IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DFPB-A-D
PRODUCT NAME: PDP-11/34 FPP DIAGNOSTIC PART 2
DATE: DECEMBER 1976
AUTHOR: ANTHONY VEZZA

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) 1976 BY DIGITAL EQUIPMENT CORPORATION
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 INTERRUPTS TEST
   8.7 ACT, APT AND XحيP COMPATIBILITY
9. PROGRAM DESCRIPTION
   9.1 DFPBA
10. LISTING
    10.1 DFPBA

1. ABSTRACT
THE THREE PROGRAMS:

DFFPA DFFPB DFFPC

ARE DESIGN TO DETECT AND REPORT LOGIC FAULTS IN THE PDP 11/34 FP11-A FLOATING POINT PROCESSOR. THE DESIGN IS AN ATTEMPT TO REACH ALL ROM STATES, TAKE ALL BRANCH MICRO TESTS (but's) AND VERIFY ALL THE LOGIC. THEY CONSIST OF 155 (0C1) INDIVIDUAL TESTS SEQUENCED TO DETECT AND ATTEMPT TO IDENTIFY FAULTS WITH A MINIMUM HARDWARE OR SOFTWARE LEVEL. THE TESTS ARE PARTITIONED INTO THREE STAND-ALONE PROGRAMS DESCRIBED BELOW.

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

A. DFFPA

DFFPA TESTS:

LDFFS
STFFS
CFCC
SEFF, SETD, SETI AND SETL
STST
LDFD AND LD (ALL SOURCE MODES)
STD (MODE 0 AND 1)
ADDF, ADDD AND SUBD (MOST CONDITIONS)

B. DFFPB

DFFPB TESTS:

ADDF, ADDD AND SUBD (ALL CONDITIONS NOT TESTED IN DFFPA)
CMPO AND CMPF
DIVD AND DIFV
MULD AND MULF
MODD AND MODF

C. DFFPC

DFFPC TESTS:

STF AND STD (ALL MODES)
STCD AND STCDF
CLRD AND CLRF
NEGF AND NEGD
2. REQUIREMENTS

2.1 EQUIPMENT

A PDP 11/34 (WITH OR WITHOUT CONSOLE), LA30 (OR EQUIVALENT), AND AN FPP1-A FLOATING POINT PROCESSOR. NOTE THAT A SPECIAL INTERRUPTS TEST MODULE IS BEING DESIGNED FOR USE IN THE MANUFACTURING ENVIRONMENT. WHEN THIS DEVICE IS PRESENT, THE PROGRAM OFFPB WILL MAKE USE OF IT TO TEST THE FPP INTERRUPT ON BUS REQUEST FUNCTIONS.

2.2 STORAGE

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

2.3 PRELIMINARY PROGRAMS

THES THREE DIAGNOSTICS WILL ASSUME THAT THE PDP 11/34 CENTRAL PROCESSOR IS FAULTLESS. THEREFORE WHEN IN DOUBT RUN THE PDP 11/34 PROCESSOR DIAGNOSTICS BEFORE THESE FPP1-A DIAGNOSTICS.

3. LOADING PROCEDURE

THE PROGRAMS WILL BE SUPPLIED ON THE 11/34 DIAGNOSTIC MEDIA. REFER TO THE YYDP 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 AND
   ERROR SUMMARY WILL BE TYPED AT THE END OF
   EVERY PASS.

5. OPERATING PROCEDURE
   ---------------

5.1 OPERATIONAL SWITCH SETTINGS

THE SWITCH SETTINGS ARE:

<table>
<thead>
<tr>
<th>OCTAL</th>
<th>DECIMAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW&lt;15&gt;=1...</td>
<td>100000</td>
<td>HALT ON ERROR</td>
</tr>
<tr>
<td>SW&lt;14&gt;=1...</td>
<td>40000</td>
<td>LOOP ON CURRENT TEST</td>
</tr>
<tr>
<td>SW&lt;13&gt;=1...</td>
<td>20000</td>
<td>INHIBIT ERROR TYPE OUTS</td>
</tr>
<tr>
<td>SW&lt;12&gt;=1...</td>
<td>10000</td>
<td>INHIBIT 1-BIT TRAPPING</td>
</tr>
<tr>
<td>SW&lt;11&gt;=1...</td>
<td>4000</td>
<td>INHIBIT ITERATIONS</td>
</tr>
<tr>
<td>SW&lt;10&gt;=1...</td>
<td>200</td>
<td>RING TTY BELL ON ERROR</td>
</tr>
<tr>
<td>SW&lt;9&gt;=1....</td>
<td>100</td>
<td>LOOP ON ERROR</td>
</tr>
<tr>
<td>SW&lt;8&gt;=1.....</td>
<td>400</td>
<td>LOOP ON TEST SPECIFIED IN SW&lt;6</td>
</tr>
<tr>
<td>SW&lt;7&gt;=1.....</td>
<td>200</td>
<td>PRINT ERROR SUMMARY EVEN IF</td>
</tr>
<tr>
<td>SW&lt;13&gt;=1...</td>
<td></td>
<td>STILL APPLIES ONLY TO</td>
</tr>
<tr>
<td>SW&lt;10&gt;=1...</td>
<td>200</td>
<td>PROGRAM DIFFER, DESELECT CORRECT</td>
</tr>
<tr>
<td>SW&lt;7&gt;=1.....</td>
<td></td>
<td>INTERRUPT TEST IN PROGRAM DFFPB.</td>
</tr>
<tr>
<td>SW&lt;7&gt;=1.....</td>
<td></td>
<td>NOTE THAT THIS TEST WILL AUTOMATI</td>
</tr>
<tr>
<td>SW&lt;13&gt;=1...</td>
<td></td>
<td>CAL BE DESELECTED BY THE</td>
</tr>
<tr>
<td>SW&lt;10&gt;=1...</td>
<td></td>
<td>ABSENCE OF THE SPECIAL TEST</td>
</tr>
<tr>
<td>SW&lt;7&gt;=1.....</td>
<td></td>
<td>EQUIPMENT DESIGNED TO CONDUCT</td>
</tr>
<tr>
<td>SW&lt;7&gt;=1.....</td>
<td></td>
<td>THIS TEST, IF THIS EQUIPMENT IS</td>
</tr>
<tr>
<td>SW&lt;7&gt;=1.....</td>
<td></td>
<td>NOT INSTALLED THERE IS NO NEED</td>
</tr>
<tr>
<td>SW&lt;7&gt;=1.....</td>
<td></td>
<td>TO DESELECT THIS TEST. THIS APPLI</td>
</tr>
<tr>
<td>SW&lt;7&gt;=1.....</td>
<td></td>
<td>S ONLY TO PROGRAM DFFPB!</td>
</tr>
</tbody>
</table>

6. ERRORS
   -----

6.1 SUMMARIES

IN PROGRAM DFFPA TESTS I AND II HAVE A SPECIAL ERROR
SUMMARY FEATURE. THESE TWO TEST RUN MANY TEST
PATTERNS THROUGH THE LOGIC. AFTER AN ERROR IS
ENCOUNTERED, ONLY THE FIRST FIVE ERRORS ARE REPORTED
6.2 Error Recovery

SW<15:9>=0...

Most errors will cause execution to go to the start of the next test after the message is typed. A few tests are in sections. In these tests an error will cause execution to go to the next section after the message is typed.

SW<15>=1...

The program will halt after typing the error message. Pressing the console continue will cause the program to continue as if SW<15>=0.

7. Restrictions

None

8. Miscellaneous

8.1 Execution Times

Less than 10 seconds for each program on any pass.

8.2 Stack Pointer

The stack pointer is initialized to 1100 in each of the three programs.

8.3 Pass Count

The program makes one pass for each end of pass message typed. The end of pass message describes the total number of passes completed and the total number of errors since the last end of pass message.

8.4 T-bit Trapping

If SW<12>=0 each program will run with trace traps on every other pass, first pass will not enable trace traps. Note SW<12>=1 disables T-bit traps.

8.5 Software Switch Register
EACH OF THE THREE PROGRAMS WILL RUN WITH OR WITHOUT A CONSOLE SWITCH REGISTER. IF A PHYSICAL CONSOLE SWITCH REGISTER IS PRESENT ON THE SYSTEM, THEN THESE PROGRAMS WILL GO AHEAD AND USE IT FOR THE SWITCH 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 C WHILE THE PROGRAM IS RUNNING. THIS CONTROL C 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 CARDS RETURN THEN THE PROGRAM WILL RESUME TESTING AT THE SAME POINT AT WHICH IT LEFT OFF WHEN THE USER TYPED CONTROL C. NOTE THAT WHEN NOT RUNNING UNDER ACT, APT OR CHAIN THE USER WILL BE ASKED FOR A SOFTWARE SWITCH REGISTER VALUE AFTER LOADING ADDRESS 200 AND STARTING THE PROGRAM THE FIRST TIME THE PROGRAM IS RUN AFTER LOADING (ONLY IF NO CONSOLE SWITCH REGISTER IS ON THE SYSTEM).

8.6 INTERRUPTS TEST

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

8.7 ACT, APT AND XXXP COMPATIBILITY

THESE PROGRAMS ARE FULLY COMPATIBLE WITH:

ACT

XXDP MONITOR AND CHAIN PROGRAMS.

9. PROGRAM DESCRIPTION
TEST 1  
ROUND\TRUNK TEST  

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

TEST 2  
OVER\UNDER TEST  

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

TEST 3  
LDCFD AND LDCDF TEST  

THIS IS A TEST OF LDCFD AND LDCDF.

TEST 4  
CMPO TEST  

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

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

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

TEST 6  
DIVF TEST  

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

TEST 7  
DIVD TEST  

THIS IS A TEST OF THE DIVD INSTRUCTION. NOTE THAT A SUBROUTINE IS USED TO SET UP THE OPERANDS, EXECUTE THE INSTRUCTION AND CHECK THE RESULTS.
TEST 10 MULF TEST
----
THIS IS A TEST OF THE MULF INSTRUCTION. IT MAKES USE OF A SUBROUTINE TO SET UP THE OPERANDS, EXECUTE THE MULF INSTRUCTION AND CHECK THE RESULTS.

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

TEST 12 UNDERFLOW, USING MULF WITH TRAPS DISABLED, TEST
----
THIS IS A TEST OF THE OVERFLOW AND UNDERFLOW CONDITIONS USING THE MULF INSTRUCTION WITH TRAPS DISABLED. NOTE THAT A SUBROUTINE IS USED TO SET UP THE OPERANDS, EXECUTE THE MULF INSTRUCTION AND CHECK THE RESULTS.

