 45676 ;AC
6910 040334 054321 54321
6911 040336 012345 12345
    
```

```

6912 040340 067654          67654
6913 040342 034567          34567          ;FSRC
6914 040344 065432          65432
6915 040346 101234          101234
6916 040350 056765          56765
6917 040352 000200          200          ;FPS BEFORE EXECUTION
6918 040354 000210          210          ;FPS AFTER EXECUTION
6919 040356 000200          200          ;ERROR FPS
6920 040360 104002          ERROR 2
6921
6922          ;TEST CMPD WITH AC POSITIVE, FSRC POSITIVE AND AC EQUAL TO FSRC
6923 040362 AAA10:          LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
6924 040362 104414          JSR          PC,@#CMPSUB
6925 040364 004737 040556          12345          ;AC
6926 040370 012345          67012
6927 040372 067012          34567
6928 040374 034567          012345
6929 040376 012345          2$: 12345          ;FSRC
6930 040400 012345          67012
6931 040402 067012          34567
6932 040404 034567          012345
6933 040406 012345          3$: 200          ;FPS BEFORE EXECUTION
6934 040410 000200          204          ;FPS AFTER EXECUTION
6935 040412 000204          200          ;ERROR FPS
6936 040414 000200
6937 040416 104002          4$: ERROR 2
6938
6939          ;TEST CMPD WITH AC POSITIVE, FSRC POSITIVE, EAC EQUAL TO EFSRC,
6940          ;AND FSRC GREATER THAN AC.
6941 040420 AAA11:          LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
6942 040420 104414          JSR          PC,@#CMPSUB
6943 040422 004737 040556          12345          ;AC
6944 040426 012345          67012
6945 040430 067012          34567
6946 040432 034567          012345
6947 040434 012345          2$: 12345          ;FSRC
6948 040436 012345          70123
6949 040440 070123          45670
6950 040442 045670          123456
6951 040444 123456          3$: 200          ;FPS BEFORE EXECUTION
6952 040446 000200          200          ;FPS AFTER EXECUTION
6953 040450 000200          210          ;ERROR FPS
6954 040452 000210
6955 040454 104002          4$: ERROR 2
6956
6957          ;TEST CMPD WITH AC POSITIVE, FSRC POSITIVE, EAC EQUAL TO EFSRC,
6958          ;AND AC GREATER THAN FSRC.
6959 040456 AAA12:          LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
6960 040456 104414          JSR          PC,@#CMPSUB
6961 040460 004737 040556          54321          ;AC
6962 040464 054321          76543
6963 040466 076543          21076
6964 040470 021076          54321
6965 040472 054321          2$: 54321          ;FSRC
6966 040474 054321
6967 040476 065432
    
```



```

6968 040500 107654          107654
6969 040502 032107          32107
6970 040504 000200          3$: 200          ;FPS BEFORE EXECUTION
6971 040506 000210          210          ;FPS AFTER EXECUTION
6972 040510 000200          200          ;ERROR FPS
6973 040512 104002          4$: ERROR 2
6974
6975          ;TEST CMPD WITH AC NEGATIVE, FSRC NEGATIVE, EAC EQUAL TO EFSRC,
6976          ;AND AC GREATER THAN FSRC
6977 040514 AAA13:
6978 040514 104414          LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
6979 040516 004737 040556          JSR PC,@#CMPSUB
6980 040522 112345          1$: 112345          ;AC
6981 040524 043210          43210
6982 040526 076543          76543
6983 040530 021076          21076
6984 040532 112345          2$: 112345          ;FSRC
6985 040534 054321          54321
6986 040536 007654          07654
6987 040540 032107          32107
6988 040542 000200          3$: 200          ;FPS BEFORE EXECUTION
6989 040544 000210          210          ;FPS AFTER EXECUTION
6990 040546 000200          200          ;ERROR FPS
6991 040550 104002          4$: ERROR 2
6992
6993
6994 040552 000137 040746          JMP @#AAADONE          ;FINISHED CMPD TEST.
6995
6996
6997
6998
6999

```

```

;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.

```

```

7000
7001
7002
7003
7004
7005
7006
7007
7008
7009
7010
7011
7012
7013
7014
7015
7016
7017
7018
7019
7020 040556 012601          CMPSUB: MOV (SP)+,R1          ;PICK UP A POINTER TO THE
7021          ;ARGUMENTS.
7022 040560 016100 000020          MOV 20(R1),R0          ;GET THE FPS BEFORE EXECUTION.
7023 040564 170100          LDFPS R0                ;LOAD IT INTO THE FPS.

```



```

7080
7081 040746
7082 040746 104413
7083
7084
7085
7086
7087
7088
7089
7090
7091
7092
7093
7094
7095
7096
7097
7098
7099 040750 000004
7100
7101
7102 040752 104414
7103 040754 012704 040200
7104
7105
7106 040760 170104
7107 040762 012737 041224 000244
7108 040770 012737 041010 001236
7109 040776 012700 041430
7110 041002 172410
7111 041004 012701 041430
7112
7113 041010 174411
7114
7115 041012 170205
7116 041014 170303
7117
7118 041016 012704 140204
7119 041022 020405
7120 041024 001131
7121
7122 041026 012702 000004
7123 041032 020203
7124 041034 001140
7125
7126
7127 041036
7128 041036 104414
7129 041040 012704 040200
7130 041044 170104
7131
7132 041046 012737 041066 001236
7133 041054 012700 041440
7134 041060 172410
7135 041062 012700 041430

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?).

:*****
:*TEST 43 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.
:*
:*****
TST43: SCOPE

:FIRST TEST DIVD WITH (FSRC=AC=0) AND TRAPS DISABLED.
LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
BBB0: MOV #40200,R4 ;SET UP FPS
;WITH INTERRUPTS
;DISABLED.
LDFPS R4
MOV #BBBER1,@#FPVECT;SET UP FOR ANY FP INTERRUPTS.
MOV #BBB1,@#STMP2
MOV #BBBP1,R0 ;SET UP ACO = 0
LDD (R0),ACO
MOV #BBBP1,R1 ;FSRC = 0
BBB1: DIVD (R1),ACO ;TEST INSTRUCTION
STFPS R5 ;GET FPS
STST R3 ;GET FEC
MOV #140204,R4 ;EXPECTED FPS.
CMP R4,R5 ;IS FPS CORRECT.
BNE BBBER2 ;IF INCORRECT BRANCH.
MOV #4,R2 ;EXPECTED FEC.
CMP R2,R3 ;IS FEC CORRECT?
BNE BBBER3 ;IF INCORRECT BRANCH.

:TEST DIVD WITH (FSRC=0) AND TRAPS DISABLED.
BBB2: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
MOV #40200,R4 ;LOAD FPS WITH TRAPS DISABLED.
LDFPS R4
MOV #BBB3,@#STMP2
MOV #BBBP2,R0 ;SET UP ACO OPERAND (NON ZERO).
LDD (R0),ACO
MOV #BBBP1,R0 ;FSRC=0
    
```

```

7136 041066 174410          BBB3: DIVD      (R0),AC0
7137
7138 041070 170205          STFPS     R5          ;GET FPS.
7139 041072 170303          STST      R3          ;GET FEC.
7140
7141 041074 012704 140200    MOV       #140200,R4   ;EXPECTED FPS.
7142 041100 020405          CMP       R4,R5       ;IS FPS CORRECT?
7143 041102 001102          BNE      BBBER2      ;IF INCORRECT BRANCH.
7144
7145 041104 012702 000004    MOV       #4,R2       ;EXPECTED FEC.
7146 041110 020203          CMP       R2,R3       ;WAS FEC CORRECT?
7147 041112 001111          BNE      BBBER3      ;IF INCORRECT BRANCH.
7148
7149          ;TEST DIVD WITH FSRC=0) AND TRAPS ENABLED.
7150          BBB4:
7151 041114 104414          LPERR
7152 041116 012704 000200    MOV       #200,R4     ;SET UP THE LOOP ON ERROR ADDRESS.
7153 041122 170104          LDFPS    R4          ;SET UP FPS. TRAP ENABLED.
7154
7155 041124 012737 041152 001236  MOV       #BBB5,@#$TMP2
7156 041132 012700 041440    MOV       #BBBP2,R0   ;SET UP ACO OPERAND (NON ZERO).
7157 041136 172410          LDD      (R0),AC0
7158
7159 041140 012737 041160 000244  MOV       #BBB6,@#FPVECT ;SET UP FOR THE EXPECTED INTERRUPT.
7160 041146 012700 041430    MOV       #BBBP1,R0   ;FSRC=0
7161
7162 041152 174410          BBB5: DIVD      (R0),AC0 ;TEST INSTRUCTION (SHOULD RESULT IN TRAP).
7163 041154 170000          CFCC
7164
7165 041156 000502          BR       BBBER4      ;GO REPORT FAILURE, NO TRAP.
7166
7167 041160 022716 041154    BBB6: CMP       #BBB5+2,(SP) ;TRAP TO HERE WHEN THE DIVISION BY 0
7168          ;OCCURS. FIRST SEE IF THE ADDRESS OF
7169          ;THE TRAP IS 2+THE ADDRESS OF THE TEST
7170          ;DIVD INSTRUCTION.
7171 041164 001402          BEQ      1$
7172 041166 000137 062524    JMP
7173
7174 041172 170205          1$: STFPS     R5          ;IF NOT THEN REPORT AN UNEXPECTED
7175 041174 170303          STST      R3          ;FP TRAP.
7176 041176 022626          CMP       (SP)+,(SP)+ ;GET FPS.
7177          ;GET FEC.
7178          ;RESET THE STACK.
7178 041200 012704 100200    MOV       #100200,R4   ;EXPECTED FPS.
7179 041204 020405          CMP       R4,R5       ;IS FPS CORRECT?
7180 041206 001040          BNE      BBBER2      ;IF INCORRECT BRANCH.
7181
7182 041210 012702 000004    MOV       #4,R2       ;EXPECTED FEC.
7183 041214 020203          CMP       R2,R3       ;IS FEC CORRECT?
7184 041216 001047          BNE      BBBER3      ;IF INCORRECT BRANCH.
7185
7186 041220 000137 041450    JMP       @#BBBDONE   ;OTHERWISE GO TO NEXT TEST.
7187
7188
7189          ;TRAP HERE IF AN UNEXPECTED INTERRUPT OCCURS.
7190 041224 062737 000002 001236 BBBER1: ADD     #2,@#$TMP2 ;SEE IF THE INTERRUPT OCCURRED
7191          ;DURING THE EXECUTION OF THE DIVD
    
```


;THE DESIRE TO CHANGE THE SOFTWARE
 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
 ;THE USER TYPED CONTROL G?).

7248
 7249
 7250
 7251
 7252
 7253
 7254
 7255
 7256
 7257
 7258
 7259
 7260
 7261
 7262 041452 000004
 7263
 7264
 7265 041454
 7266 041454 104414
 7267 041456 004767 000552
 7268 041462 000000 000000
 7269 041466 012345 067012
 7270 041472 000000 000000
 7271 041476 000000
 7272 041500 000004
 7273 041502 012345 067012
 7274 041506 104002
 7275
 7276
 7277 041510
 7278 041510 104414
 7279 041512 004737 042234
 7280 041516 065652 125252
 7281 041522 065600 000000
 7282 041526 040252 125252
 7283 041532 003000
 7284 041534 003000
 7285 041536 040052 125252
 7286 041542 104002
 7287
 7288
 7289 041544
 7290 041544 104414
 7291 041546 004767 000462
 7292 041552 076400 000000
 7293 041556 076400 000000
 7294 041562 040200 000000
 7295 041566 001000
 7296 041570 001000
 7297 041572 140200 000000
 7298
 7299 041576 104002
 7300
 7301
 7302 041600
 7303 041600 104414

```

:*****
:*TEST 44      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.
:*
:*****
TST44: SCOPE

;CHECK DIVF WITH (AC=0).
CCC1:
      LPERR
      JSR      PC,DIVFSUB      ;SET UP THE LOOP ON ERROR ADDRESS.
1$:   .WORD   0,0              ;AC
2$:   .WORD   12345,67012     ;FSRC
3$:   .WORD   0,0              ;RES
4$:   0
      ;FPS BEFORE EXECUTION.
      ;FPS AFTER EXECUTION
6$:   .WORD   12345,67012     ;ERROR RESULT
      ERROR   2

;TEST DIVF WITH AC POSITIVE, FSRC POSITIVE AND IN ROUND MODE.
CCC2:
      LPERR
      JSR      PC,@DIVFSUB    ;SET UP THE LOOP ON ERROR ADDRESS.
1$:   .WORD   65652,125252    ;AC
2$:   .WORD   65600,0         ;FSRC
3$:   .WORD   40252,125252    ;RES
4$:   3000
      ;FPS BEFORE EXECUTION.
      ;FPS AFTER EXECUTION.
6$:   .WORD   40052,125252    ;ERROR RESULT.
      ERROR   2

;TEST DIVF WITH AC POSITIVE, FSRC POSITIVE.
CCC3:
      LPERR
      JSR      PC,DIVFSUB    ;SET UP THE LOOP ON ERROR ADDRESS.
1$:   .WORD   76400,0         ;AC
2$:   .WORD   76400,0         ;FSRC
3$:   .WORD   40200,0         ;RES
4$:   1000
      ;FPS BEFORE EXECUTION.
      ;FPS AFTER EXECUTION.
6$:   .WORD   140200,0        ;ERROR RES.
      ERROR   2
      ;SIGN BAD

;TEST DIVF WITH BOTH OPERANDS POSITIVE.
CCC4:
      LPERR
      ;SET UP THE LOOP ON ERROR ADDRESS.
    
```

```

7304 041602 004737 042234      JSR      PC,@#DIVFSUB
7305 041606 056777 177777      1$:      .WORD    56777,177777      ;AC
7306 041612 054200 000000      2$:      .WORD    54200,0          ;FSRC
7307 041616 042777 177777      3$:      .WORD    42777,177777      ;RES
7308 041622 000000      4$:      0                          ;FPS BEFORE EXECUTION.
7309 041624 000000      0                          ;FPS AFTER EXECUTION.
7310 041626 002000 002000      5$:      .WORD    2000,2000          ;ERROR RES.
7311 041632 104002      6$:      ERROR      2
7312
7313      ;TEST THE DIVF INSTRUCTION:
7314 041634      CCC5:
7315 041634 104414      LPERR
7316 041636 004737 042234      JSR      PC,@#DIVFSUB      ;SET UP THE LOOP ON ERROR ADDRESS.
7317 041642 012377 177777      1$:      .WORD    12377,177777      ;AC
7318 041646 012300 000000      2$:      .WORD    12300,0          ;FSRC
7319 041652 040252 125252      3$:      .WORD    40252,125252      ;RES
7320 041656 000000      4$:      0                          ;FPS BEFORE EXECUTION.
7321 041660 000000      0                          ;FPS AFTER EXECUTION.
7322 041662 177777 177777      5$:      .WORD    -1,-1            ;ERROR RES.
7323 041666 104002      6$:      ERROR      2
7324
7325      ;TEST DIVIDE ALGORITHM. TEST ROUND CONSTANT.
7326 041670      CCC6:
7327 041670 104414      LPERR      ;SET UP THE LOOP ON ERROR ADDRESS.
7328 041672 004737 042234      JSR      PC,@#DIVFSUB
7329 041676 064600 000001      1$:      .WORD    64600,1          ;AC
7330 041702 066600 000000      2$:      .WORD    66600,0          ;FSRC
7331 041706 036200 000001      3$:      .WORD    36200,1          ;RES
7332 041712 000000      4$:      0                          ;FPS BEFORE EXECUTION.
7333 041714 000000      0                          ;FPS AFTER EXECUTION.
7334 041716 003000 003000      5$:      .WORD    3000,3000          ;ERROR RES.
7335 041722 104002      6$:      ERROR      2
7336
7337      ;TEST DIVF.
7338 041724      CCC7:
7339 041724 104414      LPERR      ;SET UP THE LOOP ON ERROR ADDRESS.
    
```

7340	041726	004737	042234		JSR	PC,@#DIVFSUB	
7341	041732	034577	177776	1\$:	.WORD	34577,177776	:AC
7342	041736	023400	000000	2\$:	.WORD	23400,0	:FSRC
7343	041742	051377	177776	3\$:	.WORD	51377,177776	:RES
7344	041746	000017		4\$:	17		:FPS BEFORE EXECUTION.
7345	041750	000000			0		:FPS AFTER EXECUTION.
7346	041752	003400	003400	5\$:	.WORD	3400,3400	:ERROR RES.
7347	041756	104002		6\$:	ERROR	2	
7348							
7349							
7350							
7351	041760						
7352	041760	104414			LPERR		:SET UP THE LOOP ON ERROR ADDRESS.
7353	041762	004737	042234		JSR	PC,@#DIVFSUB	
7354	041766	067652	125252	1\$:	.WORD	67652,125252	:AC
7355	041772	056500	000000	2\$:	.WORD	56500,0	:FSRC
7356	041776	051343	107070	3\$:	.WORD	51343,107070	:RES
7357	042002	000000		4\$:	0		:FPS BEFORE EXECUTION.
7358	042004	000000			0		:FPS AFTER EXECUTION.
7359	042006	051543	107070	5\$:	.WORD	51543,107070	:ERROR RES.
7360	042012	104002		6\$:	ERROR	2	:DIDN'T INCREMENT THE EXPONENT :AFTER DIVID NORMALIZATION.
7361							
7362							
7363							
7364	042014						
7365	042014	104414			LPERR		:SET UP THE LOOP ON ERROR ADDRESS.
7366	042016	004737	042234		JSR	PC,@#DIVFSUB	
7367	042022	140400	000000	1\$:	.WORD	140400,0	:AC
7368	042026	140500	000000	2\$:	.WORD	140500,0	:FSRC
7369	042032	040052	125253	3\$:	.WORD	040052,125253	:RES
7370	042036	000000		4\$:	0		:FPS BEFORE EXECUTION.
7371	042040	000000			0		:FPS AFTER EXECUTION.
7372	042042	140052	125253	5\$:	.WORD	140052,125253	:ERROR RES.
7373	042046	104002		6\$:	ERROR	2	:BAD SIGN.
7374							
7375							
7376	042050						
7377	042050	104414			LPERR		:SET UP THE LOOP ON ERROR ADDRESS.
7378	042052	004737	042234		JSR	PC,@#DIVFSUB	
7379	042056	160077	000000	1\$:	.WORD	160077,0	:AC
7380	042062	040277	000000	2\$:	.WORD	40277,0	:FSRC
7381	042066	160000	000000	3\$:	.WORD	160000,0	:RES
7382	042072	000007		4\$:	7		:FPS BEFORE EXECUTION.
7383	042074	000010			10		:FPS AFTER EXECUTION.
7384	042076	060000	000000	5\$:	.WORD	60000,0	:ERROR RES.
7385	042102	104002		6\$:	ERROR	2	:BAD SIGN.
7386							
7387							
7388	042104						
7389	042104	104414			LPERR		:SET UP THE LOOP ON ERROR ADDRESS.
7390	042106	004737	042234		JSR	PC,@#DIVFSUB	
7391	042112	040400	000000	1\$:	.WORD	40400,0	:AC
7392	042116	140500	000000	2\$:	.WORD	140500,0	:FSRC
7393	042122	140052	125253	3\$:	.WORD	140052,125253	:RES
7394	042126	000017		4\$:	17		:FPS BEFORE EXECUTION.
7395	042130	000010			10		:FPS AFTER EXECUTION.

7396 042132 040052 125253
 7397 042136 104002
 7398
 7399
 7400
 7401 042140
 7402 042140 104414
 7403 042142 004737 042234
 7404 042146 060100 000001
 7405 042152 040300 000000
 7406 042156 060000 000000
 7407 042162 000052
 7408 042164 000040
 7409 042166 060000 000001
 7410 042172 104002
 7411
 7412
 7413 042174
 7414 042174 104414
 7415 042176 004767 000032
 7416 042202 060100 000001
 7417 042206 040300 000000
 7418 042212 060000 000001
 7419 042216 000005
 7420 042220 000000
 7421 042222 060000 000000
 7422 042226 104002
 7423
 7424 042230 000137 042460
 7425
 7426
 7427
 7428
 7429
 7430
 7431
 7432
 7433
 7434
 7435
 7436
 7437
 7438
 7439
 7440
 7441
 7442
 7443
 7444
 7445
 7446
 7447
 7448
 7449
 7450
 7451

```

5$: .WORD 40052,125253 ;ERROR RES.
6$: ERROR 2 ;BAD SIGN.

;TEST DIVF BOTH OPERANDS POSITIVE AND TRUNCATE MODE.
CCC12:
LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
JSR PC,@#DIVFSUB
1$: .WORD 60100,1 ;AC
2$: .WORD 40300,0 ;FSRC
3$: .WORD 60000,0 ;RES
4$: 52 ;FPS BEFORE EXECUTION.
40 ;FPS AFTER EXECUTION.
5$: .WORD 60000,1 ;ERROR RES.
6$: ERROR 2 ;TRUNCATION ERROR

;DIVF WITH POSITIVE OPERANDS AND ROUND MODE.
CCC13:
LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
JSR PC,DIVFSUB
1$: .WORD 60100,1 ;AC
2$: .WORD 40300,0 ;FSRC
3$: .WORD 60000,1 ;RES
4$: 5 ;FPS BEFORE EXECUTION.
0 ;FPS AFTER EXECUTION.
5$: .WORD 60000,0 ;ERROR RES.
6$: ERROR 2 ;ROUND ERROR.

JMP @#CCCDONE ;GO TO NEXT TEST.
    
```

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

```

:
: JSR PC,@#DIVFSUB
: ACARG: .WORD X,X ;AC OPERAND
: FSRCARG: .WORD X,X ;FSRC OPERAND
: RES: .WORD X,X ;EXPECTED RESULT
: FPSB: .WORD X ;FPS BEFORE EXECUTION
: FPSA: .WORD X ;FPS AFTER EXECUTION
: ERRES: .WORD X,X ;ERROR RESULT
: ERR: ERROR X ;RESULT ERROR
: CONT: ;RETURN ADDRESS
    
```

;THE OPERANDS ARE SET UP (USING ACO FOR THE AC OPERAND). THEN
 ;FPSB IS LOADED INTO THE FPS. THE INSTRUCTION, DIVF IS EXECUTED.
 ;AFTER THE EXECUTION THE RESULT IS CHECKED AGAINST THE
 ;EXPECTED CORRECT RESULT, RES. IF IT IS CORRECT THEN THE FPS
 ;IS CHECKED WITH THE EXPECTED CORRECT FPS, FPSA. IF THE FPS WAS
 ;INCORRECT THEN IT IS REPORTED. IF THE RESULT WAS INCORRECT IT
 ;IS COMPARED WITH ERRES IN AN ATTEMPT TO ANALYSE THE ERROR. IF
 ;THE INCORRECT RESULT MATCHED ERRES THEN CONTROL IS PASSED TO
 ;THE ERROR CALL AT ERR. IF THE INCORRECT RESULT DID NOT MATCH ERRES
 ;THEN THE FAILURE IS REPORTED IN DIVFSUB AND CONTROL IS PASSED TO
 ;CONT. IF NO ERRORS ARE DETECTED THEN DIVFSUB RETURNS CONTROL
 ;TO CONT.

```

7452 042234 012601          DIVFSUB:  MOV      (SP)+,R1      ;GET A POINTER TO THE ARGUMENTS.
7453 042236 012700 000200      MOV      #200,R0      ;SET FD MODE.
7454 042242 170100          LDFPS   R0
7455 042244 010100          MOV      R1,R0        ;LOAD THE AC OPERAND.
7456 042246 172410          LDD     (R0),AC0
7457 042250 016100 000014      MOV      14(R1),R0    ;LOAD THE FPS
7458 042254 170100          LDFPS   R0
7459 042256 012737 042272 001236  MOV      #1$,@#STMP2
7460 042264 010100          MOV      R1,R0
7461 042266 062700 000004      ADD     #4,R0         ;ESTABLISH A POINTER TO FSRC.
7462
7463 042272 174410          1$:    DIVF   (R0),AC0    ;TEST INSTRUCTION.
7464
7465 042274 170204          STFPS   R4            ;GET THE FPS.
7466 042276 012700 000200      MOV      #200,R0      ;SET FD MODE
7467 042302 170100          LDFPS   R0
7468
7469 042304 012700 042450      MOV      #DIVFT,R0    ;GET THE RESULT OF THE DIVF.
7470 042310 174010          STD     AC0,(R0)
7471
7472 042312 010102          MOV      R1,R2        ;SAVE THE DATA IN CASE OF ERROR.
7473 042314 010237 001240      MOV      R2,@#STMP3
7474 042320 062702 000004      ADD     #4,R2
7475 042324 010237 001242      MOV      R2,@#STMP4
7476 042330 062702 000004      ADD     #4,R2
7477 042334 010237 001244      MOV      R2,@#STMP5
7478 042340 012737 042450 001246  MOV      #DIVFT,@#STMP6
7479 042346 010437 001250      MOV      R4,@#STMP7
7480 042352 016137 000016 001252  MOV      16(R1),@#STMP10
7481
7482 042360 021061 000010          CMP     (R0),10(R1)   ;IS THE RESULT CORRECT?
7483 042364 001011          BNE    10$           ;IF INCORRECT BRANCH.
7484 042366 026061 000002 000012  CMP     2(R0),12(R1)
7485 042374 001005          BNE    10$
7486
7487 042376 026104 000016          CMP     16(R1),R4     ;IS FPS CORRECT?
7488 042402 001020          BNE    15$           ;IF INCORRECT BRANCH.
7489 042404 000161 000026          JMP     26(R1)        ;IF NO ERRORS OCCURRED RETURN.
7490
7491 042410 021061 000020          10$:   CMP     (R0),20(R1)   ;DOES THE INCORRECT RESULT
7492 042414 001010          BNE    11$           ;MATCH THE ANTICIPATED INCORRECT RESULT.
7493 042416 026061 000002 000022  CMP     2(R0),22(R1)
7494 042424 001004          BNE    11$           ;BRANCH IF NO.
7495
7496 042426 010102          MOV     R1,R2        ;IT MATCHED SO RETURN TO THE ERROR
7497                                ;REPORT AT THE CALLING ROUTINE.
7498 042430 062702 000024          ADD     #24,R2
7499 042434 000112          JMP     (R2)
7500
7501 042436          11$:
7502 042436 104002          12$:   ERROR  2
7503 042440 000161 000026          13$:   JMP     26(R1)
7504
7505 042444          15$:
7506 042444 104002          16$:   ERROR  2
7507 042446 000774          BR     13$
    
```

```

7508
7509 042450 000000 000000 000000 DIVFT: .WORD 0,0,0,0
7510 042456 000000
7511
7512 042460          CCCDONE:
7513 042460 104413          RSETUP
7514
7515
7516
7517
7518
7519
7520
7521
7522
7523
7524
7525
7526
7527
7528 042462 000004
7529
7530
7531 042464
7532 042464 104414
7533 042466 004737 043154
7534 042472 034277 000000 000000 1$: .WORD 34277,0,0,0 ;AC
7535 042500 000000
7536 042502 040277 000000 000000 2$: .WORD 40277,0,0,0 ;FSRC
7537 042510 000000
7538 042512 034200 000000 000000 3$: .WORD 34200,0,0,0 ;RES
7539 042520 000000
7540 042522 000200          4$: 200 ;FPS BEFORE EXECUTION.
7541 042524 000200          4$: 200 ;FPS AFTER EXECUTION.
7542 042526 177777 177777 177777 5$: .WORD -1,-1,-1,-1 ;ERROR RES.
7543 042534 177777
7544 042536 104002
7545
7546
7547 042540
7548 042540 104414
7549 042542 004737 043154
7550 042546 134277 000000 000000 1$: .WORD 134277,0,0,0 ;AC
7551 042554 000000
7552 042556 040277 000000 000000 2$: .WORD 40277,0,0,0 ;FSRC
7553 042564 000000
7554 042566 134200 000000 000000 3$: .WORD 134200,0,0,0 ;RES
7555 042574 000000
7556 042576 000207          4$: 207 ;FPS BEFORE EXECUTION.
7557 042600 000210          4$: 210 ;FPS AFTER EXECUTION.
7558 042602 177777 177777 177777 5$: .WORD -1,-1,-1,-1 ;ERROR RESULT.
7559 042610 177777
7560 042612 104002
7561
7562
7563 042614
    
```

```

;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).
    
```

```

:*****
:*TEST 45          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.
:*
:*****
    
```

```

TST45: SCOPE
;DIVD TEST WITH POSITIVE OPERANDS AND IN ROUND MODE.
DDD1:
    LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
    JSR            PC,@#DIVDSUB
    .WORD          34277,0,0,0
    1$:
    .WORD          40277,0,0,0 ;FSRC
    2$:
    .WORD          34200,0,0,0 ;RES
    3$:
    200            ;FPS BEFORE EXECUTION.
    200            ;FPS AFTER EXECUTION.
    5$:
    .WORD          -1,-1,-1,-1 ;ERROR RES.
    6$:
    ERROR          2
    
```

```

;DIVD WITH AC NEGATIVE AND FSRC POSITIVE IN TRUNCATE MODE.
DDD2:
    LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
    JSR            PC,@#DIVDSUB
    .WORD          134277,0,0,0 ;AC
    1$:
    .WORD          40277,0,0,0 ;FSRC
    2$:
    .WORD          134200,0,0,0 ;RES
    3$:
    207            ;FPS BEFORE EXECUTION.
    210            ;FPS AFTER EXECUTION.
    5$:
    .WORD          -1,-1,-1,-1 ;ERROR RESULT.
    6$:
    ERROR          2
    
```

```

;DIVD TEST WITH OPERANDS BOTH NEGATIVE AND IN TRUNCATE MODE.
DDD3:
    
```

```

7564 042614 104414          LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
7565 042616 004767 000332  JSR          PC, DIVDSUB
7566 042622 134300 000000 000000 1$: .WORD      134300,0,0,1  ;AC
7567 042630 000001          .WORD          ;FSRC
7568 042632 140300 000000 000000 2$: .WORD      140300,0,0,0
7569 042640 000000          .WORD          ;RES
7570 042642 034200 000000 000000 3$: .WORD      34200,0,0,0
7571 042650 000000          .WORD          ;FPS BEFORE EXECUTION.
7572 042652 000250          4$: 250        ;FPS AFTER EXECUTION.
7573 042654 000240          .WORD          ;ERROR RES.
7574 042656 034200 000000 000000 5$: .WORD      34200,0,0,1
7575 042664 000001          .WORD          ;TRUNCATION ERROR.
7576 042666 104002          6$: ERROR      2
7577
7578          ;DIVD WITH AC POSITIVE AND FSRC NEGATIVE IN ROUND MODE.
7579          DDD4:
7580 042670 104414          LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
7581 042672 004737 043154  JSR          PC, @#DIVDSUB
7582 042676 034300 000000 000000 1$: .WORD      34300,0,0,1  ;AC
7583 042704 000001          .WORD          ;FSRC
7584 042706 140300 000000 000000 2$: .WORD      140300,0,0,0
7585 042714 000000          .WORD          ;RES
7586 042716 134200 000000 000000 3$: .WORD      134200,0,0,1
7587 042724 000001          .WORD          ;FPS BEFORE EXECUTION.
7588 042726 000207          4$: 207        ;FPS AFTER EXECUTION.
7589 042730 000210          .WORD          ;ERROR RES.
7590 042732 134200 000000 000000 5$: .WORD      134200,0,0,0
7591 042740 000000          .WORD          ;ROUND ERROR.
7592 042742 104002          6$: ERROR      2
7593
7594          ;DIVD TEST.
7595          DDD5:
7596 042744 104414          LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
7597 042746 004737 043154  JSR          PC, @#DIVDSUB
7598 042752 100400 000000 000000 1$: .WORD      100400,0,0,0  ;AC
7599 042760 000000          .WORD          ;FSRC
7600 042762 000500 000000 000000 2$: .WORD      500,0,0,0
7601 042770 000000          .WORD          ;RES
7602 042772 140052 125252  .WORD      140052,125252
7603 042776 125252 125252  .WORD      125252,125252
7604 043002 007647          4$: 7647        ;FPS BEFORE EXECUTION.
7605 043004 007650          .WORD          ;FPS AFTER EXECUTION.
7606 043006 177777 177777 177777 5$: .WORD      -1,-1,-1,-1 ;ERROR RES.
7607 043014 177777          .WORD          ;ROUND ERROR.
7608 043016 104002          6$: ERROR      2
7609
7610          ;DIVD TEST WITH AC POSITIVE AND FSRC NEGATIVE IN ROUND MODE.
7611          DDD6:
7612 043020 104414          LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
7613 043020 004737 043154  JSR          PC, @#DIVDSUB
7614 043022 000400 000000 000000 1$: .WORD      400,0,0,0  ;AC
7615 043026 000000          .WORD          ;FSRC
7616 043034 100500 000000 000000 2$: .WORD      100500,0,0,0
7617 043036 000000          .WORD          ;RES
7618 043044 140052 125252  .WORD      140052,125252
7619 043046 140052 125252  .WORD      140052,125252
    
```

```

7620 043052 125252 125253          .WORD 125252,125253
7621 043056 007707          4$: 7707          ;FPS BEFORE EXECUTION.
7622 043060 007710          7710          ;FPS AFTER EXECUTION.
7623 043062 177777 177777 177777 5$: .WORD -1,-1,-1,-1 ;ERROR RES.
7624 043070 177777
7625 043072 104002          6$: ERROR 2
7626
7627          ;DIVD TEST.
7628 043074          DDD7:
7629 043074 104414          LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
7630 043076 004737 043154          JSR PC,@#DIVDSUB
7631 043102 170360 170360          1$: .WORD 170360,170360 ;AC
7632 043106 170360 170360          .WORD 170360,170360
7633 043112 170360 170360          2$: .WORD 170360,170360 ;FSRC
7634 043116 170360 170360          .WORD 170360,170360
7635 043122 040200 000000 000000 3$: .WORD 40200,0,0,0 ;RES
7636 043130 000000
7637 043132 007717          4$: 7717          ;FPS BEFORE EXECUTION.
7638 043134 007700          7700          ;FPS AFTER EXECUTION.
7639 043136 177777 177777 177777 5$: .WORD -1,-1,-1,-1 ;ERROR RES.
7640 043144 177777
7641 043146 104002          6$: ERROR 2
7642
7643 043150 000137 043414          JMP @#DDDDONE ;GO TO NEXT TEST.
7644
7645
7646
7647
7648
7649
7650
7651
7652
7653
7654
7655
7656
7657
7658
7659
7660
7661
7662
7663
7664
7665
7666
7667
7668
7669
7670
7671
7672
7673 043154 012601
7674 043156 012700 000200
7675 043162 170100
    
```

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

```

:
:      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:
    
```

:THE OPERANDS ARE SET UP (USING ACO FOR THE AC OPERAND). THEN
 :FPSB IS LOADED INTO THE FPS. THE INSTRUCTION, DIVD IS EXECUTED.
 :AFTER THE EXECUTION THE RESULT IS CHECKED AGAINST THE
 :EXPECTED CORRECT RESULT, RES. IF IT IS CORRECT THEN THE FPS
 :IS CHECKED WITH THE EXPECTED CORRECT FPS, FPSA. IF THE FPS WAS
 :INCORRECT THEN IT IS REPORTED. IF THE RESULT WAS INCORRECT IT

:IS COMPARED WITH ERRES IN AN ATTEMPT TO ANALYSE THE ERROR. IF
 :THE INCORRECT RESULT MATCHED ERRES THEN CONTROL IS PASSED TO
 :THE ERROR CALL AT ERR. IF THE INCORRECT RESULT DID NOT MATCH ERRES
 :THEN THE FAILURE IS REPORTED IN DIVDSUB AND CONTROL IS PASSED TO
 :CONT. IF NO ERRORS ARE DETECTED THEN DIVDSUB RETURNS CONTROL
 :TO CONT.

```

DIVDSUB:  MOV      (SP)+,R1          ;GET A POINTER TO THE ARGUMENTS.
          MOV      #200,R0         ;SET FD MODE.
          LDFPS   R0
    
```

7676											
7677	043164	010100				MOV	R1,R0				;SET UP THE ACO OPERAND.
7678	043166	172410				LDD	(R0),ACO				
7679	043170	016100	000030			MOV	30(R1),R0				;LOAD THE FPS.
7680	043174	170100				LDFPS	R0				
7681											
7682	043176	012737	043212	001236		MOV	#1\$,@#STMP2				
7683	043204	010100				MOV	R1,R0				;ESTABLISH A POINTER TO FSRC.
7684	043206	062700	000010			ADD	#10,R0				
7685											
7686	043212	174410			1\$:	DIVD	(R0),ACO				;EXECUTE THE TEST INSTRUCTION.
7687											
7688	043214	170204				STFPS	R4				;GET THE FPS.
7689	043216	012700	000200			MOV	#200,R0				;SET FD MODE.
7690	043222	170100				LDFPS	R0				
7691											
7692	043224	012700	043404			MOV	#DIVDT,R0				;GET THE RESULT.
7693	043230	174010				STD	ACO,(R0)				
7694											
7695	043232	010102				MOV	R1,R2				;SAVE DATA IN CASE OF ERROR.
7696	043234	010237	001240			MOV	R2,@#STMP3				
7697	043240	062702	000010			ADD	#10,R2				
7698	043244	010237	001242			MOV	R2,@#STMP4				
7699	043250	062702	000010			ADD	#10,R2				
7700	043254	010237	001244			MOV	R2,@#STMP5				
7701	043260	012737	043404	001246		MOV	#DIVDT,@#STMP6				
7702	043266	010437	001250			MOV	R4,@#STMP7				
7703	043272	016137	000032	001252		MOV	32(R1),@#STMP10				
7704											
7705	043300	010102				MOV	R1,R2				;CHECK THE RESULT.
7706	043302	062702	000020			ADD	#20,R2				
7707	043306	012703	043404			MOV	#DIVDT,R3				
7708	043312	012705	000004			MOV	#4,R5				
7709	043316	022223			2\$:	CMP	(R2)+,(R3)+				
7710	043320	001006				BNE	10\$;BRANCH IF RESULT INCORRECT.
7711	043322	077503				SOB	R5,2\$				
7712											
7713	043324	026104	000032			CMP	32(R1),R4				;IS FPS CORRECT?
7714	043330	001023				BNE	15\$;BRANCH IF INCORRECT.
7715	043332	000161	000046			JMP	46(R1)				;RETURN.
7716											
7717	043336	010102			10\$:	MOV	R1,R2				;WAS INCORRECT RESULT ANTICIPATED?
7718	043340	062702	000034			ADD	#34,R2				
7719	043344	012703	043404			MOV	#DIVDT,R3				
7720	043350	012705	000004			MOV	#4,R5				
7721	043354	022223			11\$:	CMP	(R2)+,(R3)+				
7722	043356	001005				BNE	12\$;BRANCH IF NO.
7723	043360	077503				SOB	R5,11\$				
7724	043362	010102				MOV	R1,R2				;IF THE INCORRECT RESULT WAS
7725	043364	062702	000044			ADD	#44,R2				;ANTICIPATED RETURN TO THE
7726											;ERROR REPORT IN THE CALLING
7727	043370	000112				JMP	(R2)				;ROUTINE.
7728											
7729	043372				12\$:						;REPORT RESULT INCORRECT.
7730	043372	104002			13\$:	ERROR	2				
7731	043374	000161	000046		14\$:	JMP	46(R1)				

```

7732
7733 043400          15$:
7734 043400 104002  16$:  ERROR  2
7735 043402 000774          BR    14$
7736
7737 043404 000000 000000 000000 DIVDT: .WORD 0,0,0,0
7738 043412 000000
7739
7740 043414          DDDDONE:
7741 043414 104413          RSETUP
7742
7743
7744
7745
7746
7747
7748
7749
7750
7751
7752
7753
7754
7755
7756
7757 043416 000004
7758
7759
7760 043420
7761 043420 104414
7762 043422 004737 044200
7763 043426 000000 000000
7764 043432 000000 000000
7765 043436 000000 000000
7766 043442 007517
7767 043444 007504
7768 043446 177777 177777
7769 043452 104002
7770
7771
7772 043454
7773 043454 104414
7774 043456 004737 044200
7775 043462 071625 034435
7776 043466 000000 000000
7777 043472 000000 000000
7778 043476 000013
7779 043500 000004
7780 043502 177777 177777
7781
7782 043506 104002
7783
7784
7785 043510
7786 043510 104414
7787 043512 004737 044200
    
```

```

;REPORT FPS INCORRECT.

;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;*****
;TEST 46      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.
;
;*****
TST46:  SCOPE

;MULF WITH (FSRC=AC=0)
EEE1:
    LPERR
    JSR    PC,@#MULFSUB
    1$:   .WORD 0,0
    2$:   .WORD 0,0
    3$:   .WORD 0,0
    4$:   7517
        7504
    5$:   .WORD -1,-1
    6$:   ERROR 2
;SET UP THE LOOP ON ERROR ADDRESS.
;AC
;FSRC
;RES
;FPS BEFORE EXECUTION.
;FPS AFTER EXECUTION.

;MULF WITH (FSRC=0).
EEE2:
    LPERR
    JSR    PC,@#MULFSUB
    1$:   .WORD 71625,34435
    2$:   .WORD 0,0
    3$:   .WORD 0,0
    4$:   13
        4
    5$:   .WORD -1,-1
    6$:   ERROR 2
;SET UP THE LOOP ON ERROR ADDRESS.
;AC
;FSRC
;RES
;FPS BEFORE EXECUTION.
;FPS AFTER EXECUTION.
;ERROR RES.

;MULF WITH (AC=0)
EEE3:
    LPERR
    JSR    PC,@#MULFSUB
;SET UP THE LOOP ON ERROR ADDRESS.
    
```

7788	043516	000000	000000	1\$:	.WORD	0,0	:AC
7789	043522	071625	153443	2\$:	.WORD	071625,153443	:FSRC
7790	043526	000000	000000	3\$:	.WORD	0,0	:RES
7791	043532	007500		4\$:	7500		:FPS BEFORE EXECUTION.
7792	043534	007504			7504		:FPS AFTER EXECUTION.
7793	043536	177777	177777	5\$:	.WORD	-1,-1	:ERROR RES.
7794	043542	104002		6\$:	ERROR	2	
7795							
7796							
7797	043544						
7798	043544	104414			LPERR		:SET UP THE LOOP ON ERROR ADDRESS.
7799	043546	004737	044200		JSR	PC,@#MULFSUB	
7800	043552	040200	000000	1\$:	.WORD	40200,0	:AC
7801	043556	040177	177777	2\$:	.WORD	40177,-1	:FSRC
7802	043562	040177	177777	3\$:	.WORD	40177,-1	:RES
7803	043566	000017		4\$:	17		:FPS BEFORE EXECUTION.
7804	043570	000000			0		:FPS AFTER EXECUTION.
7805	043572	140177	177777	5\$:	.WORD	140177,-1	:ERROR RES.
7806	043576	104002		6\$:	ERROR	2	:BAD SIGN.
7807							
7808							
7809	043600						
7810	043600	104414			LPERR		:SET UP THE LOOP ON ERROR ADDRESS.
7811	043602	004767	000372		JSR	PC,MULFSUB	
7812	043606	040177	177777	1\$:	.WORD	40177,-1	:AC
7813	043612	040200	000000	2\$:	.WORD	40200,0	:FSRC
7814	043616	040177	177777	3\$:	.WORD	40177,-1	:RES
7815	043622	000040		4\$:	40		:FPS BEFORE EXECUTION.
7816	043624	000040			40		:FPS AFTER EXECUTION.
7817	043626	037777	177777	5\$:	.WORD	37777,-1	:ERROR RES.
7818	043632	104002		6\$:	ERROR	2	:ST 252 TO 044 INTO 444 (BUT Y62) :MUL. NORMALIZATION FAILURE.
7819							
7820							
7821							
7822	043634						
7823	043634	104414			LPERR		:SET UP THE LOOP ON ERROR ADDRESS.
7824	043636	004737	044200		JSR	PC,@#MULFSUB	
7825	043642	040100	000000	1\$:	.WORD	40100,0	:AC
7826	043646	040100	000000	2\$:	.WORD	40100,0	:FSRC
7827	043652	040020	000000	3\$:	.WORD	40020,0	:RES
7828	043656	000012		4\$:	12		:FPS BEFORE EXECUTION.
7829	043660	000000			0		:FPS AFTER EXECUTION.
7830	043662	042040	000000	5\$:	.WORD	42040,0	:ERROR RES.
7831	043666	104002		6\$:	ERROR	2	:ST 252 TO 444 INTO 042 (BUT Y62) :MUL. NORMALIZATION FAILURE.
7832							
7833							
7834							
7835	043670						
7836	043670	104414			LPERR		:SET UP THE LOOP ON ERROR ADDRESS.
7837	043672	004737	044200		JSR	PC,@#MULFSUB	
7838	043676	017500	000000	1\$:	.WORD	17500,0	:AC
7839	043702	023652	125252	2\$:	.WORD	23652,125252	:FSRC
7840	043706	003177	177777	3\$:	.WORD	3177,-1	:RES
7841	043712	007417		4\$:	7417		:FPS BEFORE EXECUTION.
7842	043714	007400			7400		:FPS AFTER EXECUTION.
7843	043716	177777	177777	5\$:	.WORD	-1,-1	


```

7844 043722 104002
7845
7846
7847 043724
7848 043724 104414
7849 043726 004737 044200
7850 043732 040342 000000
7851 043736 176542 000000
7852 043742 176707 102000
7853 043746 000007
7854 043750 000010
7855 043752 076507 102000
7856 043756 104002
7857
7858
7859 043760
7860 043760 104414
7861 043762 004737 044200
7862 043766 140200 000000
7863 043772 007417 007417
7864 043776 107417 007417
7865 044002 000000
7866 044004 000010
7867 044006 007417 007417
7868 044012 104002
7869
7870
7871 044014
7872 044014 104414
7873 044016 004737 044200
7874 044022 144600 000000
7875 044026 154000 000000
7876 044032 060400 000000
7877 044036 000017
7878 044040 000000
7879 044042 160400 000000
7880 044046 104002
7881
7882
7883 044050
7884 044050 104414
7885 044052 004737 044200
7886 044056 140300 000000
7887 044062 160000 000001
7888 044066 060100 000002
7889 044072 000010
7890 044074 000000
7891 044076 060100 000001
7892 044102 104002
7893
7894
7895 044104
7896 044104 104414
7897 044106 004737 044200
7898 044112 060000 000001
7899 044116 140300 000000

```

```

6$: ERROR 2
;MULF WITH AC POSITIVE AND FSRC NEGATIVE IN ROUND MODE.
EEE8: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
JSR PC,@#MULFSUB
1$: .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 2 ;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$: .WORD 140200,0 ;AC
2$: .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 2 ;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 2 ;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 2 ;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

```

7900 044122 160100 000001
 7901 044126 007547
 7902 044130 007550
 7903 044132 160100 000001
 7904 044136 104002
 7905
 7906
 7907 044140
 7908 044140 104414
 7909 044142 004737 044200
 7910 044146 040277 000000
 7911 044152 060000 000001
 7912 044156 060077 000001
 7913 044162 000014
 7914 044164 000000
 7915 044166 060077 000002
 7916 044172 104002
 7917
 7918 044174 000167 000224
 7919
 7920
 7921
 7922
 7923
 7924
 7925
 7926
 7927
 7928
 7929
 7930
 7931
 7932
 7933
 7934
 7935
 7936
 7937
 7938
 7939
 7940
 7941
 7942
 7943
 7944
 7945
 7946 044200 012601
 7947 044202 012700 000200
 7948 044206 170100
 7949 044210 010100
 7950 044212 172410
 7951 044214 016100 000014
 7952 044220 170100
 7953 044222 012737 044236 001236
 7954 044230 010100
 7955 044232 062700 000004

```

3$: .WORD 160100,1 ;RES
4$: 7547 ;FPS BEFORE EXECUTION.
7550 ;FPS AFTER EXECUTION.
5$: .WORD 160100,1 ;ERROR RES.
6$: ERROR 2 ;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 2 ;ROUND FAILURE. CONSTANT BAD.

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

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

```

:
: JSR PC,@#MULFSUB
: ACARG: .WORD X,X ;AC OPERAND
: FSRCARG: .WORD X,X ;FSRC OPERAND
: RES: .WORD X,X ;EXPECTED RESULT
: FPSB: .WORD X ;FPS BEFORE EXECUTION
: FPSA: .WORD X ;FPS AFTER EXECUTION
: ERRES: .WORD X,X ;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, MULF IS EXECUTED.
 : AFTER THE EXECUTION THE RESULT IS CHECKED AGAINST THE
 : EXPECTED CORRECT RESULT, RES. IF IT IS CORRECT THEN THE FPS
 : IS CHECKED WITH THE EXPECTED CORRECT FPS, FPSA. IF THE FPS WAS
 : INCORRECT THEN IT IS REPORTED. IF THE RESULT WAS INCORRECT IT
 : IS COMPARED WITH ERRES IN AN ATTEMPT TO ANALYSE THE ERROR. IF
 : THE INCORRECT RESULT MATCHED ERRES THEN CONTROL IS PASSED TO
 : THE ERROR CALL AT ERR. IF THE INCORRECT RESULT DID NOT MATCH ERRES
 : THEN THE FAILURE IS REPORTED IN MULFSUB AND CONTROL IS PASSED TO
 : CONT. IF NO ERRORS ARE DETECTED THEN MULFSUB RETURNS CONTROL
 : TO CONT.

```

MULFSUB:
MOV (SP)+,R1 ;GET A POINTER TO THE ARGUMENTS.
MOV #200,R0 ;SET FD MODE.
LDFPS R0
MOV R1,R0 ;LOAD THE AC OPERAND.
LDD (R0),ACO
MOV 14(R1),R0 ;LOAD THE FPS
LDFPS R0
MOV #1$,@#STMP2
MOV R1,R0
ADD #4,R0 ;ESTABLISH A POINTER TO FSRC.
    
```



```

8012
8013
8014
8015
8016
8017
8018
8019
8020
8021
8022
8023 044426 000004
8024
8025
8026 044430
8027 044430 104414
8028 044432 004737 044714
8029 044436 040200 000000 000000 1$:
8030 044444 000000
8031 044446 023777 177777 177777 2$:
8032 044454 177777
8033 044456 023777 177777 177777 3$:
8034 044464 177777
8035 044466 000217 4$:
8036 044470 000200 200
8037 044472 023777 177777 000000 5$:
8038 044500 000000
8039 044502 104002 6$:
8040
8041
8042
8043 044504
8044 044504 104414
8045 044506 004767 000202
8046 044512 065400 000000 000000 1$:
8047 044520 000001
8048 044522 037577 177777 177777 2$:
8049 044530 177776
8050 044532 064777 177777 177777 3$:
8051 044540 177777
8052 044542 000247 4$:
8053 044544 000240 240
8054 044546 065000 000000 000000 5$:
8055 044554 000000
8056 044556 104002 6$:
8057
8058
8059 044560
8060 044560 104414
8061 044562 004737 044714
8062 044566 137577 177777 177777 1$:
8063 044574 177776
8064 044576 165400 000000 000000 2$:
8065 044604 000001
8066 044606 065000 000000 000000 3$:
8067 044614 000000
    
```

```

:*****
:*TEST 47      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.
:*
:*****
    
```

```

TST47: SCOPE
;MULD TEST WITH AC POSITIVE AND FSRC POSITIVE.
FFF1:
LPERR                ;SET UP THE LOOP ON ERROR ADDRESS.
JSR PC,@#MULDSUB
.WORD 40200,0,0,0    ;AC
.WORD 23777,-1,-1,-1 ;FSRC
.WORD 23777,-1,-1,-1 ;RES
217                  ;FPS BEFORE EXECUTION.
200                  ;FPS AFTER EXECUTION.
.WORD 23777,-1,0,0  ;ERROR RES.
ERROR 2              ;BAD CONSTANT USED IN ALGORITHM
                    ;USED 24 INSTEAD OF 56.
    
```

```

;MULD TEST WITH BOTH OPERANDS POSITIVE TRUNCATION TEST.
FFF2:
LPERR                ;SET UP THE LOOP ON ERROR ADDRESS.
JSR PC,MULDSUB
.WORD 65400,0,0,1   ;AC
.WORD 37577,-1,-1,-2 ;FSRC
.WORD 64777,-1,-1,-1 ;RES
247                  ;FPS BEFORE EXECUTION.
240                  ;FPS AFTER EXECUTION.
.WORD 65000,0,0,0   ;ERROR RES.
ERROR 2              ;TRUNCATION ERROR.
    
```

```

;MULD TEST WITH BOTH OPERANDS NEGATIVE IN ROUND MODE.
FFF3:
LPERR                ;SET UP THE LOOP ON ERROR ADDRESS.
JSR PC,@#MULDSUB
.WORD 137577,-1,-1,-2 ;AC
.WORD 165400,0,0,1   ;FSRC
.WORD 65000,0,0,0   ;RES
    
```

```

8068 044616 007717 4$: 7717 ;FPS BEFORE EXECUTION.
8069 044620 007700 7700 ;FPS AFTER EXECUTION.
8070 044622 064777 177777 177777 5$: .WORD 64777,-1,-1,-1 ;ERROR RES.
8071 044630 177777
8072 044632 104002 6$: ERROR 2 ;ROUND ERROR.
8073
8074 ;MULD TEST WITH AC POSITIVE AND FSRC NEGATIVE IN ROUND MODE.
8075 044634 FFF4:
8076 044634 104414 LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
8077 044636 004737 044714 JSR PC,@MULDSUB
8078 044642 017500 000000 000000 1$: .WORD 17500,0,0,0 ;AC
8079 044650 000000
8080 044652 123652 125252 2$: .WORD 123652,125252 ;FSRC
8081 044656 125252 125252 .WORD 125252,125252
8082 044662 103177 177777 177777 3$: .WORD 103177,-1,-1,-1 ;RES
8083 044670 177777
8084 044672 000200 4$: 200 ;FPS BEFORE EXECUTION.
8085 044674 000210 210 ;FPS AFTER EXECUTION.
8086 044676 103200 000000 000000 5$: .WORD 103200,0,0,0 ;ERROR RES.
8087 044704 000000
8088 044706 104002 6$: ERROR 2 ;ROUND ERROR (BAD CONSTANT).
8089
8090 044710 000167 000240 JMP FFFDONE
    
```

:THIS SUBROUTINE, MULDSUB, IS CALLED TO SET UP, EXECUTE
 :AND CHECK THE RESULT OF A MULD 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, MULD 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.

```

8117
8118 044714 012601 MULDSUB: MOV (SP)+,R1 ;GET A POINTER TO THE ARGUMENTS.
8119 044716 012700 000200 MOV #200,R0 ;SET FD MODE.
8120 044722 170100 LDFPS R0
8121
8122 044724 010100 MOV R1,R0 ;SET UP THE ACO OPERAND.
8123 044726 172410 LDD (R0),ACO
    
```

```

8124 044730 016100 000030      MOV    30(R1),R0      ;LOAD THE FPS.
8125 044734 170100      LDFPS R0
8126
8127 044736 012737 044752 001236  MOV    #1$,@#STMP2
8128 044744 010100      MOV    R1,R0          ;ESTABLISH A POINTER TO FSRC.
8129 044746 062700 000010      ADD    #10,R0
8130
8131 044752 171010      1$:    MULDT (R0),ACO    ;EXECUTE THE TEST INSTRUCTION.
8132
8133 044754 170204      STFPS R4              ;GET THE FPS.
8134 044756 012700 000200      MOV    #200,R0        ;SET FD MODE.
8135 044762 170100      LDFPS R0
8136
8137 044764 012700 045144      MOV    #MULDT,R0      ;GET THE RESULT.
8138 044770 174010      STD    ACO,(R0)
8139
8140 044772 010102      MOV    R1,R2          ;SAVE DATA IN CASE OF ERROR.
8141 044774 010237 001240      MOV    R2,@#STMP3
8142 045000 062702 000010      ADD    #10,R2
8143 045004 010237 001242      MOV    R2,@#STMP4
8144 045010 062702 000010      ADD    #10,R2
8145 045014 010237 001244      MOV    R2,@#STMP5
8146 045020 012737 045144 001246  MOV    #MULDT,@#STMP6
8147 045026 010437 001250      MOV    R4,@#STMP7
8148 045032 016137 000032 001252  MOV    32(R1),@#STMP10
8149
8150 045040 010102      MOV    R1,R2          ;CHECK THE RESULT.
8151 045042 062702 000020      ADD    #20,R2
8152 045046 012703 045144      MOV    #MULDT,R3
8153 045052 012705 000004      MOV    #4,R5
8154 045056 022223      2$:    CMP    (R2)+,(R3)+
8155 045060 001006      BNE   10$             ;BRANCH IF RESULT INCORRECT.
8156 045062 077503      SOB   R5,2$
8157
8158 045064 026104 000032      CMP    32(R1),R4      ;IS FPS CORRECT?
8159 045070 001023      BNE   15$             ;BRANCH IF INCORRECT.
8160 045072 000161 000046      JMP    46(R1)         ;RETURN.
8161
8162 045076 010102      10$:   MOV    R1,R2          ;WAS INCORRECT RESULT ANTICIPATED?
8163 045100 062702 000034      ADD    #34,R2
8164 045104 012703 045144      MOV    #MULDT,R3
8165 045110 012705 000004      MOV    #4,R5
8166 045114 022223      11$:   CMP    (R2)+,(R3)+
8167 045116 001005      BNE   12$             ;BRANCH IF NO.
8168 045120 077503      SOB   R5,11$
8169 045122 010102      MOV    R1,R2          ;IF THE INCORRECT RESULT WAS
8170 045124 062702 000044      ADD    #44,R2        ;ANTICIPATED RETURN TO THE
8171
8172 045130 000112      JMP    (R2)           ;ERROR REPORT IN THE CALLING
8173
8174 045132      12$:   ;ROUTINE.
8175 045132 104002      13$:   ;REPORT RESULT INCORRECT.
8176 045134 000161 000046      14$:   ERROR 2
8177
8178 045140      15$:   ;
8179 045140 104002      16$:   JMP    46(R1)
      ERROR 2          ;REPORT FPS INCORRECT.
    
```

8180 045142 000774 BR 14\$
 8181
 8182 045144 000000 000000 000000 MULDT: .WORD 0,0,0,0
 8183 045152 000000

8184
 8185 045154
 8186 045154 104413

FFFDONE:
 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?).

8187
 8188
 8189
 8190
 8191
 8192
 8193
 8194
 8195
 8196
 8197
 8198
 8199
 8200
 8201
 8202
 8203 045156 000004
 8204
 8205
 8206 045160
 8207 045160 104414
 8208 045162 004737 045404
 8209 045166 020200 000000
 8210 045172 020000 000000
 8211 045176 000000 000000
 8212 045202 177777 177777
 8213 045206 000000
 8214 045210 000004
 8215 045212 000012
 8216 045214 177777
 8217 045216 104002
 8218 045220 000401
 8219 045222 104002
 8220 045224
 8221
 8222
 8223 045224
 8224 045224 104414
 8225 045226 004737 045404
 8226 045232 010200 000000
 8227 045236 010000 000000
 8228 045242 000000 000000
 8229 045246 010000 000000
 8230 045252 005013
 8231 045254 005004
 8232 045256 000012
 8233 045260 177777
 8234 045262 104002
 8235

 ;TEST 50 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.
 ;*

TST50: SCOPE

;UNDERFLOW, WITH EXPONENT OF RESULT = -129
 III1:

	LPERR		;SET UP THE LOOP ON ERROR ADDRESS.
	JSR	PC,@#OVUNFNT	
1\$:	.WORD	20200,0	;AC
2\$:	.WORD	20000,0	;FSRC
3\$:	.WORD	0,0	;RES
4\$:	.WORD	-1,-1	;ERROR RES.
5\$:	0		;FPS BEFORE EXECUTION.
	4		;FPS AFTER EXECUTION.
6\$:	12		;FEC
	-1		;FLAG
7\$:	ERROR	2	;ST 331 TO 155 INTO 115 (BUT FIU)
	BR	8\$	
	ERROR	2	
8\$:			

;UNDERFLOW, WITH EXPONENT OF RESULT = -193
 III2:

	LPERR		;SET UP THE LOOP ON ERROR ADDRESS.
	JSR	PC,@#OVUNFNT	
1\$:	.WORD	10200,0	;AC
2\$:	.WORD	10000,0	;FSRC
3\$:	.WORD	0,0	;RES
4\$:	.WORD	10000,0	;ERROR RES.
5\$:	5013		;FPS BEFORE EXECUTION.
	5004		;FPS AFTER EXECUTION.
6\$:	12		;FEC
	-1		;FLAG
7\$:	ERROR	2	;SETTING FIUV OR FIV CAUSES TRAP
			;WITH FIU CLEAR.

```

8236 045264 000401
8237 045266 104002
8238 045270
8239
8240
8241 045270
8242 045270 104414
8243 045272 004737 045404
8244 045276 060200 000000
8245 045302 060000 000000
8246 045306 000000 000000
8247 045312 060000 000000
8248 045316 000000
8249 045320 000006
8250 045322 000010
8251 045324 000000
8252 045326 104002
8253 045330 000401
8254 045332 104002
8255 045334
8256
8257
8258 045334
8259 045334 104414
8260 045336 004737 045404
8261 045342 060200 000000
8262 045346 060200 000000
8263 045352 000000 000000
8264 045356 177777 177777
8265 045362 006011
8266 045364 006006
8267 045366 000010
8268 045370 000000
8269 045372 104002
8270
8271 045374 000401
8272 045376 104002
8273 045400 000167 000410
8274
8275
8276
8277
8278
8279
8280
8281
8282
8283
8284
8285
8286
8287
8288
8289
8290
8291
    
```

```

BR 8$
ERROR 2
8$:
:OVERFLOW, EXPONENT OF RESULT = 128
1113:
LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
JSR PC,@OVUNFNT
1$: .WORD 60200,0 ;AC
2$: .WORD 60000,0 ;FSRC
3$: .WORD 0,0 ;RES
4$: .WORD 60000,0 ;ERROR RES.
5$: 0 ;FPS BEFORE EXECUTION.
6: 6 ;FPS AFTER EXECUTION.
6$: 10 ;FEC
0 ;FLAG
7$: ERROR 2 ;ST 333 TO 136 INTO 116 (BUT FIV).
BR 8$
ERROR 2
8$:
    
```

```

:OVERFLOW, EXPONENT OF RESULT = 130
1114:
LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
JSR PC,@OVUNFNT
1$: .WORD 60200,0 ;AC
2$: .WORD 60200,0 ;FSRC
3$: .WORD 0,0 ;RES
4$: .WORD -1,-1 ;ERROR RES.
5$: 6011 ;FPS BEFORE EXECUTION.
6006 ;FPS AFTER EXECUTION.
6$: 10 ;FEC
0 ;FLAG
7$: ERROR 2 ;SETTING FIUV OR FIU WITH
;FIV CLEAR CAUSES TRAP.
BR 8$
ERROR 2
8$: JMP 111DONE ;GO TO NEXT TEST.
    
```

```

:THIS SUBROUTINE, OVUNFNT, IS USED TO SET UP THE OPERANDS, EXECUTE
:THE MULF INSTRUCTION AND CHECK THE RESULTS OF AN INSTRUCTION WITH
:OPERANDS WHICH SHOULD RESULT IN EITHER OVERFLOW OR UNDERFLOW. A CALL
:TO IT IS MADE THUS:
:
:ACARG: .WORD X,X ;AC OPERAND
:FSRCARG: .WORD X,X ;FSRC OPERAND
:RES: .WORD X,X ;EXPECTED RESULT
:ERRES: .WORD X,X ;ERROR RESULT
:FPSB: .WORD X ;FPS BEFORE EXECUTION
:FPSA: .WORD X ;FPS AFTER EXECUTION
:FEC: .WORD X ;EXPECTED FEC
:FLAG: .WORD X ;0/-1,OVER/UNDER FLOW FLAG
:ERR1: ERROR X ;TRAP ERROR.
:BR CONT
    
```


8292
 8293
 8294
 8295
 8296
 8297
 8298
 8299
 8300
 8301
 8302
 8303
 8304
 8305
 8306
 8307
 8308
 8309
 8310
 8311
 8312
 8313
 8314
 8315 045404 012601
 8316 045406 012700 000200
 8317 045412 170100
 8318
 8319 045414 010100
 8320 045416 172410
 8321
 8322 045420 010102
 8323 045422 010237 001240
 8324 045426 062702 000004
 8325 045432 010237 001242
 8326 045436 062702 000004
 8327 045442 010237 001244
 8328 045446 016137 000022 001252
 8329 045454 012737 046004 001246
 8330
 8331 045462 016100 000020
 8332 045466 170100
 8333 045470 012737 045512 001236
 8334 045476 012737 045676 000244
 8335
 8336 045504 010100
 8337 045506 062700 000004
 8338
 8339 045512 171010
 8340
 8341 045514 170204
 8342 045516 170305
 8343 045520 012700 000200
 8344 045524 170100
 8345 045526 012700 046004
 8346 045532 174010
 8347 045534 010437 001250

```

ERR2:  ERROR  X           ;DATA, RESULT ERROR
CONT:                                ;RETURN ADDRESS

:THE OPERANDS ARE SET UP (USING ACO AS THE ACCUMULATOR). THEN
:THE MULF INSTRUCTION IS EXECUTED. IF NO TRAP OCCURS THEN THE
:RESULT IS CHECKED AGAINST RES. IF THE RESULT IS CORRECT THEN THE FPS IS
:COMPARED WITH FPSA IF THIS TOO IS CORRECT OVUNFNT RETURNS CONTROL
:TO THE CALLING ROUTINE AT CONT. IF THE FPS IS BAD OVUNFNT
:REPORTS THIS FAILURE AND THEN RETURNS TO CONT. IF THE RESULT OF THE
:MULF IS INCORRECT, THE INCORRECT RESULT IS COMPARED WITH THE
:ANTICIPATED FAILING DATA PATTERN, ERRES. IF THE FAILURE IN
:THE RESULT WAS ANTICIPATED CORRECTLY TO BE ERRES THEN OVUNFNT
:WILL TRANSFER CONTROL TO THE ERROR CALL AT ERR2. OTHERWISE THE
:RESULT WAS INCORRECT BUT WAS NOT ANTICIPATED AND OVUNFNT WILL
:REPORT THE FAILURE AFTER WHICH CONTROL WILL BE PASSED TO CONT.
:IF A TRAP OCCURS (IT SHOULD NOT) THEN OVUNFNT WILL READ THE FEC.
:SHOULD THE FEC MATCH THE ANTICIPATED FEC OVUNFNT WILL
:STORE ALL DATA AND TRANSFER CONTROL TO THE ERROR CALL AT ERR1. IF THE
:FEC IS NOT THE SAME AS THE ANTICIPATED FEC OVUNFNT WILL REPORT
:THE ERROR AND RETURN TO CONT. NOTE THAT OVUNFNT USES THE FLAG
:TO TELL WHETHER OR NOT THESE PARTICULAR OPERANDS WILL RESULT IN
:UNDERFLOW (FLAG=-1) OR OVERFLOW (FLAG=0).
    
```

```

OVUNFNT:  MOV      (SP)+,R1      ;GET A POINTER TO THE ARGUMENTS.
          MOV      #200,R0      ;SET FD MODE.
          LDFPS   R0
          MOV      R1,R0        ;LOAD ACO, OPERAND.
          LDD     (R0),ACO
          MOV      R1,R2        ;SAVE THE DATA PATTERNS IN CASE OF
          MOV      R2,@#STMP3   ;ERROR.
          ADD     #4,R2
          MOV      R2,@#STMP4
          ADD     #4,R2
          MOV      R2,@#STMP5
          MOV      22(R1),@#STMP10
          MOV      #OVFNNT,@#STMP6
          MOV      20(R1),R0     ;LOAD THE FPS.
          LDFPS   R0
          MOV      #1$,@#STMP2
          MOV      #25$,@#FPVECT ;SET UP THE FP TRAP VECTOR IN CASE
          ;OF ERROR.
          MOV      R1,R0        ;COMPUTE THE ADDRESS OF FSRC.
          ADD     #4,R0
          1$:    MULF   (R0),ACO  ;TEST INSTRUCTION.
          2$:    STFPS  R4        ;GET FPS.
          STST   R5            ;GET FEC.
          MOV    #200,R0      ;SET FD MODE.
          LDFPS  R0
          MOV    #OVFNNT,R0   ;GET THE RESULT.
          STD   ACO,(R0)
          MOV   R4,@#STMP7
    
```

```

8348 045540 010537 001254      MOV      R5,@#STMP11
8349
8350 045544 012700 046004      MOV      #OVFNTT,R0      ;CHECK THE RESULT.
8351 045550 010102      MOV      R1,R2
8352 045552 062702 000010      ADD      #10,R2
8353 045556 012703 000002      MOV      #2,R3
8354 045562 022022      3$:     CMP      (R0)+,(R2)+
8355 045564 001015      BNE      15$      ;BRANCH IF INCORRECT.
8356 045566 077303      SOB      R3,3$
8357
8358 045570 026104 000022      CMP      22(R1),R4      ;WAS FPS CORRECT?
8359 045574 001002      BNE      10$      ;BRANCH IF FPS IS INCORRECT.
8360
8361 045576 000161 000036      4$:     JMP      36(R1)      ;RETURN, TEST COMPLETED.
8362
8363      ;REPORT INCORRECT FPS.
8364 045602 005761 000026      10$:    TST      26(R1)      ;WAS THE RESULT OVER OR UNDER FLOW?
8365 045606 001002      BNE      12$      ;BRANCH IF UNDERFLOW.
8366
8367      ;REPORT FPS BAD AFTER OVERFLOW.
8368 045610 104002      11$:    ERROR  2
8369 045612 000771      BR       4$
8370
8371 045614      12$:
8372 045614 104002      13$:    ERROR  2      ;REPORT FPS BAD AFTER UNDERFLOW.
8373 045616 000767      BR       4$
8374
8375      ;RESULT INCORRECT.
8376 045620 012700 046004      15$:    MOV      #OVFNTT,R0      ;SEE IF FAILURE IS ANTICIPATED
8377 045624 010102      MOV      R1,R2      ;FAILURE.
8378 045626 062702 000014      ADD      #14,R2
8379 045632 012703 000002      MOV      #2,R3
8380 045636 022022      16$:    CMP      (R0)+,(R2)+
8381 045640 001007      BNE      17$      ;BRANCH IF NOT ANTICIPATED.
8382 045642 077303      SOB      R3,16$
8383
8384 045644 010102      MOV      R1,R2      ;ERROR WAS ANTICIPATED SO RETURN
8385 045646 062702 000034      ADD      #34,R2      ;TO THE ERROR REPORT IN THE CALLING
8386 045652 010237 001236      MOV      R2,@#STMP2      ;ROUTINE.
8387 045656 000112      JMP      (R2)
8388
8389 045660 005761 000026      17$:    TST      26(R1)      ;RESULT WAS NOT ANTICIPATED
8390      ;SO ERROR MUST BE REPORTED HERE.
8391      ;FIRST SEE IF ARGUMENTS SHOULD
8392      ;HAVE RESULTED IN OVERFLOW OR UNDER
8393      ;FLOW BY LOOKING AT THE FLAG.
8394 045664 001002      BNE      19$      ;BRANCH IF UNDERFLOW EXPECTED.
8395
8396      ;REPORT RESULT INCORRECT, EXPECTING
8397 045666 104002      18$:    ERROR  2      ;OVERFLOW.
8398 045670 000742      BR       4$
8399
8400      19$:
8401 045672 104002      20$:    ERROR  2      ;REPORT RESULT INCORRECT, EXPECTING
8402 045674 000740      BR       4$      ;UNDERFLOW.
8403
    
```

```

8404      ;IF AN FP TRAP OCCURS COME HERE.
8405 045676 011602      25$:  MOV    (SP),R2      ;GET ADDRESS OF TRAP.
8406 045700 022702 045514  CMP    #2$,R2      ;WAS THE TRAP DURING THE MULF INSTRUCTION?
8407 045704 001402      BEQ    26$          ;BRANCH IF YES.
8408 045706 000137 062524  JMP    @#FPSPUR    ;OTHERWISE GO REPORT A SPURIOUS
8409                                     ;FP TRAP.
8410 045712 022626      26$:  CMP    (SP)+,(SP)+  ;RESET THE STACK.
8411 045714 010237 001236  MOV    R2,@#$TMP2  ;SAVE DATA FOR ERROR REPORT.
8412 045720 170204      STFPS  R4          ;GET FPS.
8413 045722 170305      STST   R5          ;GET FEC.
8414 045724 012700 000200  MOV    #200,R0     ;SET FD MODE.
8415 045730 170100      LDFPS  R0
8416 045732 012700 046004  MOV    #OVFNIT,R0  ;GET THE RESULT.
8417 045736 174010      STD    ACO,(R0)
8418 045740 010537 001254  MOV    R5,@#$TMP11
8419 045744 020561 000024  CMP    R5,24(R1)
8420 045750 001004      BNE    27$
8421                                     ;WAS THE FEC ANTICIPATED?
8422 045752 010102      MOV    R1,R2      ;ERROR WAS ANTICIPATED SO
8423 045754 062702 000030  ADD    #30,R2     ;RETURN TO THE ERROR REPORT OF THE
8424                                     ;CALLING ROUTINE.
8425 045760 000112      JMP    (R2)
8426
8427 045762 005761 000026  27$:  TST    26(R1)   ;THE ERROR WAS NOT ANTICIPATED SO
8428                                     ;IT MUST BE REPORTED HERE. FIRST SEE IF EXPECTED
8429                                     ;OVERFLOW OR UNDER FLOW.
8430 045766 001003      BNE    29$       ;BRANCH IF EXPECTING UNDERFLOW
8431                                     ;REPORT TRAPPED ON OVERFLOW WITH FIV=0
8432
8433 045770 104002      28$:  ERROR  2
8434 045772 000161 000036  JMP    36(R1)
8435
8436 045776      29$:  ERROR  2
8437 045776 104002      30$:  ERROR  2
8438 046000 000161 000036  JMP    36(R1)
8439                                     ;REPORT TRAPPED ON UNDER FLOW WITH FIU=0
    
```

8440 046004 000000 000000 000000 OVFNTT: .WORD 0,0,0,0
 8441 046012 000000
 8442

IIIDONE:
 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?).

8443 046014
 8444 046014 104413
 8445
 8446
 8447
 8448
 8449
 8450
 8451
 8452

::*****
 :*TEST 51 UNDER/OVER FLOW, USING MULF WITH TRAP DISABLED, TEST
 :*
 :*THIS IS A TEST OF THE OVERFLOW AND UNDERFLOW CONDITIONS THAT CAN
 :*ARRISE USING THE MULF INSTRUCTION WITH TRAPS DISABLED. A SUBROUTINE IS
 :*USED TO SET UP THE OPERANDS, EXECUTE THE MULF INSTRUCTION AND
 :*CHECK THE RESULTS.
 :*
 :*****

8453
 8454
 8455
 8456
 8457
 8458
 8459
 8460
 8461

TST51: SCOPE

8462 046016 000004 .
 8463
 8464

:UNDERFLOW, EXPONENT OF RESULT=-129
 JJJ1:

8465 046020
 8466 046020 104414
 8467 046022 004737 046344
 8468 046026 020200 000000
 8469 046032 127272 000000
 8470 046036 020000 000000 000000
 8471 046044 000000
 8472 046046 000000 000000 000000
 8473 046054 000000
 8474 046056 000000 000000
 8475 046062 127272 000000
 8476 046066 000200
 8477 046070 000204
 8478 046072 000012
 8479 046074 177777
 8480 046076 104002
 8481 046100 000401
 8482 046102 104002
 8483 046104
 8484
 8485

LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
 JSR PC,@#OVUNDNT
 1\$: .WORD 20200,0 ;AC
 .WORD 127272,0
 2\$: .WORD 20000,0,0,0 ;FSRC
 3\$: .WORD 0,0,0,0 ;RES
 4\$: .WORD 0,0 ;ERROR RES.
 .WORD 127272,0
 5\$: 200 ;FPS BEFORE EXECUTION.
 204 ;FPS AFTER EXECUTION.
 6\$: 12 ;FEC
 -1 ;FLAG
 7\$: ERROR 2 ;ST 331 TO 155 INTO 115 (BUT FIU)
 BR 8\$
 8\$: ERROR 2 ;ST 115 (BUT FD)

8486 046104
 8487 046104 104414
 8488 046106 004737 046344
 8489 046112 010200 000000
 8490 046116 123456 000000
 8491 046122 010000 000000 000000
 8492 046130 000000
 8493 046132 000000 000000 000000
 8494 046140 000000
 8495 046142 000000 000000 123456

:UNDERFLOW, EXPONENT OF RESULT = -193
 JJJ2:
 LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
 JSR PC,@#OVUNDNT
 1\$: .WORD 10200,0 ;AC
 .WORD 123456,0
 2\$: .WORD 10000,0,0,0 ;FSRC
 3\$: .WORD 0,0,0,0 ;RES
 4\$: .WORD 0,0,123456,0 ;ERROR RES

```

8496 046150 000000
8497 046152 005213
8498 046154 005204
8499 046156 000012
8500 046160 177777
8501 046162 104002
8502 046164 000401
8503 046166 104002
8504 046170
8505
8506
8507 046170
8508 046170 104414
8509 046172 004737 046344
8510 046176 060200 000000
8511 046202 065432 000000
8512 046206 060000 000000 000000
8513 046214 000000
8514 046216 000000 000000 000000
8515 046224 000000
8516 046226 000000 000000 065432
8517 046234 000000
8518 046236 000200
8519 046240 000206
8520 046242 000010
8521 046244 000000
8522 046246 104002
8523 046250 000401
8524 046252 104002
8525 046254
8526
8527
8528
8529 046254
8530 046254 104414
8531 046256 004737 046344
8532 046262 060200 000000
8533 046266 125252 000000
8534 046272 060200 000000 000000
8535 046300 000000
8536 046302 000000 000000 000000
8537 046310 000000
8538 046312 000000 000000 125252
8539 046320 000000
8540 046322 006211
8541 046324 006206
8542 046326 000010
8543 046330 000000
8544 046332 104002
8545 046334 000401
8546 046336 104002
8547 046340 000137 046754
8548
8549
8550
8551
    
```

```

5$: 5213 :FPS BEFORE EXECUTION.
      5204 :FPS AFTER EXECUTION.
6$: 12 :FEC
      -1 :FLAG
7$: ERROR 2 :SETTING FIUV OR FIV BAD.
      BR 8$
      ERROR 2 :ST 115 (BUT FD)
8$:
;OVERFLOW, EXPONENT OF RESULT = 128
JJJ3:
      LPERR :SET UP THE LOOP ON ERROR ADDRESS.
      JSR PC,@#OVUNDNT
1$: .WORD 60200,0 :AC
      .WORD 65432,0
2$: .WORD 60000,0,0,0 :FSRC
3$: .WORD 0,0,0,0 :RES
4$: .WORD 0,0,65432,0 :ERROR RES.
5$: 200 :FPS BEFORE EXECUTION.
      206 :FPS AFTER EXECUTION.
6$: 10 :FEC
      0 :FLAG
7$: ERROR 2 :ST 333 TO 136 INTO 116 (BUT FIV)
      BR 8$
      ERROR 2 :ST 116 (BUT FD)
8$:
;OVERFLOW, EXPONENT OF RESULT = 130
JJJ4:
      LPERR :SET UP THE LOOP ON ERROR ADDRESS.
      JSR PC,@#OVUNDNT
1$: .WORD 60200,0 :AC
      .WORD 125252,0
2$: .WORD 60200,0,0,0 :FSRC
3$: .WORD 0,0,0,0 :RES
4$: .WORD 0,0,125252,0 :ERROR RES.
5$: 6211 :FPS BEFORE EXECUTION.
      6206 :FPS AFTER EXECUTION.
6$: 10 :FEC
      0 :FLAG
7$: ERROR 2 :SETTING FIUV OR FIV BAD.
      BR 8$
      ERROR 2 :ST 116 (BUT FD)
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
    
```

8552
8553
8554
8555
8556
8557
8558
8559
8560
8561
8562
8563
8564
8565
8566
8567
8568
8569
8570
8571
8572
8573
8574
8575
8576
8577
8578
8579
8580
8581
8582
8583
8584
8585
8586
8587
8588
8589
8590
8591
8592
8593
8594
8595
8596
8597
8598
8599
8600
8601
8602
8603
8604
8605
8606
8607

: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
 :RESULT WAS INCORRECT BUT WAS NOT ANTICIPATED AND OVUNDNT WILL
 :REPORT THE FAILURE AFTER WHICH CONTROL WILL BE PASSED TO CONT.
 :IF A TRAP OCCURS (IT SHOULD NOT) THEN OVUNDNT WILL READ THE FEC.
 :SHOULD THE FEC MATCH THE ANTICIPATED FEC OVUNDNT WILL
 :STORE ALL DATA AND TRANSFER CONTROL TO THE ERROR CALL AT ERR1. IF THE
 :FEC IS NOT THE SAME AS THE ANTICIPATED FEC OVUNDNT WILL REPORT
 :THE ERROR AND RETURN TO CONT. NOTE THAT OVUNDNT USES THE FLAG
 :TO TELL WHETHER OR NOT THESE PARTICULAR OPERANDS WILL RESULT IN
 :UNDERFLOW (FLAG=-1) OR OVERFLOW (FLAG=0).