TEST 13 UNDERFLOW, USING MULD WITH TRAPS DISABLED, TEST
----
THIS IS A TEST OF THE OVERFLOW AND UNDERFLOW CONDITIONS THAT CAN ARISE USING THE MULD INSTRUCTION WITH TRAPS DISABLED. A SUBROUTINE IS USED TO SET UP THE OPERANDS, EXECUTE THE MULD INSTRUCTION AND CHECK THE RESULTS.

TEST 14 UNDERFLOW, USING MULF WITH TRAPS ENABLED, TEST
----
THIS IS A TEST OF THE UNDERFLOW AND OVERFLOW CONDITIONS THAT CAN OCCUR USING THE MULF INSTRUCTION. A SUBROUTINE IS CALLED TO SET UP THE OPERANDS, EXECUTE THE MULF INSTRUCTION AND CHECK THE RESULTS. HERE THE PARTICULAR INTERRUPT EITHER OVERFLOW OR UNDERFLOW, IS ENABLED SO A TRAP SHOULD OCCUR.

TEST 15 UNDERFLOW, USING MULD WITH TRAPS ENABLED, TEST
----
THIS IS A TEST OF THE OVERFLOW AND UNDERFLOW CONDITIONS USING THE MULD INSTRUCTION WITH TRAPS ENABLED. A SUBROUTINE IS USED TO SET UP THE OPERANDS, EXECUTE THE MULD INSTRUCTION AND CHECK THE RESULTS.

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

TEST 17 MODD TEST

This is a test of the MODD instruction. It makes use of a subroutine to set up the arguments, execute the instruction, and check the results.

TEST 20 UNDER/OVER FLOW, USING MODD WITH TRAPS DISABLED, TEST

This is a test of the MODD overflow and underflow conditions. It makes use of a subroutine to set up the operands, execute the MODD instruction, and check the results. Traps are disabled during this test.

TEST 21 UNDER/OVER FLOW, USING MODD WITH TRAPS DISABLED, TEST

This is a test of the MODD instructions' overflow and underflow conditions. A subroutine is used to set up the operands, execute the MODD instruction, and check the results.

TEST 22 INTERRUPT CORRECT FLOWS TEST

This is a test of the 'correct' flows. This part of the micro code has as its purpose insuring that interrupt requests made during certain lengthy FPP instructions get honored. This is done in a way such that if an interrupt request occurs during one of these instructions, the state of that instruction's execution will be the same as if that instruction had never been fetched and its execution never started. Thus, the micro code will restore all registers, back up the PC and leave the FPP and ACO through ACS unmodified.

The instructions for which this is necessary are:

ADD (or SUB)
DIV
MUL
MOD
(Both Double and Floating)

All addressing modes will be tried with the ADDD instruction. Then each of the other instructions will be tried using mode 1. Note that this test needs a special interrupt module, which will probably only be present in DEC's manufacturing environment, to run. This special equipment is designed to raise an interrupt request in the processor if a bit is set in its status register and only when an FPP instruction is encountered.
Therefore this test will be run conditionally
dependent upon whether or not the Status Register
of the test equipment times out when referenced.
This test can also be deselected by turning switch 7
of the switch register (Physical or Virtual) on.
The test assumes that the test equipment's Status
Register is at location 7777774 (Note that all
references to this location are made indirect
through this program's Location Corint, so that if
the user has modified the test equipment's Status
Register to respond to a different address location
Corint must be made to contain that Status
Register's new address). This program assumes that
the trap vector for the test equipment is 110.
Again note that all references to this trap vector
are indirect. Through this program's Location Cortrap
(if the test equipment is made to trap to a
different vector location Cortrap must contain the
address of this vector).

10. Listing

\$00266
\$00002

\$.LIST ME
\$.LIST MD,MC,CMD
.ENABL ABS

.TITLE MAINDEC-11-DFFPB-A  PDP 11/34 FPP DIAGNOSTIC PART 2

COPYRIGHT (C) SEP 1976
DIGITAL EQUIPMENT CORP.

MAYNARD, MASS. 01754

PROG BY ANTHONY S. VEZZA

THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYMSAC
PACKAGE (MAINDEC-11-DZ0AC-C2), SEPT 14, 1976.

$TN=1
$SWR=160000  ;;HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYPOUT

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

SBBT. BASIC DEFINITIONS

INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***

STACK= 1100
EQUIV EM,ERROR  ;;BASIC DEFINITION OF ERROR CALL
EQUIV IOT,SCOPE  ;;BASIC DEFINITION OF SCOPE CALL

MISCELLANEOUS DEFINITIONS

HT= 11  ;;CODE FOR HORIZONTAL TAB
LF= 12  ;;CODE FOR LINE FEED
CR= 15  ;;CODE FOR CARRIAGE RETURN
CRLF= 200  ;;CODE FOR CARRIAGE RETURN-LINE FEED
PS= 177776  ;;PROCESSOR STATUS WORD
EQUIV PS,PSW
STKLMT= 177774  ;;STACK LIMIT REGISTER
NO1

MAINDEC-11-DDFPB-A  PDP 11/34 FPP DIAGNOSTIC PART 2 MACYII 27/1006  01-NOV-76  21:12  PAGE 13

BASIC DEFINITIONS

673  177772  ; PROGRAM INTERRUPT REQUEST REGISTER
674  177570  ; HARDWARE SWITCH REGISTER
675  177570  ; HARDWARE DISPLAY REGISTER

; GENERAL PURPOSE REGISTER DEFINITIONS
676
677  678  000000  ; GENERAL REGISTER
679  679  000001
680  681  000002
682  683  000003
684  685  000004
686  687  000005
688  689  000006
690  691  000007
692  693  000008
694  695  000009
696  697  000010
698  699  000011

; PRIORITY LEVEL DEFINITIONS
700
701  702  000000  ; PRIORITY LEVEL 0
703  704  000040
705  706  000100
707  708  000140
709  710  000200
711  712  000260
713  714  000300
715  716  000340
717

; 'SWITCH REGISTER' SWITCH DEFINITIONS
718  719  720  721  722  723  724  725  726  727  728  000000  ; PRIORITY LEVEL 1
729  730  731  732  733  734  735  736  737  738  100000
739  740  741  742  743  744  745  746  747  748  749  800000
750  751  752  753  754  755  756  757  758  759  760

; EQUIV SW0, SW1
761
762  763  764  765  766  767  768  769  770  771  772
773

; DATA BIT DEFINITIONS (BIT00 TO BIT15)
774  775  776  777

BIT15 = 100000
BIT14= 40000
BIT13= 20000
BIT12= 10000
BIT11= 4000
BIT10= 2000
BIT09= 1000
BIT08= 400
BIT07= 200
BIT06= 100
BIT05= 40
BIT04= 20
BIT03= 10
BIT02= 4
BIT01= 2
BIT00= 1

.EQUIV BIT09,BIT9
.EQUIV BIT08,BIT8
.EQUIV BIT07,BIT7
.EQUIV BIT06,BIT6
.EQUIV BIT05,BIT5
.EQUIV BIT04,BIT4
.EQUIV BIT03,BIT3
.EQUIV BIT02,BIT2
.EQUIV BIT01,BIT1
.EQUIV BIT00,BITO

 Femme "CPU" TRAP VECTOR ADDRESSES

ERRVEC= 4  
RESVEC= 10  
TBITVEC=14  
TBTRVEC= 14  
BPTVEC= 14  
INPUT-OUTPUT TRAP (IOT) **Scope**
POWERF= 24  
EMTVEC= 30  
"TRAP" TRAP
EMTVEC= 34  
"TRAP" TRAP
EMTVEC= 64  
TTY KEYBOARD VECTOR
EMTVEC= 64  
TTY PRINTER VECTOR
PIRQVEC=240  

.SBTTL FFP REGISTER DEFINITIONS
AC0= 20
AC1= 21
AC2= 22
AC3= 23
AC4= 24
AC5= 25
AC6= 26
AC7= 27

.SBTTL TRAP CATCHER

000000

;#ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A "#2,HALT"
;#SEQUENCE TO CATCH ILLEGAL TRAPS AND INTURPTS
;#LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS

000174
C02

DISPREG: .WORD 0 ::SOFTWARE DISPLAY REGISTER
SREG: .WORD 0 ::SOFTWARE SWITCH REGISTER
SBTYL STARTING ADDRESS(ES)