```

OVUNDNT:  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 THE DATA PATTERNS IN CASE OF
          MOV      R2,@#STMP3 ;ERROR.
          ADD     #10,R2
          MOV      R2,@#STMP4
          ADD     #10,R2
          MOV      R2,@#STMP5
          MOV      42(R1),@#STMP10
          MOV      #OVDNTT,@#STMP6
          MOV      40(R1),R0   ;LOAD THE FPS.
          LDFPS   R0
          MOV      #1$,@#STMP2
          MOV      #25$,@#FPVECT ;SET UP THE FP TRAP VECTOR IN CASE
          ;OF ERROR.
  
```

```

046344 012601
046346 012700 000200
046352 170100
046354 010100
046356 172410
046360 010102
046362 010237 001240
046366 062702 000010
046372 010237 001242
046376 062702 000010
046402 010237 001244
046406 016137 000042 001252
046414 012737 046744 001246
046422 016100 000040
046426 170100
046430 012737 046452 001236
046436 012737 046636 000244
  
```

8608	046444	010100		MOV	R1,R0		;COMPUTE THE ADDRESS OF FSRC.
8609	046446	062700	000010	ADD	#10,R0		
8610							
8611	046452	171010		1\$:	MULD	(R0),ACO	;TEST INSTRUCTION.
8612							
8613	046454	170204		2\$:	STFPS	R4	;GET FPS.
8614	046456	170305			STST	R5	;GET FEC.
8615	046460	012700	000200		MOV	#200,R0	;SET FD MODE.
8616	046464	170100			LDFPS	R0	
8617	046466	012700	046744		MOV	#OVDNTT,R0	;GET THE RESULT.
8618	046472	174010			STD	ACO,(R0)	
8619	046474	010437	001250		MOV	R4,@#STMP7	
8620	046500	010537	001254		MOV	R5,@#STMP11	
8621							
8622	046504	012700	046744		MOV	#OVDNTT,R0	;CHECK THE RESULT.
8623	046510	010102			MOV	R1,R2	
8624	046512	062702	000020		ADD	#20,R2	
8625	046516	012703	000004		MOV	#4,R3	
8626	046522	022022		3\$:	CMP	(R0)+,(R2)+	
8627	046524	001015			BNE	15\$;BRANCH IF INCORRECT.
8628	046526	077303			SOB	R3,3\$	
8629							
8630	046530	026104	000042		CMP	42(R1),R4	;WAS FPS CORRECT?
8631	046534	001002			BNE	10\$;BRANCH IF FPS IS INCORRECT.
8632							
8633	046536	000161	000056	4\$:	JMP	56(R1)	;RETURN, TEST COMPLETED.
8634							
8635					:REPORT	INCORRECT FPS.	
8636	046542	005761	000046	10\$:	TST	46(R1)	;WAS THE RESULT OVER OR UNDER FLOW?
8637	046546	001002			BNE	12\$;BRANCH IF UNDERFLOW.
8638							
8639							;REPORT FPS BAD AFTER OVERFLOW.
8640	046550	104002		11\$:	ERROR	2	
8641	046552	000771			BR	4\$	
8642							
8643	046554			12\$:			;REPORT FPS BAD AFTER UNDERFLOW.
8644	046554	104002		13\$:	ERROR	2	
8645	046556	000767			BR	4\$	
8646							
8647					:RESULT	INCORRECT.	
8648	046560	012700	046744	15\$:	MOV	#OVDNTT,R0	;SEE IF FAILURE IS ANTICIPATED
8649	046564	010102			MOV	R1,R2	;FAILURE.
8650	046566	062702	000030		ADD	#30,R2	
8651	046572	012703	000004		MOV	#4,R3	
8652	046576	022022		16\$:	CMP	(R0)+,(R2)+	
8653	046600	001007			BNE	17\$;BRANCH IF NOT ANTICIPATED.
8654	046602	077303			SOB	R3,16\$	
8655							
8656	046604	010102			MOV	R1,R2	;ERROR WAS ANTICIPATED SO RETURN
8657	046606	062702	000054		ADD	#54,R2	;TO THE ERROR REPORT IN THE CALLING
8658	046612	010237	001236		MOV	R2,@#STMP2	;ROUTINE.
8659	046616	000112			JMP	(R2)	
8660							
8661	046620	005761	000046	17\$:	TST	46(R1)	;RESULT WAS NOT ANTICIPATED
8662							;SO ERROR MUST BE REPORTED HERE.
8663							;FIRST SEE IF ARGUMENTS SHOULD

```

8664                                     ;HAVE RESULTED IN OVERFLOW OR UNDER
8665                                     ;FLOW BY LOOKING AT THE FLAG.
8666 046624 001002                       BNE      19$      ;BRANCH IF UNDERFLOW EXPECTED.
8667                                     ;REPORT RESULT INCORRECT, EXPECTING
8668                                     ;OVERFLOW.
8669 046626 104002                       18$:  ERROR  2
8670 046630 000742                       BR      4$
8671                                     ;REPORT RESULT INCORRECT, EXPECTING
8672 046632                               19$:
8673 046632 104002                       20$:  ERROR  2
8674 046634 000740                       BR      4$
8675
8676                                     ;IF AN FP TRAP OCCURS COME HERE.
8677 046636 011602                               25$:  MOV      (SP),R2      ;GET ADDRESS OF TRAP.
8678 046640 022702 046454                   CMP      #2$,R2        ;WAS THE TRAP DURING THE MULF INSTRUCTION?
8679 046644 001402                               BEQ      26$          ;BRANCH IF YES.
8680 046646 000137 062524                   JMP      @#FPSPUR     ;OTHERWISE GO REPORT A SPURIOUS
8681                                     ;FP TRAP.
8682 046652 022626                               26$:  CMP      (SP)+,(SP)+ ;RESET THE STACK.
8683 046654 010237 001236                   MOV      R2,@#STMP2   ;SAVE DATA FOR ERROR REPORT.
8684 046660 170204                               STFPS   R4            ;GET FPS.
8685 046662 170305                               STST    R5            ;GET FEC.
8686 046664 012700 000200                   MOV      #200,R0      ;SET FD MODE.
8687 046670 170100                               LDFPS   R0
8688 046672 012700 046744                   MOV      #OVDNTT,R0   ;GET THE RESULT.
8689 046676 174010                               STD     AC0,(R0)
8690 046700 010537 001254                   MOV      R5,@#STMP11
8691 046704 020561 000044                   CMP      R5,44(R1)
8692 046710 001004                               BNE     27$
8693                                     ;WAS THE FEC ANTICIPATED?
8694 046712 010102                               MOV      R1,R2        ;ERROR WAS ANTICIPATED SO
8695 046714 062702 000050                   ADD     #50,R2        ;RETURN TO THE ERROR REPORT OF THE
8696                                     ;CALLING ROUTINE.
8697 046720 000112                               JMP     (R2)
8698
8699 046722 005761 000026                               27$:  TST     26(R1)
8700                                     ;THE ERROR WAS NOT ANTICIPATED SO
8701                                     ;IT MUST BE REPORTED HERE. FIRST SEE IF EXPECTED
8702 046726 001003                               BNE     29$          ;OVERFLOW OR UNDER FLOW.
8703                                     ;BRANCH IF EXPECTING UNDERFLOW
8704                                     ;REPORT TRAPPED ON OVERFLOW WITH FIV=0
8705 046730 104002                               28$:  ERROR  2
8706 046732 000161 000056                   JMP     56(R1)
8707
8708 046736                               29$:
8709 046736 104002                               30$:  ERROR  2
8710 046740 000161 000056                   JMP     56(R1)
8711
8712 046744 000000 000000 000000 OVDNTT: .WORD 0,0,0,0
8713 046752 000000
8714
8715                                     JJJDONE:
8716 046754 104413                               RSETUP
8717
8718                                     ;GO INITIALIZE THE FPS AND STACK; AND
8719                                     ;SEE IF THE USER HAS EXPRESSED
8719                                     ;THE DESIRE TO CHANGE THE SOFTWARE
8719                                     ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
    
```


;THE USER TYPED CONTROL G?).

8720
8721
8722
8723
8724
8725
8726
8727
8728
8729
8730
8731
8732
8733
8734
8735
8736
8737 046756 000004
8738
8739
8740 046760
8741 046760 104414
8742 046762 004737 047204
8743 046766 020123 045676
8744 046772 020200 000000
8745 046776 000123 045676
8746 047002 177777 177777
8747 047006 002000
8748 047010 102004
8749 047012 000012
8750 047014 177777
8751 047016 104002
8752
8753 047020 000401
8754 047022 104002
8755 047024
8756
8757
8758 047024
8759 047024 104414
8760 047026 004737 047204
8761 047032 010127 127272
8762 047036 010200 000000
8763 047042 060127 127272
8764 047046 177777 177777
8765 047052 007017
8766 047054 107000
8767 047056 000012
8768 047060 177777
8769 047062 104002
8770 047064 000401
8771 047066 104002
8772 047070
8773
8774
8775 047070

:TEST 52 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.
:*

TST52: SCOPE

;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 2 ;ST 331 (BUT FIU) NO TRAP.
BR 8\$
ERROR 2
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 2 ;ST 331 (BUT FIU) NO TRAP.
BR 8\$
ERROR 2
8\$:

;OVERFLOW, EXPONENT OF THE RESULT = 128

KKK4:

```

8776 047070 104414
8777 047072 004737 047204
8778 047076 060252 125252
8779 047102 060000 000000
8780 047106 000052 125252
8781 047112 177777 177777
8782 047116 001000
8783 047120 101006
8784 047122 000010
8785 047124 000000
8786 047126 104002
8787 047130 000401
8788 047132 104002
3789 047134
8790
8791

```

```

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.
5$: 1000 ;FPS BEFORE EXECUTION.
101006 ;FPS AFTER EXECUTION.
6$: 10 ;FEC
0 ;FLAG
7$: ERROR 2 ;ST 333 (BUT FIV) NO TRAP
BR 8$
ERROR 2
8$:

```

```

;OVERFLOW, EXPONENT OF RESULT = 130
KKK5:

```

```

8792 047134 104414
8793 047134 004737 047204
8794 047136 060345 067654
8795 047142 060200 000000
8796 047146 060200 000000
8797 047152 000345 067654
8798 047156 177777 177777
8799 047162 007015
8800 047164 107002
8801 047166 000010
8802 047170 000000
8803 047172 104002
8804 047174 000401
8805 047176 104002
8806 047200 000167 000412
8807
8808
8809
8810
8811
8812
8813
8814
8815
8816
8817
8818
8819
8820
8821
8822
8823
8824
8825
8826
8827
8828
8829
8830
8831

```

```

LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
JSR PC,@#OVUNFT
1$: .WORD 60345,67654 ;AC
2$: .WORD 60200,0 ;FSRC
3$: .WORD 345,67654 ;RES
4$: .WORD -1,-1 ;ERROR RES.
5$: 7015 ;FPS BEFORE EXECUTION.
107002 ;FPS AFTER EXECUTION.
6$: 10 ;FEC
0 ;FLAG
7$: ERROR 2 ;ST 133 (BUT FIV) NO TRAP
BR 8$
ERROR 2
8$: JMP KKKDONE

```

```

;THIS SUBROUTINE, OVUNFT, IS USED TO SET UP THE OPERANDS, EXECUTE
;THE MULF INSTRUCTION AND CHECK THE RESULTS OF AN INSTRUCTION WITH
;OPERANDS WHICH SHOULD RESULT IN EITHER OVERFLOW OR UNDERFLOW. A CALL
;TO IT IS MADE THUS:

```

```

:
: ACARG: .WORD X,X ;AC OPERAND
: FSRCARG: .WORD X,X ;FSRC OPERAND
: RES: .WORD X,X ;EXPECTED RESULT
: ERRES: .WORD X,X ;ERROR RESULT
: FPSB: .WORD X ;FPS BEFORE EXECUTION
: FPSA: .WORD X ;FPS AFTER EXECUTION
: FEC: .WORD X ;EXPECTED FEC
: FLAG: .WORD X ;0/-1,OVER/UNDER FLOW FLAG
: ERR1: ERROR X ;TRAP ERROR.
: BR CONT
: ERR2: ERROR X ;DATA, RESULT ERROR
: CONT: ;RETURN ADDRESS

```

```

;THE OPERANDS ARE SET UP (USING ACO AS THE ACCUMULATOR). THEN
;THE MULF INSTRUCTION IS EXECUTED. IF THE TRAP OCCURS THEN THE
;RESULT IS CHECKED AGAINST RES. IF THE RESULT IS CORRECT THEN THE FPS IS
;COMPARED WITH FPSA IF THIS TOO IS CORRECT OVUNFT RETURNS CONTROL
;TO THE CALLING ROUTINE AT CONT. IF THE FPS IS BAD OVUNFT
;REPORTS THIS FAILURE AND THEN RETURNS TO CONT. THE FEC IS TREATED

```

```

8832      ;IN THE SAME WAY. IF THE RESULT OF THE
8833      ;MULF IS INCORRECT, THE INCORRECT RESULT IS COMPARED WITH THE
8834      ;ANTICIPATED FAILING DATA PATTERN, ERRES. IF THE FAILURE IN
8835      ;THE RESULT WAS ANTICIPATED CORRECTLY TO BE ERRES THEN OVUNFT
8836      ;WILL TRANSFER CONTROL TO THE ERROR CALL AT ERR2. OTHERWISE THE
8837      ;RESULT WAS INCORRECT BUT WAS NOT ANTICIPATED AND OVUNFT WILL
8838      ;REPORT THE FAILURE AFTER WHICH CONTROL WILL BE PASSED TO CONT.
8839      ;IF NO TRAP OCCURS CONTROL IS PASSED TO ERR1.
8840      ;NOTE THAT OVUNFT USES THE FLAG
8841      ;TO TELL WHETHER OR NOT THESE PARTICULAR OPERANDS WILL RESULT IN
8842      ;UNDERFLOW (FLAG=-1) OR OVERFLOW (FLAG=0).
8843
8844      047204 012601      OVUNFT: MOV      (SP)+,R1      ;GET A POINTER TO THE ARGUMENTS.
8845      047206 012700 000200      MOV      #200,R0      ;SET FD MODE.
8846      047212 170100      LDFPS   R0
8847
8848      047214 010100      MOV      R1,R0      ;LOAD ACO, OPERAND.
8849      047216 172410      LDD     (R0),ACO
8850
8851      047220 010102      MOV      R1,R2      ;SAVE THE DATA PATTERNS IN CASE OF
8852      047222 010237 001240      MOV      R2,@#STMP3  ;ERROR.
8853      047226 062702 000004      ADD     #4,R2
8854      047232 010237 001242      MOV      R2,@#STMP4
8855      047236 062702 000004      ADD     #4,R2
8856      047242 010237 001244      MOV      R2,@#STMP5
8857      047246 016137 000022 001252      MOV      22(R1),@#STMP10
8858      047254 012737 047606 001246      MOV      #OVFTT,@#STMP6
8859
8860      047262 016100 000020      MOV      20(R1),R0  ;LOAD THE FPS.
8861      047266 170100      LDFPS   R0
8862      047270 012737 047312 001236      MOV      #1$,@#STMP2
8863
8864      047276 012737 047322 000244      MOV      #50$,@#FPVECT ;SET UP THE FP TRAP VECTOR IN CASE
8865      ;OF ERROR.
8866      047304 010100      MOV      R1,R0      ;COMPUTE THE ADDRESS OF FSRC.
8867      047306 062700 000004      ADD     #4,R0
8868
8869      047312 171010      1$:    MULF   (R0),ACO  ;TEST INSTRUCTION. SHOULD CAUSE TRAP.
8870      047314 170000      2$:    CFCC
8871
8872      047316 000137 047546      JMP     @#25$      ;FAILURE, NO TRAP.
8873
8874      047322 011602      50$:   MOV     (SP),R2      ;TRAP TO HERE AND SEE IF THE PC OF THE
8875      047324 020227 047314      CMP     R2,#2$      ;TRAP WAS THAT OF THE MULF INSTRUCTION.
8876      047330 001402      BEQ    51$          ;BRANCH IF YES.
8877      047332 000137 062524      JMP     @#FPSPUR   ;OTHERWISE REPORT SPURIOUS FP ERROR.
8878
8879      047336 022626      51$:   CMP     (SP)+,(SP)+ ;RESET THE STACK
8880      047340 170204      STFPS  R4          ;GET FPS.
8881      047342 170305      STST   R5          ;GET FEC.
8882      047344 012700 000200      MOV     #200,R0     ;SET FD MODE.
8883      047350 170100      LDFPS  R0
8884      047352 012700 047606      MOV     #OVFTT,R0   ;GET THE RESULT.
8885      047356 174010      STD    ACO,(R0)
8886      047360 010437 001250      MOV     R4,@#STMP7
8887      047364 010537 001254      MOV     R5,@#STMP11
    
```

```

8888
8889 047370 012700 047606      MOV    #OVFTT,R0      ;CHECK THE RESULT.
8890 047374 010102      MOV    R1,R2
8891 047376 062702 000010      ADD    #10,R2
8892 047402 012703 000002      MOV    #2,R3
8893 047406 022022      3$:  CMP    (R0)+,(R2)+
8894 047410 001027      BNE   15$             ;BRANCH IF INCORRECT.
8895 047412 077303      SOB   R3,3$
8896
8897 047414 026104 000022      CMP    22(R1),R4     ;WAS FPS CORRECT?
8898 047420 001014      BNE   10$             ;BRANCH IF FPS IS INCORRECT.
8899
8900 047422 026105 000024      CMP    24(R1),R5     ;IS FEC CORRECT?
8901 047426 001002      BNE   5$              ;IF INCORRECT BRANCH.
8902 047430 000161 000036      4$:  JMP    36(R1)        ;RETURN, TEST COMPLETED.
8903
8904      :REPORT INCORRECT FEC.
8905 047434 005761 000026      5$:  TST    26(R1)
8906 047440 001002      BNE   7$              ;WAS THE RESULT OVERFLOW OR UNDERFLOW?
8907      :BRANCH IF UNDERFLOW.
8908      :REPORT BAD FEC ON EXPECTED OVERFLOW.
8909 047442 104002      6$:  ERROR 2
8910 047444 000771      BR    4$
8911
8912 047446      7$:
8913 047446 104002      8$:  ERROR 2
8914 047450 000767      BR    4$
8915      :REPORT BAD FEC ON EXPECTED UNDERFLOW.
8916      :REPORT INCORRECT FPS.
8917 047452 005761 000026      10$: TST    26(R1)
8918 047456 001002      BNE   12$             ;WAS THE RESULT OVER OR UNDER FLOW?
8919      :BRANCH IF UNDERFLOW.
8920      :REPORT FPS BAD AFTER OVERFLOW.
8921 047460 104002      11$: ERROR 2
8922 047462 000762      BR    4$
8923
8924 047464      12$:
8925 047464 104002      13$: ERROR 2
8926 047466 000760      BR    4$
8927      :REPORT FPS BAD AFTER UNDERFLOW.
8928      :RESULT INCORRECT.
8929 047470 012700 047606      15$: MOV    #OVFTT,R0
8930 047474 010102      MOV    R1,R2
8931 047476 062702 000014      ADD    #14,R2
8932 047502 012703 000002      MOV    #2,R3
8933 047506 022022      16$: CMP    (R0)+,(R2)+
8934 047510 001007      BNE   17$             ;BRANCH IF NOT ANTICIPATED.
8935 047512 077303      SOB   R3,16$
8936
8937 047514 010102      MOV    R1,R2
8938 047516 062702 000034      ADD    #34,R2
8939 047522 010237 001236      MOV    R2,@#STMP2
8940 047526 000112      JMP    (R2)
8941
8942 047530 005761 000026      17$: TST    26(R1)
8943      :RESULT WAS NOT ANTICIPATED
      :SO ERROR MUST BE REPORTED HERE.
    
```

```

8944                                     ;FIRST SEE IF ARGUMENTS SHOULD
8945                                     ;HAVE RESULTED IN OVERFLOW OR UNDER
8946                                     ;FLOW BY LOOKING AT THE FLAG.
8947 047534 001002                       BNE      19$      ;BRANCH IF UNDERFLOW EXPECTED.
8948
8949                                     ;REPORT RESULT INCORRECT, EXPECTING
8950 047536 104002 18$: ERROR 2          ;OVERFLOW.
8951 047540 000733 BR      4$
8952
8953 047542 19$: ERROR 2                  ;REPORT RESULT INCORRECT, EXPECTING
8954 047542 104002 20$: ERROR 2          ;UNDERFLOW.
8955 047544 000731 BR      4$
8956
8957                                     ;IF NO FP TRAP OCCURS COME HERE.
8958 047546 170204 25$: STFPS R4          ;GET FPS.
8959 047550 170305 STST R5              ;GET FEC.
8960 047552 012700 000200 MOV #200,R0 ;SET FD MODE.
8961 047556 170100 LDFPS R0
8962 047560 012700 047606 MOV #OVFTT,R0 ;GET THE RESULT.
8963 047564 174010 STD ACO,(R0)
8964 047566 010437 001250 MOV R4,@#STMP7
8965 047572 010537 001254 MOV R5,@#STMP11
8966 047576 010102 MOV R1,R2          ;ERROR WAS ANTICIPATED SO
8967 047600 062702 000030 ADD #30,R2      ;RETURN TO THE ERROR REPORT OF THE
8968                                     ;CALLING ROUTINE.
8969 047604 000112 JMP (R2)
8970
8971 047606 000000 000000 000000 OVFTT: .WORD 0,0,0,0
8972 047614 000000
8973
8974 047616 KKKDONE:
8975 047616 104413 RSETUP          ;GO INITIALIZE THE FPS AND STACK; AND
8976                                     ;SEE IF THE USER HAS EXPRESSED
8977                                     ;THE DESIRE TO CHANGE THE SOFTWARE
8978                                     ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
8979                                     ;THE USER TYPED CONTROL G?).
8980
8981
8982
8983
8984
8985 ::*****
8986 ;*TEST 53 UNDER/OVER FLOW, USING MULD WITH TRAPS ENABLED, TEST
8987 ;*
8988 ;*THIS IS A TEST OF THE OVER FLOW AND UNDER FLOW CONDITIONS USING THE
8989 ;*MULD INSTRUCTION WITH TRAPS ENABLED. A SUBROUTINE IS USED TO SET UP
8990 ;*THE OPERANDS, EXECUTE THE MULD INSTRUCTION AND CHECK THE RESULTS.
8991 ;*
8992 ;******
8992 047620 000004 TST53: SCOPE
8993
8994 ;UNDERFLOW, EXPONENT OF RESULT = -129
8995 LLL1:
8996 047622 LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
8997 047624 004737 050146 JSR PC,@#OVUNDT
8998 047630 020052 125252 .WORD 20052,125252 ;AC
8999 047634 125252 125252 .WORD 125252,125252
    
```

```

9000 047640 020300 000000 000000 2$: .WORD 20300,0,0,0 ;FSRC
9001 047646 000000
9002 047650 000177 177777 177777 3$: .WORD 177,-1,-1,-1 ;RES
9003 047656 177777
9004 047660 000177 177777 4$: .WORD 177,-1 ;ERROR RES.
9005 047664 125252 125252 .WORD 125252,125252
9006 047670 002200 5$: 2200 ;FPS BEFORE EXECUTION.
9007 047672 102204 102204 ;FPS AFTER EXECUTION.
9008 047674 000012 6$: 12 ;FEC
9009 047676 177777 -1 ;FLAG
9010 047700 104002 7$: ERROR 2 ;ST 331 (BUT FIU) NO TRAP.
9011 047702 000401 BR 8$
9012 047704 104002 ERROR 2 ;ST 155 (BUT FD)
9013 047706
9014
9015 ;UNDERFLOW, EXPONENT OF THE RESULT = -193
9016 047706 LLL2: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
9017 047706 104414 JSR PC,@#OVUNDT
9018 047710 004737 050146 .WORD 10327,127272 ;AC
9019 047714 010327 127272 1$: .WORD 36363,45454
9020 047720 036363 045454 .WORD 10000,0,0,0 ;FSRC
9021 047724 010000 000000 000000 2$: .WORD 60127,127272 ;RES
9022 047732 000000 3$: .WORD 36363,45454
9023 047734 060127 127272 .WORD -1,-1,-1,-1 ;ERROR RES.
9024 047740 036363 045454 4$: .WORD 7217 ;FPS BEFORE EXECUTION.
9025 047744 177777 177777 177777 5$: 107200 ;FPS AFTER EXECUTION.
9026 047752 177777 6$: 12 ;FEC
9027 047754 007217 -1 ;FLAG
9028 047756 107200 7$: ERROR 2 ;ST 137 (BUT FIU) NO TRAP.
9029 047760 000012 BR 8$
9030 047762 177777 ERROR 2
9031 047764 104002
9032 047766 000401
9033 047770 104002
9034 047772
9035
9036 ;OVERFLOW, EXPONENT OF THE RESULT = 128
9037 047772 LLL3: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
9038 047772 104414 JSR PC,@#OVUNDT
9039 047774 004737 050146 .WORD 60252,125252 ;AC
9040 050000 060252 125252 1$: .WORD 125252,125252 ;FSRC
9041 050004 125252 125252 .WORD 160100,0,0,0 ;FSRC
9042 050010 160100 000000 000000 2$: .WORD 100177,-1,-1,-1 ;RES
9043 050016 000000 3$: .WORD 100177,-1 ;ERROR RES.
9044 050020 100177 177777 177777 4$: .WORD 125252,125252
9045 050026 177777 5$: 1200 ;FPS BEFORE EXECUTION.
9046 050030 100177 177777 101216 ;FPS AFTER EXECUTION.
9047 050034 125252 125252 6$: 10 ;FEC
9048 050040 001200 0 ;FLAG
9049 050042 101216 7$: ERROR 2 ;ST 333 (BUT FIV) NO TRAP.
9050 050044 000010 BR 8$
9051 050046 000000 ERROR 2 ;ST 700 (BUT FD).
9052 050050 104002
9053 050052 000401
9054 050054 104002
9055 050056
    
```

```

9056
9057 ;OVERFLOW, EXPONENT OF THE RESULT = 130
9058 LLL4: LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
9059 050056 104414 JSR PC,@#OVUNDT
9060 050060 004737 050146 1$: .WORD 60345,67654 ;AC
9061 050064 060345 067654 .WORD 56765,45676
9062 050070 056765 045676 .WORD 60200,0,0,0 ;FSRC
9063 050074 060200 000000 000000 2$: .WORD
9064 050102 000000 3$: .WORD 345,67654 ;RES
9065 050104 000345 067654 .WORD 56765,45676
9066 050110 056765 045676 4$: .WORD -1,-1,-1,-1 ;ERROR RES.
9067 050114 177777 177777 177777 5$: 7215 ;FPS BEFORE EXECUTION.
9068 050122 177777 6$: 107202 ;FPS AFTER EXECUTION.
9069 050124 007215 7$: 10 ;FEC
9070 050126 107202 8$: 0 ;FLAG
9071 050130 000010 9$: ERROR 2 ;ST 133 (BUT FIV) NO TRAP
9072 050132 000000 BR 8$
9073 050134 104002 ERROR 2
9074 050136 000401 BR 8$
9075 050140 104002 ERROR 2
9076 050142 000137 050560 JMP @#LLLDONE
    
```

: THIS SUBROUTINE, OVUNDT, IS USED TO SET UP THE OPERANDS, EXECUTE THE MULD INSTRUCTION AND CHECK THE RESULTS OF AN INSTRUCTION WITH OPERANDS WHICH SHOULD RESULT IN EITHER OVERFLOW OR UNDERFLOW. A CALL TO IT IS MADE THUS:

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 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

9077
9078
9079
9080
9081
9082
9083
9084
9085
9086
9087
9088
9089
9090
9091
9092
9093
9094
9095
9096
9097
9098
9099
9100
9101
9102
9103
9104
9105
9106
9107
9108
9109
9110
9111

:UNDERFLOW (FLAG=-1) OR OVERFLOW (FLAG=0).

9112							
9113							
9114	050146	012601					
9115	050150	012700	000200				
9116	050154	170100					
9117							
9118	050156	010100					
9119	050160	172410					
9120							
9121	050162	010102					
9122	050164	010237	001240				
9123	050170	062702	000010				
9124	050174	010237	001242				
9125	050200	062702	000010				
9126	050204	010237	001244				
9127	050210	016137	000042	001252			
9128	050216	012737	050550	001246			
9129							
9130	050224	016100	000040				
9131	050230	170100					
9132	050232	012737	050254	001236			
9133	050240	012737	050264	000244			
9134							
9135	050246	010100					
9136	050250	062700	000010				
9137							
9138	050254	171010					
9139	050256	170000					
9140							
9141	050260	000137	050510				
9142							
9143	050264	011602					
9144	050266	020227	050256				
9145	050272	001402					
9146	050274	000137	062524				
9147							
9148	050300	022626					
9149	050302	170204					
9150	050304	170305					
9151	050306	012700	000200				
9152	050312	170100					
9153	050314	012700	050550				
9154	050320	174010					
9155	050322	010437	001250				
9156	050326	010537	001254				
9157							
9158	050332	012700	050550				
9159	050336	010102					
9160	050340	062702	000020				
9161	050344	012703	000004				
9162	050350	022022					
9163	050352	001027					
9164	050354	077303					
9165							
9166	050356	026104	000042				
9167	050362	001014					

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 THE DATA PATTERNS IN CASE OF
 MOV R2,@#STMP3 ;ERROR.
 ADD #10,R2

MOV R2,@#STMP4
 ADD #10,R2
 MOV R2,@#STMP5
 MOV 42(R1),@#STMP10
 MOV #OVDTT,@#STMP6

MOV 40(R1),R0 ;LOAD THE FPS.
 LDFPS R0

MOV #1\$,@#STMP2 ;SET UP THE FP TRAP VECTOR IN CASE
 MOV #50\$,@#FPVECT ;OF ERROR.
 ;COMPUTE THE ADDRESS OF FSRC.

MOV R1,R0
 ADD #10,R0
 1\$: MULD (R0),ACO ;TEST INSTRUCTION. SHOULD CAUSE TRAP.
 2\$: CFCC

JMP @#25\$;FAILURE, NO TRAP.

50\$: MOV (SP),R2 ;TRAP TO HERE AND SEE IF THE PC OF THE
 CMP R2,#2\$;TRAP WAS THAT OF THE MULF INSTRUCTION.
 BEQ 51\$;BRANCH IF YES.
 JMP @#FPSPUR ;OTHERWISE REPORT SPURIOUS FP ERROR.

51\$: CMP (SP)+,(SP)+ ;RESET THE STACK
 STFPS R4 ;GET FPS.
 STST R5 ;GET FEC.
 MOV #200,R0 ;SET FD MODE.
 LDFPS R0

MOV #OVDTT,R0 ;GET THE RESULT.
 STD ACO,(R0)
 MOV R4,@#STMP7
 MOV R5,@#STMP11

MOV #OVDTT,R0 ;CHECK THE RESULT.

MOV R1,R2
 ADD #20,R2
 MOV #4,R3
 3\$: CMP (R0)+,(R2)+ ;BRANCH IF INCORRECT.
 BNE 15\$
 SOB R3,3\$

CMP 42(R1),R4 ;WAS FPS CORRECT?
 BNE 10\$;BRANCH IF FPS IS INCORRECT.

9168									
9169	050364	026105	000044		CMP	44(R1),R5		: IS FEC CORRECT?	
9170	050370	001002			BNE	5\$: IF INCORRECT BRANCH.	
9171	050372	000161	000056	4\$:	JMP	56(R1)		: RETURN. TEST COMPLETED.	
9172									
9173					:REPORT	INCORRECT FEC.			
9174	050376	005761	000046	5\$:	TST	46(R1)		: WAS THE RESULT OVERFLOW OR UNDERFLOW?	
9175	050402	001002			BNE	7\$: BRANCH IF UNDERFLOW.	
9176									
9177								:REPORT BAD FEC ON EXPECTED OVERFLOW.	
9178	050404	104002		6\$:	ERROR	2			
9179	050406	000771			BR	4\$			
9180									
9181	050410			7\$:				:REPORT BAD FEC ON EXPECTED UNDERFLOW.	
9182	050410	104002		8\$:	ERROR	2			
9183	050412	000767			BR	4\$			
9184									
9185					:REPORT	INCORRECT FPS.			
9186	050414	005761	000046	10\$:	TST	46(R1)		: WAS THE RESULT OVER OR UNDER FLOW?	
9187	050420	001002			BNE	12\$: BRANCH IF UNDERFLOW.	
9188								:REPORT FPS BAD AFTER OVERFLOW.	
9189									
9190	050422	104002		11\$:	ERROR	2			
9191	050424	000762			BR	4\$			
9192									
9193	050426			12\$:				:REPORT FPS BAD AFTER UNDERFLOW.	
9194	050426	104002		13\$:	ERROR	2			
9195	050430	000760			BR	4\$			
9196									
9197					:RESULT	INCORRECT.			
9198	050432	012700	050550	15\$:	MOV	#OVDTT,R0		: SEE IF FAILURE IS ANTICIPATED	
9199	050436	010102			MOV	R1,R2		: FAILURE.	
9200	050440	062702	000030		ADD	#30,R2			
9201	050444	012703	000004		MOV	#4,R3			
9202	050450	022022		16\$:	CMP	(R0)+,(R2)+		: BRANCH IF NOT ANTICIPATED.	
9203	050452	001007			BNE	17\$			
9204	050454	077303			SOB	R3,16\$			
9205									
9206	050456	010102			MOV	R1,R2		: ERROR WAS ANTICIPATED SO RETURN	
9207	050460	062702	000054		ADD	#54,R2		: TO THE ERROR REPORT IN THE CALLING	
9208	050464	010237	001236		MOV	R2,@#STMP2		: ROUTINE.	
9209	050470	000112			JMP	(R2)			
9210									
9211	050472	005761	000046	17\$:	TST	46(R1)		: RESULT WAS NOT ANTICIPATED	
9212								: SO ERROR MUST BE REPORTED HERE.	
9213								: FIRST SEE IF ARGUMENTS SHOULD	
9214								: HAVE RESULTED IN OVERFLOW OR UNDER	
9215								: FLOW BY LOOKING AT THE FLAG.	
9216	050476	001002			BNE	19\$: BRANCH IF UNDERFLOW EXPECTED.	
9217									
9218								:REPORT RESULT INCORRECT, EXPECTING	
9219	050500	104002		18\$:	ERROR	2		: OVERFLOW.	
9220	050502	000733			BR	4\$			
9221									
9222	050504			19\$:				:REPORT RESULT INCORRECT, EXPECTING	
9223	050504	104002		20\$:	ERROR	2		: UNDERFLOW.	