JMP @START ::JUMP TO STARTING ADDRESS OF PROGRAM
<table>
<thead>
<tr>
<th>シンタグ</th>
<th>サイズ</th>
<th>内容</th>
</tr>
</thead>
<tbody>
<tr>
<td>001500</td>
<td>000000</td>
<td>コモンタグ</td>
</tr>
<tr>
<td>STSMM</td>
<td>WORD</td>
<td>テストナンバー</td>
</tr>
<tr>
<td>SFRDLG</td>
<td>BYTE</td>
<td>エラーフラグ</td>
</tr>
<tr>
<td>SCNT</td>
<td>WORD</td>
<td>パブテストイテレーション</td>
</tr>
<tr>
<td>SLPADR</td>
<td>WORD</td>
<td>スコープループアドレス</td>
</tr>
<tr>
<td>SLPEPR</td>
<td>WORD</td>
<td>スコープリターンフォーバラス</td>
</tr>
<tr>
<td>SRTTL</td>
<td>WORD</td>
<td>エラーベイテクス</td>
</tr>
<tr>
<td>SLTMB</td>
<td>BYTE</td>
<td>アイテムコントロールバイテ</td>
</tr>
<tr>
<td>SERRAX</td>
<td>BYTE</td>
<td>マックスエラーパテスト</td>
</tr>
<tr>
<td>SERRPC</td>
<td>WORD</td>
<td>パックエラーバラス</td>
</tr>
<tr>
<td>SIGDR</td>
<td>WORD</td>
<td>'GOOD'バラス</td>
</tr>
<tr>
<td>SGDAT</td>
<td>WORD</td>
<td>'GOOD'データ</td>
</tr>
<tr>
<td>SBDAT</td>
<td>WORD</td>
<td>'BAD'データ</td>
</tr>
<tr>
<td>SBOAT</td>
<td>WORD</td>
<td>預約--未使用</td>
</tr>
<tr>
<td>SAVOR</td>
<td>BYTE</td>
<td>自動モードインディケータ</td>
</tr>
<tr>
<td>SINTAG</td>
<td>BYTE</td>
<td>インタリュートモードインディケータ</td>
</tr>
<tr>
<td>SDR</td>
<td>WORD</td>
<td>スイッチレジスタ</td>
</tr>
<tr>
<td>DSWRA</td>
<td>WORD</td>
<td>DDIスリップレジスタ</td>
</tr>
<tr>
<td>SKS</td>
<td>000000</td>
<td>TTYキーボード設定</td>
</tr>
<tr>
<td>SKSB</td>
<td>177562</td>
<td>TTYキーボードバッファー</td>
</tr>
<tr>
<td>SKSB</td>
<td>177564</td>
<td>TTYプリンターステータスレジスタアドレス</td>
</tr>
<tr>
<td>$TPB</td>
<td>177566</td>
<td>TTYプリンタバッファレジスタアドレス</td>
</tr>
<tr>
<td>$SNUL</td>
<td>000000</td>
<td>ユニットチャレサバラス</td>
</tr>
<tr>
<td>$Fills</td>
<td>000000</td>
<td>コンテンツフォーフィラー</td>
</tr>
<tr>
<td>$DFLGL</td>
<td>12</td>
<td>「TAMRO」バラス</td>
</tr>
<tr>
<td>$REG0</td>
<td>000000</td>
<td>コンテナ (0)</td>
</tr>
<tr>
<td>$REG1</td>
<td>000000</td>
<td>コンテナ (1)</td>
</tr>
<tr>
<td>$REG2</td>
<td>000000</td>
<td>コンテナ (2)</td>
</tr>
<tr>
<td>$REG3</td>
<td>000000</td>
<td>コンテナ (3)</td>
</tr>
<tr>
<td>$REG4</td>
<td>000000</td>
<td>コンテナ (4)</td>
</tr>
<tr>
<td>$REG5</td>
<td>000000</td>
<td>コンテナ (5)</td>
</tr>
<tr>
<td>$REG6</td>
<td>000000</td>
<td>コンテナ (6)</td>
</tr>
<tr>
<td>$REG7</td>
<td>000000</td>
<td>コンテナ (7)</td>
</tr>
<tr>
<td>$REG8</td>
<td>000000</td>
<td>コンテナ (8)</td>
</tr>
<tr>
<td>$REG9</td>
<td>000000</td>
<td>コンテナ (9)</td>
</tr>
<tr>
<td>$REG10</td>
<td>000000</td>
<td>コンテナ (10)</td>
</tr>
<tr>
<td>$REG11</td>
<td>000000</td>
<td>コンテナ (11)</td>
</tr>
<tr>
<td>$REG12</td>
<td>000000</td>
<td>コンテナ (12)</td>
</tr>
<tr>
<td>$REG13</td>
<td>000000</td>
<td>コンテナ (13)</td>
</tr>
<tr>
<td>$REG14</td>
<td>000000</td>
<td>コンテナ (14)</td>
</tr>
<tr>
<td>$REG15</td>
<td>000000</td>
<td>コンテナ (15)</td>
</tr>
<tr>
<td>$REG16</td>
<td>000000</td>
<td>コンテナ (16)</td>
</tr>
<tr>
<td>$REG17</td>
<td>000000</td>
<td>コンテナ (17)</td>
</tr>
<tr>
<td>$REG18</td>
<td>000000</td>
<td>コンテナ (18)</td>
</tr>
<tr>
<td>$REG19</td>
<td>000000</td>
<td>コンテナ (19)</td>
</tr>
<tr>
<td>$REG20</td>
<td>000000</td>
<td>コンテナ (20)</td>
</tr>
</tbody>
</table>
E02

**COMMON TAGS**

- SREG1: .WORD O
  - CONTAINS (SREG1+42)
- SREG2: .WORD O
  - CONTAINS (SREG1+44)
- SREG3: .WORD O
  - CONTAINS (SREG1+45)
- SREG4: .WORD O
  - USER DEFINED
- SREG5: .WORD O
  - USER DEFINED
- SREG6: .WORD O
  - USER DEFINED
- SREG7: .WORD O
  - USER DEFINED
- SREG8: .WORD O
  - USER DEFINED
- SREG9: .WORD O
  - USER DEFINED
- SREG10: .WORD O
  - USER DEFINED
- SREG11: .WORD O
  - USER DEFINED
- SREG12: .WORD O
  - USER DEFINED
- SREG13: .WORD O
  - USER DEFINED
- SREG14: .WORD O
  - USER DEFINED
- SREG15: .WORD O
  - USER DEFINED
- SREG16: .WORD O
  - USER DEFINED
- SREG17: .WORD O
  - USER DEFINED
- SREG18: .WORD O
  - USER DEFINED
- SREG19: .WORD O
  - USER DEFINED
- SREG20: .WORD O
  - USER DEFINED
- SREG21: .WORD O
  - USER DEFINED
- SREG22: .WORD O
  - USER DEFINED
- SREG23: .WORD O
  - USER DEFINED
- STMP0: .WORD O
  - USER DEFINED
- STMP1: .WORD O
  - USER DEFINED
- STMP2: .WORD O
  - USER DEFINED
- STMP3: .WORD O
  - USER DEFINED
- STMP4: .WORD O
  - USER DEFINED
- STMP5: .WORD O
  - USER DEFINED
- STMP6: .WORD O
  - USER DEFINED
- STMP7: .WORD O
  - USER DEFINED
- STMP8: .WORD O
  - USER DEFINED
- STMP9: .WORD O
  - USER DEFINED
- STMP10: .WORD O
  - USER DEFINED
- STMP11: .WORD O
  - USER DEFINED
- STMP12: .WORD O
  - USER DEFINED
- STMP13: .WORD O
  - USER DEFINED
- STMP14: .WORD O
  - USER DEFINED
- STMP15: .WORD O
  - USER DEFINED
- STMP16: .WORD O
  - USER DEFINED
- STMP17: .WORD O
  - USER DEFINED
- STMP18: .WORD O
  - USER DEFINED
- STMP19: .WORD O
  - USER DEFINED
- STMP20: .WORD O
  - USER DEFINED
- STMP21: .WORD O
  - USER DEFINED
- STMP22: .WORD O
  - USER DEFINED
- STMP23: .WORD O
  - USER DEFINED
- STMP24: .WORD O
  - USER DEFINED
- STMP25: .WORD O
  - USER DEFINED
- STMP26: .WORD O
  - USER DEFINED
- STMP27: .WORD O
  - USER DEFINED
- STMP28: .WORD O
  - USER DEFINED
- STMP29: .WORD O
  - USER DEFINED
- STMP30: .WORD O
  - USER DEFINED
- STMP31: .WORD O
  - USER DEFINED
- TIMES: 0
  - MAX. NUMBER OF ITERATIONS
- ESCAPE: O
  - ESCAPE ON ERROR ADDRESS
- $BELL: .ASCIIZ (207, 377, 377)
  - CODE FOR BELL
- $?1: .ASCIIZ O
  - QUESTION MARK
- $?RI: .ASCIIZ O
  - CARRIAGE RETURN
- SLF: .ASCIIZ O
  - LINE FEED

**SBJTL APT MAILBOX-ETABLE**

**EVEN**

- Email: .BYTE AMSGTY
  - APT MAILBOX MESSAGE TYPE CODE
- FATAL: .WORD AMFLAT
  - FATAL ERROR NUMBER
- TESTN: .WORD AMTEST
  - TEST NUMBER
- SPS: .WORD AMPASS
  - PASS COUNT
- $DEVCT: .WORD AMDEVCT
  - DEVICE COUNT
- $UNIT: .WORD AMUNIT
  - I/O UNIT NUMBER
- $SSMIG: .WORD AMMSG
  - MESSAGE ADDRESS
- $SSMGL: .WORD AMMSGL
  - MESSAGE LENGTH
- $TABLE: .WORD AMTABLE
  - APT ENVIRONMENT TABLE
- $ENV: .BYTE AEMAP
  - ENVIRONMENT BYTE
- $FAP: .BYTE AENMP
  - ENVIRONMENT MODE Bits
- $ASREG: .WORD AASREG
  - APT SWITCH REGISTER
- $ASW: .WORD AASW
  - USER SWITCHES
- $CPUTP: .WORD ACPUOP
  - CPU TYPE OPTIONS
  - BIT 15-11=CPU TYPE
  - BIT 10=REAL TIME CLOCK
  - BIT 9=FLOATING POINT PROCESSOR
  - BIT 8=MEMORY MANAGEMENT
- $MAMS1: .BYTE AMAMS1
  - HIGH ADDRESS, M.S. BYTE
SBTL: 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 $ITEM$. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
* NOTE: IF $ITEM$ IS 0 THE ONLY PERTINENT DATA IS ($ERRTB$).
* NOTE: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

* EM  . WORD EM1, DH1, DT1, DF1
* DH  . WORD EM2, DH2, DT2, DF2
* DT  . WORD EM3, DH3, DT3, DF3
* DF  . WORD EM4, DH4, DT4, DF4

001442
001444 037660 366216 370540
001450 070012
001452 037712 066306 073576
001460 070030
001462 037716 366216 070540
001470 070012
001472 040053 366216 070540
001500 070012
001502 040160 366216 070540
001510 070012
001512 040265 066216 070540
001520 070012
001522 040377 066216 070540
001530 070012
001532 040477 066216 070540
001540 070012
001542 040564 066216 070540
001550 070012
001552 040713 066216 070540
001560 070012
001562 041022 066216 070540
001570 070012
001572 041336 066216 070540
001600 070012
001602 041250 066216 070540
001610 070012
001612 041362 066216 070540
<table>
<thead>
<tr>
<th>ITEM</th>
<th>Address 1</th>
<th>Address 2</th>
<th>Address 3</th>
<th>Address 4</th>
<th>Address 5</th>
<th>Word 1</th>
<th>Word 2</th>
<th>Word 3</th>
<th>Word 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>37347</td>
<td>370647</td>
<td>066347</td>
<td>370624</td>
<td></td>
<td>EM17, DH17, DT17, DF17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>066216</td>
<td>070844</td>
<td></td>
<td></td>
<td></td>
<td>EM20, DH20, DT20, DF20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>065422</td>
<td>370666</td>
<td></td>
<td></td>
<td></td>
<td>EM21, DH21, DT21, DF21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>066511</td>
<td>070710</td>
<td></td>
<td></td>
<td></td>
<td>EM22, DH22, DT22, DF22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>066511</td>
<td>070710</td>
<td></td>
<td></td>
<td></td>
<td>EM23, DH23, DT23, DF23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>066216</td>
<td>070726</td>
<td></td>
<td></td>
<td></td>
<td>EM24, DH24, DT24, DF24</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>066216</td>
<td>070726</td>
<td></td>
<td></td>
<td></td>
<td>EM25, DH25, DT25, DF25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>066216</td>
<td>070726</td>
<td></td>
<td></td>
<td></td>
<td>EM26, DH26, DT26, DF26</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>066216</td>
<td>070726</td>
<td></td>
<td></td>
<td></td>
<td>EM27, DH27, DT27, DF27</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>066216</td>
<td>070726</td>
<td></td>
<td></td>
<td></td>
<td>EM30, DH30, DT30, DF30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>066216</td>
<td>070726</td>
<td></td>
<td></td>
<td></td>
<td>EM31, DH31, DT31, DF31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>066216</td>
<td>070726</td>
<td></td>
<td></td>
<td></td>
<td>EM32, DH32, DT32, DF32</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>066216</td>
<td>070726</td>
<td></td>
<td></td>
<td></td>
<td>EM33, DH33, DT33, DF33</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>066216</td>
<td>070726</td>
<td></td>
<td></td>
<td></td>
<td>EM34, DH34, DT34, DF34</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>066216</td>
<td>070726</td>
<td></td>
<td></td>
<td></td>
<td>EM35, DH35, DT35, DF35</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>066216</td>
<td>070726</td>
<td></td>
<td></td>
<td></td>
<td>EM36, DH36, DT36, DF36</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>066216</td>
<td>070726</td>
<td></td>
<td></td>
<td></td>
<td>EM37, DH37, DT37, DF37</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>066216</td>
<td>070726</td>
<td></td>
<td></td>
<td></td>
<td>EM40, DH40, DT40, DF40</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>09000</td>
<td>070061</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADDR</td>
<td>DATA</td>
<td>PMT</td>
<td>SPT</td>
<td>COMPL</td>
<td>CO</td>
<td>TEXT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>-----</td>
<td>-----</td>
<td>-------</td>
<td>---</td>
<td>------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02242</td>
<td>042631</td>
<td>66216</td>
<td>07076</td>
<td>WORD</td>
<td>EM41, DH41, DT41, DF41</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02250</td>
<td>070261</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02252</td>
<td>042704</td>
<td>66216</td>
<td>07076</td>
<td>ITEM 42</td>
<td>EM42, DH42, DT42, DF42</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02260</td>
<td>070061</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02262</td>
<td>043035</td>
<td>66216</td>
<td>07076</td>
<td>ITEM 43</td>
<td>EM43, DH43, DT43, DF43</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02270</td>
<td>070061</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02272</td>
<td>043166</td>
<td>66216</td>
<td>07076</td>
<td>ITEM 44</td>
<td>EM44, DH44, DT44, DF44</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02280</td>
<td>070061</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02290</td>
<td>043234</td>
<td>66216</td>
<td>07076</td>
<td>ITEM 45</td>
<td>EM45, DH45, DT45, DF45</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02298</td>
<td>070061</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02292</td>
<td>043307</td>
<td>66216</td>
<td>07076</td>
<td>ITEM 46</td>
<td>EM46, DH46, DT46, DF46</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02300</td>
<td>070105</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02302</td>
<td>043364</td>
<td>66216</td>
<td>07076</td>
<td>ITEM 47</td>
<td>EM47, DH47, DT47, DF47</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02308</td>
<td>070105</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02312</td>
<td>043520</td>
<td>66216</td>
<td>07076</td>
<td>ITEM 50</td>
<td>EM50, DH50, DT50, DF50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02318</td>
<td>070105</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02318</td>
<td>043573</td>
<td>66216</td>
<td>07076</td>
<td>ITEM 51</td>
<td>EM51, DH51, DT51, DF51</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02320</td>
<td>070105</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02322</td>
<td>043641</td>
<td>66216</td>
<td>07076</td>
<td>ITEM 52</td>
<td>EM52, DH52, DT52, DF52</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02324</td>
<td>070105</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02328</td>
<td>051471</td>
<td>66216</td>
<td>071054</td>
<td>ITEM 53</td>
<td>EM53, DH53, DT53, DF53</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02330</td>
<td>070201</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02332</td>
<td>051522</td>
<td>66216</td>
<td>071054</td>
<td>ITEM 54</td>
<td>EM54, DH54, DT54, DF54</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02334</td>
<td>070201</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02338</td>
<td>051437</td>
<td>66216</td>
<td>071054</td>
<td>ITEM 55</td>
<td>EM55, DH55, DT55, DF55</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02340</td>
<td>070201</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02342</td>
<td>051552</td>
<td>66216</td>
<td>071054</td>
<td>ITEM 56</td>
<td>EM56, DH56, DT56, DF56</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02344</td>
<td>070201</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02346</td>
<td>051643</td>
<td>66216</td>
<td>071054</td>
<td>ITEM 57</td>
<td>EM57, DH57, DT57, DF57</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02348</td>
<td>070201</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02352</td>
<td>051733</td>
<td>66216</td>
<td>071054</td>
<td>ITEM 60</td>
<td>EM60, DH60, DT60, DF60</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02354</td>
<td>070201</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02356</td>
<td>052035</td>
<td>66216</td>
<td>071054</td>
<td>ITEM 61</td>
<td>EM61, DH61, DT61, DF61</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02358</td>
<td>070201</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02362</td>
<td>052231</td>
<td>66216</td>
<td>071054</td>
<td>ITEM 62</td>
<td>EM62, DH62, DT62, DF62</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02364</td>
<td>070201</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02366</td>
<td>052425</td>
<td>66216</td>
<td>071054</td>
<td>ITEM 63</td>
<td>EM63, DH63, DT63, DF63</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02368</td>
<td>070201</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
J02

ERROR POINTER TABLE

000272 052536 066216 071054  .ITEM 64  .WORD  EM64,DM64,DT64,DF64
000272 052536 066216 071054  .ITEM 65  .WORD  EM65,DM65,DT65,DF65
0002302 052655 066216 071054  .ITEM 66  .WORD  EM66,DM66,DT66,DF66
0002310 070201  .ITEM 67  .WORD  EM67,DM67,DT67,DF67
0002312 052770 066216 071054  .ITEM 70  .WORD  EM70,DM70,DT70,DF70
0002302 052655 066216 071054  .ITEM 71  .WORD  EM71,DM71,DT71,DF71
0002310 070201  .ITEM 72  .WORD  EM72,DM72,DT72,DF72
0002312 052770 066216 071054  .ITEM 73  .WORD  EM73,DM73,DT73,DF73
0002302 052655 066216 071054  .ITEM 74  .WORD  EM74,DM74,DT74,DF74
0002310 070201  .ITEM 75  .WORD  EM75,DM75,DT75,DF75
0002312 052770 066216 071054  .ITEM 76  .WORD  EM76,DM76,DT76,DF76
0002302 052655 066216 071054  .ITEM 77  .WORD  EM77,DM77,DT77,DF77
0002310 070201  .ITEM 100  .WORD  EM100,DM100,DT100,DF100
0002440 070233  .ITEM 101  .WORD  EM101,DM101,DT101,DF101
0002440 070233  .ITEM 102  .WORD  EM102,DM102,DT102,DF102
0002440 070233  .ITEM 103  .WORD  EM103,DM103,DT103,DF103
0002440 070233  .ITEM 104  .WORD  EM104,DM104,DT104,DF104
0002440 070233  .ITEM 105  .WORD  EM105,DM105,DT105,DF105
0002440 070233  .ITEM 106  .WORD  EM106,DM106,DT106,DF106
ITEM 107
.WORD EM107, DH107, DT107, DF107