```

9224 050506 000731          BR      4$
9225
9226          :IF NO FP TRAP OCCURS COME HERE.
9227 050510 170204      25$: STFPS  R4          :GET FPS.
9228 050512 170305      STST   R5          :GET FEC.
9229 050514 012700 000200  MOV   #200,R0      :SET FD MODE.
9230 050520 170100      LDFPS  R0
9231 050522 012700 050550  MOV   #OVDTT,R0    :GET THE RESULT.
9232 050526 174010      STD    ACO,(R0)
9233 050530 010437 001250  MOV   R4,@#STMP7
9234 050534 010537 001254  MOV   R5,@#STMP11
9235 050540 010102      MOV   R1,R2          :ERROR WAS ANTICIPATED SO
9236 050542 062702 000050  ADD   #50,R2      :RETURN TO THE ERROR REPORT OF THE
9237                                     :CALLING ROUTINE.
9238 050546 000112      JMP    (R2)
9239
9240 050550 000000 000000 000000 OVDTT: .WORD 0,0,0,0
9241 050556 000000
9242
9243 050560          LLLDONE:
9244 050560 104413      RSETUP          :GO INITIALIZE THE FPS AND STACK; AND
9245                                     :SEE IF THE USER HAS EXPRESSED
9246                                     :THE DESIRE TO CHANGE THE SOFTWARE
9247                                     :VIRTUAL CONSOLE SWITCH REGISTER (HAS
9248                                     :THE USER TYPED CONTROL G?).
9249
9250
9251
9252
9253
9254          :*****
9255          :*TEST 54      MODF TEST
9256          :*
9257          :*THIS IS A TEST OF THE MODF INSTRUCTION, WHICH MAKES USE OF
9258          :*A SUBROUTINE TO SET UP THE OPERANDS, EXECUTE THE MODF INSTRUCTION
9259          :*AND CHECK THE RESULTS.
9260          :*
9261          :*****
9262 050562 000004      TST54: SCOPE
9263
9264          :MODF WITH (FSRC=AC=0)
9265          GGG1:
9266 050564 104414          LPERR          :SET UP THE LOOP ON ERROR ADDRESS.
9267 050566 004737 051650  JSR    PC,@#MODFSUB
9268 050572 000000 000000  1$:   .WORD 0,0          :AC
9269 050576 000000 000000  2$:   .WORD 0,0          :FSRC
9270 050602 000000 000000  3$:   .WORD 0,0          :FRACTIONAL RES.
9271 050606 000000 000000  4$:   .WORD 0,0          :INTEGER RES.
9272 050612 177777 177777  5$:   .WORD -1,-1       :ERROR FRACTIONAL RES.
9273 050616 177777 177777  6$:   .WORD -1,-1       :ERROR INGETER RES.
9274 050622 000013          7$:   13
9275 050624 000004          8$:   4
9276 050626 104002          8$:   ERROR 2
9277 050630 000401          BR    9$
9278 050632 104002          ERROR 2
9279 050634          9$:
    
```

9280
 9281
 9282 050634
 9283 050634 104414
 9284 050636 004737 051650
 9285 050642 123456 076543
 9286 050646 000000 000000
 9287 050652 000000 000000
 9288 050656 000000 000000
 9289 050662 123456 076543
 9290 050666 177777 177777
 9291 050672 000000
 9292 050674 000004
 9293 050676 104002
 9294 050700 000401
 9295 050702 104002
 9296 050704
 9297
 9298
 9299 050704
 9300 050704 104414
 9301 050706 004737 051650
 9302 050712 000000 000000
 9303 050716 076543 021234
 9304 050722 000000 000000
 9305 050726 000000 000000
 9306 050732 000000 000000
 9307 050736 177777 177777
 9308 050742 000003
 9309 050744 000004
 9310 050746 104002
 9311 050750 000401
 9312 050752 104002
 9313 050754
 9314
 9315
 9316 050754
 9317 050754 104414
 9318 050756 004737 051650
 9319 050762 046252 125252
 9320 050766 040300 000000
 9321 050772 000000 000000
 9322 050776 046377 177777
 9323 051002 046252 125252
 9324 051006 040300 000000
 9325 051012 000013
 9326 051014 000004
 9327 051016 104002
 9328 051020 000401
 9329 051022 104002
 9330 051024
 9331
 9332
 9333 051024
 9334 051024 104414
 9335 051026 004737 051650

:MODF TEST, WITH (FSRC=0)
 GGG2:

LPERR
 JSR PC,@#MODFSUB
 1\$: .WORD 123456,76543
 2\$: .WORD 0,0
 3\$: .WORD 0,0
 4\$: .WORD 0,0
 5\$: .WORD 123456,76543
 6\$: .WORD -1,-1
 7\$: 0
 8\$: ERROR 2
 BR 9\$
 ERROR 2
 9\$:

:SET UP THE LOOP ON ERROR ADDRESS.
 :AC
 :FSRC
 :FRACTIONAL RES.
 :INTEGER RESULT.
 :ERROR FRACTIONAL RES.
 :ERROR INTEGER RES.
 :FPS BEFORE EXECUTION.
 :FPS AFTER EXECUTION.
 :STORE ZERO FAILURE.

:MODF TEST WITH (AC=0)
 GGG3:

LPERR
 JSR PC,@#MODFSUB
 1\$: .WORD 0,0
 2\$: .WORD 76543,21234
 3\$: .WORD 0,0
 4\$: .WORD 0,0
 5\$: .WORD 0,0
 6\$: .WORD -1,-1
 7\$: 3
 8\$: ERROR 2
 BR 9\$
 ERROR 2
 9\$:

:SET UP THE LOOP ON ERROR ADDRESS.
 :AC
 :FSRC
 :FRACTIONAL RES.
 :INTEGER RES.
 :ERROR FRACTIONAL RES.
 :ERROR INTEGER RES.
 :FPS BEFORE EXECUTION.
 :FPS AFTER EXECUTION.
 :RES.BAD

:MODF TEST WITH EXPONENT OF THE RESULT = 25
 GGG4:

LPERR
 JSR PC,@#MODFSUB
 1\$: .WORD 46252,125252
 2\$: .WORD 40300,0
 3\$: .WORD 0,0
 4\$: .WORD 46377,-1
 5\$: .WORD 46252,125252
 6\$: .WORD 40300,0
 7\$: 13
 8\$: ERROR 2
 BR 9\$
 ERROR 2
 9\$:

:SET UP THE LOOP ON ERROR ADDRESS.
 :AC
 :FSRC
 :FRACTIONAL RES.
 :INTEGER RES.
 :ERROR FRACTIONAL RES.
 :ERROR INTEGER RES.
 :FPS BEFORE EXECUTION.
 :FPS AFTER EXECUTION.
 :ST 134

:MODF TEST WITH EXPONENT OF THE RESULT = 127
 GGG5:

LPERR
 JSR PC,@#MODFSUB

:SET UP THE LOOP ON ERROR ADDRESS.

9336 051032 077652 125252
 9337 051036 040300 000000
 9338 051042 000000 000000
 9339 051046 077777 177777
 9340 051052 077652 125252
 9341 051056 040300 000000
 9342 051062 000000
 9343 051064 000004
 9344 051066 104002
 9345 051070 000401
 9346 051072 104002
 9347 051074

1\$: .WORD 77652,125252 ;AC
 2\$: .WORD 40300,0 ;FSRC
 3\$: .WORD 0,0 ;FRACTIONAL RES.
 4\$: .WORD 77777,-1 ;INTEGER RES.
 5\$: .WORD 77652,125252 ;ERROR FRACTIONAL RES.
 6\$: .WORD 40300,0 ;ERROR INTEGER RES.
 7\$: 0 ;FPS BEFORE EXECUTION.
 4 ;FPS AFTER EXECUTION.
 8\$: ERROR 2
 BR 9\$
 ERROR 2
 9\$:

;MODF TEST WITH EXPONENT OF RESULT = 25
 GGG6:

9350 051074
 9351 051074 104414
 9352 051076 004737 051650
 9353 051102 046200 000001
 9354 051106 040340 000000
 9355 051112 000000 000000
 9356 051116 046340 000001
 9357 051122 040000 000000
 9358 051126 177777 177777
 9359 051132 000013
 9360 051134 000004
 9361 051136 104002
 9362
 9363 051140 000401
 9364 051142 104002
 9365 051144
 9366
 9367

LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
 JSR PC,@#MODFSUB
 1\$: .WORD 46200,1 ;AC
 2\$: .WORD 40340,0 ;FSRC
 3\$: .WORD 0,0 ;FRACTIONAL RES.
 4\$: .WORD 46340,1 ;INTEGER RES.
 5\$: .WORD 40000,0 ;ERROR FRACTIONAL RES.
 6\$: .WORD -1,-1 ;ERROR INTEGER RES.
 7\$: 13 ;FPS BEFORE EXECUTION.
 4 ;FPS AFTER EXECUTION.
 8\$: ERROR 2 ;BAD CONSTANT (NOT 24),
 BR 9\$;OR ST 525 TO 050 INTO 150.
 ERROR 2
 9\$:

;MODF TEST WITH EXPONENT OF THE RESULT = 24
 GGG7:

9368 051144
 9369 051144 104414
 9370 051146 004737 051650
 9371 051152 046000 000001
 9372 051156 040340 000000
 9373 051162 040100 000000
 9374 051166 046140 000001
 9375 051172 000000 000000
 9376 051176 177777 177777
 9377 051202 000000
 9378 051204 000000
 9379 051206 104002
 9380
 9381 051210 000401
 9382 051212 104002
 9383 051214
 9384

LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
 JSR PC,@#MODFSUB
 1\$: .WORD 46000,1 ;AC
 2\$: .WORD 40340,0 ;FSRC
 3\$: .WORD 40100,0 ;FRACTIONAL RES.
 4\$: .WORD 46140,1 ;INTEGER RESULT.
 5\$: .WORD 0,0 ;ERROR FRACTIONAL RES.
 6\$: .WORD -1,-1 ;ERROR INTEGER RES.
 7\$: 0 ;FPS BEFORE EXECUTION.
 0 ;FPS AFTER EXECUTION.
 8\$: ERROR 2 ;BAD CONSTANT USED (NOT 24)
 BR 9\$;OR ST 525 TO 150 INTO 050
 ERROR 2
 9\$:

;MODF TEST WITH EXPONENT OF THE RESULT = 10
 GGG8:

9385
 9386 051214
 9387 051214 104414
 9388 051216 004737 051650
 9389 051222 042577 177777
 9390 051226 040200 000000
 9391 051232 040177 176000

LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
 JSR PC,@#MODFSUB
 1\$: .WORD 42577,-1 ;AC
 2\$: .WORD 40200,0 ;FSRC
 3\$: .WORD 40177,176000 ;FRACTIONAL RES.

9392 051236 042577 140000
 9393 051242 177777 177777
 9394 051246 177777 177777
 9395 051252 000000
 9396 051254 000000
 9397 051256 104002
 9398 051260 000401
 9399 051262 104002
 9400 051264

4\$: .WORD 42577,140000 ; INTEGER RES.
 5\$: .WORD -1,-1 ; ERROR FRACTIONAL RES.
 6\$: .WORD -1,-1 ; ERROR INTEGER RES.
 7\$: 0 ; FPS BEFORE EXECUTION.
 0 ; FPS AFTER EXECUTION.
 8\$: ERROR 2
 BR 9\$
 ERROR 2
 9\$:

;MODF TEST WITH THE EXPONENT OF THE RESULT = 10

9401
 9402
 9403 051264
 9404 051264 104414
 9405 051266 004737 051650
 9406 051272 042577 140001
 9407 051276 040200 000000
 9408 051302 034600 000000
 9409 051306 042577 140000
 9410 051312 000000 000000
 9411 051316 177777 177777
 9412 051322 000000
 9413 051324 000000
 9414 051326 104002
 9415 051330 000401
 9416 051332 104002
 9417 051334

GGG9: LPERR ; SET UP THE LOOP ON ERROR ADDRESS.
 JSR PC,@#MODFSUB
 1\$: .WORD 42577,140001 ; AC
 2\$: .WORD 40200,0 ; FSRC
 3\$: .WORD 34600,0 ; FRACTIONAL RES.
 4\$: .WORD 42577,140000 ; INTEGER RES.
 5\$: .WORD 0,0 ; ERROR FRACTIONAL RES.
 6\$: .WORD -1,-1 ; ERROR INTEGER RES.
 7\$: 0 ; FPS BEFORE EXECUTION.
 0 ; FPS AFTER EXECUTION.
 8\$: ERROR 2 ; ST 532 TO 122 INTO NORMALIZE.
 BR 9\$
 ERROR 2
 9\$:

;MODF TEST WITH EXPONENT OF THE RESULT = 9

9418
 9419
 9420 051334
 9421 051334 104414
 9422 051336 004737 051650
 9423 051342 042377 100000
 9424 051346 040200 000000
 9425 051352 000000 000000
 9426 051356 042377 100000
 9427 051362 177777 177777
 9428 051366 177777 177777
 9429 051372 000013
 9430 051374 000004
 9431 051376 104002
 9432 051400 000401
 9433 051402 104002
 9434 051404

GGG10: LPERR ; SET UP THE LOOP ON ERROR ADDRESS.
 JSR PC,@#MODFSUB
 1\$: .WORD 42377,100000 ; AC
 2\$: .WORD 40200,0 ; FSRC
 3\$: .WORD 0,0 ; FRACTIONAL RES.
 4\$: .WORD 42377,100000 ; INTEGER RES.
 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 2
 BR 9\$
 ERROR 2
 9\$:

;MODF TEST WITH EXPONENT OF THE RESULT = 0

9435
 9436
 9437 051404
 9438 051404 104414
 9439 051406 004737 051650
 9440 051412 040177 177777
 9441 051416 040200 000000
 9442 051422 040177 177777
 9443 051426 000000 000000
 9444 051432 000000 000000
 9445 051436 040177 177777
 9446 051442 000017
 9447 051444 000000

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.

```

9448 051446 104002      8$:      ERROR      2      ;ST 041 TO 046 INTO 246.
9449 051450 000401      BR          9$
9450 051452 104002      ERROR      2
9451 051454
9452
9453      ;MODF TEST WITH EXPONENT OF THE RESULT = -15
9454 051454      GGG12:
9455 051454 104414      LPERR
9456 051456 004737 051650      JSR      PC,@#MODFSUB      ;SET UP THE LOOP ON ERROR ADDRESS.
9457 051462 034377 177777      1$:      .WORD      34377,-1      ;AC
9458 051466 040200 000000      2$:      .WORD      40200,0      ;FSRC
9459 051472 034377 177777      3$:      .WORD      34377,-1      ;FRACTIONAL RES.
9460 051476 000000 000000      4$:      .WORD      0,0      ;INTEGER RES.
9461 051502 000000 000000      5$:      .WORD      0,0      ;ERROR FRACTIONAL RES.
9462 051506 034377 177777      6$:      .WORD      34377,-1      ;ERROR INTEGER RES.
9463 051512 000000      7$:      0      ;FPS BEFORE EXECUTION.
9464 051514 000000      0      ;FPS AFTER EXECUTION.
9465 051516 104002      8$:      ERROR      2
9466 051520 000401      BR          9$
9467 051522 104002      ERROR      2
9468 051524
9469
9470      ;MODF TEST WITH EXPONENT OF RESULT = -64, IN ROUND MODE
9471 051524      GGG13:
9472 051524 104414      LPERR
9473 051526 004737 051650      JSR      PC,@#MODFSUB      ;SET UP THE LOOP ON ERROR ADDRESS.
9474 051532 020000 000001      1$:      .WORD      20000,1      ;AC
9475 051536 040300 000000      2$:      .WORD      40300,0      ;FSRC
9476 051542 020100 000002      3$:      .WORD      20100,2      ;FRACTIONAL RES.
9477 051546 000000 000000      4$:      .WORD      0,0      ;INTEGER RES.
9478 051552 020100 000001      5$:      .WORD      20100,1      ;ERROR FRACTIONAL RES.
9479 051556 000000 000000      6$:      .WORD      0,0      ;ERROR INTEGER RES.
9480 051562 000000      7$:      0      ;FPS BEFORE EXECUTION.
9481 051564 000000      0      ;FPS AFTER EXECUTION.
9482 051566 104002      8$:      ERROR      2      ;ROUND TRUNK, ST 126 INTO ROUND.
9483 051570 000401      BR          9$
9484 051572 104002      ERROR      2
9485 051574
9486
9487      ;MODF TEST WITH EXPONENT OF RESULT = 11
9488 051574      GGG14:
9489 051574 104414      LPERR
9490 051576 004737 051650      JSR      PC,@#MODFSUB      ;SET UP THE LOOP ON ERROR ADDRESS.
9491 051602 142777 170000      1$:      .WORD      142777,170000      ;AC
9492 051606 040200 000000      2$:      .WORD      40200,0      ;FSRC
9493 051612 140000 000000      3$:      .WORD      140000,0      ;FRACTIONAL RES.
9494 051616 142777 160000      4$:      .WORD      142777,160000      ;INTEGER RES.
9495 051622 040000 000000      5$:      .WORD      40000,0      ;ERROR FRACTIONAL RES.
9496 051626 042777 160000      6$:      .WORD      42777,160000      ;ERROR INTEGER RES.
9497 051632 000007      7$:      7      ;FPS BEFORE EXECUTION.
9498 051634 000010      10      ;FPS AFTER EXECUTION.
9499 051636 104002      8$:      ERROR      2      ;SIGN OF FRACTION.
9500 051640 000401      BR          9$
9501 051642 104002      ERROR      2
9502 051644 000167 000366      9$:      JMP          GGGDONE      ;SIGN OF INTEGER.
9503      ;GO TO NEXT TEST.
    
```

9504
9505
9506
9507
9508
9509
9510
9511

: 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
:

```

9512          :          INTRES: .WORD  X,X          ; INTEGER RESULT
9513          :          ERFRES: .WORD  X,X          ; ERROR FRACTION RFSULT
9514          :          ERINTRES: .WORD  X,X          ; ERROR INTEGER RESULT
9515          :          FPSB:      .WORD  X             ; FPS BEFORE EXECUTION
9516          :          FPSA:      .WORD  X             ; FPS AFTER EXECUTION
9517          :          ERR1:     ERROR  X             ; FRACTION ERROR
9518          :          BR       CONT                    ;
9519          :          ERR2:     ERROR  X             ; INTEGER ERROR
9520          :          CONT:                    ; RETURN ADDRESS
9521          :
9522          : THE OPERANDS ARE SET UP (USING ACO FOR THE AC ARGUMENT). THE MODF
9523          : INSTRUCTION IS EXECUTED. THEN THE RESULTS ARE RETRIEVED.
9524          : THE FRACTION PART OF THE RESULT IS COMPARED WITH FRES. IF THIS IS CORRECT
9525          : THEN THE INTEGER PART IS COMPARED WITH INTRES. IF BOTH OF THESE ARE CORRECT
9526          : THEN THE FPS IS COMPARED WITH FPSA. AFTER EXECUTION IF NO ERRORS OCCURRED
9527          : THEN MODFSUB WILL RETURN TO CONT. IF THE FPS WAS INCORRECT
9528          : IT IS REPORTED HERE. IF THE FRACTION IS INCORRECT IT IS COMPARED WITH
9529          : THE ANTICIPATED BAD FRACTION, ERFRES. IF THIS DOESN'T MATCH
9530          : THE TRUE RESULT THEN THE ERROR IS REPORTED HERE. IF THE ANTICIPATED
9531          : FAILURE MATCHES THE TRUE RESULT THEN MODFSUB PASSES CONTROL TO THE
9532          : ERROR CALL AT ERR1. LIKewise IF THE INTEGER PART OF THE RESULT IS
9533          : NOT CORRECT THEN IT IS COMPARED WITH THE ANTICIPATED INTEGER
9534          : FAILURE. IF THIS DOESN'T MATCH THEN THE ERROR IS REPORTED HERE.
9535          : IF A MATCH IS MADE HOWEVER, MODFSUB WILL RETURN CONTROL TO THE ERROR
9536          : CALL AT ERR2.
9537

```

```

9538 051650 012601          MODFSUB:      MOV      (SP)+,R1          ;GET A POINTER TO THE ARGUMENTS
9539 051652 012700 000200      MOV      #200,R0          ;SET FD MODE.
9540 051656 170100          LDFPS   R0
9541 051660 010100          MOV      R1,R0          ;SET UP ACO
9542 051662 172410          LDD     (R0),ACO
9543 051664 012700 052226      MOV      #MODP1,R0          ;PUT A BACKGROUND PATTERN INTO AC1.
9544 051670 172510          LDD     (R0),AC1
9545 051672 016100 000030      MOV      30(R1),R0          ;SET UP THE FPS.
9546 051676 170100          LDFPS   R0
9547 051700 012737 051714 001236      MOV      #1$,@#STMP2
9548 051706 010100          MOV      R1,R0          ;COMPUTE THE ADDRESS OF THE FSRC.
9549 051710 062700 000004      ADD     #4,R0
9550
9551 051714 171410          1$:    MODF   (R0),ACO          ;EXECUTE THE TEST INSTRUCTION.
9552
9553 051716 170204          STFPS   R4          ;GET THE FPS.
9554 051720 012700 000200      MOV      #200,R0          ;SET FD MODE.
9555 051724 170100          LDFPS   R0
9556 051726 012700 052206      MOV      #MODFT0,R0          ;GET THE FRACTIONAL RESULT.
9557 051732 174010          STD     ACO,(R0)
9558 051734 012700 052216      MOV      #MODFT1,R0          ;GET THE INTEGER RESULT.
9559 051740 174110          STD     AC1,(R0)
9560
9561 051742 010102          MOV      R1,R2          ;SAVE THE DATA IN CASE OF ERROR.
9562 051744 010237 001240      MOV      R2,@#STMP3
9563 051750 062702 000004      ADD     #4,R2
9564 051754 010237 001242      MOV      R2,@#STMP4
9565 051760 062702 000004      ADD     #4,R2
9566 051764 010237 001244      MOV      R2,@#STMP5
9567 051770 062702 000004      ADD     #4,R2

```



```

9568 051774 010237 001246      MOV      R2,@#STMP6
9569 052000 012737 052206 001250  MOV      #MODFT0,@#STMP7
9570 052006 012737 052216 001252  MOV      #MODFT1,@#STMP10
9571 052014 010437 001254      MOV      R4,@#STMP11
9572 052020 016137 000032 001256  MOV      32(R1),@#STMP12
9573
9574 052026 012702 052206      MOV      #MODFT0,R2      ;CHECK THE FRACTIONAL RESULT.
9575 052032 026112 000010      CMP      10(R1),(R2)
9576 052036 001022      BNE      10$            ;BRANCH IF INCORRECT.
9577 052040 026162 000012 000002  CMP      12(R1),2(R2)
9578 052046 001016      BNE      10$
9579
9580 052050 012702 052216      MOV      #MODFT1,R2      ;CHECK THE INTEGER RESULT.
9581 052054 026112 000014      CMP      14(R1),(R2)
9582 052060 001026      BNE      15$            ;BRANCH IF INCORRECT.
9583 052062 026162 000016 000002  CMP      16(R1),2(R2)
9584 052070 001022      BNE      15$
9585
9586 052072 026104 000032      CMP      32(R1),R4      ;CHECK THE FPS.
9587 052076 001034      BNE      20$            ;BRANCH IF INCORRECT.
9588
9589 052100 000161 000042      9$:     JMP      42(R1)        ;RETURN.
9590
9591      ;FRACTIONAL ERROR.
9592 052104 026112 000020      10$:    CMP      20(R1),(R2)      ;WAS THE ERROR ANTICIPATED?
9593 052110 001010      BNE      11$            ;BRANCH IF NOT ANTICIPATED.
9594 052112 026162 000022 000002  CMP      22(R1),2(R2)
9595 052120 001004      BNE      11$
9596 052122 010102      MOV      R1,R2          ;THE ERROR WAS ANTICIPATED SO
9597 052124 062702 000034      ADD      #34,R2          ;RETURN TO THE ERROR REPORT AT THE
9598      ;CALLING ROUTINE.
9599      JMP      (R2)
9600
9601 052132      11$:    ;THE ERROR WAS NOT ANTICIPATED SO
9602 052132 104002      12$:    ERROR 2          ;REPORT THE INCORRECT FRACTION HERE.
9603 052134 000761      BR      9$
9604
9605      ;INTEGER ERROR.
9606 052136 026112 000024      15$:    CMP      24(R1),(R2)      ;WAS THIS ERROR ANTICIPATED?
9607 052142 001010      BNE      16$            ;BRANCH IF NOT.
9608 052144 026162 000026 000002  CMP      26(R1),2(R2)
9609 052152 001004      BNE      16$
9610 052154 010102      MOV      R1,R2          ;THE ERROR WAS ANTICIPATED SO RETURN
9611 052156 062702 000040      ADD      #40,R2          ;TO THE ERROR REPORT IN THE CALLING
9612      ;ROUTINE.
9613 052162 000112      JMP      (R2)
9614
9615 052164      16$:    ;THE ERROR WAS NOT ANTICIPATED SO REPORT
9616 052164 104002      17$:    ERROR 2          ;THE INTEGER FAILURE HERE.
9617 052166 000744      BR      9$
9618
9619      ;FPS INCORRECT.
9620 052170 010437 001254 001256  20$:    MOV      R4,@#STMP11      ;REPORT INCORRECT FPS.
9621 052174 016137 000032      MOV      32(R1),@#STMP12
9622 052202 104002      21$:    ERROR 2
9623 052204 000735      BR      9$
    
```

```

9624
9625 052206 000000 000000 000000 MODFT0: .WORD 0,0,0,0
9626 052214 000000
9627
9628 052216 000000 000000 000000 MODFT1: .WORD 0,0,0,0
9629 052224 000000
9630
9631 052226 177777 177777 177777 MODP1: .WORD -1,-1,-1,-1
9632 052234 177777
9633
    
```

```

GGGDONE:
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?).
    
```

```

9634 052236 104413
9635 052236
9636
9637
9638
9639
9640
9641
9642
9643
9644
9645
9646
9647
9648
9649
9650
9651
9652 052240 000004
9653
9654
9655 052242
9656 052242 104414
9657 052244 004737 053746
9658 052250 000000 000000 000000 1$:
9659 052256 000000
9660 052260 000000 000000 000000 2$:
9661 052266 000000
9662 052270 000000 000000 000000 3$:
9663 052276 000000
9664 052300 000000 000000 000000 4$:
9665 052306 000000
9666 052310 000000 000000 000000 5$:
9667 052316 000000
9668 052320 000000 000000 177777 6$:
9669 052326 177777
9670 052330 000200 7$:
9671 052332 000204
9672 052334 104002 8$:
9673 052336 000401
9674 052340 104002 9$:
9675 052342
9676
9677
9678 052342
9679 052342 104414
    
```

```

;*****
;*TEST 55      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.
;*
;*****
TST55: SCOPE
    
```

```

;MODD WITH (FS, C=AC=0)
HHH1:
        LPERR                ;SET UP THE LOOP ON ERROR ADDRESS.
        JSR      PC,@#MODDSUB
        .WORD    0,0,0,0      ;AC
        .WORD    0,0,0,0      ;FSRC
        .WORD    0,0,0,0      ;FRACTIONAL RES.
        .WORD    0,0,0,0      ;INTEGER RES.
        .WORD    0,0,0,0      ;ERROR FRACTIONAL RES.
        .WORD    0,0,-1,-1    ;ERROR INTEGER RES.
        200
        204
        ERROR    2           ;FPS BEFORE EXECUTION.
        BR      9$           ;FPS AFTER EXECUTION.
        ERROR    2           ;ST 231 TO 142 INTO 143
    
```

```

;MODD TEST WITH FSRC=0
HHH2:
        LPERR                ;SET UP THE LOOP ON ERROR ADDRESS.
    
```

```

9680 052344 004737 053746 JSR PC,@#MODDSUB
9681 052350 012345 067012 1$: .WORD 012345,67012 ;AC
9682 052354 034567 012345 .WORD 34567,012345
9683 052360 000000 000000 000000 2$: .WORD 0,0,0,0 ;FSRC
9684 052366 000000 .WORD 0,0,0,0 ;FRACTIONAL RES.
9685 052370 000000 000000 000000 3$: .WORD 0,0,0,0 ;INTEGER RES.
9686 052376 000000 .WORD 0,0,0,0 ;ERROR FRACTIONAL RES.
9687 052400 000000 000000 000000 4$: .WORD 0,0,0,0 ;ERROR INTEGER RES.
9688 052406 000000 .WORD 012345,67012 ;FPS BEFORE EXECUTION.
9689 052410 012345 067012 .WORD 34567,012345 ;FPS AFTER EXECUTION.
9690 052414 034567 012345 .WORD -1,-1,-1,-1 ;STORE DOUBLE ZERO
9691 052420 177777 177777 177777 6$: .WORD -1,-1,-1,-1 ;AC V 1 <= ZERO ST 143
9692 052426 177777
9693 052430 000213 7$: 213 ;FPS BEFORE EXECUTION.
9694 052432 000204 204 ;FPS AFTER EXECUTION.
9695 052434 104002 8$: ERROR 2 ;STORE DOUBLE ZERO
9696 052436 000401 BR 9$
9697 052440 104002 ERROR 2
9698 052442 9$:
9699
9700 ;MODD TEST WITH (AC=0)
9701 052442 HHH3:
9702 052442 104414 LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
9703 052444 004737 053746 JSR PC,@#MODDSUB
9704 052450 000000 000000 000000 1$: .WORD 0,0,0,0 ;AC
9705 052456 000000 .WORD 72727,127272 ;FSRC
9706 052460 072727 127272 2$: .WORD 72727,127272 ;FRACTIONAL RES.
9707 052464 072727 127272 .WORD 0,0,0,0 ;INTEGER RES.
9708 052470 000000 000000 000000 3$: .WORD 0,0,0,0 ;ERROR FRACTIONAL RES.
9709 052476 000000 .WORD -1,-1,-1,-1 ;ERROR INTEGER RES.
9710 052500 000000 000000 000000 4$: .WORD -1,-1,-1,-1 ;FPS BEFORE EXECUTION.
9711 052506 000000 7$: 213 ;FPS AFTER EXECUTION.
9712 052510 177777 177777 177777 5$: .WORD -1,-1,-1,-1 ;STORE DOUBLE ZERO
9713 052516 177777 8$: ERROR 2 ;AC V 1 <= ZERO ST 143
9714 052520 177777 177777 177777 6$: .WORD -1,-1,-1,-1 ;FPS BEFORE EXECUTION.
9715 052526 177777 204 ;FPS AFTER EXECUTION.
9716 052530 000213 8$: ERROR 2
9717 052532 000204 BR 9$
9718 052534 104002 ERROR 2
9719 052536 000401
9720 052540 104002 9$:
9721 052542
9722
9723 ;MODD TEST WITH EXPONENT OF THE RESULT = 57
9724 052542 HHH4:
9725 052542 104414 LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
9726 052544 004737 053746 JSR PC,@#MODDSUB
9727 052550 056252 125252 1$: .WORD 56252,125252 ;AC
9728 052554 125252 125250 .WORD 125252,125250
9729 052560 040300 000000 000000 2$: .WORD 40300,0,0,0 ;FSRC
9730 052566 000000 .WORD 0,0,0,0 ;FRACTIONAL RES.
9731 052570 000000 000000 000000 3$: .WORD 0,0,0,0 ;INTEGER RES.
9732 052576 000000 .WORD 56377,-1,-1,-4 ;ERROR FRACTIONAL RES.
9733 052600 056377 177777 177777 4$: .WORD 56377,-1,-1,-4 ;ERROR INTEGER RES.
9734 052606 177774 .WORD 0,0
9735 052610 000000 000000 5$: .WORD 0,0
    
```

9736	052614	125252	125252			.WORD	125252,125252	
9737	052620	056377	177777	177777	6\$:	.WORD	56377,-1,-1,-1	;ERROR INTEGER RES.
9738	052626	177777						
9739	052630	000213			7\$:	213		;FPS BEFORE EXECUTION.
9740	052632	000204				204		;FPS AFTER EXECUTION.
9741	052634	104002			8\$:	ERROR	2	;ST 526 TO 134 INTO 135
9742	052636	000401				BR	9\$	
9743	052640	104002				ERROR	2	
9744	052642				9\$:			
9745								
9746								
9747	052642							
9748	052642	104414				LPERR		;SET UP THE LOOP ON ERROR ADDRESS.
9749	052644	004737	053746			JSR	PC,@#MODDSUB	
9750	052650	140240	000000	000000	1\$:	.WORD	140240,0,0,0	;AC
9751	052656	000000						
9752	052660	063714	146314		2\$:	.WORD	63714,146314	;FSRC
9753	052664	133572	167737			.WORD	133572,167737	
9754	052670	000000	000000	000000	3\$:	.WORD	0,0,0,0	;FRACTIONAL RES.
9755	052676	000000						
9756	052700	163777	177777		4\$:	.WORD	163777,-1	;INTEGER RES.
9757	052704	162531	125726			.WORD	162531,125726	
9758	052710	177777	177777	177777	5\$:	.WORD	-1,-1,-1,-1	;ERROR FRACTIONAL RES.
9759	052716	177777						
9760	052720	063777	177777		6\$:	.WORD	63777,-1	;ERROR INTEGER RES.
9761	052724	162531	125726			.WORD	162531,125726	
9762	052730	000210			7\$:	210		;FPS BEFORE EXECUTION.
9763	052732	000204				204		;FPS AFTER EXECUTION.
9764	052734	104002			8\$:	ERROR	2	
9765	052736	000401				BR	9\$	
9766	052740	104002				ERROR	2	;ST 526 BAD SIGN
9767	052742				9\$:			
9768								
9769								
9770	052742							
9771	052742	104414				LPERR		;SET UP THE LOOP ON ERROR ADDRESS.
9772	052744	004737	053746			JSR	PC,@#MODDSUB	
9773	052750	056200	000000	000000	1\$:	.WORD	56200,0,0,1	;AC
9774	052756	000001						
9775	052760	040340	000000	000000	2\$:	.WORD	40340,0,0,0	;FSRC
9776	052766	000000						
9777	052770	000000	000000	000000	3\$:	.WORD	0,0,0,0	;FRACTIONAL RES.
9778	052776	000000						
9779	053000	056340	000000	000000	4\$:	.WORD	56340,0,0,1	;INTEGER RES.
9780	053006	000001						
9781	053010	040000	000000	000000	5\$:	.WORD	40000,0,0,0	;ERROR FRACTIONAL RES.
9782	053016	000000						
9783	053020	056340	000000	000000	6\$:	.WORD	56340,0,0,1	;ERROR INTEGER RES.
9784	053026	000001						
9785	053030	000213			7\$:	213		;FPS BEFORE EXECUTION.
9786	053032	000204				204		;FPS AFTER EXECUTION.
9787	053034	104002			8\$:	ERROR	2	;CONSTANT BAD (NOT 56)
9788								;OR ST 525 TO 050 INTO 150
9789	053036	000401				BR	9\$	
9790	053040	104002				ERROR	2	
9791	053042				9\$:			

```

9792
9793      ;MODD TEST WITH EXPONENT OF THE RESULT = 56
9794 053042 HHH7: LPERR      ;SET UP THE LOOP ON ERROR ADDRESS.
9795 053042 104414 JSR      PC,@#MODDSUB
9796 053044 004737 053746 .WORD 56000,0,0,1 ;AC
9797 053050 056000 000000 000000 1$: .WORD
9798 053056 000001 .WORD 40340,0,0,0 ;FSRC
9799 053060 040340 000000 000000 2$: .WORD
9800 053066 000000 .WORD 40100,0,0,0 ;FRACTIONAL RES.
9801 053070 040100 000000 000000 3$: .WORD
9802 053076 000000 .WORD 56140,0,0,1 ;INTEGER RES.
9803 053100 056140 000000 000000 4$: .WORD
9804 053106 000001 .WORD 0,0,0,0 ;ERROR FRACTIONAL RES.
9805 053110 000000 000000 000000 5$: .WORD
9806 053116 000000 .WORD 56140,0,0,1 ;ERROR INTEGER RES.
9807 053120 056140 000000 000000 6$: .WORD
9808 053126 000001 7$: 213 ;FPS BEFORE EXECUTION.
9809 053130 000213 200 ;FPS AFTER EXECUTION.
9810 053132 000200 8$: ERROR 2 ;BAD CONSTANT (NOT 56) OR
9811 053134 104002 ;ST 525 TO 150 INTO 050
9812
9813 053136 000401 BR 9$
9814 053140 104002 ERROR 2
9815 053142 9$:
9816
9817      ;MODD TEST WITH EXPONENT OF THE RESULT = 36
9818 053142 HHH8: LPERR      ;SET UP THE LOOP ON ERROR ADDRESS.
9819 053142 104414 JSR      PC,@#MODDSUB
9820 053144 004737 053746 .WORD 51177,-1,-1,-1 ;AC
9821 053150 051177 177777 177777 1$: .WORD
9822 053156 177777 .WORD 40200,0,0,0 ;FSRC
9823 053160 040200 000000 000000 2$: .WORD
9824 053166 000000 .WORD 40177,-20,0,0 ;FRACTIONAL RES.
9825 053170 040177 177760 000000 3$: .WORD
9826 053176 000000 .WORD 51177,-1,-20,0 ;INTEGER RES.
9827 053200 051177 177777 177760 4$: .WORD
9828 053206 000000 .WORD -1,-1,-1,-1 ;ERROR FRACTIONAL RES.
9829 053210 177777 177777 177777 5$: .WORD
9830 053216 177777 .WORD -1,-1,-1,-1 ;ERROR INTEGER RES.
9831 053220 177777 177777 177777 6$: .WORD
9832 053226 177777 7$: 217 ;FPS BEFORE EXECUTION.
9833 053230 000217 200 ;FPS AFTER EXECUTION.
9834 053232 000200 8$: ERROR 2
9835 053234 104002 BR 9$
9836 053236 000401 ERROR 2
9837 053240 104002 9$:
9838 053242
9839
9840      ;MODD TEST WITH EXPONENT OF THE RESULT = 30
9841 053242 HHH9: LPERR      ;SET UP THE LOOP ON ERROR ADDRESS.
9842 053242 104414 JSR      PC,@#MODDSUB
9843 053244 004737 053746 .WORD 40200,0,0,0 ;AC
9844 053250 040200 000000 000000 1$: .WORD
9845 053256 000000 .WORD 47577,-1 ;FSRC
9846 053260 047577 177777 2$: .WORD
9847 053264 176000 000001 .WORD 176000,1
    
```

9848	053270	031600	000000	000000	3\$:	.WORD	31600,0,0,0	:FRACTIONAL RES.
9849	053276	000000						
9850	053300	047577	177777		4\$:	.WORD	47577,-1	:INTEGER RES.
9851	053304	176000	000000			.WORD	176000,0	
9852	053310	000000	000000	000000	5\$:	.WORD	0,0,0,0	:ERROR FRACTIONAL RES.
9853	053316	000000						
9854	053320	047577	177777	177777	6\$:	.WORD	47577,-1,-1,-1	:ERROR INTEGER RES.
9855	053326	177777						
9856	053330	000200			7\$:		200	:FPS BEFORE EXECUTION.
9857							200	:FPS AFTER EXECUTION.
9858	053332	000200						
9859	053334	104002			8\$:	ERROR	2	:(NORMALIZE) ST 532 TO 122 :INTO NORM.
9860								
9861	053336	000401				BR	9\$	
9862	053340	104002				ERROR	2	:AC V 1 <= X14 :OR ST 733 TO 156 INTO 157.
9863								
9864	053342				9\$:			
9865								
9866								
9867	053342							
9868	053342	104414				HHH10:		:MODD TEST WITH EXPONENT OF THE RESULT = 31
9869	053344	004737	053746			LPERR		:SET UP THE LOOP ON ERROR ADDRESS.
9870	053350	047777	177777		1\$:	JSR	PC,@#MODDSUB	
9871	053354	177000	000000			.WORD	47777,-1	:AC
9872	053360	040200	000000	000000	2\$:	.WORD	177000,0	
9873	053366	000000				.WORD	40200,0,0,0	:FSRC
9874	053370	000000	000000	000000	3\$:	.WORD	0,0,0,0	:FRACTIONAL RES.
9875	053376	000000						
9876	053400	047777	177777		4\$:	.WORD	47777,-1	:INTEGER RES.
9877	053404	177000	000000			.WORD	177000,0	
9878	053410	000000	000000	177000	5\$:	.WORD	0,0,177000,0	:ERROR FRACTIONAL RES.
9879	053416	000000						
9880	053420	177777	177777	177777	6\$:	.WORD	-1,-1,-1,-1	:ERROR INTEGER RES.
9881	053426	177777						
9882	053430	000213			7\$:		213	:FPS BEFORE EXECUTION.
9883	053432	000204					204	:FPS AFTER EXECUTION.
9884	053434	104002			8\$:	ERROR	2	:(BUT FD) STORE X10
9885	053436	000401				BR	9\$	
9886	053440	104002				ERROR	2	
9887	053442				9\$:			
9888								
9889								
9890	053442							
9891	053442	104414				HHH11:		:MODD TEST WITH EXPONENT OF THE RESULT = 0
9892	053444	004737	053746			LPERR		:SET UP THE LOOP ON ERROR ADDRESS.
9893	053450	040200	000000	000000	1\$:	JSR	PC,@#MODDSUB	
9894	053456	000000				.WORD	40200,0,0,0	:AC
9895	053460	040177	072727		2\$:	.WORD	40177,72727	:FSRC
9896	053464	127272	072727			.WORD	127272,72727	
9897	053470	040177	072727		3\$:	.WORD	40177,72727	:FRACTIONAL RES.
9898	053474	127272	072727			.WORD	127272,72727	
9899	053500	000000	000000	000000	4\$:	.WORD	0,0,0,0	:INTEGER RES.
9900	053506	000000						
9901	053510	177777	177777	177777	5\$:	.WORD	-1,-1,-1,-1	:ERROR FRACTIONAL RES.
9902	053516	177777						
9903	053520	000000	000000	177777	6\$:	.WORD	0,0,-1,-1	:ERROR INTEGER RES.