ITEM 110
.WORD EM10, DH110, DT110, DF110

ITEM 111
.WORD EM111, DH111, DT111, DF111

ITEM 112
.WORD EM112, DH112, DT112, DF112

ITEM 113
.WORD EM113, DH113, DT113, DF113

ITEM 114
.WORD EM114, DH114, DT114, DF114

ITEM 115
.WORD EM115, DH115, DT115, DF115

ITEM 116
.WORD EM116, DH116, DT116, DF116

ITEM 117
.WORD EM117, DH117, DT117, DF117

ITEM 120
.WORD EM120, DH120, DT120, DF120

ITEM 121
.WORD EM121, DH121, DT121, DF121

ITEM 122
.WORD EM122, DH122, DT122, DF122

ITEM 123
.WORD EM123, DH123, DT123, DF123

ITEM 124
.WORD EM124, DH124, DT124, DF124

ITEM 125
.WORD EM125, DH125, DT125, DF125

ITEM 126
.WORD EM126, DH126, DT126, DF126

ITEM 127
.WORD EM127, DH127, DT127, DF127

ITEM 130
.WORD EM130, DH130, DT130, DF130

ITEM 131
<table>
<thead>
<tr>
<th>Address</th>
<th>Field1</th>
<th>Field2</th>
<th>Field3</th>
<th>Field4</th>
<th>Field5</th>
<th>Field6</th>
<th>Field7</th>
<th>Field8</th>
</tr>
</thead>
<tbody>
<tr>
<td>002742</td>
<td>095573</td>
<td>066557</td>
<td>071000</td>
<td>.WORD</td>
<td>EM131, DH131, DT131, DF131</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>002750</td>
<td>070155</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>002752</td>
<td>095731</td>
<td>066216</td>
<td>070726</td>
<td>.WORD</td>
<td>EM132, DH132, DT132, DF132</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>002760</td>
<td>070105</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>002762</td>
<td>096067</td>
<td>066557</td>
<td>071000</td>
<td>.WORD</td>
<td>EM133, DH133, DT133, DF133</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>002770</td>
<td>070155</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>002772</td>
<td>096225</td>
<td>066557</td>
<td>071000</td>
<td>.WORD</td>
<td>EM134, DH134, DT134, DF134</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003000</td>
<td>070155</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003002</td>
<td>096362</td>
<td>066216</td>
<td>070726</td>
<td>.WORD</td>
<td>EM135, DH135, DT135, DF135</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003010</td>
<td>070105</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003012</td>
<td>096517</td>
<td>066557</td>
<td>071000</td>
<td>.WORD</td>
<td>EM136, DH136, DT136, DF136</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003020</td>
<td>070155</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003022</td>
<td>096654</td>
<td>066557</td>
<td>071000</td>
<td>.WORD</td>
<td>EM137, DH137, DT137, DF137</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003030</td>
<td>070131</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003032</td>
<td>096742</td>
<td>066557</td>
<td>071000</td>
<td>.WORD</td>
<td>EM140, DH140, DT140, DF140</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003040</td>
<td>070131</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003042</td>
<td>093731</td>
<td>066557</td>
<td>071000</td>
<td>.WORD</td>
<td>EM141, DH141, DT141, DF141</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003050</td>
<td>070131</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003052</td>
<td>094006</td>
<td>066557</td>
<td>071000</td>
<td>.WORD</td>
<td>EM142, DH142, DT142, DF142</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003060</td>
<td>070131</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003062</td>
<td>094731</td>
<td>066557</td>
<td>071000</td>
<td>.WORD</td>
<td>EM143, DH143, DT143, DF143</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003070</td>
<td>070131</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003072</td>
<td>097107</td>
<td>066557</td>
<td>071000</td>
<td>.WORD</td>
<td>EM144, DH144, DT144, DF144</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003100</td>
<td>070131</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003102</td>
<td>097166</td>
<td>066216</td>
<td>070726</td>
<td>.WORD</td>
<td>EM145, DH145, DT145, DF145</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003110</td>
<td>070061</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003112</td>
<td>097333</td>
<td>066216</td>
<td>070726</td>
<td>.WORD</td>
<td>EM146, DH146, DT146, DF146</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003120</td>
<td>070061</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003122</td>
<td>097500</td>
<td>066216</td>
<td>070726</td>
<td>.WORD</td>
<td>EM147, DH147, DT147, DF147</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003130</td>
<td>070061</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003132</td>
<td>097644</td>
<td>066216</td>
<td>070726</td>
<td>.WORD</td>
<td>EM150, DH150, DT150, DF150</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003140</td>
<td>070061</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003142</td>
<td>050010</td>
<td>066557</td>
<td>071000</td>
<td>.WORD</td>
<td>EM151, DH151, DT151, DF151</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003150</td>
<td>070155</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003152</td>
<td>050076</td>
<td>066557</td>
<td>071000</td>
<td>.WORD</td>
<td>EM152, DH152, DT152, DF152</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003160</td>
<td>070155</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003162</td>
<td>095144</td>
<td>066557</td>
<td>071000</td>
<td>.WORD</td>
<td>EM153, DH153, DT152, DF153</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003170</td>
<td>070155</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
NO2

;ITEM 177
.WORD EM177, DH177, DT177, DF177

;ITEM 200
.WORD EM200, DH200, DT200, DF200

;ITEM 201
.WORD EM201, DH201, DT201, DF201

;ITEM 202
.WORD EM202, DH202, DT202, DF202

;ITEM 203
.WORD EM203, DH203, DT203, DF203

;ITEM 204
.WORD EM204, DH204, DT204, DF204

;ITEM 205
.WORD EM205, DH205, DT205, DF205

;ITEM 206
.WORD EM206, DH206, DT206, DF206

;ITEM 207
.WORD EM207, DH207, DT207, DF207

;ITEM 210
.WORD EM210, DH210, DT210, DF210

;ITEM 211
.WORD EM211, DH211, DT211, DF211

;ITEM 212
.WORD EM212, DH212, DT212, DF212

;ITEM 213
.WORD EM213, DH213, DT213, DF213

;ITEM 214
.WORD EM214, DH214, DT214, DF214

;ITEM 215
.WORD EM215, DH215, DT215, DF215

;ITEM 216
.WORD EM216, DH216, DT216, DF216

;ITEM 217
.WORD EM217, DH217, DT217, DF217

;ITEM 220
.WORD EM220, DH220, DT220, DF220

;ITEM 221
<table>
<thead>
<tr>
<th>Offset</th>
<th>Address</th>
<th>Value</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>004072</td>
<td>06443</td>
<td>067300</td>
<td>071426</td>
</tr>
<tr>
<td>004130</td>
<td>070434</td>
<td>067300</td>
<td>071426</td>
</tr>
<tr>
<td>004102</td>
<td>064447</td>
<td>067300</td>
<td>071426</td>
</tr>
<tr>
<td>004110</td>
<td>070444</td>
<td></td>
<td></td>
</tr>
<tr>
<td>004112</td>
<td>063341</td>
<td>066216</td>
<td>070326</td>
</tr>
<tr>
<td>004120</td>
<td>0703105</td>
<td></td>
<td></td>
</tr>
<tr>
<td>004122</td>
<td>064560</td>
<td>067645</td>
<td>071514</td>
</tr>
<tr>
<td>004130</td>
<td>070474</td>
<td></td>
<td></td>
</tr>
<tr>
<td>004132</td>
<td>064614</td>
<td>067712</td>
<td>071532</td>
</tr>
<tr>
<td>004140</td>
<td>070474</td>
<td></td>
<td></td>
</tr>
<tr>
<td>004142</td>
<td>064646</td>
<td>067712</td>
<td>071532</td>
</tr>
<tr>
<td>004150</td>
<td>070474</td>
<td></td>
<td></td>
</tr>
<tr>
<td>004152</td>
<td>064701</td>
<td>066306</td>
<td>071544</td>
</tr>
<tr>
<td>004160</td>
<td>070502</td>
<td></td>
<td></td>
</tr>
<tr>
<td>004162</td>
<td>064772</td>
<td>066216</td>
<td>071556</td>
</tr>
<tr>
<td>004170</td>
<td>070506</td>
<td></td>
<td></td>
</tr>
<tr>
<td>004172</td>
<td>065056</td>
<td>066216</td>
<td>071556</td>
</tr>
<tr>
<td>004200</td>
<td>070506</td>
<td></td>
<td></td>
</tr>
<tr>
<td>004202</td>
<td>065143</td>
<td>066216</td>
<td>071556</td>
</tr>
<tr>
<td>004210</td>
<td>070506</td>
<td></td>
<td></td>
</tr>
<tr>
<td>004212</td>
<td>065231</td>
<td>066216</td>
<td>071556</td>
</tr>
<tr>
<td>004220</td>
<td>070506</td>
<td></td>
<td></td>
</tr>
<tr>
<td>004222</td>
<td>065320</td>
<td>066216</td>
<td>071556</td>
</tr>
<tr>
<td>004230</td>
<td>070506</td>
<td></td>
<td></td>
</tr>
<tr>
<td>004232</td>
<td>065406</td>
<td>066216</td>
<td>071556</td>
</tr>
<tr>
<td>004240</td>
<td>070506</td>
<td></td>
<td></td>
</tr>
<tr>
<td>004242</td>
<td>065475</td>
<td>066216</td>
<td>071556</td>
</tr>
<tr>
<td>004250</td>
<td>070506</td>
<td></td>
<td></td>
</tr>
<tr>
<td>004252</td>
<td>065565</td>
<td>066216</td>
<td>071556</td>
</tr>
<tr>
<td>004260</td>
<td>070506</td>
<td></td>
<td></td>
</tr>
<tr>
<td>004262</td>
<td>065666</td>
<td>066216</td>
<td>071556</td>
</tr>
<tr>
<td>004270</td>
<td>070506</td>
<td></td>
<td></td>
</tr>
<tr>
<td>004272</td>
<td>065743</td>
<td>066216</td>
<td>071556</td>
</tr>
<tr>
<td>004300</td>
<td>070506</td>
<td></td>
<td></td>
</tr>
<tr>
<td>004302</td>
<td>066030</td>
<td>066216</td>
<td>071556</td>
</tr>
<tr>
<td>004310</td>
<td>070506</td>
<td></td>
<td></td>
</tr>
<tr>
<td>004312</td>
<td>066115</td>
<td>067752</td>
<td>071544</td>
</tr>
</tbody>
</table>
D03

.SBTTL ACT11 Hooks

*******************************************************************************

.Hooks required by ACT11
   $5VPC= . ; SAVE PC
   0= 46
   $5VAD= . ; 1) SET LOC. 46 TO ADDRESS OF SENDAC IN .SE CI
   52
   0= 313
   .WORD 0 . ; 2) SET LOC. 52 TO ZERO
   $5VPC . ; RESTORE PC
   .SBTTL APT Parameter Block

*******************************************************************************

.SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
*******************************************************************************

   $5X=. ; SAVE CURRENT LOCATION
   0= 24
   $5VAD=. ; SET POWER FAIL TO POINT TO START OF PROGRAM
   200 . ; FOR APT START UP
   0= 44 . ; POINT TO APT INDIRECT ADDRESS PTR.
   .APHTDR=. ; POINT TO APT HEADER BLOCK
   $5X . ; RESET LOCATION COUNTER

*******************************************************************************

.SETUP APT Parameter Block As Defined In The APT-PDP11 Diagnostic
; INTERFACE SPEC.

   $APHTDR:
   $5X:
   $5VAD:
   $5VAD:
   $5VAD:
   $5VAD:
   $5VAD:
   $5VAD:
   $5VAD:
   $5VAD:

START:
   .SBTTL INITIALIZE THE COMMON TAGS
; CLEAR THE COMMON TAGS (SCTAG) AREA

   MOV #SCTAG, R6 . ; FIRST LOCATION TO BE CLEARED
   CLR . ; CLEAR MEMORY LOCATION
   CMP #SMR, R6 . ; DONE?
   BNE -6 . ; LOOP BACK IF NO
   MOV #STACK SP . ; SETUP THE STACK POINTER

; INITIALIZE A FEW VECTORS

   MOV #$SCTAG, #IOTVEC . ; IOT VECTOR FOR SCOPE ROUTINE
   MOV #340, #IOTVECL+2 . ; LEVEL 7
   MOV #$SCTAG, #ENTVEC . ; ENT VECTOR FOR ERROR ROUTINE
   MOV #340, #ENTVECL+2 . ; LEVEL 7
   MOV #$SCTAG, #TRAPVEC . ; TRAP VECTOR FOR TRAP CALLS
   MOV #340, #TRAPVECL+2 . ; LEVEL 7
   MOV #$SCTAG, #PRVVEC . ; POWER FAILURE VECTOR
   MOV #340, #PRVVECL+2 . ; LEVEL 7
   MOV #$SCTAG, #ENDPCT . ; SETUP END-OF-PROGRAM COUNTER
   CLR $TIMES . ; INITIALIZE NUMBER OF ITERATIONS
INITIALIZE THE COMMON TAGS

CLR $ESCAPE
MOV #ERRV, $ESCAPE
MOV $ESCAPE, #ERRV

INITIALIZE THE "1-BIT" TRAP VECTOR, THEN LOAD LOCATION "$TRAN", THE "END-OF-PASS" ($EOP) ROUTINE, WITH A "R" OR "RT"

MOV #SATRN, @BITVEC
SET "1" BIT VECTOR TO $TRAN

MOV #ERRV, @BITVEC
SET "2" BIT VECTOR TO $TRAN

MOV $TRAN, $RT
TRY TO DO A RT

CLR -(SP)
DUMMY PS
MOV $TRAN, -(SP)
AND PC

RT
TRY THE RTT

MOV $TRAN, $RTT
RTT IS LEGAL--SET $TRAN TO A RTT

ADD @10, SP
RTT ILLEGAL--CLEAN OFF THE STACK

MOV #RESVEC + 2, @RESVEC
RESTORE TRAP CATCHER

CR
CLEAR "1" BIT TIDY

MOV $TRAP, $TRAP
INITIALIZE THE LOOP ADDRESS FOR SCOPE

MOV $TRAP, $TOPOR
SET THE ERROR LOOP ADDRESS

SIZE OF A HARDWARE SWITCH REGISTER, IF NOT FOUND OR IT IS
EQUAL TO A "1", SETUP FOR A SOFTWARE SWITCH REGISTER.