```

9904 053526 177777
9905 053530 000200      7$:      200      :FPS BEFORE EXECUTION.
9906 053532 000200      :FPS AFTER EXECUTION.
9907 053534 104002      8$:      ERROR    2
9908 053536 000401      BR        9$
9909 053540 104002      ERROR    2      :ST 246 TO 126 INTO 127 (BUi FD)
9910 053542
9911
9912      ;MODD TEST WITH EXPONENT OF THE RESULT = -115
9913      HHH12:
9914 053542 104414      LPERR
9915 053544 004737 053746      JSR      PC,@#MODDSUB      :SET UP THE LOOP ON ERROR ADDRESS.
9916 053550 003377 177777      1$:      .WORD    3377,-1      :AC
9917 053554 177777 052525      .WORD    -1,52525
9918 053560 040200 000000 000000 2$:      .WORD    40200,0,0,0      :FSRC
9919 053566 000000
9920 053570 003377 177777      3$:      .WORD    3377,-1      :FRACTIONAL RES.
9921 053574 177777 052525      .WORD    -1,52525
9922 053600 000000 000000 000000 4$:      .WORD    0,0,0,0      :INTEGER RES.
9923 053606 000000
9924 053610 177777 177777 177777 5$:      .WORD    -1,-1,-1,-1      :ERROR FRACTIONAL RES.
9925 053616 177777
9926 053620 000000 000000 177777 6$:      .WORD    0,0,-1,-1      :ERROR INTEGER RES.
9927 053626 177777
9928 053630 000200      7$:      200      :FPS BEFORE EXECUTION.
9929 053632 000200      :FPS AFTER EXECUTION.
9930 053634 104002      8$:      ERROR    2
9931 053636 000401      BR        9$
9932 053640 104002      ERROR    2      :ST 446 TO 126 INTO 127 (BUT FD)
9933 053642
9934
9935      ;MODD TEST WITH EXPONENT OF THE RESULT = -63, IN ROUND MODE.
9936 053642      HHH13:
9937 053642 104414      LPERR
9938 053644 004737 053746      JSR      PC,@#MODDSUB      :SET UP THE LOOP ON ERROR ADDRESS.
9939 053650 040300 000000 000000 1$:      .WORD    40300,0,0,0      :AC
9940 053656 000000
9941 053660 020200 000000 000000 2$:      .WORD    20200,0,0,1      :FSRC
9942 053666 000001
9943 053670 020300 000000 000000 3$:      .WORD    20300,0,0,2      :FRACTIONAL RES.
9944 053676 000002
9945 053700 000000 000000 000000 4$:      .WORD    0,0,0,0      :INTEGER RES.
9946 053706 000000
9947 053710 000000 000000 177777 5$:      .WORD    0,0,-1,-1      :ERROR FRACTIONAL RES.
9948 053716 177777
9949 053720 177777 177777 177777 6$:      .WORD    -1,-1,-1,-1      :ERROR INTEGER RES.
9950 053726 177777
9951 053730 000200      7$:      200      :FPS BEFORE EXECUTION.
9952 053732 000200      :FPS AFTER EXECUTION.
9953 053734 104002      8$:      ERROR    2      :ST 127 INTO RND/TR
9954 053736 000401      BR        9$
9955 053740 104002      ERROR    2
9956 053742 000137 054344      9$:      JMP      @#HHHDONE      :GO TO THE NEXT TEST.
9957
9958      ;THIS SUBROUTINE, MODDSUB, IS CALLED TO SETUP THE
9959      ;OPERANDS, EXECUTE THE MODD INSTRUCTION AND CHECK THE RESULTS.
    
```

9960
9961
9962
9963
9964
9965
9966
9967
9968
9969
9970
9971
9972
9973
9974
9975
9976
9977
9978
9979
9980
9981
9982
9983
9984
9985
9986
9987
9988
9989
9990
9991
9992
9993
9994
9995
9996
9997
9998
9999
10000
10001
10002
10003
10004
10005
10006
10007
10008
10009
10010
10011
10012
10013
10014
10015

053746 012601
053750 012700 000200
053754 170100
053756 010100
053760 172410
053762 012700 052226
053766 172510
053770 016100 000060
053774 170100
053776 012737 054012 001236
054004 010100
054006 062700 000010
054012 171410
054014 170204
054016 012700 000200
054022 170100
054024 012700 054324
054030 174010
054032 012700 054334
054036 174110
054040 010102

:IT IS CALLED THUS:

```

ACARG: .WORD X,X,X,X      :AC OPERAND
FSRCARG: .WORD X,X,X,X    :FSRC OPERAND
FRES: .WORD X,X,X,X      :FRACTIONAL RESULT
INTRES: .WORD X,X,X,X    :INTEGER RESULT
ERFRES: .WORD X,X,X,X    :ERROR FRACTION RESULT
ERINTRES: .WORD X,X,X,X  :ERROR INTEGER RESULT
FPSB: .WORD X            :FPS BEFORE EXECUTION
FPSA: .WORD X            :FPS AFTER EXECUTION
ERR1: ERROR X           :FRACTION ERROR
      BR CONT
ERR2: ERROR X           :INTEGER ERROR
CONT:                   :RETURN ADDRESS
    
```

:THE OPERANDS ARE SET UP (USING ACO FOR THE AC ARGUMENT). THE MODD INSTRUCTION IS EXECUTED. THEN THE RESULTS ARE RETRIEVED. THE FRACTION PART OF THE RESULT IS COMPARED WITH FRES. IF THIS IS CORRECT THEN THE INTEGER PART IS COMPARED WITH INTRES. IF BOTH OF THESE ARE CORRECT THEN THE FPS IS COMPARED WITH FPSA. AFTER EXECUTION IF NO ERRORS OCCURRED THEN MODDSUB WILL RETURN TO CONT. IF THE FPS WAS INCORRECT IT IS REPORTED HERE. IF THE FRACTION IS INCORRECT IT IS COMPARED WITH THE ANTICIPATED BAD FRACTION, ERFRES. IF THIS DOESN'T MATCH THE TRUE RESULT THEN THE ERROR IS REPORTED HERE. IF THE ANTICIPATED FAILURE MATCHES THE TRUE RESULT THEN MODDSUB PASSES CONTROL TO THE ERROR CALL AT ERR1. LIKewise IF THE INTEGER PART OF THE RESULT IS NOT CORRECT THEN IT IS COMPARED WITH THE ANTICIPATED INTEGER FAILURE. IF THIS DOESN'T MATCH THEN THE ERROR IS REPORTED HERE.

:IF A MATCH IS MADE HOWEVER, MODDSUB WILL RETURN CONTROL TO THE ERROR CALL AT ERR2.

```

MODDSUB:  MOV      (SP)+,R1      ;GET A POINTER TO THE ARGUMENTS
          MOV      #200,R0      ;SET FD MODE.
          LDFPS   R0
          MOV      R1,R0       ;SET UP ACO
          LDD     (R0),ACO
          MOV      #MODP1,R0    ;PUT A BACKGROUND PATTERN INTO AC1.
          LDD     (R0),AC1
          MOV      60(R1),R0    ;SET UP THE FPS.
          LDFPS   R0
          MOV      #1$,@#$TMP2 ;COMPUTE THE ADDRESS OF THE FSRC.
          MOV      R1,R0
          ADD     #10,R0
1$:      MODD    (R0),ACO      ;EXECUTE THE TEST INSTRUCTION.
          STFPS   R4           ;GET THE FPS.
          MOV      #200,R0     ;SET FD MODE.
          LDFPS   R0
          MOV      #MODDT0,R0  ;GET THE FRACTIONAL RESULT.
          STD     ACO,(R0)
          MOV      #MODDT1,R0  ;GET THE INTEGER RESULT.
          STD     AC1,(R0)
          MOV      R1,R2       ;SAVE THE DATA IN CASE OF ERROR.
    
```


10016	054042	010237	001240	MOV	R2,@#STMP3	
10017	054046	062702	000010	ADD	#10,R2	
10018	054052	010237	001242	MOV	R2,@#STMP4	
10019	054056	062702	000010	ADD	#10,R2	
10020	054062	010237	001244	MOV	R2,@#STMP5	
10021	054066	062702	000010	ADD	#10,R2	
10022	054072	010237	001246	MOV	R2,@#STMP6	
10023	054076	012737	054324	MOV	#MODDT0,@#STMP7	001250
10024	054104	012737	054334	MOV	#MODDT1,@#STMP10	001252
10025	054112	016137	000062	MOV	62(R1),@#STMP12	001256
10026	054120	010437	001254	MOV	R4,@#STMP11	
10027						
10028	054124	012702	054324	MOV	#MODDT0,R2	
10029	054130	010103		MOV	R1,R3	:CHECK THE FRACTIONAL RESULT.
10030	054132	062703	000020	ADD	#20,R3	
10031	054136	012705	000004	MOV	#4,R5	
10032	054142	022223		2\$: CMP	(R2)+,(R3)+	
10033	054144	001020		BNE	10\$:BRANCH IF INCORRECT.
10034	054146	077503		SOB	R5,2\$	
10035						
10036	054150	012702	054334	MOV	#MODDT1,R2	:CHECK THE INTEGER RESULT.
10037	054154	010103		MOV	R1,R3	
10038	054156	062703	000030	ADD	#30,R3	
10039	054162	012705	000004	MOV	#4,R5	
10040	054166	022223		3\$: CMP	(R2)+,(R3)+	
10041	054170	001026		BNE	15\$:BRANCH IF INCORRECT.
10042	054172	077503		SOB	R5,3\$	
10043						
10044						
10045	054174	026104	000062	CMP	62(R1),R4	:CHECK THE FPS.
10046	054200	001042		BNE	20\$:BRANCH IF INCORRECT.
10047						
10048	054202	000161	000072	9\$: JMP	72(R1)	:RETURN.
10049						
10050						:FRACTIONAL ERROR.
10051	054206	012702	054324	10\$: MOV	#MODDT0,R2	:WAS THE FRACTIONAL ERROR ANTICIPATED?
10052	054212	010103		MOV	R1,R3	
10053	054214	062703	000040	ADD	#40,R3	
10054	054220	012705	000004	MOV	#4,R5	
10055	054224	022223		50\$: CMP	(R2)+,(R3)+	
10056	054226	001005		BNE	11\$:BRANCH IF NOT ANTICIPATED.
10057	054230	077503		SOB	R5,50\$	
10058	054232	010102		MOV	R1,R2	:THE ERROR WAS ANTICIPATED SO
10059	054234	062702	000064	ADD	#64,R2	:RETURN TO THE ERROR REPORT AT THE
10060						:CALLING ROUTINE.
10061	054240	000112		JMP	(R2)	
10062						
10063	054242			11\$: ERROR	2	:THE ERROR WAS NOT ANTICIPATED SO
10064	054242	104002		12\$: BR	9\$:REPORT THE INCORRECT FRACTION HERE.
10065	054244	000756				
10066						
10067						:INTEGER ERROR.
10068	054246	012702	054334	15\$: MOV	#MODDT1,R2	:WAS THE INTEGER ERROR ANTICIPATED?
10069	054252	010103		MOV	R1,R3	
10070	054254	062703	000050	ADD	#50,R3	
10071	054260	012705	000004	MOV	#4,R5	

```

10072 054264 022223          60$:  CMP      (R2)+,(R3)+
10073 054266 001005          BNE      17$      ;BRANCH IF NOT ANTICIPATED.
10074 054270 077503          SOB      R5,60$
10075 054272 010102          MOV      R1,R2    ;THE ERROR WAS ANTICIPATED SO RETURN
10076 054274 062702 000070  ADD      #70,R2   ;TO THE ERROR REPORT IN THE CALLING
10077                                ;ROUTINE.
10078 054300 000112          JMP      (R2)
10079
10080 054302          16$:
10081 054302 104002          17$:  ERROR   2      ;THE ERROR WAS NOT ANTICIPATED SO REPORT
10082 054304 000736          BR      9$      ;THE INTEGER FAILURE HERE.
10083
10084                                ;FPS INCORRECT.
10085 054306 010437 001254 20$:  MOV      R4,@#$TMP11 ;REPORT INCORRECT FPS.
10086 054312 016137 000062 001256  MOV      62(R1),@#$TMP12
10087 054320 104002          21$:  ERROR   2
10088 054322 000727          BR      9$
10089
10090 054324 000000 000000 000000 MODDT0: .WORD 0,0,0,0
10091 054332 000000
10092
10093 054334 000000 000000 000000 MODDT1: .WORD 0,0,0,0
10094 054342 000000
10095
10096 054344          HHHDONE:
10097 054344 104413          RSETUP      ;GO INITIALIZE THE FPS AND STACK; AND
10098                                ;SEE IF THE USER HAS EXPRESSED
10099                                ;THE DESIRE TO CHANGE THE SOFTWARE
10100                                ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
10101                                ;THE USER TYPED CONTROL G?).
10102
10103
10104
10105
10106                                ;*****
10107                                ;*TEST 56      UNDER/OVER FLOW, USING MODF WITH TRAPS DISABLED, TEST
10108                                ;*
10109                                ;*THIS IS A TEST OF THE MODF OVERFLOW AND UNDERFLOW CONDITIONS. IT MAKES
10110                                ;*USE OF A SUBROUTINE TO SETUP THE OPERANDS, EXECUTE THE MODF INSTRUCTION
10111                                ;*AND CHECK THE RESULTS. TRAPS ARE DISABLED DURING THIS TEST.
10112                                ;*
10113                                ;*****
10114 054346 000004          TST56:  SCOPE
10115
10116                                ;UNDERFLOW TEST, WITH EXPONENT OF THE RESULT = -129, FIU = 1, FID = 1
10117 054350          MMM1:
10118 054350 104414          LPERR      ;SET UP THE LOOP ON ERROR ADDRESS.
10119 054352 004767 000320  JSR      PC,MODFOV
10120 054356 020123 045676  1$:  .WORD 20123,45676 ;AC
10121 054362 020200 000000  2$:  .WORD 20200,0     ;FSRC
10122 054366 000123 045676  3$:  .WORD 123,45676   ;FRACTIONAL RES.
10123 054372 000000 000000  4$:  .WORD 0,0         ;INTEGER RES.
10124 054376 177777 177777  5$:  .WORD -1,-1      ;ERROR FRACTIONAL RES.
10125 054402 177777 177777  6$:  .WORD -1,-1      ;ERROR INTEGER RES.
10126 054406 042000          7$:  42000           ;FPS BEFORE EXECUTION.
10127 054410 142004                   142004         ;FPS AFTER EXECUTION.
    
```

```
10128 054412 000012          12          :FEC
10129 054414 104002      8$:  ERROR 2          :FEC INCORRECT, UNDERFLOW.
10130 054416 000401          BR 9$
10131 054420 104002          ERROR 2          :AC V 1 (2,3) <= ZERO, ST 126.
10132 054422      9$:
10133
10134      :UNDERFLOW EXP OF RESULT = -193, FIU = 0, FID = 1
10135 054422      MMM2:
10136 054422 104414          LPERR          :SET UP THE LOOP ON ERROR ADDRESS.
10137 054424 004737 054676      JSR PC,@#MODFOV
10138 054430 010200 000000      1$: .WORD 10200,0          :AC
10139 054434 010000 000000      2$: .WORD 10000,0          :FSRC
10140 054440 000000 000000      3$: .WORD 0,0            :FRACTIONAL RES.
10141 054444 000000 000000      4$: .WORD 0,0            :INTEGER RES.
10142 054450 177777 177777      5$: .WORD -1,-1          :ERROR FRACTIONAL RES.
10143 054454 177777 177777      6$: .WORD -1,-1          :ERROR INTEGER RES.
10144 054460 005013          7$: 5013              :FPS BEFORE EXECUTION.
10145 054462 005004          5004              :FPS AFTER EXECUTION.
10146 054464 000012          12                :FEC
10147 054466 000240      8$:  NOP
10148 054470 000401          BR 9$
10149 054472 104002          ERROR 2
10150 054474      9$:
10151
10152      :OVERFLOW TEST WITH EXPONENT OF THE RESULT = 128, FIV = 1, FID = 1
10153 054474      MMM3:
10154 054474 104414          LPERR          :SET UP THE LOOP ON ERROR ADDRESS.
10155 054476 004737 054676      JSR PC,@#MODFOV
10156 054502 060052 125252      1$: .WORD 60052,125252    :AC
10157 054506 060200 000000      2$: .WORD 60200,0        :FSRC
10158 054512 000000 000000      3$: .WORD 0,0            :FRACTIONAL RES.
10159 054516 000052 125252      4$: .WORD 52,125252      :INTEGER RES.
10160 054522 000000 000000      5$: .WORD 0,0            :ERROR FRACTIONAL RES.
10161 054526 000000 000000      6$: .WORD 0,0            :ERROR INTEGER RES.
10162 054532 041000          7$: 41000              :FPS BEFORE EXECUTION.
10163 054534 141006          141006           :FPS AFTER EXECUTION.
10164 054536 000010          10                :FEC
10165 054540 104002      8$:  ERROR 2          :BAD FEC ON OVERFLOW.
10166 054542 000401          BR 9$
10167 054544 104002          ERROR 2
10168
10169 054546      9$:
10170
10171      :OVERFLOW TEST WITH EXPONENT OF THE RESULT = 130, FIV = 0, FID = 1
10172 054546      MMM4:
10173 054546 104414          LPERR          :SET UP THE LOOP ON ERROR ADDRESS.
10174 054550 004737 054676      JSR PC,@#MODFOV
10175 054554 060345 067654      1$: .WORD 60345,67654    :AC
10176 054560 060200 000000      2$: .WORD 60200,0        :FSRC
10177 054564 000000 000000      3$: .WORD 0,0            :FRACTIONAL RES.
10178 054570 000000 000000      4$: .WORD 0,0            :INTEGER RES.
10179 054574 000000 000000      5$: .WORD 0,0            :ERROR FRACTIONAL RES.
10180 054600 000345 067654      6$: .WORD 345,67654      :ERROR INTEGER RES.
10181 054604 006011          7$: 6011              :FPS BEFORE EXECUTION.
10182 054606 006006          6006              :FPS AFTER EXECUTION.
10183 054610 000010          10                :FEC
```

10184 054612 000240
 10185 054614 000401
 10186 054616 104002
 10187 054620
 10188
 10189
 10190
 10191 054620
 10192 054620 104414
 10193 054622 004737 054676
 10194 054626 160252 125252
 10195 054632 060000 000000
 10196 054636 000000 000000
 10197 054642 100052 125252
 10198 054646 000000 000000
 10199 054652 000052 125252
 10200 054656 041000
 10201 054660 141006
 10202 054662 000010
 10203 054664 104002
 10204 054666 000401
 10205 054670 104002
 10206 054672 000137 055272

8\$: NOP
 BR 9\$
 ERROR 2 ;ST 520 TO 162 INTO STORE ZERO TWICE.
 9\$:
 ;OVERFLOW TEST WITH EXPONENT OF THE RESULT = 128, RESULT NEGATIVE
 ;AND FIV = 1, FID = 1
 MMS:
 LPERR ;SET UP THE LOOP ON ERROR ADDRESS.
 JSR PC,@#MODFOV
 1\$: .WORD 160252,125252 ;AC
 2\$: .WORD 60000,0 ;FSRC
 3\$: .WORD 0,0 ;FRACTIONAL RES.
 4\$: .WORD 100052,125252 ;INTEGER RES.
 5\$: .WORD 0,0 ;ERROR FRACTIONAL RES.
 6\$: .WORD 52,125252 ;ERROR INTEGER RES.
 7\$: 41000 ;FPS BEFORE EXECUTION.
 141006 ;FPS AFTER EXECUTION.
 10 ;FEC
 8\$: ERROR 2
 BR 9\$
 ERROR 2 ;ST 517, BAD SIGN.
 9\$: JMP @#MMDONE ;GO TO THE NEXT TEST.

;THIS SUBROUTINE, MODFOV, IS CALLED TO SETUP THE
 ;OPERANDS, EXECUTE THE MODF INSTRUCTION AND CHECK THE RESULTS.
 ;IT IS CALLED THUS:

```

ACARG: .WORD X,X ;AC OPERAND
FSRCARG: .WORD X,X ;FSRC OPERAND
FRES: .WORD X,X ;FRACTIONAL RESULT
INTRES: .WORD X,X ;INTEGER RESULT
ERFRES: .WORD X,X ;ERROR FRACTION RESULT
ERINTRES: .WORD X,X ;ERROR INTEGER RESULT
FPSB: .WORD X ;FPS BEFORE EXECUTION
FPSA: .WORD X ;FPS AFTER EXECUTION
FEC: .WORD X ;FEC
ERR1: ERROR X ;FEC ERROR
BR CONT
ERR2: ERROR X ;INTEGER ERROR
CONT: ;RETURN ADDRESS
  
```

;THE OPERANDS ARE SET UP (USING ACO FOR THE AC ARGUMENT). THE MODF
 ;INSTRUCTION IS EXECUTED. THEN THE RESULTS ARE RETRIEVED.
 ;THE FRACTION PART OF THE RESULT IS COMPARED WITH FRES. IF THIS IS CORRECT
 ;THEN THE INTEGER PART IS COMPARED WITH INTRES. IF BOTH OF THESE ARE CORRECT
 ;THEN THE FPS IS COMPARED WITH FPSA. AFTER EXECUTION IF NO ERRORS OCCURRED
 ;THEN MODFOV WILL RETURN TO CONT. IF THE FPS WAS INCORRECT
 ;IT IS REPORTED HERE. IF THE FRACTION IS INCORRECT IT IS COMPARED WITH
 ;THE ANTICIPATED BAD FRACTION, ERFRES. IF THIS DOESN'T MATCH
 ;THE TRUE RESULT THEN THE ERROR IS REPORTED HERE. IF THE ANTICIPATED
 ;FAILURE MATCHES THE TRUE RESULT THEN MODFOV PASSES CONTROL TO THE
 ;ERROR CALL AT ERR1. LIKewise IF THE INTEGER PART OF THE RESULT IS
 ;NOT CORRECT THEN IT IS COMPARED WITH THE ANTICIPATED INTEGER
 ;FAILURE. IF THIS DOESN'T MATCH THEN THE ERROR IS REPORTED HERE.
 ;IF A MATCH IS MADE HOWEVER, MODFOV WILL RETURN CONTROL TO THE ERROR

10207
 10208
 10209
 10210
 10211
 10212
 10213
 10214
 10215
 10216
 10217
 10218
 10219
 10220
 10221
 10222
 10223
 10224
 10225
 10226
 10227
 10228
 10229
 10230
 10231
 10232
 10233
 10234
 10235
 10236
 10237
 10238
 10239

```
10240 ;CALL AT ERR2.
10241
10242 054676 012601 MODFOV MOV (SP)+,R1 ;GET A POINTER TO THE ARGUMENTS
10243 054700 012700 000200 MOV #200,R0 ;SET FD MODE.
10244 054704 170100 LDFPS R0
10245 054706 010100 MOV R1,R0 ;SET UP ACO
10246 054710 172410 LDD (R0),ACO
10247 054712 012700 052226 MOV #MODP1,R0 ;PUT A BACKGROUND PATTERN INTO AC1.
10248 054716 172510 LDD (R0),AC1
10249 054720 016100 000030 MOV 30(R1),R0 ;SET UP THE FPS.
10250 054724 170100 LDFPS R0
10251 054726 012737 054742 001236 MOV #1$,@#$TMP2
10252 054734 010100 MOV R1,R0 ;COMPUTE THE ADDRESS OF THE FSRC.
10253 054736 062700 000004 ADD #4,R0
10254
10255 054742 171410 1$: MODF (R0),ACO ;EXECUTE THE TEST INSTRUCTION.
10256
10257 054744 170204 STFPS R4 ;GET THE FPS.
10258 054746 170305 STST R5 ;GET FEC.
10259 054750 012700 000200 MOV #200,R0 ;SET FD MODE.
10260 054754 170100 LDFPS R0
10261 054756 012700 055252 MOV #MODFDO,R0 ;GET THE FRACTIONAL RESULT.
10262 054762 174010 STD ACO,(R0)
10263 054764 012700 055262 MOV #MODFD1,R0 ;GET THE INTEGER RESULT.
10264 054770 174110 STD AC1,(R0)
10265
10266 054772 010102 MOV R1,R2 ;SAVE THE DATA IN CASE OF ERROR.
10267 054774 010237 001240 MOV R2,@#$TMP3
10268 055000 062702 000004 ADD #4,R2
10269 055004 010237 001242 MOV R2,@#$TMP4
10270 055010 062702 000004 ADD #4,R2
10271 055014 010237 001244 MOV R2,@#$TMP5
10272 055020 062702 000004 ADD #4,R2
10273 055024 010237 001246 MOV R2,@#$TMP6
10274 055030 012737 055252 001250 MOV #MODFDO,@#$TMP7
10275 055036 012737 055262 001252 MOV #MODFD1,@#$TMP10
10276 055044 010437 001254 MOV R4,@#$TMP11
10277 055050 016137 000032 001256 MOV 32(R1),@#$TMP12
10278 055056 010537 001260 MOV R5,@#$TMP13
10279 055062 016137 000034 001262 MOV 34(R1),@#$TMP14
10280
10281 055070 012702 055252 MOV #MODFDO,R2 ;CHECK THE FRACTIONAL RESULT.
10282 055074 026112 000010 CMP 10(R1),(R2)
10283 055100 001025 BNE! 10$ ;BRANCH IF INCORRECT.
10284 055102 026162 000012 000002 CMP 12(R1),2(R2)
10285 055110 001021 BNE 10$
10286
10287 055112 012702 055262 MOV #MODFD1,R2 ;CHECK THE INTEGER RESULT.
10288 055116 026112 000014 CMP 14(R1),(R2)
10289 055122 001016 BNE 15$ ;BRANCH IF INCORRECT.
10290 055124 026162 000016 000002 CMP 16(R1),2(R2)
10291 055132 001012 BNE 15$
10292
10293 055134 026104 000032 CMP 32(R1),R4 ;CHECK THE FPS.
10294 055140 001024 BNE 20$ ;BRANCH IF INCORRECT.
10295
```

```

10296 055142 026105 000034          CMP    34(R1),R5      ;CHECK THE FEC.
10297 055146 001030          BNE    25$           ;BRANCH IF INCORRECT.
10298
10299 055150 000161 000044      9$:    JMP    44(R1)      ;RETURN.
10300
10301          ;FRACTIONAL ERROR.
10302 055154          10$:
10303 055154 104002      12$:    ERROR    2      ;THE ERROR WAS NOT ANTICIPATED SO
10304 055156 000774          BR      9$          ;REPORT THE INCORRECT FRACTION HERE.
10305
10306          ;INTEGER ERROR.
10307 055160 026112 000024      15$:    CMP    24(R1),(R2) ;WAS THIS ERROR ANTICIPATED?
10308 055164 001010          BNE    16$          ;BRANCH IF NOT.
10309 055166 026162 000026 000002      CMP    26(R1),2(R2)
10310 055174 001004          BNE    16$
10311 055176 010102          MOV    R1,R2        ;THE ERROR WAS ANTICIPATED SO RETURN
10312 055200 062702 000042      ADD    #42,R2       ;TO THE ERROR REPORT IN THE CALLING
10313          ;ROUTINE.
10314 055204 000112          JMP    (R2)
10315
10316 055206          16$:
10317 055206 104002      17$:    ERROR    2      ;THE ERROR WAS NOT ANTICIPATED SO REPORT
10318 055210 000757          BR      9$          ;THE INTEGER FAILURE HERE.
10319
10320          ;FPS INCORRECT.
10321 055212 010437 001254      20$:    MOV    R4,@#$TMP11 ;REPORT INCORRECT FPS.
10322 055216 016137 000032 001256      MOV    32(R1),@#$TMP12
10323 055224 104002      21$:    ERROR    2
10324 055226 000750          BR      9$
10325
10326          ;REPORT FEC ERROR.
10327 055230 010537 001260      25$:    MOV    R5,@#$TMP13
10328 055234 016137 000034 001262      MOV    34(R1),@#$TMP14
10329 055242 010102          MOV    R1,R2
10330 055244 062702 000036          ADD    #36,R2
10331 055250 000112          JMP    (R2)
10332
10333 055252 000000 000000 000000 MODFD0: .WORD 0,0,0,0
10334 055260 000000
10335
10336 055262 000000 000000 000000 MODFD1: .WORD 0,0,0,0
10337 055270 000000
10338
10339 055272          MMDONE:
10340 055272 104413          RSETUP
10341
10342          ;GO INITIALIZE THE FPS AND STACK; AND
10343          ;SEE IF THE USER HAS EXPRESSED
10344          ;THE DESIRE TO CHANGE THE SOFTWARE
10345          ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
10346          ;THE USER TYPED CONTROL G?).
10347
10348
10349          ;*****
10350          ;*TEST 57 UNDER\OVER FLOW, USING MODD WITH TRAPS DISABLED, TEST
10351          ;*

```

```

10352      ;*THIS IS A TEST OF THE MODD INSTRUCTION'S OVER FLOW AND UNDER FLOW
10353      ;*CONDITIONS. A SUBROUTINE IS USED TO SET UP THE OPERANDS, EXECUTE THE
10354      ;*MODD INSTUCTION AND CHECK THE RESULTS.
10355      ;*
10356      ;*****
10357 055274 000004      TST57: SCOPE
10358
10359      ;UNDERFLOW TEST WITH EXPONENT OF THE RESULT = -129, FIU = 1, FID = 1
10360      NNN1:
10361 055276 104414      LPERR      ;SET UP THE LOOP ON ERROR ADDRESS.
10362 055300 004737 055712      JSR      PC,@#MODDOV
10363 055304 020252 125252      1$:      .WORD 20252,125252      ;AC
10364 055310 125252 125252      .WORD 125252,125252
10365 055314 020100 000000 000000      2$:      .WORD 20100,0,0,0      ;FSRC
10366 055322 000000
10367 055324 000177 177777 177777      3$:      .WORD 177,-1,-1,-1      ;FRACTIONAL RES.
10368 055332 177777
10369 055334 000000 000000 000000      4$:      .WORD 0,0,0,0      ;INTEGER RES.
10370 055342 000000
10371 055344 020252 125252      5$:      .WORD 20252,125252      ;ERROR FRACTIONAL RES.
10372 055350 125252 125252      .WORD 125252,125252
10373 055354 000000 000000 177777      6$:      .WORD 0,0,-1,-1      ;ERROR INTEGER RES.
10374 055362 177777
10375 055364 042200      7$:      42200      ;FPS BEFORE EXECUTION.
10376 055366 142204      142204      ;FPS AFTER EXECUTION.
10377 055370 000012      12      ;FEC
10378 055372 104002      8$:      ERROR 2      ;FEC INCORRECT ON UNDERFLOW.
10379 055374 000401      BR 9$
10380 055376 104002      ERROR 2      ;ST 155 (BUT FD)
10381 055400      9$:
10382
10383      ;UNDERFLOW TEST WITH EXPONENT OF THE RESULT = -193, FIU = 0, FID = 1
10384      NNN2:
10385 055400 104414      LPERR      ;SET UP THE LOOP ON ERROR ADDRESS.
10386 055402 004737 055712      JSR      PC,@#MODDOV
10387 055406 010000 000000      1$:      .WORD 10000,0      ;AC
10388 055412 123456 000000      .WORD 123456,0
10389 055416 010200 000000 000000      2$:      .WORD 10200,0,0,0      ;FSRC
10390 055424 000000
10391 055426 000000 000000 000000      3$:      .WORD 0,0,0,0      ;FRACTIONAL RES.
10392 055434 000000
10393 055436 000000 000000 000000      4$:      .WORD 0,0,0,0      ;INTEGER RES.
10394 055444 000000
10395 055446 000000 000000 000000      5$:      .WORD 0,0,0,0      ;ERROR FRACTIONAL RES.
10396 055454 000000
10397 055456 000000 000000      6$:      .WORD 0,0      ;ERROR INTEGER RES.
10398 055462 123456 000000      .WORD 123456,0
10399 055466 005213      7$:      5213      ;FPS BEFORE EXECUTION.
10400 055470 005204      5204      ;FPS AFTER EXECUTION.
10401 055472 000012      12
10402 055474 000240      8$:      NOP
10403 055476 000401      BR 9$
10404 055500 104002      ERROR 2      ;ST 047 (BUT FD).
10405 055502      9$:
10406
10407      ;OVERFLOW TEST WITH EXPONENT OF THE RESULT = 128, FIV = 1, FID = 1

```

```

10408 055502          NNN3:
10409 055502 104414  LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
10410 055504 004737 055712 JSR          PC,@#MODDOV
10411 055510 060252 125252 1$: .WORD      60252,125252 ;AC
10412 055514 125252 125252 .WORD      125252,125252
10413 055520 060100 000000 000000 2$: .WORD      60100,0,0,0 ;FSRC
10414 055526 000000 .WORD      0,0,0,0 ;FRACTIONAL RES.
10415 055530 000000 000000 000000 3$: .WORD      0,0,0,0 ;FRACTIONAL RES.
10416 055536 000000 .WORD      177,-1,-1,-1 ;INTEGER RES.
10417 055540 000177 177777 177777 4$: .WORD      177,-1,-1,-1 ;INTEGER RES.
10418 055546 177777 .WORD      0,0,0,0 ;ERROR FRACTIONAL RES.
10419 055550 000000 000000 000000 5$: .WORD      0,0,0,0 ;ERROR FRACTIONAL RES.
10420 055556 000000 .WORD      177,-1 ;ERROR INTEGER RES.
10421 055560 000177 177777 6$: .WORD      177,-1 ;ERROR INTEGER RES.
10422 055564 125252 125252 .WORD      125252,125252
10423 055570 041200 7$: 41200 ;FPS BEFORE EXECUTION.
10424 055572 141206 141206 ;FPS AFTER EXECUTION.
10425 055574 000010 10 ;FEC
10426 055576 104002 8$: ERROR      2 ;FEC BAD ON OVERFLOW.
10427 055600 000401 BR          9$
10428 055602 104002 ERROR      2 ;ST 520 TO 162 INTO 163 (BUT FD).
10429 055604 9$:
10430
10431 ;OVERFLOW TEST WITH EXPONENT OF THE RESULT = 130, FIV = 0, FID = 1
10432 055604          NNN4:
10433 055604 104414  LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
10434 055606 004737 055712 JSR          PC,@#MODDOV
10435 055612 060200 000000 1$: .WORD      60200,0 ;AC
10436 055616 125252 000000 .WORD      125252,0
10437 055622 060200 000000 000000 2$: .WORD      60200,0,0,0 ;FSRC
10438 055630 000000 .WORD      0,0,0,0 ;FRACTIONAL RES.
10439 055632 000000 000000 000000 3$: .WORD      0,0,0,0 ;FRACTIONAL RES.
10440 055640 000000 .WORD      0,0,0,0 ;INTEGER RES.
10441 055642 000000 000000 000000 4$: .WORD      0,0,0,0 ;INTEGER RES.
10442 055650 000000 .WORD      0,0,0,0 ;ERROR FRACTIONAL RES.
10443 055652 000000 000000 000000 5$: .WORD      0,0,0,0 ;ERROR FRACTIONAL RES.
10444 055660 000000 .WORD      400,0 ;ERROR INTEGER RES.
10445 055662 000400 000000 6$: .WORD      400,0 ;ERROR INTEGER RES.
10446 055666 125252 000000 .WORD      125252,0
10447 055672 006211 7$: 6211 ;FPS BEFORE EXECUTION.
10448 055674 006206 6206 ;FPS AFTER EXECUTION.
10449 055676 000010 10 ;FEC
10450 055700 000240 8$: NOP
10451 055702 000401 BR          9$
10452 055704 104002 ERROR      2 ;ST 520 TO 162 INTO STORE ZERO TWICE.
10453 055706 000137 056320 9$: JMP          @#NNNDONE ;GO TO NEXT TEST.
10454
10455 ;THIS SUBROUTINE, MODDOV, IS CALLED TO SETUP THE
10456 ;OPERANDS, EXECUTE THE MODD INSTRUCTION AND CHECK THE RESULTS.
10457 ;IT IS CALLED THUS:
10458 ;
10459 ;
10460 ;
10461 ;
10462 ;
10463 ;
ACARG: .WORD      X,X,X,X ;AC OPERAND
FSRCARG: .WORD     X,X,X,X ;FSRC OPERAND
FRES: .WORD       X,X,X,X ;FRACTIONAL RESULT
INTRES: .WORD      X,X,X,X ;INTEGER RESULT
ERFRES: .WORD      X,X,X,X ;ERROR FRACTION RESULT
    
```


B 1

10464
10465
10466
10467
10468
10469
10470
10471
10472
10473
10474
10475
10476
10477
10478
10479
10480
10481
10482
10483
10484
10485
10486
10487
10488
10489
10490
10491
10492
10493
10494
10495
10496
10497
10498
10499
10500
10501
10502
10503
10504
10505
10506
10507
10508
10509
10510
10511
10512
10513
10514
10515
10516
10517
10518
10519

055712 012601
055714 012700 000200
055720 170100
055722 010100
055724 172410
055726 012700 052226
055732 172510
055734 016100 000060
055740 170100
055742 012737 055756 001236
055750 010100
055752 062700 000010

055756 171410

055760 170305
055762 170204
055764 012700 000200
055770 170100
055772 012700 056300
055776 174010
056000 012700 056310
056004 174110

056006 010102
056010 010237 001240
056014 062702 000010
056020 010237 001242
056024 062702 000010
056030 010237 001244
056034 062702 000010
056040 010237 001246

```

ERINTRES: .WORD X,X,X,X      ;ERROR INTEGER RESULT
FPSB:      .WORD X            ;FPS BEFORE EXECUTION
FPSA:      .WORD X            ;FPS AFTER EXECUTION
ERR1:      ERROR X           ;FRACTION ERROR
           BR CONT
ERR2:      ERROR X           ;INTEGER ERROR
CONT:
:THE OPERANDS ARE SET UP (USING ACO FOR THE AC ARGUMENT). THE MODD
:INSTRUCTION IS EXECUTED. THEN THE RESULTS ARE RETRIEVED.
:THE FRACTION PART OF THE RESULT IS COMPARED WITH FRES. IF THIS IS CORRECT
:THEN THE INTEGER PART IS COMPARED WITH INTRES. IF BOTH OF THESE ARE CORRECT
:THEN THE FPS IS COMPARED WITH FPSA. AFTER EXECUTION IF NO ERRORS OCCURRED
:THEN MODDOV WILL RETURN TO CONT. IF THE FPS WAS INCORRECT
:IT IS REPORTED HERE. IF THE FRACTION IS INCORRECT IT IS COMPARED WITH
:THE ANTICIPATED BAD FRACTION, ERFRES. IF THIS DOESN'T MATCH
:THE TRUE RESULT THEN THE ERROR IS REPORTED HERE. IF THE ANTICIPATED
:FAILURE MATCHES THE TRUE RESULT THEN MODDOV PASSES CONTROL TO THE
:ERROR CALL AT ERR1. LIKEWISE IF THE INTEGER PART OF THE RESULT IS
:NOT CORRECT THEN IT IS COMPARED WITH THE ANTICIPATED INTEGER
:FAILURE. IF THIS DOESN'T MATCH THEN THE ERROR IS REPORTED HERE.
:IF A MATCH IS MADE HOWEVER, MODDOV WILL RETURN CONTROL TO THE ERROR
:CALL AT ERR2.

MODDOV: MOV (SP)+,R1          ;GET A POINTER TO THE ARGUMENTS
        MOV #200,R0          ;SET FD MODE.
        LDFPS R0
        MOV R1,R0           ;SET UP ACO
        LDD (R0),ACO
        MOV #MODP1,R0       ;PUT A BACKGROUND PATTERN INTO AC1.
        LDD (R0),AC1
        MOV 60(R1),R0       ;SET UP THE FPS.
        LDFPS R0
        MOV #1$,@#STMP2    ;COMPUTE THE ADDRESS OF THE FSRC.
        MOV R1,R0
        ADD #10,R0

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

        STST R5             ;GET THE FPS.
        STFPS R4           ;GET THE FPS.
        MOV #200,R0        ;SET FD MODE.
        LDFPS R0
        MOV #MODDD0,R0     ;GET THE FRACTIONAL RESULT.
        STD ACO,(R0)
        MOV #MODDD1,R0    ;GET THE INTEGER RESULT.
        STD AC1,(R0)

        MOV R1,R2          ;SAVE THE DATA IN CASE OF ERROR.
        MOV R2,@#STMP3
        ADD #10,R2
        MOV R2,@#STMP4
        ADD #10,R2
        MOV R2,@#STMP5
        ADD #10,R2
        MOV R2,@#STMP6
    
```

```

10520 056044 012737 056300 001250      MOV      #MODDD0,@#STMP7
10521 056052 012737 056310 001252      MOV      #MODDD1,@#STMP10
10522 056060 010437 001254      MOV      R4,@#STMP11
10523 056064 016137 000062 001256      MOV      62(R1),@#STMP12
10524 056072 010537 001260      MOV      R5,@#STMP13
10525 056076 016137 000064 001262      MOV      64(R1),@#STMP14
10526
10527 056104 012702 056300      MOV      #MODDD0,R2      ;CHECK THE FRACTIONAL RESULT.
10528 056110 010103      MOV      R1,R3
10529 056112 062703 000020      ADD      #20,R3
10530 056116 012700 000004      MOV      #4,R0
10531 056122 022223      2$:      CMP      (R2)+,(R3)+
10532 056124 001023      BNE      10$      ;BRANCH IF INCORRECT.
10533 056126 077003      SOB      R0,2$
10534
10535 056130 012702 056310      MOV      #MODDD1,R2      ;CHECK THE INTEGER RESULT.
10536 056134 010103      MOV      R1,R3
10537 056136 062703 000030      ADD      #30,R3
10538 056142 012700 000004      MOV      #4,R0
10539 056146 022223      3$:      CMP      (R2)+,(R3)+
10540 056150 001013      BNE      15$      ;BRANCH IF INCORRECT.
10541 056152 077003      SOB      R0,3$
10542
10543
10544 056154 026104 000062      CMP      62(R1),R4      ;CHECK THE FPS.
10545 056160 001027      BNE      20$      ;BRANCH IF INCORRECT.
10546
10547 056162 026105 000064      CMP      64(R1),R5      ;CHECK THE FEC.
10548 056166 001033      BNE      25$
10549
10550 056170 000161 000074      9$:      JMP      74(R1)      ;RETURN.
10551
10552      ;FRACTIONAL ERROR.
10553 056174      10$:
10554 056174 104002      12$:      ERROR   2      ;THE ERROR WAS NOT ANTICIPATED SO
10555 056176 000774      BR      9$      ;REPORT THE INCORRECT FRACTION HERE.
10556
10557      ;INTEGER ERROR.
10558 056200 012702 056310      15$:      MOV      #MODDD1,R2      ;WAS THE INTEGER ERROR ANTICIPATED?
10559 056204 010103      MOV      R1,R3
10560 056206 062703 000050      ADD      #50,R3
10561 056212 012705 000004      MOV      #4,R5
10562 056216 022223      60$:      CMP      (R2)+,(R3)+
10563 056220 001005      BNE      17$      ;BRANCH IF NOT ANTICIPATED.
10564 056222 077503      SOB      R5,60$
10565 056224 010102      MOV      R1,R2      ;THE ERROR WAS ANTICIPATED SO RETURN
10566 056226 062702 000072      ADD      #72,R2      ;TO THE ERROR REPORT IN THE CALLING
10567      ;ROUTINE.
10568 056232 000112      JMP      (R2)
10569
10570 056234      16$:
10571 056234 104002      17$:      ERROR   2      ;THE ERROR WAS NOT ANTICIPATED SO REPORT
10572 056236 000754      BR      9$      ;THE INTEGER FAILURE HERE.
10573
10574      ;FPS INCORRECT.
10575 056240 010437 001254      20$:      MOV      R4,@#STMP11      ;REPORT INCORRECT FPS.
    
```

```

10576 056244 016137 000062 001256      MOV      62(R1),@#$TMP12
10577 056252 104002      21$:    ERROR  2
10578 056254 000745      BR      9$
10579
10580      :REPORT FEC ERROR.
10581 056256 010537 001260      25$:    MOV      R5,@#$TMP13
10582 056262 016137 000064 001262      MOV      64(R1),@#$TMP14
10583 056270 010102      MOV      R1,R2
10584 056272 062702 000066      ADD     #66,R2
10585 056276 000112      JMP     (R2)
10586
10587 056300 000000 000000 000000 MODDD0: .WORD  0,0,0,0
10588 056306 000000
10589
10590 056310 000000 000000 000000 MODDD1: .WORD  0,0,0,0
10591 056316 000000
10592
10593 056320      NNNDONE:
10594 056320 104413      RSETUP
10595
10596
10597
10598
10599
10600
10601
10602
10603
10604
10605
10606
10607 056322 000004
10608 056324
10609 056324 104414
10610 056326 012737 056342 000244 XT1:    LPERR
10611 056334 170227 000000      MOV     #XT1A,@#244 ;SET UP THE LOOP ON ERROR ADDRESS.
10612 056340 000401      STFPS  #0
10613 056342 104001      BR     XT2
10614
10615 056344 012700 177777 XT1A:   ERROR  1 ;SHOULD NOT TRAP
10616 056350 170127 000000 XT2:    MOV     #-1,R0
10617 056354 170200      LDFPS  #0
10618 056356 005700      STFPS  R0
10619 056360 001401      TST    R0
10620 056362 104001      BEQ    XT2A ;FPS IS NOT ZEROED
10621 056364 012700 057124 XT2A:   MOV     #XPAT0,R0
10622 056370 172440      LDF    -(R0),AC0
10623 056372 022700 057120      CMP    #XPAT0-4,R0
10624 056376 001401      BEQ    XT2B
10625 056400 104001      ERROR  1 ;RO WAS NOT DECR BY 4
10626 056402 170200      STFPS  R0
10627 056404 022700 000004 XT2B:   CMP    #4,R0 ;CHECK IF FZ IS SET?
10628 056410 001401      BEQ    XT3
10629 056412 104001      ERROR  1
10630
10631 056414 170127 000000 XT3:    LDFPS  #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?).

 :*TEST 60 MORE MICROCODES COVERAGE
 :*THIS TEST WILL PROVIDE ADDITIONAL MICRO-CODE LOCATIONS COVERAGE
 :*IN FPP1 AND FPP2.
 :*
 :*****

10632	056420	012700	057124		MOV	#XPAT0,R0	
10633	056424	174040			STF	AC0,-(R0)	
10634	056426	022700	057120		CMP	#XPAT0-4,R0	
10635	056432	001401			BEQ	XT3A	;R0 WAS NOT DECR BY 4
10636	056434	104001			ERROR	1	
10637	056436	170200		XT3A:	STFPS	R0	
10638	056440	005700			TST	R0	
10639	056442	001401			BEQ	XT4	
10640	056444	104001			ERROR	1	
10641							
10642	056446	170127	000000		LDFPS	#0	
10643	056452	012737	056476	000244	MOV	#XT4A,@#244	
10644	056460	170127	004000		LDFPS	#04000	;INTRPT ON UNDEFINED VARIABLE

10645	056464	172437	057124		LDF	@#XPATO,ACO	
10646	056470	174437	057154		DIVF	@#XPAT3,ACO	;GET UNDEFINED VARIABLE, -0
10647	056474	104001			ERROR	1	;MISSING INTERRUPT TO 244-0
10648	056476	170200		XT4A:	STFPS	R0	
10649	056500	022700	104004		CMP	#104004,R0	;CHECK: FER,FIUV,FZ ARE SET?
10650	056504	001401			BEQ	XT4B	
10651	056506	104001			ERROR	1	
10652	056510	012700	057114	XT4B:	MOV	#XBUF,R0	
10653	056514	174010			STF	ACO,(R0)	
10654	056516	005737	057114		TST	@#XBUF	
10655	056522	001401			BEQ	XT5	
10656	056524	104001			ERROR	1	;ACO SHOULD REMAIN ZEROED
10657							
10658	056526	012737	056546	000244	XT5:	MOV	#XT5A,@#244
10659	056534	170127	004000		LDFPS	#04000	;INTRPT ON UNDEFINED VARIBALE
10660	056540	177437	057154		LDCDF	@#XPAT3,ACO	;GET UNDEFINED VARIABLE, -0
10661	056544	104001			ERROR	1	;MISSING INTERRUPT TO 244-0
10662	056546	170200		XT5A:	STFPS	R0	
10663	056550	022700	104014		CMP	#104014,R0	;CHECK: FER,FIUV,FN,FZ ARE SET?
10664	056554	001401			BEQ	XT5B	
10665	056556	104001			ERROR	1	
10666	056560	012700	057114	XT5B:	MOV	#XBUF,R0	
10667	056564	174010			STF	ACO,(R0)	
10668	056566	005737	057114		TST	@#XBUF	
10669	056572	001401			BEQ	XT6	
10670	056574	104001			ERROR	1	;ACO SHOULD BE ZEROED
10671							
10672	056576			XT6:			
10673	056576	104414			LPERR		;SET UP THE LOOP ON ERROR ADDRESS.
10674	056600	012737	056624	000244	MOV	#XT6A,@#244	
10675	056606	170127	004000		LDFPS	#04000	;INTRPT ON UNDEFINED VARIBALE
10676	056612	172437	057124		LDF	@#XPATO,ACO	
10677	056616	172037	057154		ADDF	@#XPAT3,ACO	
10678	056622	104002			ERROR	2	;MISSING INTERRUPT TO 244
10679	056624	170200		XT6A:	STFPS	R0	
10680	056626	022700	104004		CMP	#104004,R0	;CHECK: FER,FIUV,FZ ARE SET?
10681	056632	001401			BEQ	XT6B	
10682	056634	104002			ERROR	2	
10683	056636	012700	057114	XT6B:	MOV	#XBUF,R0	
10684	056642	174010			STF	ACO,(R0)	
10685	056644	005737	057114		TST	@#XBUF	
10686	056650	001401			BEQ	XT7	
10687	056652	104002			ERROR	2	;ACO SHOULD BE ZEROED
10688							
10689	056654	170127	000000	XT7:	LDFPS	#0	
10690	056660	172437	057164		LDF	@#XPAT4,ACO	
10691	056664	175437	057214		STCFI	ACO,@#XPATO	
10692	056670	022737	000002	057214	CMP	#2,@#XPATO	;CHECK DATA
10693	056676	001401			BEQ	XT8	
10694	056700	104001			ERROR	1	
10695							
10696	056702	170127	000100	XT8:	LDFPS	#100	;SET FL
10697	056706	172437	057164		LDF	@#XPAT4,ACO	
10698	056712	175467	000276		STCFI	ACO,XPATO	
10699	056716	022737	000002	057216	CMP	#2,@#XPATO+2	
10700	056724	001401			BEQ	XT9	

```

10701 056726 104001          ERROR 1
10702
10703          ;START OF FPP2
10704 056730 170127 000000  XT9:  LDFPS #0
10705 056734 172437 057124      LDF @#XPAT0,ACO
10706 056740 172037 057164      ADDF @#XPAT4,ACO
10707 056744 170200          STFPS R0
10708 056746 005700          TST R0
10709 056750 001401          BEQ XT10
10710 056752 104002          ERROR 2
10711
10712 056754 170127 000000  XT10: LDFPS #0
10713 056760 172437 057164      LDF @#XPAT4,ACO
10714 056764 173037 057164      SUBF @#XPAT4,ACO
10715 056770 170200          STFPS R0
10716 056772 022700 000004      CMP #4,R0
10717 056776 001401          BEQ XT11
10718 057000 104002          ERROR 2
10719
10720 057002 170127 000000  XT11: LDFPS #0
10721 057006 172437 057164      LDF @#XPAT4,ACO
10722 057012 173437 057164      CMPF @#XPAT4,ACO
10723 057016 170200          STFPS R0
10724 057020 022700 000004      CMP #4,R0          ;CHECK IF FZ IS SET?
10725 057024 001401          BEQ XT12
10726 057026 104002          ERROR 2
10727
10728 057030 170127 000000  XT12: LDFPS #0
10729 057034 172437 057164      LDF @#XPAT4,ACO
10730 057040 174437 057144      DIVF @#XPAT2,ACO
10731 057044 012700 057114      MOV #XBUF,R0
10732 057050 174010          STF ACO,(R0)
10733 057052 022737 040176 057114  CMP #040176,@#XBUF ;CHECK DATA
10734 057060 001401          BEQ XT13
10735 057062 104002          ERROR 2
10736
10737 057064 170127 000000  XT13: LDFPS #0
10738 057070 172437 057174      LDF @#XPAT5,ACO
10739 057074 174437 057204      DIVF @#XPAT6,ACO
10740 057100 170200          STFPS R0
10741 057102 022700 000004      CMP #4,R0
10742 057106 001401          BEQ XT13B
10743 057110 104002          ERROR 2
10744 057112 000444          XT13B: BR XNEXT
10745
10746
10747 057114 000000 000000 000000 XBUF: .WORD 0,0,0,0
10748 057122 000000          XPAT0: .WORD 0,0,0,0
10749 057124 000000 000000 000000
10750 057132 000000          XPAT1: .WORD 1,1,1,1
10751 057134 000001 000001 000001
10752 057142 000001          XPAT2: .WORD 40401,0,0,0
10753 057144 040401 000000 000000
10754 057152 000000          XPAT3: .WORD 100000,0,0,0
10755 057154 100000 000000 000000
10756 057162 000600
    
```

10757	057164	040400	000000	000000	XPAT4:	.WORD	040400,0,0,0
10758	057172	000000					
10759	057174	000207	000000	000000	XPAT5:	.WORD	207,0,0,0
10760	057202	000000					
10761	057204	077007	000000	000000	XPAT6:	.WORD	77007,0,0,0
10762	057212	000000					
10763	057214	000000	000000	000000	XPAT0:	.WORD	0,0,0,0
10764	057222	000000					
10765							
10766	057224				XNEXT:		

```

10767 .SBTTL END OF PASS ROUTINE
10768
10769
10770
10771
10772
10773
10774
10775
10776
10777 057224 000004
10778 057226 005067 121650
10779 057232 005067 122044
10780 057236 005267 122062
10781 057242 042767 100000 122054
10782 057250 005327
10783 057252 000001
10784 057254 003045
10785 057256 012737
10786 057260 000001
10787 057262 057252
10788 057264 104401 057437
10789 057270 016746 122030
10790 057274 104405
10791 057276 104401 057434
10792 057302 013700 000042
10793 057306 001411
10794 057310 005046
10795 057312 012746 057320
10796 057316 000442
10797
10798 057320 000005
10799 057322 004710
10800 057324 000240
10801 057326 000240
10802 057330 000240
10803 057332 013737 000004 001232
10804 057340 012737 057356 000004
10805 057346 012737 000001 164000
10806 057354 000402
10807 057356 062706 000004 1$:
10808 057362 013737 001232 000004 2$:
10809 057370 104400
10810 057372 042716 000020
10811 057376 032777 010000 121534
10812 057404 001005
10813 057406 005167 000020
10814 057412 100402
10815 057414 052716 000020
10816 057420 012746 057426 1$:
10817 057424 000002
10818
10819 057426 000137
10820 057430 002160
10821 057432 000000
10822 057434 377 377 000

*****
*INCREMENT THE PASS NUMBER ($PASS)
*INDICATE END-OF-PROGRAM AFTER 1 PASS THRU THE PROGRAM
*TYPE 'END PASS #XXXXX' (WHERE XXXXX IS A DECIMAL NUMBER)
*IF SW12=1 INHIBIT TRACE TRAP
*IF THERE IS A MONITOR GO TO IT
*IF THERE ISN'T JUMP TO LOOP
*****
$EOP: SCOPE
      CLR $STNM ;ZERO THE TEST NUMBER
      CLR $TIMES ;ZERO THE NUMBER OF ITERATIONS
      INC $PASS ;INCREMENT THE PASS NUMBER
      BIC #100000,$PASS ;DON'T ALLOW A NEGATIVE PASS NUMBER
      DEC (PC)+ ;LOOP?
$EOPCT: .WORD 1 ;1 PASS FIRST TIME (QV)!!!
      BGT $DOAGN ;YES
      MOV (PC)+,@(PC)+ ;RESTORE COUNTER
$ENDCT: .WORD 1
      $EOPCT
      TYPE $SENDMG ;TYPE 'END PASS #'
      MOV $PASS,-(SP) ;SAVE PASS COUNT FOR TYPEOUT
      TYPDS ;TYPE PASS COUNT IN DECIMAL
      TYPE $ENULL ;TYPE A NULL CHARACTER STRING
$GET42: MOV @#42,R0 ;GET MONITOR ADDRESS
      BEQ DOAGIN ;BRANCH IF NO MONITOR
      CLR -(SP) ;INSURE THE 'T' BIT IS CLEAR
      MOV #$.CLR.T,-(SP) ;SETUP FOR AN RTI OR RTT
      BR $RTRN ;GO DO AN RTI OR RTT TO LOAD THE PSW
      ;WITH A CLEARED 'T' BIT
      $CLR.T: RESET ;CLEAR THE WORLD
$ENDAD: JSR PC,(R0) ;GO TO THE MONITOR
      NOP ;SAVE ROOM
      NOP ;FOR
      NOP ;ACT11
DOAGIN: MOV @#4,@$TMP0 ;SAVE CONTENTS OF LOCATION 4
      MOV #1$,@#4 ;SET UP INCASE OF TRAP
      MOV #1,@#164000 ;NOTIFY MULTI-TESTER
      BR 2$ ;NO TRAP SO DON'T RESET STACK
1$: ADD #4,SP ;RESET STACK AFTER TRAP
2$: MOV @$TMP0,@#4 ;RESTORE CONTENTS OF LOCATION 4
$DOAGN: TRAP ;PUSH OLD PSW AND PC ON STACK
      BIC #20,(SP) ;CLEAR THE 'T' BIT
      BIT #BIT12,@SWR ;RUN WITH TRACE TRAP?
      BNE 1$ ;BRANCH IF NO
      COM $TBIT ;IS IT TIME FOR TRACE TRAP
      BMI 1$ ;BRANCH IF NO
      BIS #20,(SP) ;SET TRACE TRAP
1$: MOV #$.LOOP,-(SP) ;RETURN HERE FROM RTI
$RTRN: RTI ;RETURN--THIS IS CHANGED TO AN 'RTT'
      ;IF IT IS A LEGAL INSTRUCTION
      ;RETURN TO TESTING
$.LOOP: JMP @(PC)+
$RTNAD: .WORD LOOP
$.TBIT: .WORD 0 ;'T' BIT STATE INDICATOR
$.ENULL: .BYTE -1,-1,0 ;NULL CHARACTER STRING
    
```



```

10823 057437 015 042412 042116 $ENDMG: .ASCIZ <15><12>/END PASS #/
10824 057444 050040 051501 020123
10825 057452 000043
10826
10827 .EVEN
10828 .SBTTL SCOPE HANDLER ROUTINE
10829
10830 ::*****
10831 ::THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
10832 ::AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
10833 ::AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
10834 ::THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
10835 ::*SW14=1 LOOP ON TEST
10836 ::*SW11=1 INHIBIT ITERATIONS
10837 ::*SW09=1 LOOP ON ERROR
10838 ::*SW08=1 LOOP ON TEST IN SWR<7:0>
10839 ::*CALL
10840 ::* SCOPE ;:SCOPE=IOT
10841 $SCOPE:
10842 057454 104407 CKSWR ;:TEST FOR CHANGE IN SOFT-SWR
10843 057456 032777 040000 121454 1$: BIT #BIT14,@SWR ;:LOOP ON PRESENT TEST?
10844 057464 001114 BNE $OVER ;:YES IF SW14=1
10845 :#####START OF CODE FOR THE XOR TESTER#####
10846 057466 000416 $XTSTR: BR 6$ ;:IF RUNNING ON THE 'XOR' TESTER CHANGE
10847 ;:THIS INSTRUCTION TO A 'NOP' (NOP=240)
10848 057470 013746 000004 MOV @#ERRVEC,-(SP) ;:SAVE THE CONTENTS OF THE ERROR VECTOR
10849 057474 012737 057514 000004 MOV #5$,@#ERRVEC ;:SET FOR TIMEOUT
10850 057502 005737 177060 TST @#177060 ;:TIME OUT ON XOR?
10851 057506 012637 000004 MOV (SP)+,@#ERRVEC ;:RESTORE THE ERROR VECTOR
10852 057512 000463 BR $SVLAD ;:GO TO THE NEXT TEST
10853 057514 022626 5$: CMP (SP)+,(SP)+ ;:CLEAR THE STACK AFTER A TIME OUT
10854 057516 012637 000004 MOV (SP)+,@#ERRVEC ;:RESTORE THE ERROR VECTOR
10855 057522 000423 BR 7$ ;:LOOP ON THE PRESENT TEST
10856 057524 6$:;#####END OF CODE FOR THE XOR TESTER#####
10857 057524 032777 000400 121406 BIT #BIT08,@SWR ;:LOOP ON SPEC. TEST?
10858 057532 001404 BEQ 2$ ;:BR IF NO
10859 057534 127767 121400 121340 CMPB @SWR,$TSTNM ;:ON THE RIGHT TEST? SWR<7:0>
10860 057542 001465 BEQ $OVER ;:BR IF YES
10861 057544 105767 121333 2$: TSTB $ERFLG ;:HAS AN ERROR OCCURRED?
10862 057550 001421 BEQ 3$ ;:BR IF NO
10863 057552 126767 121337 121323 CMPB $ERMAX,$ERFLG ;:MAX. ERRORS FOR THIS TEST OCCURRED?
10864 057560 101015 BHI 3$ ;:BR IF NO
10865 057562 032777 001000 121350 BIT #BIT09,@SWR ;:LOOP ON ERROR?
10866 057570 001404 BEQ 4$ ;:BR IF NO
10867 057572 016767 121312 121306 7$: MOV $LPERR,$LPADR ;:SET LOOP ADDRESS TO LAST SCOPE
10868 057600 000446 BR $OVER
10869 057602 105067 121275 4$: CLRB $ERFLG ;:ZERO THE ERROR FLAG
10870 057606 005067 121470 CLR $TIMES ;:CLEAR THE NUMBER OF ITERATIONS TO MAKE
10871 057612 000415 BR 1$ ;:ESCAPE TO THE NEXT TEST
10872 057614 032777 004000 121316 3$: BIT #BIT11,@SWR ;:INHIBIT ITERATIONS?
10873 057622 001011 BNE 1$ ;:BR IF YES
10874 057624 005767 121474 TST $PASS ;:IF FIRST PASS OF PROGRAM
10875 057630 001406 BEQ 1$ ;: INHIBIT ITERATIONS
10876 057632 005267 121246 INC $ICNT ;:INCREMENT ITERATION COUNT
10877 057636 026767 121440 121240 CMP $TIMES,$ICNT ;:CHECK THE NUMBER OF ITERATIONS MADE
10878 057644 002024 BGE $OVER ;:BR IF MORE ITERATION REQUIRED
    
```

```

10879 057646 012767 000001 121230 1$: MOV #1,$ICNT ;;REINITIALIZE THE ITERATION COUNTER
10880 057654 016767 000052 121420 MOV $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
10881 057662 105267 121214 $SVLAD: INCB $TSTNM ;;COUNT TEST NUMBERS
10882 057666 116767 121210 121426 MOVB $TSTNM,$TESTN ;;SET TEST NUMBER IN APT MAILBOX
10883 057674 011667 121206 MOV (SP),$LPADR ;;SAVE SCOPE LOOP ADDRESS
10884 057700 011667 121204 MOV (SP),$LPERR ;;SAVE ERROR LOOP ADDRESS
10885 057704 005067 121374 CLR $ESCAPE ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
10886 057710 112767 000001 121177 MOVB #1,$ERMAX ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
10887 057716 016777 121160 121216 $OVER: MOV $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
10888 057724 016716 121156 MOV $LPADR,(SP) ;;FUDGE RETURN ADDRESS
10889 057730 000002 RTI ;;FIXES PS
10890 057732 000001 $MXCNT: 1 ;;MAX. NUMBER OF ITERATIONS
    
```

.SBTTL ERROR HANDLER ROUTINE

```

10891
10892
10893
10894
10895
10896
10897
10898
10899
10900
10901
10902
10903
10904
10905
    *****
    *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
    * ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER
    
```

```

10906 057734 $ERROR:
10907 057734 104407 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
10908 057736 105267 121141 7$: INCB $ERFLG ;;SET THE ERROR FLAG
10909 057742 001775 BEQ 7$ ;;DON'T LET THE FLAG GO TO ZERO
10910 057744 016777 121132 121170 MOV $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
10911 057752 032777 002000 121160 BIT #BIT10,@SWR ;;BELL ON ERROR?
10912 057760 001402 BEQ 1$ ;;NO - SKIP
10913 057762 104401 001306 TYPE $BELL ;;RING BELL
10914 057766 005267 121120 1$: INC $ERTTL ;;COUNT THE NUMBER OF ERRORS
10915 057772 011667 121120 MOV (SP),$ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
10916 057776 162767 000002 121112 SUB #2,$ERRPC
10917 060004 117767 121106 121102 MOVB @ $ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
10918 060012 032777 020000 121120 BIT #BIT13,@SWR ;;SKIP TYPEOUT IF SET
10919 060020 001004 BNE 20$ ;;SKIP TYPEOUTS
10920 060022 004767 002314 JSR PC,ERTYPE ;;GO TO USER ERROR ROUTINE
10921 060026 104401 001313 TYPE $CRLF
10922 060032 20$:
10923 060032 122767 000001 121276 CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
10924 060040 001007 BNE 2$ ;;NO,SKIP APT ERROR REPORT
10925 060042 116767 121046 000004 MOVB $ITEMB,21$ ;;SET ITEM NUMBER AS ERROR NUMBER
10926 060050 004767 001126 JSR PC,$ATY4 ;;REPORT FATAL ERROR TO APT
10927 060054 000 .BYTE 0
10928 060055 000 .BYTE 0
10929 060056 000777 21$: BR 22$ ;;APT ERROR LOOP
10930 060060 005777 121054 22$: TST @SWR ;;HALT ON ERROR
10931 060064 100002 2$: BPL 3$ ;;SKIP IF CONTINUE
10932 060066 000000 HALT ;;HALT ON ERROR!
10933 060070 104407 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
10934 060072 032777 001000 121040 3$: BIT #BIT09,@SWR ;;LOOP ON ERROR SWITCH SET?
    
```

```

10935 060100 001402          BEQ      4$          ;;BR IF NO
10936 060102 016716 121002  MOV      $LPERR,(SP) ;;FUDGE RETURN FOR LOOPING
10937 060106 005767 121172  4$:    TST      $ESCAPE  ;;CHECK FOR AN ESCAPE ADDRESS
10938 060112 001402          BEQ      5$          ;;BR IF NONE
10939 060114 016716 121164  MOV      $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
10940 060120
10941 060120 022737 057322 000042 5$:    CMP      #SENDAD,@#42 ;;ACT-11 AUTO-ACCEPT?
10942 060126 001001          BNE      6$          ;;BRANCH IF NO
10943 060130 000000          HALT
10944 060132
10945 060132 000002          6$:    RTI           ;;RETURN
    
```

.SRTL 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
    
```

\$SAVREG:

```

MOV      R0,-(SP)      ;;PUSH R0 ON STACK
MOV      R1,-(SP)      ;;PUSH R1 ON STACK
MOV      R2,-(SP)      ;;PUSH R2 ON STACK
MOV      R3,-(SP)      ;;PUSH R3 ON STACK
MOV      R4,-(SP)      ;;PUSH R4 ON STACK
MOV      R5,-(SP)      ;;PUSH R5 ON STACK
MOV      22(SP),-(SP)  ;;SAVE PS OF MAIN FLOW
MOV      22(SP),-(SP)  ;;SAVE PC OF MAIN FLOW
MOV      22(SP),-(SP)  ;;SAVE PS OF CALL
MOV      22(SP),-(SP)  ;;SAVE PC OF CALL
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
    
```

```

10964 060134
10965 060134 010046
10966 060136 010146
10967 060140 010246
10968 060142 010346
10969 060144 010446
10970 060146 010546
10971 060150 016646 000022
10972 060154 016646 000022
10973 060160 016646 000022
10974 060164 016646 000022
10975 060170 000002
10976
10977
10978
10979
10980 060172
10981 060172 012666 000022
10982 060176 012666 000022
10983 060202 012666 000022
10984 060206 012666 000022
10985 060212 012605
10986 060214 012604
10987 060216 012603
10988 060220 012602
10989 060222 012601
10990 060224 012600
    
```

```

10991 060226 000002          RTI
10992
10993          .SBTTL TYPE ROUTINE
10994
10995          ::*****
10996          ::*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
10997          ::*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
10998          ::*NOTE1:          $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
10999          ::*NOTE2:          $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
11000          ::*NOTE3:          $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
11001          ::*
11002          ::*CALL:
11003          ::*1) USING A TRAP INSTRUCTION
11004          ::*          TYPE          ,MESADR          ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
11005          ::*OR
11006          ::*          TYPE
11007          ::*          MESADR
11008          ::*
11009
11010 060230 105767 120723      $TYPE:  TSTB          $TPFLG          ;;IS THERE A TERMINAL?
11011 060234 100002          BPL          1$          ;;BR IF YES
11012 060236 000000          HALT          ;;HALT HERE IF NO TERMINAL
11013 060240 000430          BR          3$          ;;LEAVE
11014 060242 010046          1$:  MOV          R0,-(SP)          ;;SAVE R0
11015 060244 017600 000002  MOV          @2(SP),R0          ;;GET ADDRESS OF ASCIZ STRING
11016 060250 122767 000001 121060  CMPB          #APTENV,$ENV          ;;RUNNING IN APT MODE
11017 060256 001011          BNE          62$          ;;NO,GO CHECK FOR APT CONSOLE
11018 060260 132767 000100 121051  BITB          #APTSPOOL,$ENVM          ;;SPOOL MESSAGE TO APT
11019 060266 001405          BEQ          62$          ;;NO,GO CHECK FOR CONSOLE
11020 060270 010067 000004          MOV          R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
11021 060274 004767 000672          JSR          PC,$ATY3          ;;SPOOL MESSAGE TO APT
11022 060300 000000          .WORD          0          ;;MESSAGE ADDRESS
11023 060302 132767 000040 121027  61$:  BITB          #APTCSUP,$ENVM          ;;APT CONSOLE SUPPRESSED
11024 060310 001003          BNE          60$          ;;YES,SKIP TYPE OUT
11025 060312 112046          2$:  MOVB          (R0)+,-(SP)          ;;PUSH CHARACTER TO BE TYPED ONTO STACK
11026 060314 001005          BNE          4$          ;;BR IF IT ISN'T THE TERMINATOR
11027 060316 005726          TST          (SP)+          ;;IF TERMINATOR POP IT OFF THE STACK
11028 060320 012600          60$:  MOV          (SP)+,R0          ;;RESTORE R0
11029 060322 062716 000002          3$:  ADD          #2,(SP)          ;;ADJUST RETURN PC
11030 060326 000002          RTI          ;;RETURN
11031 060330 122716 000011          4$:  CMPB          #HT,(SP)          ;;BRANCH IF <HT>
11032 060334 001430          BEQ          8$          ;;BRANCH IF NOT <CRLF>
11033 060336 122716 000200          CMPB          #CRLF,(SP)          ;;BRANCH IF NOT <CRLF>
11034 060342 001006          BNE          5$          ;;BRANCH IF NOT <CRLF>
11035 060344 005726          TST          (SP)+          ;;POP <CR><LF> EQUIV
11036 060346 104401          TYPE          ;;TYPE A CR AND LF
11037 060350 001313          $CRLF
11038 060352 105067 000130          CLRB          $CHARCNT          ;;CLEAR CHARACTER COUNT
11039 060356 000755          BR          2$          ;;GET NEXT CHARACTER
11040 060360 004767 000056          5$:  JSR          PC,$TYPEC          ;;GO TYPE THIS CHARACTER
11041 060364 126726 120566          6$:  CMPB          $FILLC,(SP)+          ;;IS IT TIME FOR FILLER CHARS.?
11042 060370 001350          BNE          2$          ;;IF NO GO GET NEXT CHAR.
11043 060372 016746 120556          MOV          $NULL,-(SP)          ;;GET # OF FILLER CHARS. NEEDED
11044          ;;AND THE NULL CHAR.
11045 060376 105366 000001          7$:  DECB          1(SP)          ;;DOES A NULL NEED TO BE TYPED?
11046 060402 002770          BLT          6$          ;;BR IF NO--GO POP THE NULL OFF OF STACK
    