MOV #ERRV, -(SP)
SAVE ERROR VECTOR
MOV #ERRV, @ERRV
SET UP ERROR VECTOR

MOV #DISP, #DISP
DISP FOR A HARDWARE SWITCH REGISTER

MOV #DISP, #DISP
DISP FOR A HARDWARE SWITCH REGISTER

MOV #DISP, #DISP
DISP FOR A HARDWARE SWITCH REGISTER

CLR $PASS
CLEAR PASS COUNT

BITB $APTSIZE, $ENV
TEST USER SIZE UNDER APT

MOV #SWREG, $SWREG
NO USE APT SWITCH REGISTER

SBSTL TYPE PROGRAM NAME

SBSTL GET VALUE FOR SOFTWARE SWITCH REGISTER

TYPE THE NAME OF THE PROGRAM IF FIRST PASS

TST #64
ARE WE RUNNING UNDER XOP, ACT

BRANCH IF YES

TST #64
ARE WE RUNNING UNDER APT?

BRANCH IF YES

TST #64
SOFTWARE SWITCH REG SELECTED

BRANCH IF NO

GET SOFT-SW SWITCH SETTINGS

MOV #2, $SWREG

SBSTL TYPE ASCIZ STRING

INC $1
"FIRST TIME"

BRANCH IF NO

CMP $ENAD, #242
BRANCH IF YES

BEQ $71
ACT-11

TYPE 72

SBSTL GET VALUE FOR SOFTWARE SWITCH REGISTER

TYPE THE NAME OF THE PROGRAM IF FIRST PASS

INC $1
"FIRST TIME"

BRANCH IF NO

CMP $ENAD, #42
BRANCH IF YES

BEQ $71
ACT-11

TYPE 72
TYPE ASCII STRING

MOV #1, $AUTOB
SET AUTO-MODE INDICATOR
; TEST THAT CC ARE CLEARED BY R/T

HH17: LPERR
      MOV #00200,R4
      ; SET UP THE LOOP ON ERROR ADDRESS.
      ; SET FIV, FIV, AND FC
      DFPS R4
      MOV #HH19, #$TMP2
      MOV #CPSPUR, #$VPVECT
      MOV #HHP8, RO
      ADD (RO), ACO
      MOV #HHP8, RO
      ; SET ACO OPERAND
      FSRC
      ; TEST INSTRUCTION
      GET FPS
      ; GET THE RESULT
      ADD (RO), ACO
      MOV #HHP10, R1
      ; IS IT CORRECT
      CMP #R2
      ; IS IT CORRECT
      BEQ HH20
      JMP @HHERR7
      ; SET FIV, FIV, AND FD
      MOV #HHP2
      DFPS R4
      MOV #HHP3, RO
      ADD (RO), ACO
      MOV #HHP3, RO
      ; SET ACO OPERAND
      FSRC
      ; TEST INSTRUCTION
      GET FPS
      ; GET THE RESULT
      ADD (RO), ACO
      MOV #HHP11, R1
      ; IS IT CORRECT
      CMP #R2
      ; IS IT CORRECT
      BEQ HH21
      JMP @HHERR8
      ; TEST THAT N IS SET BY R/T
      MOV #HHP7
      DFPS R4
      MOV #HHP8, RO
      ADD (RO), ACO
      MOV #HHP8, RO
      ; SET ACO OPERAND
      FSRC
      ; TEST INSTRUCTION
      GET FPS
      ; GET THE RESULT
      ADD (RO), ACO
      MOV #HHP11, R1
      ; IS IT CORRECT
      CMP #R2
      ; IS IT CORRECT
      BEQ HH22
      JMP @HHERR9
      ; FPS CORRECT?
      MOV #HHP6
      DFPS R4
      MOV #HHP7, RO
      ADD (RO), ACO
      MOV #HHP7, RO
      ; SET ACO OPERAND
      FSRC
      ; TEST INSTRUCTION
      GET FPS
      ; GET THE RESULT
      ADD (RO), ACO
      MOV #HHP10, R1
      ; IS IT CORRECT
      CMP #R2
      ; IS IT CORRECT
      BEQ HH23
      JMP @HHERR10
      ; FPS CORRECT?
015: ERROR 215

MOV R5, @STMP10
MOV R4, @STMP11
MOV @HRP9, @STMP3
MOV @HRP8, @STMP4
MOV @HRD10, @STMP5
MOV @HRP7, @STMP4

1%: ERROR 207

JMP @HHDONE

MOV R5, @STMP10
MOV R4, @STMP11
MOV @HRP8, @STMP3
MOV @HRP6, @STMP4
MOV @HRD10, @STMP5
MOV @HRP10, @STMP6

1%: ERROR 212

JMP @HHDONE

MOV R5, @STMP10
MOV R4, @STMP11
MOV @HRP5, @STMP3
MOV @HRP4, @STMP4
MOV @HRD10, @STMP5
MOV @HRP11, @STMP6

1%: ERROR 207

JMP @HHDONE

MOV R5, @STMP10
MOV R4, @STMP11
MOV @HRP2, @STMP3
MOV @HRP1, @STMP4
MOV @HRD10, @STMP5
MOV @HRP11, @STMP6

1%: ERROR 213

JMP @HHDONE

MOV @HRP2, @STMP2, R3

; WAS THE TRAP TO 244
; ON THE INSTRUCTION
; BEING TESTED

MOV @SP, @STMP2

; FAILURE OF FPS INTERRUPT
; DISABLE BIT (FD=1)
; TO INHIBIT TRAP.
ERROR 217

HHP0: 452
125252
125253

HHP1: 252
125252
125253

HHP2: 0

HHP3: 0

HHP4: 0

HHP5: 100200

HHP6: 300

HHP7: 100000

HHP8: 200

HHP9: 100300

HHP10: 400

HHP11: 100400

HHDONE: RSETUP

): HHP0 + HHP1 WITH
: PROPER NORMALIZATION

): HHP0 + HHP1 WITH
: BAD NORMALIZATION

): HHP7 = HHP8 + HHP9

): = HHP5 + HHP6

): HHP10 = HHP8 + HHP8

): HHP11 = HHP5 + HHP5

GO INITIALIZE THE FPS AND STACK: ANC
L03

**MAINEC-11-0FPB-A**

**PDP :1 34 FPP DIAGNOSTIC PART 2 MACY11 27 (1006) 01-NOV-76 21:12 PAGE 37**

**OFFPB.PI. 01-NOV-76 21:06**

**TI ROUND/TRUNK TEST**

SEE IF THE USER HAS EXPRESSED
THE DESIRE TO CHANGE THE SOFTWARE
VIRTUAL CONSOLE SWITCH REGISTER WAS
THE USER TYPED CONTROL G?.


******************************************************************************

UNTIL:

* 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
* IF ICE=1 OR FIV=1) AND ALSO WITH TRAPS
* DISABLED (FIV=0 OR FIV=0).
* *

******************************************************************************

TST2: SCOPE

;TEST OVERFLOW CONDITION WITH TRAP DISABLER FIV=0

GG1:

LPERR R4,2000 ~SET UP THE LOOP ON ERROR ADDRESS.

MOV #200,R4 ~CLEAR FIU, FIV, AND SET FD

LDFS R4

MOV #G2,2*TMP2

MOV #GGER,2*FPVEC;

MOV #GP5,AC

LD (RO),AC

MOV #GP5,R0 ~FSRC

ADD (RO),ACU ~TEST INSTRUCTION

GET FPS

GET THE RESULT

MOV #GGER,RO

MOV #R2

IS IT CORRECT

CMP (RO)+(R1)+

BEQ GGH

JMP @GGER1

SOC RO,GG3

BIS R6,R4 ~FPS CORRECT?

CMP R4,R5

BEQ GGH

JMP @GGER2 ~TEST OVERFLOW WITH TRAPS ENABLED

;FIV = 1

GG5:

LPERR R4,1200 ~SET UP THE LOOP ON ERROR ADDRESS.

MOV #1200,R4 ~CLEAR FIU, SET FIV, AND FD

LDFS R4

MOV #GP5,R0 ~SET ACO OPERAND

MOV #GP6,R0 ~ACO (RO)

MOV #GP6,RO ~ACU (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,R0 ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)

MOV #GP6,RO ~ACR (RO)
MOV #Ggps, R0 ;ESPc
ADDX (R0), ACO ;TEST INSTRUCTION
MOV #Gmdat0, R0
STD ACO, (R0)
JMP #Ggerr3
MOV #STM2, R3
ADC #R2, R3
CMP R3, (RSP)
BEQ 1$ ; Gmpysp
MOV (RSP), #STM2
CMP (RSP), (RSP) + 1, R5
STFPS R5
MOV #Gmdat0, R0 ;GET THE RESULT
STD ACO, (R0)
MOV #Ggps, R1 ;IS IT CORRECT
MOV #R2, R4
CMF (R0)*, (R1)*
BEQ #Ggerr4
JMP #Ggerr4
MOV R2, R4
MOV R2, GGB
CMP R4, R5
BEQ 1$ ;FPS CORRECT?
MOV #IO, R4
;CHECK FEC
SST R5
CMP R4, R5
BEQ #Gg10
JMP #Ggerr5
; CHECK UNDERFLOW CONDITION WITH TRAPS DISABLED (FIIU = 0)
LPERR
MOV #IO, R4
;SET UP THE LOOP ON ERROR ADDRESS.
MOV R4, R4
LDFPS R4
MOV #Gerr1, #STM2
MOV #Gmp2, #FPvec
MOV #Gmdat0, R0
STD ACO, (R0)
MOV #Ggps, R1 ;IS IT CORRECT
MOV #R4, R2
CMF (R0), (R1)*
BEQ #Gg13
JMP #Ggerr8
MOV R2, R4
CMP R4, R5
BEQ #Gg14
; FPS CORRECT?
; CHECK UNDERFLOW CONDITION WITH TRAPS
; DISABLED (FIU = 0)

GG19: LPERR LI #2000,R4 ; SET UP THE LOOP ON ERROR ADDRESS.
JMP @GG19
;
; MOV @GG15, #@TMP2
; MOV @GG16, #@FPV
; MOV @GG2, @R0
; LDD (RD), ACO
; FSPC
; MOV @GG3, @R0
;
; ADDD (RD), ACO
; ; TEST INSTRUCTION

GG20: ADDD (RD), ACO ; TEST INSTRUCTION

; CHECK UNDERFLOW CONDITION WITH TRAPS
; TRAP ENABLED (FIU = 1)

GG14: LPERR LI #2200,R4 ; SET UP THE LOOP ON ERROR ADDRESS.
JMP @GG15
;
; MOV @GG15, @R0
; STD ACO,(RD)
; JMP @GG16
;
; MOV @GG16, #@TMP2,R3
; ADD R1, R0
; CMP @SP, R1
;
; JMP @FGSPUR
;
; MOV @SP, #@TMP2
; CMP @SP,(SP)+,(SP)+
;
; STFPS R5 ; GET FPS
; MOV @GG17, @R0
; STD ACO,(RD)
; MOV @GG18, @R1
; MOV @GG19, @R2
; CMP (R0)+,(R1)+
;
; BEQ @GG18
;
; CMP #100000,R4
; CMP R4,R5
;
; BEQ @2S
;
; CMP @SP, #12,R4
;
; ; CHECK FEC
; STST R5
; CMP R4,R5
; BEQ @GG19
;
; CMP @SP, #GG13
; JMP @GG19

GG15: ADDD (RD), ACO ; TEST INSTRUCTION

; CHECK UNDERFLOW CONDITION WITH TRAPS
; TRAP ENABLED (FIU = 1)

GG14: LPERR LI #2200,R4 ; SET UP THE LOOP ON ERROR ADDRESS.
JMP @GG15
;
; MOV @GG15, @R0
; STD ACO,(RD)
; JMP @GG16
;
; MOV @GG16, #@TMP2
; MOV @GG17, #@FPV
; MOV @GG2, @R0
; LDD (RD), ACO
; FSPC
; MOV @GG3, @R0
;
; ADDD (RD), ACO
; ; TEST INSTRUCTION

GG20: ADDD (RD), ACO ; TEST INSTRUCTION

; CHECK UNDERFLOW CONDITION WITH TRAPS
; TRAP ENABLED (FIU = 1)
<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>BYTE1</th>
<th>BYTE2</th>
<th>BYTE3</th>
<th>BYTE4</th>
</tr>
</thead>
<tbody>
<tr>
<td>010046</td>
<td>00137</td>
<td>011674</td>
<td></td>
<td></td>
</tr>
<tr>
<td>010050</td>
<td>013701</td>
<td>001236</td>
<td></td>
<td></td>
</tr>
<tr>
<td>010054</td>
<td>027001</td>
<td>000002</td>
<td></td>
<td></td>
</tr>
<tr>
<td>010062</td>
<td>020116</td>
<td>001402</td>
<td></td>
<td></td>
</tr>
<tr>
<td>010066</td>
<td>001137</td>
<td>0036614</td>
<td></td>
<td></td>
</tr>
<tr>
<td>170301</td>
<td>001001</td>
<td>0011554</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011624</td>
<td>001242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12737</td>
<td>011634</td>
<td>001246</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
GGER10:
MOV R5, @STMP10
MOV R4, @STMP11
MOV @GP3, @STMP3
MOV @GP2, @STMP4
MOV @GDAT0, @STMP5
MOV @GP7, @STMP6
1S: ERROR 225
JMP @GGDONE

GGER11:
MOV R5, @STMP10
MOV R4, @STMP11
MOV @GP3, @STMP3
MOV @GP2, @STMP4
MOV @GDAT0, @STMP5
MOV @GP7, @STMP6
1S: ERROR 207
JMP @GGDONE

GGER12:
MOV R5, @STMP10
MOV R4, @STMP11
MOV @GP3, @STMP3
MOV @GP2, @STMP4
MOV @GDAT0, @STMP5
MOV @GP7, @STMP6
1S: ERROR 231
JMP @GGDONE

GGER13:
MOV R5, @STMP10
MOV R4, @STMP11
MOV @GP3, @STMP3
MOV @GP2, @STMP4
MOV @GDAT0, @STMP5
MOV @GP7, @STMP6
1S: ERROR 230
JMP @GGDONE

GGER14:
MOV @STMP2, R1
ADD #2, R1
CMP R1, (SP)
BEO 10$:
JMP @FSPUR

10$:
STST R1
CMP R1, #12
BNE 5$:
CMP (SP)+, (SP)+
MOV @GDAT0, RO
STD ACO, (RO)

5$:
JMP @FSPUR
GO

MOV RH, #TMP11
MOV #GP2, #TMP3
MOV #GP8, #TMP4
MOV #GDATA0, #TMP5
MOV #GP9, #TMP6

1$: ERROR 23!
JMP #GDONE

GDATA0: 0

GP1: 300
GP2: 100200
GP3: 200
GP4: 10200
GP5: 77600

; OVERFLOW = GP5 + GP5

; OVERFLOW RESULT
; UNDERFLOW RESULT
; GP6 = GP4 + GP5
; GP7 = GP3 + GP2 (FIU = 0)
; GP8 = GP3 + GP1
; GP9 = GP3 + GP2 (FIU = 1)

GP6: 0
GP7: 62400
GP8: 340
GP9: 100
GDATA1: 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 3  LDCFD AND LDCOF TEST
; *
; THIS IS A TEST OF LDCFD AND LDCOF.
; *
; *************************************************************
;
; ST3: SCOPE
; TEST FOR CORRECT AUTO INCREMENT CONSTANT.

HX1:    LPERR  MOV @200,R4
        LDFPP R4
        MOV @XP1,RO
        LDD (RO),ACO
        MOV @XP2,RO
        MOV @X2,#$TMP2
        HX2:  LDCFD (RO)+,ACO
              CMP RO, @XP2+4
              BEQ HX3
              JMP @HXR1
        HX3:  STFPS R5
              MOV @XDAT0,RO
              STD ACO,(RO)
              MOV @XP7,R1
              MOV @4,R2
              HX4:  CMP (R1)+,(RO)+
                    BEQ HX7
                    MOV @XP2,R1
                    MOV @XDAT0,RO
                    MOV @4,R2
                    HX5:  CMP (R1)+,(RO)+
                          BEQ HX6
                          JMP @HXR2
                          HX6:  JMP @HXR3
                          HX7:  JMP @HXR3
                          HX8:  JMP @HXR8
                          HX9:  JMP @HXR8
                          HX10: MOV @200,R4
                               LDFPP R4
                               HX11: MOV @XP1,RO
                                      LDD (RO),ACO
                                      MOV @XP2,RO
                                      MOV @X9,#$TMP2
                                      SETF
012072 177420  HX9:  LDCOF  (RO)+,AC0  ;TEST INSTRUCTION
012074 020027 013334  CMP  RO, #HXP2+10  ;HAS A GOOD
012076 061402 012712  BEQ  HX10  ;CONSTANT USED
012078 000137 012712  JMP  #@HXERS  ;TO INCREMENT RO?
012106 170206  HX10:  STFPS  R5
012110 012700 013304  MOV  #HxDato,RO
012114 170110  SETD  ACO,(RO)  ;GET RESULT
012116 170110  STD  ACO,(RO)
012120 012701 013404  MOV  #HXP8,R1
012124 012702 000004  MOV  R4,R2
012126 022120  HX11:  CMP  (R1)+,(RO)+  ;IS IT CORRECT?
012128 000145  BEQ  HX14
012130 012701 013374  MOV  #HXP7,R1
012134 012700 013304  MOV  #HxDato,RO
012138 012702 000004  MOV  R4,R2
012140 022110  HX12:  CMP  (R1)+,(RO)  ;DID FD FAIL TO GET
012144 000140  BEQ  HX13  ;COMPLIMENTED?
012146 000137 013046  JMP  #@HXERS6
012150 072006  HX13:  SOB  R2,HX12
012154 000137 013076  JMP  #@HXER7
012158 072200  HX14:  SOB  R2,HX11
012162 012700 000000  MOV  #0,R4  ;FPS CORRECT?
012166 020406  CMP  R4,R5
012170 012704 000000  BEQ  HX15
012174 001402  JMP  #@HXER8
012178 000137 013030  ;TEST GR7 IMMEDIATE MODE CONSTANT