```

```

11047 060404 004767 000032          JSR    PC,$TYPEC    ;;GO TYPE A NULL
11048 060410 105367 000072          DECB   $CHARCNT     ;;DO NOT COUNT AS A COUNT
11049 060414 000770                   BR     7$           ;;LOOP
11050
11051          ;HORIZONTAL TAB PROCESSOR
11052
11053 060416 112716 000040          8$:    MOVB   #' (SP)    ;;REPLACE TAB WITH SPACE
11054 060422 004767 000014          9$:    JSR    PC,$TYPEC  ;;TYPE A SPACE
11055 060426 132767 000007 000052  BITB   #7,$CHARCNT   ;;BRANCH IF NOT AT
11056 060434 001372                   BNE    9$           ;;TAB STOP
11057 060436 005726                   TST   (SP)+        ;;POP SPACE OFF STACK
11058 060440 000724                   BR     2$           ;;GET NEXT CHARACTER
11059 060442 105777 120502          $TYPEC: TSTB   @ $TPS   ;;WAIT UNTIL PRINTER IS READY
11060 060446 100375                   BPL   $TYPEC
11061 060450 116677 000002 120474  MOVB   2(SP),@ $TPB  ;;LOAD CHAR TO BE TYPED INTO DATA REG.
11062 060456 122766 000015 000002  CMPB   #CR,2(SP)    ;;IS CHARACTER A CARRIAGE RETURN?
11063 060464 001003                   BNE    1$           ;;BRANCH IF NO
11064 060466 105067 000014          CLRB   $CHARCNT     ;;YES--CLEAR CHARACTER COUNT
11065 060472 000406                   BR     $TYPEX      ;;EXIT
11066 060474 122766 000012 000002  1$:    CMPB   #LF,2(SP)  ;;IS CHARACTER A LINE FEED?
11067 060502 001402                   BEQ   $TYPEX      ;;BRANCH IF YES
11068 060504 105227                   INCB   (PC)+       ;;COUNT THE CHARACTER
11069 060506 000000          $CHARCNT: .WORD 0   ;;CHARACTER COUNT STORAGE
11070 060510 000207          $TYPEX: RTS    PC
11071
11072
11073          .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
11074
11075          ;*****
11076          ;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
11077          ;OCTAL (ASCII) NUMBER AND TYPE IT.
11078          ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
11079          ;*CALL:
11080          ;*      MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
11081          ;*      TYPOS   ;;CALL FOR TYPEOUT
11082          ;*      .BYTE  N              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
11083          ;*      .BYTE  M              ;;M=1 OR 0
11084          ;*                                  ;;1=TYPE LEADING ZEROS
11085          ;*                                  ;;0=SUPPRESS LEADING ZEROS
11086          ;*
11087          ;*$STYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
11088          ;*$TYPOS OR $TYPOC
11089          ;*CALL:
11090          ;*      MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
11091          ;*      TYPON   ;;CALL FOR TYPEOUT
11092          ;*
11093          ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
11094          ;*CALL:
11095          ;*      MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
11096          ;*      TYPOC   ;;CALL FOR TYPEOUT
11097          ;*
11098          ;*$TYPOS: MOV     @ (SP),-(SP)  ;;PICKUP THE MODE
11099          ;*      MOVB   1(SP),$OFILL   ;;LOAD ZERO FILL SWITCH
11100          ;*      MOVB   (SP)+,$OMODE+1 ;;NUMBER OF DIGITS TO TYPE
11101          ;*      ADD    #2,(SP)       ;;ADJUST RETURN ADDRESS
11102          ;*      BR     $STYPON
    
```

11103	060536	112767	000001	000171	\$TYPOC: MOV	#1,\$OFILL	::SET THE ZERO FILL SWITCH
11104	060544	112767	000006	000165	MOV	#6,\$OMODE+1	::SET FOR SIX(6) DIGITS
11105	060552	112767	000005	000154	\$TYPON: MOV	#5,\$OCNT	::SET THE ITERATION COUNT
11106	060560	010346			MOV	R3,-(SP)	::SAVE R3
11107	060562	010446			MOV	R4,-(SP)	::SAVE R4
11108	060564	010546			MOV	R5,-(SP)	::SAVE R5
11109	060566	116704	000145		MOV	\$OMODE+1,R4	::GET THE NUMBER OF DIGITS TO TYPE
11110	060572	005404			NEG	R4	
11111	060574	062704	000006		ADD	#6,R4	::SUBTRACT IT FOR MAX. ALLOWED
11112	060600	110467	000132		MOV	R4,\$OMODE	::SAVE IT FOR USE
11113	060604	116704	000125		MOV	\$OFILL,R4	::GET THE ZERO FILL SWITCH
11114	060610	016605	000012		MOV	12(SP),R5	::PICKUP THE INPUT NUMBER
11115	060614	005003			CLR	R3	::CLEAR THE OUTPUT WORD
11116	060616	006105			1\$: ROL	R5	::ROTATE MSB INTO 'C'
11117	060620	000404			BR	3\$::GO DO MSB
11118	060622	006105			2\$: ROL	R5	::FORM THIS DIGIT
11119	060624	006105			ROL	R5	
11120	060626	006105			ROL	R5	
11121	060630	010503			MOV	R5,R3	
11122	060632	006103			3\$: ROL	R3	::GET LSB OF THIS DIGIT
11123	060634	105367	000076		DECB	\$OMODE	::TYPE THIS DIGIT?
11124	060640	100016			BPL	7\$::BR IF NO
11125	060642	042703	177770		BIC	#177770,R3	::GET RID OF JUNK
11126	060646	001002			BNE	4\$::TEST FOR 0
11127	060650	005704			TST	R4	::SUPPRESS THIS 0?
11128	060652	001403			BEQ	5\$::BR IF YES
11129	060654	005204			4\$: INC	R4	::DON'T SUPPRESS ANYMORE 0'S
11130	060656	052703	000060		BIS	#'0,R3	::MAKE THIS DIGIT ASCII
11131	060662	052703	000040		5\$: BIS	#',R3	::MAKE ASCII IF NOT ALREADY
11132	060666	110367	000040		MOV	R3,8\$::SAVE FOR TYPING
11133	060672	104401	060732		TYPE	8\$::GO TYPE THIS DIGIT
11134	060676	105367	000032		7\$: DECB	\$OCNT	::COUNT BY 1
11135	060702	003347			BGT	2\$::BR IF MORE TO DO
11136	060704	002402			BLT	6\$::BR IF DONE
11137	060706	005204			INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
11138	060710	000744			BR	2\$::GO DO THE LAST DIGIT
11139	060712	012605			6\$: MOV	(SP)+,R5	::RESTORE R5
11140	060714	012604			MOV	(SP)+,R4	::RESTORE R4
11141	060716	012603			MCV	(SP)+,R3	::RESTORE R3
11142	060720	016666	000002	000004	MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
11143	060726	012616			MOV	(SP)+,(SP)	
11144	060730	000002			RTI		::RETURN
11145	060732	000			8\$: .BYTE	0	::STORAGE FOR ASCII DIGIT
11146	060733	000			.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
11147	060734	000			\$OCNT: .BYTE	0	::OCTAL DIGIT COUNTER
11148	060735	000			\$OFILL: .BYTE	0	::ZERO FILL SWITCH
11149	060736	000000			\$OMODE: .WORD	0	::NUMBER OF DIGITS TO TYPE

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

11150
 11151
 11152
 11153
 11154
 11155
 11156
 11157
 11158

```

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

```

11159          : *CALL:
11160          : *      MOV      NUM,-(SP)      ::PUT THE BINARY NUMBER ON THE STACK
11161          : *      TYPDS                    ::GO TO THE ROUTINE
11162
11163          $TYPDS:
11164 060740      MOV      R0,-(SP)      ::PUSH R0 ON STACK
11165 060740 010046  MOV      R1,-(SP)      ::PUSH R1 ON STACK
11166 060742 010146  MOV      R2,-(SP)      ::PUSH R2 ON STACK
11167 060744 010246  MOV      R3,-(SP)      ::PUSH R3 ON STACK
11168 060746 010346  MOV      R5,-(SP)      ::PUSH R5 ON STACK
11169 060750 010546  MOV      #20200,-(SP)  ::SET BLANK SWITCH AND SIGN
11170 060752 012746 020200  MOV      20(SP),R5    ::GET THE INPUT NUMBER
11171 060756 016605 000020  BPL      1$          ::BR IF INPUT IS POS.
11172 060762 100004      NEG      R5          ::MAKE THE BINARY NUMBER POS.
11173 060764 005405      MOVB    #'-,1(SP)   ::MAKE THE ASCII NUMBER NEG.
11174 060766 112766 000055 000001 1$:  CLR      R0          ::ZERO THE CONSTANTS INDEX
11175 060774 005000      MOV     #$DBLK,R3   ::SETUP THE OUTPUT POINTER
11176 060776 012703 061154  MOVB    #' ,(R3)+    ::SET THE FIRST CHARACTER TO A BLANK
11177 061002 112723 000040 2$:  CLR      R2          ::CLEAR THE BCD NUMBER
11178 061006 005002      MOV     $DTBL(R0),R1  ::GET THE CONSTANT
11179 061010 016001 061144 3$:  SUB      R1,R5       ::FORM THIS BCD DIGIT
11180 061014 160105      BLT     4$          ::BR IF DONE
11181 061016 002402      INC     R2          ::INCREASE THE BCD DIGIT BY 1
11182 061020 005202      BR      3$
11183 061022 000774      ADD     R1,R5       ::ADD BACK THE CONSTANT
11184 061024 060105 4$:  TST     R2          ::CHECK IF BCD DIGIT=0
11185 061026 005702      BNE     5$          ::FALL THROUGH IF 0
11186 061030 001002      TSTB   (SP)        ::STILL DOING LEADING 0'S?
11187 061032 105716      BMI     7$          ::BR IF YES
11188 061034 100407      BMI     7$
11189 061036 106316 5$:  ASLB   (SP)        ::MSD?
11190 061040 103003      BCC     6$          ::BR IF NO
11191 061042 116663 000001 177777 6$:  MOVB   1(SP),-1(R3)  ::YES--SET THE SIGN
11192 061044 052702 000060 7$:  BIS    #'0,R2       ::MAKE THE BCD DIGIT ASCII
11193 061046 052702 000040 7$:  BIS    #' ,R2       ::MAKE IT A SPACE IF NOT ALREADY A DIGIT
11194 061060 110223      MOVB   R2,(R3)+     ::PUT THIS CHARACTER IN THE OUTPUT BUFFER
11195 061062 005720      TST    (R0)+        ::JUST INCREMENTING
11196 061064 020027 000010      CMP    R0,#10      ::CHECK THE TABLE INDEX
11197 061070 002746      BLT    2$          ::GO DO THE NEXT DIGIT
11198 061072 003002      BGT    8$          ::GO TO EXIT
11199 061074 010502      MOV    R5,R2       ::GET THE LSD
11200 061076 000764 8$:  BR     6$          ::GO CHANGE TO ASCII
11201 061100 105726      TSTB  (SP)+        ::WAS THE LSD THE FIRST NON-ZERO?
11202 061102 100003 9$:  BPL    9$          ::BR IF NO
11203 061104 116663 177777 177776 9$:  MOVB  -1(SP),-2(R3)  ::YES--SET THE SIGN FOR TYPING
11204 061112 105013      CLRB  (R3)         ::SET THE TERMINATOR
11205 061114 012605      MOV   (SP)+,R5     ::POP STACK INTO R5
11206 061116 012603      MOV   (SP)+,R3     ::POP STACK INTO R3
11207 061120 012602      MOV   (SP)+,R2     ::POP STACK INTO R2
11208 061122 012601      MOV   (SP)+,R1     ::POP STACK INTO R1
11209 061124 012600      MOV   (SP)+,R0     ::POP STACK INTO R0
11210 061126 104401 061154      TYPE  $DBLK        ::NOW TYPE THE NUMBER
11211 061132 016666 000002 000004      MOV   2(SP),4(SP)  ::ADJUST THE STACK
11212 061140 012616      MOV   (SP)+,(SP)
11213 061142 000002      RTI
11214 061144 023420      $DTBL: 10000.
          1000.
    
```

```

11215 061150 000144          100.
11216 061152 000012          10.
11217 061154 000004          $DBLK: .BLKW 4
11218
11219          .SBTTL APT COMMUNICATIONS ROUTINE
11220
11221          ::*****
11222 061164 112767 000001 000236 $ATY1:  MOVB  #1,$FFLG          ;;TO REPORT FATAL ERROR
11223 061172 112767 000001 000226 $ATY3:  MOVB  #1,$MFLG          ;;TO TYPE A MESSAGE
11224 061200 000403
11225 061202 112767 000001 000220 $ATY4:  MOVB  #1,$FFLG          ;;TO ONLY REPORT FATAL ERROR
11226 061210
11227 061210 010046          MOV      R0,-(SP)          ;;PUSH R0 ON STACK
11228 061212 010146          MOV      R1,-(SP)          ;;PUSH R1 ON STACK
11229 061214 105767 000206          TSTB    $MFLG          ;;SHOULD TYPE A MESSAGE?
11230 061220 001450          BEQ     5$              ;;IF NOT: BR
11231 061222 122767 000001 120106          CMPB    #APTENV,$ENV          ;;OPERATING UNDER APT?
11232 061230 001031          BNE     3$              ;;IF NOT: ER
11233 061232 132767 000100 120077          BITB    #APTSPOOL,$ENVM      ;;SHOULD SPOOL MESSAGES?
11234 061240 001425          BEQ     3$              ;;IF NOT: BR
11235 061242 017600 000004          MOV     @4(SP),R0          ;;GET MESSAGE ADDR.
11236 061246 062766 000002 000004          ADD     #2,4(SP)          ;;BUMP RETURN ADDR.
11237 061254 005767 120036          1$:    TST     $MSGTYPE          ;;SEE IF DONE W/ LAST XMISSION?
11238 061260 001375          BNE     1$              ;;IF NOT: WAIT
11239 061262 010067 120044          MOV     R0,$MSGAD          ;;PUT ADDR IN MAILBOX
11240 061266 105720          2$:    TSTB    (R0)+          ;;FIND END OF MESSAGE
11241 061270 001376          BNE     2$
11242 061272 166700 120034          SUB     $MSGAD,R0          ;;SUB START OF MESSAGE
11243 061276 006200          ASR     R0              ;;GET MESSAGE LGTH IN WORDS
11244 061300 010067 120030          MOV     R0,$MSGGLT          ;;PUT LENGTH IN MAILBOX
11245 061304 012767 000004 120004          MOV     #4,$MSGTYPE          ;;TELL APT TO TAKE MSG.
11246 061312 000413          BR      5$
11247 061314 017667 000004 000016 3$:    MOV     @4(SP),4$          ;;PUT MSG ADDR IN JSR LINKAGE
11248 061322 062766 000002 000004          ADD     #2,4(SP)          ;;BUMP RETURN ADDRESS
11249 061330 016746 116442          MOV     177776,-(SP)       ;;PUSH 177776 ON STACK
11250 061334 004767 176670          JSR     PC,$TYPE          ;;CALL TYPE MACRO
11251 061340 000000          4$:    .WORD    0
11252 061342          5$:
11253 061342 105767 000062          10$:   TSTB    $FFLG          ;;SHOULD REPORT FATAL ERROR?
11254 061346 001416          BEQ     12$            ;;IF NOT: BR
11255 061350 005767 117762          TST     $ENV            ;;RUNNING UNDER APT?
11256 061354 001413          BEQ     12$            ;;IF NOT: BR
11257 061356 005767 117734          11$:   TST     $MSGTYPE          ;;FINISHED LAST MESSAGE?
11258 061362 001375          BNE     11$            ;;IF NOT: WAIT
11259 061364 017667 000004 117726          MOV     @4(SP),$FATAL      ;;GET ERROR #
11260 061372 062766 000002 000004          ADD     #2,4(SP)          ;;BUMP RETURN ADDR.
11261 061400 005267 117712          INC     $MSGTYPE          ;;TELL APT TO TAKE ERROR
11262 061404 105067 000020          12$:   CLRB    $FFLG          ;;CLEAR FATAL FLAG
11263 061410 105067 000013          CLRB    $LFLG          ;;CLEAR LOG FLAG
11264 061414 105067 000006          CLRB    $MFLG          ;;CLEAR MESSAGE FLAG
11265 061420 012601          MOV     (SP)+,R1          ;;POP STACK INTO R1
11266 061422 012600          MOV     (SP)+,R0          ;;POP STACK INTO R0
11267 061424 000207          RTS     PC              ;;RETURN
11268 061426          000          $MFLG: .BYTE    0          ;;MESSG. FLAG
11269 061427          000          $LFLG: .BYTE    0          ;;LOG FLAG
11270 061430          000          $FFLG: .BYTE    0          ;;FATAL FLAG
    
```



```

11271      061432
11272      000200
11273      000001
11274      000100
11275      000040
11276
11277      .SBTTL  TTY INPUT ROUTINE
11278
11279      ::*****
11280      .ENABL  LSB
11281
11282      ::*****
11283      ::*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
11284      ::*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
11285      ::*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
11286      ::*WHEN OPERATING IN TTY FLAG MODE.
11287 061432 022767 000176 117500 $CKSWR:  CMP    #SWREG,SWR      :: IS THE SOFT-SWR SELECTED?
11288 061440 001074          BNE    15$              :: BRANCH IF NO
11289 061442 105777 117476          TSTB   @STKS             :: CHAR THERE?
11290 061446 100071          BPL    15$              :: IF NO, DON'T WAIT AROUND
11291 061450 117746 117472          MOVB  @STKB,-(SP)       :: SAVE THE CHAR
11292 061454 042716 177600          BIC   #^C177,(SP)     :: STRIP-OFF THE ASCII
11293 061460 022726 000007          CMP   #7,(SP)+        :: IS IT A CONTROL G?
11294 061464 001062          BNE   15$              :: NO, RETURN TO USER
11295 061466 126727 117442 000001  CMPB  $AUTOB,#1       :: ARE WE RUNNING IN AUTO-MODE?
11296 061474 001456          BEQ   15$              :: BRANCH IF YES
11297
11298 061476 104401 062041          TYPE  ,SCNTLG         :: ECHO THE CONTROL-G (^G)
11299 061502 104401 062046          $GTSWR: TYPE ,SMSWR      :: TYPE CURRENT CONTENTS
11300 061506 016746 116464          MOV   SWREG,-(SP)    :: SAVE SWREG FOR TYPEOUT
11301 061512 104402          TYPOC                :: GO TYPE--OCTAL ASCII(ALL DIGITS)
11302 061514 104401 062057          TYPE  ,SMNEW         :: PROMPT FOR NEW SWR
11303 061520 005046          19$:  CLR   -(SP)      :: CLEAR COUNTER
11304 061522 005046          CLR   -(SP)          :: THE NEW SWR
11305 061524 105777 117414          7$:   TSTB  @STKS     :: CHAR THERE?
11306 061530 100375          BPL   7$              :: IF NOT TRY AGAIN
11307
11308 061532 117746 117410          MOVB  @STKB,-(SP)    :: PICK UP CHAR
11309 061536 042716 177600          BIC   #^C177,(SP)   :: MAKE IT 7-BIT ASCII
11310
11311
11312
11313 061542 021627 000025          9$:   CMP   (SP),#25   :: IS IT A CONTROL-U?
11314 061546 001005          BNE   10$            :: BRANCH IF NOT
11315 061550 104401 062034          TYPE  ,SCNTLU        :: YES, ECHO CONTROL-U (^U)
11316 061554 062706 000006          20$:  ADD   #6,SP     :: IGNORE PREVIOUS INPUT
11317 061560 000757          BR    19$            :: LET'S TRY IT AGAIN
11318
11319
11320 061562 021627 000015          10$:  CMP   (SP),#15   :: IS IT A <CR>?
11321 061566 001022          BNE   16$            :: BRANCH IF NO
11322 061570 005766 000004          TST   4(SP)         :: YES, IS IT THE FIRST CHAR?
11323 061574 001403          BEQ   11$            :: BRANCH IF YES
11324 061576 016677 000002 117334  MOV   2(SP),@SWR     :: SAVE NEW SWR
11325 061604 062706 000006          11$:  ADD   #6,SP     :: CLEAR UP STACK
11326 061610 104401 001313          14$:  TYPE  ,$CRLF    :: ECHO <CR> AND <LF>
    
```

11327	061614	126727	117315	000001		CMPB	\$INTAG,#1	::RE-ENABLE TTY KBD INTERRUPTS?
11328	061622	001003				BNE	15\$::BRANCH IF NOT
11329	061624	012777	000100	117312		MOV	#100,@\$TKS	::RE-ENABLE TTY KBD INTERRUPTS
11330	061632	000002			15\$:	RTI		::RETURN
11331	061634	004767	176602		16\$:	JSR	PC,\$TYPEC	::ECHO CHAR
11332	061640	021627	000060			CMP	(SP),#60	::CHAR < 0?
11333	061644	002420				BLT	18\$::BRANCH IF YES
11334	061646	021627	000067			CMP	(SP),#67	::CHAR > 7?
11335	061652	003015				BGT	18\$::BRANCH IF YES
11336	061654	042726	000060			BIC	#60,(SP)+	::STRIP-OFF ASCII
11337	061660	005766	000002			TST	2(SP)	::IS THIS THE FIRST CHAR
11338	061664	001403				BEQ	17\$::BRANCH IF YES
11339	061666	006316				ASL	(SP)	::NO, SHIFT PRESENT
11340	061670	006316				ASL	(SP)	::CHAR OVER TO MAKE
11341	061672	006316				ASL	(SP)	::ROOM FOR NEW ONE.
11342	061674	005266	000002		17\$:	INC	2(SP)	::KEEP COUNT OF CHAR
11343	061700	056616	177776			BIS	-2(SP),(SP)	::SET IN NEW CHAR
11344	061704	000707				BR	7\$::GET THE NEXT ONE
11345	061706	104401	001312		18\$:	TYPE	\$QUES	::TYPE ?<CR><LF>
11346	061712	000720				BR	20\$::SIMULATE CONTROL-U
11347						.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
 *

11358	061714	011646				\$RDCHR: MOV	(SP),-(SP)	::PUSH DOWN THE PC
11359	061716	016666	000004	000002		MOV	4(SP),2(SP)	::SAVE THE PS
11360	061724	105777	117214		1\$:	TSTB	@\$TKS	::WAIT FOR
11361	061730	100375				BPL	1\$::A CHARACTER
11362	061732	117766	117210	000004		MOVB	@\$TKB,4(SP)	::READ THE TTY
11363	061740	042766	177600	000004		BIC	#^C<177>,4(SP)	::GET RID OF JUNK IF ANY
11364	061746	026627	000004	000023		CMP	4(SP),#23	::IS IT A CONTROL-S?
11365	061754	001013				BNE	3\$::BRANCH IF NO
11366	061756	105777	117162		2\$:	TSTB	@\$TKS	::WAIT FOR A CHARACTER
11367	061762	100375				BPL	2\$::LOOP UNTIL ITS THERE
11368	061764	117746	117156			MOVB	@\$TKB,-(SP)	::GET CHARACTER
11369	061770	042716	177600			BIC	#^C<177>,(SP)	::MAKE IT 7-BIT ASCII
11370	061774	022627	000021			CMP	(SP)+,#21	::IS IT A CONTROL-Q?
11371	062000	001366				BNE	2\$::IF NOT DISCARD IT
11372	062002	000750				BR	1\$::YES, RESUME
11373	062004	026627	000004	000140	3\$:	CMP	4(SP),#140	::IS IT UPPER CASE?
11374	062012	002407				BLT	4\$::BRANCH IF YES
11375	062014	026627	000004	000175		CMP	4(SP),#175	::IS IT A SPECIAL CHAR?
11376	062022	003003				BGT	4\$::BRANCH IF YES
11377	062024	042766	000040	000004		BIC	#40,4(SP)	::MAKE IT UPPER CASE
11378	062032	000002			4\$:	RTI		::GO BACK TO USER
11379	062034	052536	005015	000		\$CNTLU: .ASCIZ	/^U/<15><12>	::CONTROL 'U'
11380	062041	136	006507	000012		\$CNTLG: .ASCIZ	/^G/<15><12>	::CONTROL 'G'
11381	062046	005015	053523	020122		\$MSWR: .ASCIZ	<15><12>/SWR = /	
11382	062054	020075	000					

11383 062057 040 047040 053505 \$MNEW: .ASCIZ / NEW = /
 11384 062064 036440 000040
 11385
 11386
 11387
 11388
 11389
 11390
 11391
 11392
 11393
 11394 062070 010046
 11395 062072 016600 000002
 11396 062076 005740
 11397 062100 111000
 11398 062102 006300
 11399 062104 016000 062124
 11400 062110 000200
 11401
 11402
 11403
 11404
 11405 062112 011646
 11406 062114 016666 000004 000002
 11407 062122 000002
 11408
 11409
 11410
 11411
 11412
 11413
 11414
 11415
 11416 062124 062112
 11417 062126 060230
 11418 062130 060536
 11419 062132 060512
 11420 062134 060552
 11421 062136 060740
 11422
 11423 062140 061502
 11424
 11425 062142 061432
 11426 062144 061714
 11427 062146 060134
 11428 062150 060172
 11429 062152 062620
 11430 062154 062612
 11431 000032
 11432
 11433
 11434
 11435
 11436
 11437 062156 012737 062334 000024
 11438 062164 012737 000340 000026

.SBTTL TRAP DECODER

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

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

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

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

.SBTTL TRAP TABLE

*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
 *BY THE 'TRAP' INSTRUCTION.

ROUTINE	STARTING ADDRESS	DESCRIPTION
\$TRPAD: .WORD \$TRAP2	TRAP+1(104401)	TTY TYPEOUT ROUTINE
\$TYPE	TRAP+2(104402)	TYPE OCTAL NUMBER (WITH LEADING ZEROS)
\$TYPOC	TRAP+3(104403)	TYPE OCTAL NUMBER (NO LEADING ZEROS)
\$TYPOS	TRAP+4(104404)	TYPE OCTAL NUMBER (AS PER LAST CALL)
\$TYPON	TRAP+5(104405)	TYPE DECIMAL NUMBER (WITH SIGN)
\$TYPDS		
\$GTSWR	TRAP+6(104406)	GET SOFT-SWR SETTING
\$CKSWR	TRAP+7(104407)	TEST FOR CHANGE IN SOFT-SWR
\$RDCHR	TRAP+10(104410)	TTY TYPEIN CHARACTER ROUTINE
\$SAVREG	TRAP+11(104411)	SAVE R0-R5 ROUTINE
\$RESREG	TRAP+12(104412)	RESTORE R0-R5 ROUTINE
.RSET	TRAP+13(104413)	ROUTINE TO RESET STACK AND FPS
.LPER	TRAP+14(104414)	ROUTINE TO SET LOOP ON ERROR ADDRESS

.SBTTL POWER DOWN AND UP ROUTINES

 *POWER DOWN ROUTINE
 \$PWRDN: MOV #\$ILLUP,@#PWRVEC ;;SET FOR FAST UP
 MOV #340,@#PWRVEC+2 ;;PRIO:7

```

11439 062172 010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
11440 062174 010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
11441 062176 010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
11442 062200 010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
11443 062202 010446      MOV      R4,-(SP)      ;;PUSH R4 ON STACK
11444 062204 010546      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
11445 062206 017746 116726      MOV      @SWR,-(SP)    ;;PUSH @SWR ON STACK
11446 062212 010667 000122      MOV      SP,$SAVR6    ;;SAVE SP
11447 062216 012737 062230 000024      MOV      #PWRUP,@PWRVEC ;;SET UP VECTOR
11448 062224 000000      HALT
11449 062226 000776      BR       -2           ;;HANG UP
11450
11451
11452
11453 062230 012737 062334 000024      :*****
:POWER UP ROUTINE
SPWRUP: MOV      #SILLUP,@PWRVEC ;;SET FOR FAST DOWN
11454 062236 016706 000076      MOV      $SAVR6,SP    ;;GET SP
11455 062242 005067 000072      CLR      $SAVR6      ;;WAIT LOOP FOR THE TTY
11456 062246 005267 000066      1$: INC      $SAVR6    ;;WAIT FOR THE INC
11457 062252 001375      BNE      1$          ;;OF WORD
11458 062254 012677 116660      MOV      (SP)+,@SWR   ;;POP STACK INTO @SWR
11459 062260 012605      MOV      (SP)+,R5    ;;POP STACK INTO R5
11460 062262 012604      MOV      (SP)+,R4    ;;POP STACK INTO R4
11461 062264 012603      MOV      (SP)+,R3    ;;POP STACK INTO R3
11462 062266 012602      MOV      (SP)+,R2    ;;POP STACK INTO R2
11463 062270 012601      MOV      (SP)+,R1    ;;POP STACK INTO R1
11464 062272 012600      MOV      (SP)+,R0    ;;POP STACK INTO R0
11465 062274 012737 062156 000024      MOV      #SPWRDN,@PWRVEC ;;SET UP THE POWER DOWN VECTOR
11466 062302 012737 000340 000026      MOV      #340,@PWRVEC+2 ;;PRIO:7
11467 062310 104401      TYPE
11468 062312 062670      SPWRMG: .WORD  POWERM  ;;REPORT THE POWER FAILURE
11469 062314 012716      MOV      (PC)+,(SP)  ;;POWER FAIL MESSAGE POINTER
11470 062316 001466      SPWRAD: .WORD  START   ;;RESTART AT START
11471 062320 042766 000020 000002      BIC      #20,2(SP)   ;;RESTART ADDRESS
11472 062326 005067 175100      CLR      $TBIT      ;;CLEAR 'T' BIT
11473 062332 000002      RTI
11474 062334 000000      $ILLUP: HALT        ;;THE POWER UP SEQUENCE WAS STARTED
11475 062336 000776      BR       -2           ;; BEFORE THE POWER DOWN WAS COMPLETE
11476 062340 000000      $SAVR6: 0            ;;PUT THE SP HERE
11477
11478
11479
11480
11481
11482
11483
11484
11485
11486
11487 062342 104401      :*****
: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
:TYPE A CRLF
11488 062344 001313      .WORD  $CRLF
11489 062346 113737 001102 001232      MOV      @PWRUP,@PWRVEC
11490 062354 042737 177400 001232      BIC      #177400,@PWRVEC
11491 062362 013737 001116 001234      MOV      @PWRUP,@PWRVEC
11492 062370 010046      MOV      R0,-(SP)    ;;GET PC OF CALL
11493
11494 062372 113700 001114      MOV      @PWRUP,@PWRVEC ;;SAVE R0
:GET THE ITEM NUMBER.
    
```

```

11495 062376 042700 177400          BIC      #177400,R0
11496 062402 001007          BNE      MULT
11497 062404 104401 062456      PCTYP:  TYPE      ,EMSG
11498
11499 062410 013746 001116          MOV      @#$ERRPC,-(SP)      ;IF ZERO THEN JUST
11500 062414 104402          TYPOC   ;PRINT THE PC
11501 062416 000137 062452          JMP      @#ERT5
11502 062422 005300          MULT:  DEC      RO          ;OTHERWISE MULT RO BY 2 TO
11503                                     ;GET ERROR MESSAGE POINTER
11504 062424 006300          ASL      RO
11505 062426 062700 001442          ADD      #$ERRTB,RO
11506
11507 062432 011037 062442          MOV      (RO),@#2$          ;PICK UP THE ADDRESS
11508 062436 001405          BEQ      ERT5              ;OF THE EM, ERROR MESSAGE
11509 062440 104401
11510 062442 000000          2$:    .WORD   0
11511 062444 104401          TYPE
11512 062446 001313          .WORD   $CRLF
11513 062450 000755          BR      PCTYP
11514
11515 062452 012600          ERT5:  MOV      (SP)+,RO      ;RESTORE RO.
11516 062454 000207          RTS      PC                ;AND RETURN.
11517
11518 062456 046106 040517 044524      MSG:   .ASCIZ  /FLOATING POINT ERROR, STOPPED AT PC= /
11519 062464 043516 050040 044517
11520 062472 052116 042440 051122
11521 062500 051117 020054 052123
11522 062506 050117 042520 020104
11523 062514 052101 050040 036503
11524 062522 000040
11525                                     .EVEN
11526
11527
11528
11529
11530                                     .SBTTL  FPP SPURIOUS TRAP TO 244 HANDLER
11531                                     :*****
11532                                     :*****
11533                                     :*THIS ROUTINE HANDLES UNEXPECTED TRAPS TO THE FPP TRAP VECTOR AT 244.
11534                                     :*THE LAST FPP INSTRUCTION EXECUTED AND ITS ADDRESS HAS BEEN RECORDED
11535                                     :*THESE ALONG WITH THE FEC, FPS AND PC OF TRAP ARE REPORTED.
11536                                     :*
11536 062524 011637 001236      FPSPUR: MOV      (SP),@#$TMP2      ;SAVE PC OF TRAP.
11537 062530 022626          CMP      (SP)+,(SP)+        ;RESTORE SP.
11538 062532 170200          STFPS   RO                  ;GET FPS
11539 062534 010037 001240          ; MOV      RO,@#$TMP3
11540 062540 170300          STST    RO                  ;GET FEC
11541 062542 010037 001242          MOV      RO,@#$TMP4
11542 062546 104001          1$:    ERROR  1
11543 062550 104413          RSETUP
11544
11545
11546
11547
11548 062552 000137 057224          JMP      @#$EOP
11549
11550
    
```

```

;MOST LIKELY BAD FP1 CHIP
;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?).
    
```

11551
 11552
 11553
 11554
 11555
 11556 062556 011637 001236
 11557 062562 022626
 11558 062564 104004
 11559 062566 104413
 11560
 11561
 11562
 11563
 11564 062570 000137 057224
 11565
 11566
 11567
 11568
 11569
 11570
 11571
 11572 062574 011637 001236
 11573 062600 022626
 11574 062602 104002
 11575 062604 104413
 11576
 11577
 11578
 11579
 11580 062606 000137 057224
 11581
 11582
 11583
 11584
 11585
 11586
 11587
 11588
 11589 062612 011637 001110
 11590 062616 000002
 11591
 11592
 11593
 11594
 11595
 11596
 11597
 11598
 11599
 11600
 11601
 11602
 11603 062620 023727 001140 177570
 11604
 11605 062626 001001
 11606 062630 104407

```

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

```

.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 2
        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
    
```

```

.SBTTL SET LOOP ON ERROR ADDRESS ROUTINE
*****
*****
*
.LPER: MOV (SP),@#$LPERR
        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
    
```

```

11607
11608
11609
11610 062632 012737 062524 000244 1$: MOV #FPSPUR,@#FPVECT
11611 062640 012737 062556 000004 MOV #CPSPUR,@#ERRVECT
11612 062646 012737 062574 000010 MOV #CPTWO,@#10
11613 062654 011600 MOV (SP),R0
11614 062656 012706 001100 MOV #STACK,SP
11615 062662 005004 CLR R4
11616 062664 170104 LDFPS R4
11617 062666 000110 JMP (R0)
11618
11619 .NLIST BEX
    
```

;OF THE PROGRAM VIRTUAL SWITCH REGISTER
 ;AND GIVE THE USER A CHANCE TO
 ;MODIFY IT.

;SAVE RETURN ADDRESS.
 ;RESET THE STACK POINTER.
 ;CLEAR THE FPS.

;RETURN.

;SPECIAL MESSAGES:

```

062670 050200 053517 051105 POWERM: .ASCIZ <CRLF>'POWER FAILURE. PROGRAM RESTARTING.'<CRLF>
062735 ; 000 NULL: .BYTE 0
062736 000011 $TAB: .ASCIZ <TAB>
062740 020040 000 SPACE: .ASCIZ ' '
    
```

;ERROR MESSAGES:

```

062743 120 047522 040502 EM1: .ASCIZ /PROBABLY BAD FP1 CHIP/
062771 120 047522 040502 EM2: .ASCIZ /PROBABLY BAD HYBRID FP CHIP/
063025 120 047522 040502 EM3: .ASCIZ /PROBABLY BAD MMU CHIP/
063053 120 047522 040502 EM4: .ASCIZ /PROBABLY BAD MMU OR HYBRID FP CHIPS/
000001 .END
    
```


ABASE = 000000	271	312			
ACDW1 = 000000	271	314			
ACDW2 = 000000	271	315			
ACPUOP= 000000	271	286			
ADDW0 = 000000	271	316			
ADDW1 = 000000	271	317			
ADDW10= 000000	271	326			
ADDW11= 000000	271	327			
ADDW12= 000000	271	328			
ADDW13= 000000	271	329			
ADDW14= 000000	271	330			
ADDW15= 000000	271	331			
ADDW2 = 000000	271	318			
ADDW3 = 000000	271	319			
ADDW4 = 000000	271	320			
ADDW5 = 000000	271	321			
ADDW6 = 000000	271	322			
ADDW7 = 000000	271	323			
ADDW8 = 000000	271	324			
ADDW9 = 000000	271	325			
ADEVCT= 000000	271	277			
ADEVM = 000000	271	313			
ADONE 002444	524	532	545	558	564#
AENV = 000000	271	282			
AENVM = 000000	271	283			
AERR1 002336	499	528#			
AERR2 002350	500	535#			
AERR3 002402	501	548#			
AFATAL= 000000	271	274			
AMADR1= 000000	271	299			
AMADR2= 000000	271	303			
AMADR3= 000000	271	306			
AMADR4= 000000	271	309			
AMAMS1= 000000	271	293			
AMAMS2= 000000	271	301			
AMAMS3= 000000	271	304			
AMAMS4= 000000	271	307			
AMSGAD= 000000	271	279			
AMSGLG= 000000	271	280			
AMSGTY= 000000	271	273			
AMTYP1= 000000	271	294			
AMTYP2= 000000	271	302			
AMTYP3= 000000	271	305			
AMTYP4= 000000	271	308			
APASS = 000000	271	276			
APRIOR= 000000	271				
APTCSU= 000040	11023	11275#			
APTENV= 000001	10923	11016	11231	11273#	
APTSIZ= 000200	446	11272#			
APTSPO= 000100	11018	11233	11274#		
ASWREG= 000000	271	284			
ATESTN= 000000	271	275			
ALUNIT = 000000	271	278			
AUSWR = 000000	271	285			
AVECT1= 000000	271	310			
AVECT2= 000000	271	311			

BB24	025360	4833	4835#		
BB25	025366	4828	4837#		
BB26	025400	4839	4842#		
BB27	025430	4844	4850#		
BB3	024774	4746#			
BB30	025452	4856#	4859		
BB31	025462	4857	4859#		
BB32	025474	4861	4864#		
BB33	025524	4866	4872#		
BB34	025546	4878#	4881		
BB35	025556	4879	4881#		
BB4	025012	4750#	4753		
BB5	025022	4751	4753#		
BB6	025034	4756	4759#		
BB7	025064	4763	4767#		
BDONE	002546	594	614#		
BERR	002502	591	596#		
BERR1	002532	601	608#		
BIT0	= 000001	146#			
BIT00	= 000001	136#	146		
BIT01	= 000002	135#	145		
BIT02	= 000004	134#	144		
BIT03	= 000010	133#	143		
BIT04	= 000020	132#	142		
BIT05	= 000040	131#	141		
BIT06	= 000100	130#	140		
BIT07	= 000200	129#	139		
BIT08	= 000400	128#	138	10857	
BIT09	= 001000	127#	137	10865	10934
BIT1	= 000002	145#			
BIT10	= 002000	126#	10911		
BIT11	= 004000	125#	10872		
BIT12	= 010000	124#	10811		
BIT13	= 020000	123#	10918		
BIT14	= 040000	122#	10843		
BIT15	= 100000	121#			
BIT2	= 000004	144#			
BIT3	= 000010	143#			
BIT4	= 000020	142#			
BIT5	= 000040	141#			
BIT6	= 000100	140#			
BIT7	= 000200	139#			
BIT8	= 000400	138#			
BIT9	= 001000	137#			
BPTVEC	= 000014	153#			
B1	002456	582#	593		
B2	002460	585#	598		
B3	002476	593#	606	612	
CCCDON	042460	7424	7512#		
CCC1	041454	7265#			
CCC10	042050	7376#			
CCC11	042104	7388#			
CCC12	042140	7401#			
CCC13	042174	7413#			
CCC2	041510	7277#			
CCC3	041544	7289#			

HA5R	011716	2243	2300#															
HA5W	011646	2143	2206	2289#														
HCLR	011532	2123	2225	2260#														
HCLR1	011542	2262#	2263															
HCMP	011474	2149	2161	2174	2187	2200	2213	2249#										
HCMP1	011514	2253#	2256															
HCMP2	011524	2254	2256#															
HDAT1	011726	2118	2303#															
HDAT2	011736	2305#																
HDAT3	011746	2307#																
HDAT4	011756	2309#																
HDAT5	011766	2311#																
HDONE	011776	2219	2275	2314#														
HERROR	011550	2255	2267#															
HFLAG	011604	2116*	2217	2221*	2279#													
HHDATO	033132	5606	5612	5641	5666	5672	5697	5720	5738	5747	5756	5765	5774	5783				
		5792	5801	5810	5819	5828	5837	5855#										
HHDONE	033302	5732	5741	5750	5759	5768	5777	5786	5795	5804	5813	5822	5831	5840				
		5854	5907#															
HHERO	032152	5617	5733#															
HHERO0	032266	5623	5751#															
HHER1	032220	5619	5742#															
HHER10	033014	5731	5832#															
HHER2	032334	5647	5760#															
HHER3	032402	5652	5769#															
HHER4	032450	5677	5778#															
HHER5	032516	5679	5787#															
HHER6	032564	5684	5796#															
HHER7	032632	5703	5805#															
HHER8	032700	5708	5814#															
HHER9	032746	5726	5823#															
HHHDON	054344	9956	10096#															
HHH1	052242	9655#																
HHH10	053342	9867#																
HHH11	053442	9890#																
HHH12	053542	9913#																
HHH13	053642	9936#																
HHH2	052342	9678#																
HHH3	052442	9701#																
HHH4	052542	9724#																
HHH5	052642	9747#																
HHH6	052742	9770#																
HHH7	053042	9794#																
HHH8	053142	9818#																
HHH9	053242	9841#																
HHP0	033142	5601	5737	5746	5755	5859#												
HHP1	033152	5603	5736	5745	5754	5863#												
HHP10	033262	5699	5811	5820	5899#													
HHP11	033272	5722	5829	5838	5903#													
HHP2	033162	5608	5739	5748	5757	5867#												
HHP3	033172	5613	5871#															
HHP4	033202	5643	5673	5766	5775	5875#												
HHP5	033212	5636	5715	5717	5764	5773	5826	5827	5835	5836	5879#							
HHP6	033222	5638	5763	5772	5883#													
HHP7	033232	5668	5784	5793	5802	5887#												
HHP8	033242	5661	5692	5694	5782	5791	5800	5808	5809	5817	5818	5891#						

HX1	036224	6423#							
HX10	036432	6473	6476#						
HX11	036454	6483#	6495						
HX12	036474	6489#	6492						
HX13	036504	6490	6492#						
HX14	036512	6484	6495#						
HX15	036530	6499	6504#						
HX16	036556	6509	6512#						
HX165	036562	6513#	6685	6696					
HX17	036610	6518	6521#						
HX18	036646	6526	6531#						
HX19	036666	6537#	6540						
HX2	036254	6430	6431#						
HX20	036676	6538	6540#						
HX21	036700	6543#							
HX22	036730	6547	6551#						
HX23	036750	6558#	6571						
HX24	036770	6564#	6567						
HX25	037000	6565	6567#						
HX26	037006	6559	6571#						
HX27	037010	6575#							
HX28	037042	6584	6586#						
HX29	037064	6595#	6598						
HX3	036270	6433	6435#						
HX30	037074	6596	6598#						
HX31	037112	6602	6607#						
HX32	037142	6615	6617#						
HX33	037164	6626#	6629						
HX34	037174	6627	6629#						
HX35	037212	6633	6635#						
HX4	036310	6441#	6451						
HX5	036330	6446#	6449						
HX6	036340	6447	6449#						
HX7	036346	6442	6451#						
HX8	036364	6454	6457#						
HX9	036416	6466	6470#						
H1	011060	2116#							
H10	011322	2196#	2201						
H11	011354	2209#	2214						
H12	011406	2218	2221#						
H2	011100	2120#	2121						
H3	011110	2125#	2222						
H4	011162	2147#							
H5	011204	2157#	2162						
H6	011236	2170#	2175						
H7	011270	2183#	2188						
IDATIO	006274	1325	1331	1348	1376	1381	1409	1445	1486#
IDATI1	006276	1487#							
IDATI2	006300	1406*	1488#						
IDATI3	006302	1407*	1489#						
IDAT00	006264	1344	1347	1403	1408	1446	1481#		
IDAT01	006266	1482#							
IDAT02	006270	1353	1358	1414	1483#				
IDAT03	006272	1417	1484#						
IDONE	006304	1423	1441	1448	1456	1468	1491#		
IERR0	006044	1338	1426#						

IERR1	006126	1362	1416	1444#				
IERR2	006146	1366	1451#					
IERR25	006170	1454#	1460					
IERR3	006220	1398	1463#					
IERR4	006174	1419	1457#					
IIIDON	046014	8273	8443#					
III1	045160	8206#						
III2	045224	8223#						
III3	045270	8241#						
III4	045334	8258#						
IOTVEC=	000020	154#	401*	402*				
IPAT10	006244	1326	1375	1471#				
IPAT11	006246	1472#						
IPAT12	006250	1414	1473#					
IPAT13	006252	1417	1474#					
IPAT20	006254	1334	1384	1476#				
IPAT21	006256	1477#						
IPAT22	006260	1478#						
IPAT23	006262	1479#						
I1	005464	1323#						
I10	005640	1356	1365#					
I105	005646	1351	1368#					
I106	005642	1364	1366#					
I11	005650	1372#						
I12	005652	1375#						
I13	005666	1378#	1379					
I14	005722	1388	1391#	1464				
I15	005724	1392#	1430					
I16	005726	1393#	1432					
I17	005744	1397	1400#					
I2	005502	1328#	1329					
I20	006004	1411#	1421					
I21	006020	1416#	1418					
I22	006024	1415	1417#					
I23	006040	1412	1421#					
I3	005542	1339	1340#					
I4	005544	1341#	1426					
I5	005546	1342#	1428					
I6	005572	1350#	1368					
I7	005606	1355#	1365					
JBUF0	012766	2516	2565#					
JBUF1	012770	2566#						
JBUF2	012772	2567#						
JBUF3	012774	2568#						
JDAT10	012776	2505	2520	2528	2552	2560	2570#	
JDAT11	013000	2571#						
JDAT12	013002	2572#						
JDAT13	013004	2573#						
JDAT00	013006	2513	2521	2561	2575#			
JDAT0	013016	2502	2580#					
JDAT01	013010	2576#						
JDAT02	013012	2577#						
JDAT03	013014	2578#						
JDAT1	013020	2581#						
JDAT2	013022	2582#						
JDAT3	013024	2583#						

JDONE	013026	2532	2547	2554	2563	2586#
JERRO	012646	2507	2536#			
JERR1	012714	2518	2530	2549#		
JERR2	012740	2525	2558#			
JJJDON	046754	8547	8715#			
JJJ1	046020	8465#				
JJJ2	046104	8486#				
JJJ3	046170	8507#				
JJJ4	046254	8529#				
J1	012540	2499#				
J10	012666	2537	2539	2542#		
J2	012564	2509#	2550	2559		
J3	012566	2510#	2536			
J4	012570	2511#	2538			
J5	012624	2523#	2526			
J6	012632	2524	2526#			
J7	012644	2529	2532#			
KBUF0	013270	2610	2621	2633	2674#	
KBUF1	013272	2675#				
KBUF2	013274	2676#				
KBUF3	013276	2677#				
KDATI0	013260	2625	2656	2664	2669#	
KDATI1	013262	2670#				
KDATI2	013264	2671#				
KDATI3	013266	2672#				
KDAT00	013300	2618	2626	2665	2679#	
KDAT01	013302	2680#				
KDAT02	013304	2681#				
KDAT03	013306	2682#				
KDONE	013320	2637	2651	2658	2667	2689#
KERRO	013142	2612	2641#			
KERR1	013206	2623	2635	2653#		
KERR2	013232	2630	2662#			
KKKDON	047616	8806	8974#			
KKK1	046760	8740#				
KKK3	047024	8758#				
KKK4	047070	8775#				
KKK5	047134	8792#				
KPAT0	013310	2607	2684#			
KPAT1	013312	2685#				
KPAT2	013314	2686#				
K1	013034	2604#				
K10	013162	2642	2644	2647#		
K2	013060	2614#	2654	2663		
K3	013062	2615#	2641			
K4	013064	2616#	2643			
K5	013120	2628#	2631			
K6	013126	2629	2631#			
K7	013140	2634	2637#			
LDATI0	013620	2713	2720	2739	2759	2771 2786#
LDATI1	013622	2787#				
LDATI2	013624	2733	2737*	2753	2788#	
LDATI3	013626	2738*	2789#			
LDAT00	013632	2730	2740	2760	2765	2791#
LDAT01	013634	2772	2792#			
LDAT02	013636	2793#				

SW13	=	020000	95#							
SW14	=	040000	94#							
SW15	=	100000	93#							
SW2	=	000004	116#							
SW3	=	000010	115#							
SW4	=	000020	114#							
SW5	=	000040	113#							
SW6	=	000100	112#							
SW7	=	000200	111#							
SW8	=	000400	110#							
SW9	=	001000	109#							
S1		012002	2331#							
S10		012150	2379#							
S11		012174	2387#	2390						
S12		012204	2388	2390#						
S2		012036	2345#	2433	2440					
S3		012040	2346#	2394	2448#					
S4		012044	2348#	2450						
S5		012070	2356#	2359						
S6		012100	2357	2359#						
S7		012106	2364#	2398						
S8		012142	2376#	2436	2443	2459				
S9		012144	2377#	2413	2461					
TAB	=	000011	48#	11619						
TBITVE	=	000014	151#	415*	416*					
TDAT10		007066	1510	1515	1532	1553	1558	1587	1623	1661#
TDAT11		007070	1662#							
TDAT12		007072	1584*	1663#						
TDAT13		007074	1585*	1664#						
TDAT00		007056	1528	1531	1580	1586	1624	1656#		
TDAT01		007060	1657#							
TDAT02		007062	1537	1592	1658#					
TDAT03		007064	1595	1659#						
TDONE		007076	1600	1619	1626	1633	1644	1666#		
TERR0		006644	1522	1604#						
TERR1		006726	1541	1594	1622#					
TERR2		006746	1543	1629#						
TERR25		006762	1631#	1636						
TERR3		007012	1575	1639#						
TERR4		006774	1597	1634#						
TKVEC	=	000060	158#							
TPAT10		007036	1509	1552	1646#					
TPAT11		007040	1647#							
TPAT12		007042	1592	1648#						
TPAT13		007044	1595	1649#						
TPAT20		007046	1518	1561	1651#					
TPAT21		007050	1652#							
TPAT22		007052	1653#							
TPAT23		007054	1654#							
TPVEC	=	000064	159#							
TRAPVE	=	000034	157#	405*	406*					
TRIVEC	=	000014	152#							
TST1		002220	496#							
TST10		006306	1505#							
TST11		007100	1686#							
TST12		011054	2113#							

T14	006526	1565	1567#							
T15	006530	1568#	1608	1640						
T16	006532	1569#	1610							
T17	006550	1574	1577#							
T2	006330	1512#	1513							
T20	006610	1589#	1599							
T21	006624	1594#	1596							
T22	006626	1593	1595#							
T23	006640	1590	1599#							
T3	006370	1523	1524#							
T4	006372	1525#	1604							
T5	006374	1526#	1606							
T6	006420	1534#	1545							
T7	006434	1539#	1542							
UDONE	017676	3620	3629	3647	3662	3673	3678	3710#		
UERR0	017342	3511	3623#							
UERR1	017366	3532	3593	3633#						
UERR10	017374	3635#	3649							
UERR11	017436	3644#								
UERR2	017452	3555	3574	3648#						
UERR20	017460	3634	3650#							
UERR21	017522	3659#								
UERR3	017536	3608	3665#							
UERR4	017566	3672	3674#							
UFLAG	017662	3503*	3594	3601*	3704#					
UPAT00	017612	3504	3516	3539	3560	3579	3597	3611	3679#	
UPAT01	017614	3680#								
UPAT02	017616	3681#								
UPAT03	017620	3682#								
UPAT10	017622	3522	3684#							
UPAT11	017624	3685#								
UPAT12	017626	3686#								
UPAT13	017630	3687#								
UPAT20	017632	3545	3689#							
UPAT21	017634	3690#								
UPAT22	017636	3691#								
UPAT23	017640	3692#								
UPAT30	017642	3565	3694#							
UPAT31	017644	3695#								
UPAT32	017646	3696#								
UPAT33	017650	3697#								
UPAT40	017652	3584	3613	3699#						
UPAT41	017654	3700#								
UPAT42	017656	3701#								
UPAT43	017660	3702#								
UROM1	017670	3518*	3541*	3562*	3581*	3650	3707#			
UROM2	017672	3519*	3542*	3563*	3582*	3651	3708#			
UROM3	017674	3520*	3543*	3564*	3583*	3636	3709#			
UTMP1	017664	3509*	3518	3581	3602*	3705#				
UTMP2	017666	3510*	3541	3562	3603*	3706#				
U0	016600	3506#	3507							
U1	016630	3512#	3604							
U10	017166	3585#	3588							
U11	017170	3586#	3623							
U12	017222	3592	3594#							
U13	017240	3599#	3600							

YDAT03	021552	4094#							
YDONE	021604	4055	4070	4078	4084	4111#			
YERR1	021404	4060	4063#						
YERR2	021446	4062	4071#						
YERR3	021510	4047	4079#						
YFLAG	021534	4024*	4048	4050*	4086#				
YPAT00	021554	4025	4052	4096#					
YPAT01	021556	4097#							
YPAT02	021560	4098#							
YPAT03	021562	4099#							
YPAT10	021564	4026	4051	4101#					
YPAT11	021566	4102#							
YPAT12	021570	4103#							
YPAT13	021572	4104#							
YPAT20	021574	4032	4066	4074	4106#				
YPAT21	021576	4107#							
YPAT22	021600	4108#							
YPAT23	021602	4109#							
YTMP1	021536	4025*	4034	4051*	4057	4065	4073	4087#	
YTMP2	021540	4026*	4041	4052*	4068	4076	4088#		
YTMP3	021542	4027*	4045	4053*	4082	4089#			
Y1	021232	4028#	4054						
Y2	021254	4035#	4064	4072	4080				
Y3	021300	4042#	4044						
Y4	021316	4046	4048#						
Y5	021356	4049	4055#						
Y6	021360	4043	4056#						
Y7	021374	4059#	4061						
ZDAT00	022046	4141	4163	4178#					
ZDAT01	022050	4179#							
ZDAT02	022052	4180#							
ZDAT03	022054	4181#							
ZDONE	022106	4158	4166	4172	4198#				
ZERR1	021752	4147	4159#						
ZERR2	022014	4151	4167#						
ZFLAG	022040	4128*	4152	4154*	4174#				
ZPAT00	022056	4129	4183#						
ZPAT01	022060	4184#							
ZPAT02	022062	4185#							
ZPAT03	022064	4186#							
ZPAT10	022066	4155	4188#						
ZPAT11	022070	4189#							
ZPAT12	022072	4190#							
ZPAT13	022074	4191#							
ZPAT20	022076	4135	4162	4193#					
ZPAT21	022100	4194#							
ZPAT22	022102	4195#							
ZPAT23	022104	4196#							
ZTMP1	022042	4129*	4137	4144	4155*	4161	4164	4175#	
ZTMP2	022044	4130*	4149	4156*	4170	4176#			
Z1	021630	4131#	4157						
Z2	021652	4138#	4160	4168					
Z3	021676	4145#	4148						
Z4	021704	4146	4148#						
Z5	021716	4150	4152#						
Z6	021750	4153	4158#						

\$EOPCT	057252	409*	10783#	10787															
\$ERFLG	001103	192#	10832	10861	10863	10869*	10891	10908*	10946										
\$ERMAX	001115	198#	412*	10863	10886*	10891													
\$ERROR	057734	403	10906#																
\$ERRPC	001116	199#	10915*	10916*	10917	10946	11491	11499											
\$ERRTB	001442	350#	11505																
\$ERTTL	001112	196#	10914*	10946															
\$ESCAP	001304	262#	411*	10885*	10937	10939	10946												
\$ETABL	001336	281#																	
\$ETEND	001442	334#	389																
\$FATAL	001320	274#	11259*																
\$FFLG	061430	11222*	11225*	11253	11262*	11270#													
\$FILLC	001156	217#	11041	11072															
\$FILLS	001155	216#	11072																
\$GDADR	001120	200#																	
\$GDDAT	001124	202#																	
\$GET42	057302	10792#																	
\$GT\$WR	061502	11299#	11423																
\$HD =	000003	41	42																
\$HIBTS	001452	384#																	
\$ICNT	001104	193#	10876*	10877	10879*	10890													
\$ILLUP	062334	11437	11453	11474#															
\$INTAG	001135	207#	11327	11385															
\$ITEMB	001114	197#	10917*	10925	10946	11494													
\$LF	001314	266#	10946	11072															
\$LFLG	061427	11263*	11269#																
\$LOOP	057426	10816	10819#																
\$LPADR	001106	194#	427*	10867*	10883*	10888	10890												
\$LPERR	001110	195#	428*	10867	10884*	10890	10936	11589*											
\$MADR1	001350	299#																	
\$MADR2	001354	303#																	
\$MADR3	001360	306#																	
\$MADR4	001364	309#																	
\$MAIL	001316	272#	385	389	445	460	10882	10923	11016										
\$MAMS1	001346	293#																	
\$MAMS2	001352	301#																	
\$MAMS3	001356	304#																	
\$MAMS4	001362	307#																	
\$MBADR	001454	385#																	
\$MFLG	061426	11223*	11229	11264*	11268#														
\$MNEW	062057	11302	11383#																
\$MSGAD	001332	279#	11239*	11242															
\$MSGLG	001334	280#	11244*																
\$MSGTY	001316	273#	11237	11245*	11257	11261*													
\$MSWR	062046	11299	11381#																
\$MTYP1	001347	294#																	
\$MTYP2	001353	302#																	
\$MTYP3	001357	305#																	
\$MTYP4	001363	308#																	
\$MXCNT	057732	10880	10890#																
\$NULL	001154	215#	11043	11072															
\$NWTST=	000001	485#	487	572#	574	621#	623	769#	771	867#	869	926#	928	1314#					
		1316	1499#	1501	1675#	1677	2106#	2108	2322#	2324	2489#	2491	2594#	2596					
		2697#	2699	2804#	2806	2914#	2916	3064#	3066	3217#	3219	3343#	3345	3487#					
		3489	3713#	3715	3856#	3858	4015#	4017	4117#	4119	4206#	4208	4414#	4416					
		4721#	4723	4989#	4991	5327#	5329	5461#	5463	5581#	5583	5915#	5917	6415#					

	6417	6766#	6768	7091#	7093	7254#	7256	7521#	7523	7749#	7751	8015#	8017
	8194#	8196	8453#	8455	8725#	8727	8984#	8986	9254#	9256	9644#	9646	10106#
\$OCNT 060734	10108	10349#	10351	10601#	10603								
\$OMODE 060736	11105*	11134*	11147#										
\$OVER 057716	11100*	11104*	11109	11112*	11123*	11149#							
\$PASS 001324	10844	10860	10868	10878	10887#								
\$PASTM 001460	276#	445*	10780*	10781*	10789	10874	10891						
\$PWAD 062316	387#												
\$PWDRN 062156	11470#												
\$PWDRM 062312	407	11437#	11465										
\$PWRRUP 062230	11468#												
\$QUES 001312	11447	11453#											
\$RDCHR 061714	264#	10946	11072	11345									
\$RDDEC= ***** U	11358#	11426											
\$RDLIN= ***** U	11427												
\$RDOCT= ***** U	11427												
\$RDSZ = 000001	11379#												
\$REGAD 001160	219#												
\$REG0 001162	221#												
\$REG1 001164	222#												
\$REG10 001202	229#												
\$REG11 001204	230#												
\$REG12 001206	231#												
\$REG13 001210	232#												
\$REG14 001212	233#												
\$REG15 001214	234#												
\$REG16 001216	235#												
\$REG17 001220	236#												
\$REG2 001166	223#												
\$REG20 001222	237#												
\$REG21 001224	238#												
\$REG22 001226	239#												
\$REG23 001230	240#												
\$REG3 001170	224#												
\$REG4 001172	225#												
\$REG5 001174	226#												
\$REG6 001176	227#												
\$REG7 001200	228#												
\$RESRE 060172	10980#	11428											
\$RTNAD 057430	10820#												
\$RTRN 057424	415	417*	422*	10796	10817#								
\$R2A = ***** U	11429												
\$SAVRE 060134	10964#	11427											
\$SAVR6 062340	11446*	11454	11455*	11456*	11476#								
\$SCOPE 057454	401	10841#											
\$SETUP= 000137	170#	400	401	403	405	407	409	410	411	413	427	454	457
	10842	10907	10933	10941	11282	11385	11472						
\$STUP = 177777	170#												
\$SVLAD 057662	10852	10881#											
\$SVPC = 001452	362#	367											
\$SWR = 177400	41	42#	46#	261	262	263	410	411	413	427	428	497	579
	633	790	876	940	1321	1506	1687	2114	2331	2497	2602	2705	2813
	2922	3073	3224	3351	3503	3722	3864	4024	4128	4215	4426	4733	4998
	5336	5472	5593	5930	6422	6775	7100	7263	7529	7758	8024	8204	8463
	8738	8993	9263	9653	10115	10358	10608	10833	10834	10835	10836	10837	10843

	10855	10857	10858	10861	10862	10863	10870	10871	10872	10884	10887	10890	10898
\$SWREG 001340	10899	10900	10901	10902	10911	10918	10930	10934	10946	11471			
\$SWRMK= 000000	284#	448											
\$SWRMS= 000200	10837	10838	10859										
STAB 062736	47#												
\$TBIT 057432	11619#												
\$TERM = 000032	426*	10813*	10821#	11472*									
\$TESTN 001322	11431#												
\$TIMES 001302	275#	10882*											
\$TKB 001146	261#	410*	10779*	10870*	10877	10880*	10890						
\$TKS 001144	212#	11280	11291	11308	11362	11368							
\$TMP0 001232	211#	11280	11289	11305	11329*	11360	11366						
\$TMP1 001234	241#	10803*	10808	11489*	11490*								
\$TMP10 001252	242#	11491*											
	249#	1207*	1238*	1243*	1273*	3524*	3548*	3568*	3587*	5734*	5743*	5752*	5761*
	5770*	5779*	5788*	5797*	5806*	5815*	5824*	5833*	6155*	6165*	6175*	6187*	6197*
	6226*	6236*	6246*	6256*	6266*	6276*	6306*	6316*	6326*	6336*	6346*	6356*	7234*
	7480*	7703*	7974*	8148*	8328*	8600*	8857*	9127*	9570*	10024*	10275*	10521*	
\$TMP11 001254	250#	5735*	5744*	5753*	5762*	5771*	5780*	5789*	5798*	5807*	5816*	5825*	5834*
	6156*	6166*	6176*	6188*	6198*	6227*	6237*	6247*	6257*	6267*	6277*	6307*	6317*
	6327*	6337*	6347*	6357*	8348*	8418*	8620*	8690*	8887*	8965*	9156*	9234*	9571*
	9620*	10026*	10085*	10276*	10321*	10522*	10575*						
\$TMP12 001256	251#	9572*	9621*	10025*	10086*	10277*	10322*	10523*	10576*				
\$TMP13 001260	252#	10278*	10327*	10524*	10581*								
\$TMP14 001262	253#	10279*	10328*	10525*	10582*								
\$TMP15 001264	254#	1206*	1242*										
\$TMP16 001266	255#												
\$TMP17 001270	256#												
\$TMP2 001236	243#	528*	542*	555*	560*	598*	639*	656*	672*	686*	703*	718*	734*
	749*	800*	854*	881*	912*	943*	986*	1208*	1245*	1339*	1388*	1436*	1464*
	1523*	1565*	1614*	1640*	1689*	1717*	1745*	1774*	1802*	1830*	1858*	1886*	1914*
	1942*	1970*	1998*	2269	2400*	2433*	2436*	2440*	2443*	2454*	2465*	2544*	2550*
	2559*	2649*	2654*	2663*	2751*	2758*	2770*	2857*	2869*	2874*	2886*	2996*	3001*
	3011*	3023*	3028*	3147*	3163*	3176*	3181*	3290*	3295*	3298*	3306*	3313*	3428*
	3433*	3436*	3449*	3454*	3525*	3549*	3569*	3588*	3617*	3626*	3674*	3728*	3756*
	3785*	3813*	3871*	3895*	3919*	3943*	4064*	4072*	4080*	4160*	4168*	4223*	4253*
	4289*	4315	4328*	4332*	4431*	4460*	4487*	4517*	4549*	4578*	4739*	4763*	4790*
	4814*	4844*	4866*	5003*	5033*	5056*	5079*	5108*	5137*	5167*	5341*	5372*	5477*
	5500*	5600*	5634*	5660*	5690*	5714*	5841	5846*	5936*	5961*	5971	5976*	6003*
	6028*	6038	6043*	6071*	6096*	6106	6111*	6135	6206	6285	6430*	6466*	6509*
	6526*	6547*	6584*	6615*	6689*	7025*	7108*	7132*	7155*	7190*	7193	7459*	7682*
	7953*	8127*	8333*	8386*	8411*	8605*	8658*	8683*	8862*	8939*	9132*	9208*	9547*
	10001*	10251*	10497*	11536*	11556*	11572*							
\$TMP20 001272	257#												
\$TMP21 001274	258#												
\$TMP22 001276	259#												
\$TMP23 001300	260#												
\$TMP3 001240	244#	603*	609*	807*	830*	844*	896*	902*	915*	1054*	1062*	1092*	1100*
	1124*	1139*	1154*	1170*	1185*	1193*	1201*	1209*	1244*	1437*	1465*	1616*	1641*
	2421*	2427*	2542*	2551*	2560*	2647*	2655*	2664*	2752*	2759*	2771*	2858*	2868*
	2875*	2884*	3021*	3029*	3152*	3174*	3182*	3300*	3307*	3438*	3455*	3644*	3659*
	3676*	3747*	3776*	3804*	3834*	3964*	3969*	3979*	3984*	4065*	4073*	4081*	4161*
	4169*	4340*	4346*	4354*	4360*	4364*	4370*	4604*	4608*	4611*	4617*	4623*	4626*
	4632*	4638*	4644*	4647*	4653*	4656*	4659*	4662*	4886*	4895*	4905*	4914*	4920*
	4929*	5195*	5199*	5206*	5213*	5220*	5227*	5234*	5241*	5248*	5255*	5262*	5269*
	5276*	5399*	5403*	5410*	5417*	5424*	5519*	5525*	5533*	5736*	5745*	5754*	5763*

ACCMAC	19#	1688	1716	1744	1773	1801	1829	1857	1885	1913	1941	1969	1997	5402	5409
ASMAC1	13#	5198	5205	5212	5219	5226	5233	5240	5247	5254	5261	5268	5275	5808	5817
	5416	5423	5524	5532	5736	5745	5754	5763	5772	5781	5790	5799	5808	5817	5826
	5835	6147	6157	6167	6177	6189	6199	6218	6228	6238	6248	6258	6268	6278	6298
ASMAC2	6308	6318	6328	6338	6348	6358									
	14#	5733	5742	5751	5760	5769	5778	5787	5796	5805	5814	5823	5832	6154	6164
	6174	6186	6196	6225	6235	6245	6255	6265	6275	6305	6315	6325	6335	6345	6355
COMMEN	161#														
ENDCOM	161#														
ENDPAS	3#														
ERROR	55#	520	531	544	557	562	605	611	649	664	680	695	712	727	743
	758	808	832	846	856	898	904	916	1058	1078	1082	1093	1115	1119	1128
	1143	1159	1174	1189	1197	1203	1222	1232	1239	1258	1268	1275	1440	1447	1455
	1467	1618	1625	1632	1643	2062	2274	2422	2428	2434	2437	2441	2444	2456	2467
	2546	2553	2562	2650	2657	2666	2754	2761	2773	2860	2870	2877	2888	2998	3006
	3012	3024	3031	3149	3157	3164	3177	3184	3292	3302	3309	3317	3430	3440	3450
	3457	3619	3628	3646	3661	3677	3740	3748	3769	3777	3797	3805	3827	3835	3967
	3973	3981	3986	4069	4077	4083	4165	4171	4330	4334	4342	4348	4356	4362	4368
	4374	4605	4609	4615	4621	4630	4636	4642	4651	4887	4896	4907	4916	4922	4931
	5196	5203	5210	5217	5224	5231	5238	5245	5252	5259	5266	5273	5280	5400	5407
	5414	5421	5428	5521	5529	5537	5740	5749	5758	5767	5776	5785	5794	5803	5812
	5821	5830	5839	5853	6151	6161	6171	6181	6193	6203	6222	6232	6242	6252	6262
	6272	6282	6302	6312	6322	6332	6342	6352	6362	6641	6646	6652	6658	6663	6668
	6673	6680	6691	6700	6709	6787	6803	6818	6833	6849	6866	6884	6902	6920	6937
	6955	6973	6991	7057	7073	7205	7214	7222	7235	7274	7286	7299	7311	7323	7335
	7347	7360	7373	7385	7397	7410	7422	7502	7506	7544	7560	7576	7592	7608	7625
	7641	7730	7734	7769	7782	7794	7806	7818	7831	7844	7856	7868	7880	7892	7904
	7916	7996	8000	8039	8056	8072	8088	8175	8179	8217	8219	8234	8237	8252	8254
	8269	8272	8368	8372	8397	8401	8433	8437	8480	8482	8501	8503	8522	8524	8544
	8546	8640	8644	8669	8673	8705	8709	8751	8754	8769	8771	8786	8788	8803	8805
	8909	8913	8921	8925	8950	8954	9010	9012	9031	9033	9052	9054	9073	9075	9178
	9182	9190	9194	9219	9223	9276	9278	9293	9295	9310	9312	9327	9329	9344	9346
	9361	9364	9379	9382	9397	9399	9414	9416	9431	9433	9448	9450	9465	9467	9482
	9484	9499	9501	9602	9616	9622	9672	9674	9695	9697	9718	9720	9741	9743	9764
	9766	9787	9790	9811	9814	9835	9837	9859	9862	9884	9886	9907	9909	9930	9932
	9953	9955	10064	10081	10087	10129	10131	10149	10165	10167	10186	10203	10205	10303	10317
	10323	10378	10380	10404	10426	10428	10452	10554	10571	10577	10613	10620	10625	10629	10636
	10640	10647	10651	10656	10661	10665	10670	10678	10682	10687	10694	10701	10710	10718	10726
	10735	10743	11542	11558	11574										
ESCAPE	161#														
GETPRI	161#														
GETSWR	161#	457#													
HMAC1	15#	2126	2130	2134	2138	2142									
HMAC2	17#	2226	2230	2234	2238	2242									
HMAC3	18#	2151	2164	2177	2190	2203									
LOADTP	4#														
LPER	23#	497	579	633	651	666	682	697	714	729	745	790	876	941	984
	1321	1372	1506	1549	1692	1720	1748	1777	1805	1833	1861	1889	1917	1945	1973
	2001	2114	2331	2364	2497	2602	2705	3512	3534	3556	3575	3605	3722	3750	3779
	3807	3864	3888	3912	3936	4028	4131	4215	4247	4282	4427	4456	4485	4515	4545
	4574	4734	4759	4788	4812	4842	4864	4999	5029	5052	5075	5104	5133	5163	5337
	5368	5473	5496	5596	5629	5655	5686	5710	5932	5957	5999	6024	6067	6092	6423
	6457	6504	6521	6543	6575	6607	6777	6791	6806	6821	6837	6852	6870	6888	6906
	6923	6941	6959	6977	7102	7127	7150	7265	7277	7289	7302	7314	7326	7338	7351
	7364	7376	7388	7401	7413	7531	7547	7563	7579	7595	7612	7628	7760	7772	7785
	7797	7809	7822	7835	7847	7859	7871	7883	7895	7907	8026	8043	8059	8075	8206

TYPOCT	161#	11300													
TYPTXT	161#														
\$\$CMRE	182#	221	222	223	224	225	226	227	228	229	230	231	232	233	234
	235	236	237	238	239	240									
\$\$CMTM	182#	241	242	243	244	245	246	247	248	249	250	251	252	253	254
	255	256	257	258	259	260									
\$\$ESCA	161#														
\$\$NEWT	161#	485	572	621	769	867	926	1314	1499	1675	2106	2322	2489	2594	2697
	2804	2914	3064	3217	3343	3487	3713	3856	4015	4117	4206	4414	4721	4989	5327
	5461	5581	5915	6415	6766	7091	7254	7521	7749	8015	8194	8453	8725	8984	9254
	9644	10106	10349	10601											
\$\$SET	11409#	11418	11419	11420	11421	11423	11425	11426	11427	11428	11429	11430			
\$\$SETM	445#														
\$\$SETU	413#														
\$\$SKIP	161#														
.EQUAT	29#	51													
.HEADE	29#	31													
.SETUP	29#	170													
.SWRHI	29#														
.SACT1	29#	358													
.SAPT8	29#	268#													
.SAPTH	29#	368													
.SAPTY	29#	11219													
.SCATC	29#	171													
.SCMTA	29#	182													
.SEOP	29#														
.SERRO	29#	10892													
.SPOWE	29#	11433													
.SREAD	29#	11277													
.\$SAVE	29#	10947													
.\$SCOP	29#	10827													
.\$STRAP	29#	11386													
.\$TYPD	29#	11151													
.\$TYPE	29#	10993													
.\$TYPO	29#	11073													

. ABS. 063117 000

ERRORS DETECTED: 0

CJKDCB.BIN,CJKDCB.LST/CRF/SOL/NL:TOC=CJKDCB.P11
 RUN-TIME: 112 89 15 SECONDS
 RUN-TIME RATIO: 889/217=4.0
 CORE USED: 30K (59 PAGES)