012204 104413  HX15:  LPERR  ;SET UP THE LOOP ON ERROR ADDRESS.
012208 012704 002000  MOV  #200,R4
012212 170104  LDFPS  R4  ;SET FD
012216 012737 012232 001236  MOV  #Hx16,#@LMP12
012220 012737 013126 000004  MOV  #@HXERS4,#@ERRVECT
012224 005001  CLR  R1
012228 177427 043243  HX16:  LDCFD  #5201,AC0
012232 005201  HX165:  INCR  R1
012236 005201  INCR  R1
012240 005201  INCR  R1
012244 012737 036646 000004  MOV  #@PSP12,#@ERRVECT
012248 020127 000003  CMP  R1,R3  ;SEE IF PC WAS
012252 001402  BEQ  HX17  ;CORRECT
012256 000137 013162  JMP  #@HXER10
012260 104413  HX17:  LPERR  ;SET UP THE LOOP ON ERROR ADDRESS.
012264 012704 002000  MOV  #200,R4
012268 104413
01272  172104  LDFS  R4
01274  01237  012322  001236  MOV  #HX18,#TMP2
01280  012700  013364  MOV  #HPX6,RO
01300  012410  172410  LDD  (RO),ACO
01310  012232  036646  000004  MOV  #CPSPUR,#ERRVECT
01320  013240  013234  MOV  #HPX2,RO
01332  172410  HX18: LDCFD  (RO),ACO
012324  012700  013304  MOV  #HXDAT0,RO
012330  174010  STD  ACO,(RO)  ;GET RESULT.
012332  012701  013374  MOV  #HPX7,R1
012336  012702  000004  MOV  #4,R2
012342  022021  HX19: CMP  (RO)+(R1)+  ;IS RESULT CORRECT??
012344  011402  BEQ  HX20
012346  000137  012732  JMP  #&XER2
012356  077205  HX20: SOB  R2,HX19
012354  104413  HX21: LPERR  #200,R4
012360  012744  000200  MOV  #200,R4
012364  175104  LDFS  R4
012369  012737  012404  001236  MOV  #HX22,#TMP2
012377  012700  013364  MOV  #HPX6,RO
012375  172410  LDD  (RO),ACO
012379  012700  013344  MOV  #HPX4,RO
012404  172410  HX22: LDCFD  (RO),ACO
012406  012700  013304  MOV  #HXDAT0,RO
012412  174010  STD  ACO,(RO)  ;GET RESULT
012420  012701  013354  MOV  #HPX5,R1
012422  012702  000004  MOV  #4,R2
012426  022120  HX23: CMP  (R1)+(RO)+
012432  001415  BEQ  HX26
012436  012701  013374  MOV  #HPX7,R1
012444  012700  013304  MOV  #HXDAT0,RO
012440  012702  000004  MOV  #4,R2
012444  022120  HX24: CMP  (R1)+(RO)+  ;WAS SIGN INCORRECT
012448  001402  BEQ  HX25
012450  001371  JMP  #&XER11
012452  000137  013214  HX25: SOB  R2,HX24
012454  077205  JMP  #&XER12
012462  077220  HX26: SOB  R2,HX23
012464  104413  HX27: LPERR  #200,R4
012469  012704  000200  MOV  #200,R4
012472  170104  LDFS  R4
012474  012700  013314  MOV  #HPX1,RO
012500  172410  LDD  (RO),ACO.
K04

MOV @XH28, &STMP2
MOV @XHP1, R0

ADDCA (RO), ACO

STFPS RS

MOV @XHDATO, R0
STD ACO, (RO)

; GET RESULT

MOV @XHP1, R1
MOV #4, R2

MOV (R1)+, (RO)+

; IS IT 0?

H29: CMP R1, R2
JMP @XHP1R13

H30: SOB R2, H29

MOV #204, R4

; FPS CORRECT

H31: LPERR
MOV #200, R4
LDFPS R4

;SET UP THE LOOP ON ERROR ADDRESS.

MOV @XHP6, R0
LDD (RO), ACO

MOV +01216, 01236
MOV #X32, &STMP2
MOV @XHP1, R0

LDCFD (RO), ACO

; GET RESULT

MOV (R1)+, (RO)+

; IS IT ZERO?

H33: CMP R1, R2
JMP @XHP1R13

H34: SOB R2, H33

MOV #204, R4

; FPS CORRECT

H35: JMP @XHDER4

; RO INCORRECT
L04

MINDC-11-01FPB-A

01-Nov-76 21:12 PAGE 5C

MOV XERP1: MOV @XERP1+4, @STMP4
MOV RO, @STMP3

MOV 1$: ERROR 234
JMP @XDONE

MOV XER5: MOV @XER5+10, @STMP4
MOV RO, @STMP3

MOV 1$: ERROR 237
JMP @XDONE

; REPORT BAD DATA

MOV XER2: MOV @XER2+8, @STMP5
MOV @XER2+4, @STMP7

MOV 1$: ERROR 233
JMP @XDONE

MOV XER3: MOV @XER3+8, @STMP5
MOV @XER3+4, @STMP7

MOV 1$: ERROR 241
JMP @XDONE

MOV XER4: MOV @XER4+8, @STMP3
MOV @XER4+4, @STMP4

MOV 1$: ERROR 240
JMP @XDONE

MOV XER8: MOV @XER8+8, @STMP3
MOV @XER8+4, @STMP4

MOV 1$: ERROR 242
JMP @XDONE

MOV XER6: MOV @XER6+8, @STMP5
MOV @XER6+4, @STMP7

MOV 1$: ERROR 244
JMP @XDONE

MOV XER7: MOV @XER7+8, @STMP5
MOV @XER7+4, @STMP7

MOV 1$: ERROR 243
JMP @XDONE

MOV XER9: BIT #1, (SP)

; SEE IF IT

; AN ODD ADDRESS

MOV XER10: SUB #3, R1
**TEST 4**  
**CMPD TEST**

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

**TEST THE CMPD INSTRUCTION WITH (FSRC=AC=0)**

**AAAR1:**

LPERR ;SET UP THE LOOP ON ERROR ADDRESS.

JSR PC,2*CMPSUB  ;AC

1$: .WORD 0,0,0,0

2$: .WORD 0,0,0,0

3$: 200

4$: ERROR 1

**TEST CMPD WITH (AC=0) AND FSRC POSITIVE.**

**AAAR2:**

LPERR ;SET UP THE LOOP ON ERROR ADDRESS.

JSR PC,2*CMPSUB  ;AC

1$: .WORD 0,0,0,0

2$: 25252

3$: 200

4$: ERROR 3

**TEST CMPD WITH (AC=0) AND FSRC NEGATIVE**

**AAAR3:**
; TEST CMPD WITH AC NEGATIVE AND FSRC POSITIVE
AAAA7: LPERR PC.0#CMPSUB ;SET UP THE LOOP ON ERROR ADDRESS.
1$: JSR AC
2$: ;FSRC
3$: ;FPS BEFORE EXECUTION
4$: ;FPS AFTER EXECUTION
 ;FPS ERROR

; TEST CMPD WITH AC POSITIVE AND FSRC POSITIVE
; AND EAC LESS THAN EFSCR.
AAAA8: LPERR PC.0#CMPSUB ;SET UP THE LOOP ON ERROR ADDRESS.
1$: JSR AC
2$: ;FSRC
3$: ;FPS BEFORE EXECUTION
4$: ;FPS AFTER EXECUTION
 ;FPS ERROR

; TEST CMPD WITH AC POSITIVE, FSRC POSITIVE AND EAC GREATER THAN EFSCR
AAAA9: LPERR PC.0#CMPSUB ;SET UP THE LOOP ON ERROR ADDRESS.
1$: JSR AC
2$: ;FSRC
3$: ;FPS BEFORE EXECUTION
4$: ;FPS AFTER EXECUTION
;TEST CMPD WITH AC NEGATIVE, FSRC NEGATIVE, EAC EQUAL TO EFRC,
;AND AC GREATER THAN FSRC
AAA13:
      LPERR          ;SET UP THE LOOP ON ERROR ADDRESS.
      PC, #CMPSUB
      ;AC
      1$: 115345
          43210
          76543
          21076
      2$: 115345
          54321
          07654
          32107
      3$: 200
          310
          400
          400
      4$: ERROR 16
          JMP #AAA1AONE ;FINISHED CMPD TEST.

;THIS SUBROUTINE, CMPSUB, IS CALLED TO SET UP, EXECUTE
;AND CHECK THE RESULTS OF A CMPD INSTRUCTION.
;IT IS CALLED AS:

       JSR PC, #CMPSUB
   ACARG: .WORD X,X,X,X
   FSRCARG: .WORD X,X,X,X
       ;AC OPERAND
   FSRC: .WORD X
       ;FSRC OPERAND
   FPSB: .WORD X
       ;FPS BEFORE EXECUTION
   FPXA: .WORD X
       ;FPS AFTER EXECUTION
   FPSE: .WORD X
       ;ERROR FPS
   ERR: .WORD X
       ;ERROR FPS
   CONT: .WORD X
       ;RETURN ADDRESS

   ;THE OPERANDS ARE SET UP (USING AC FOR THE AC OPERAND). THEN
   ;FPSB IS LOADED INTO THE FPS. THE INSTRUCTION, CMPD, IS EXECUTED.
   ;AFTER THE EXECUTION THE FPS IS CHECKED AGAINST FPXA. IF IT IS A MATCH
   ;THEN THERE WAS NO ERROR AND CONTROL IS RETURNED TO CONT. IF
   ;THE FPS IS INCORRECT IT IS COMPARED WITH FPSE IN AN ATTEMPT TO ANALYSE
   ;THE FAILURE. IF THE FPS IS THE SAME AS FPSE THEN CONTROL IS
   ;RETURNED TO THE ERROR CALL AT LOCATION ERR. IF THE FPS WAS
   ;NOT CORRECT BUT DIDN'T MATCH FPSE A GENERAL ERROR IS REPORTED
   ;AND CONTROL IS PASSED TO CONT.

   CMPSUB: MOV (SP)+, R1
           ;PICK UP A POINTER TO THE
           ;ARGUMENTS.
           MOV R1, R1
           ;GET THE FPS BEFORE EXECUTION.
           ADD R1, #0
           ;LOAD IT INTO THE FPS.
           MOV R1, R1
           ;Adresse OF CMPD INSTRUCTION.
           LDD R1
           ;GET ADDRESS OF AC OPERAND.
           ADD R0, (RO)
           ;LOAD AC OPERAND
           MOV R1, R1
           ;ADDRESS
           AD
F05

; FOR SCOPING.
; EXECUTE THE TEST INSTRUCTION.
; SAVE FPS AFTER INSTRUCTION.
; GET EXPECTED FPS.
; IF INCORRECT SET UP FOR AN ERROR CALL.

; WAS FPS CORRECT?
; BRANCH IF YES.

; WAS THE FPS THE SAME AS THE EXPECTED INCORRECT FPS?
; BRANCH IF NO MATCH.

; IF THE EXPECTED INCORRECT FPS MATCHED THE RESULTANT FPS
; RETURN TO THE ERROR CALL;
; IN THE CALLING ROUTINE.

; OTHERWISE REPORT INCORRECT FPS.

; IF FPS WAS CORRECT MAKE SURE ACD WAS NOT AFFECTED BY CMPD.

; REPORT ACD MODIFIED BY CMPD.

; 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?)
**G05**

**TEST 5  DIVD WITH (FSRC=0) AND (BUT FC) TEST**

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

`LST: SCOPE`

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

```assembly
LPERR 600000
014424 000000
014426 104413
014430 012704 040200

BBB0:  MOV  #40200,R4  ;SET UP THE LOOP ON ERROR ADDRESS.
        MOV  168H,4  ;SET UP FPS WITH TRAPS DISABLED.

LDFPS  170104
014434 170104
014436 012704 014700 000244
014444 012737 014464 001236
014452 012700 015104
014456 172410
014460 012701 015104

BBB1:  DIVD  (R1),ACO  ;TEST INSTRUCTION

STFPS  R5  ;SET FPS
STST  R3  ;SET FEC

MOV  #140204,R4  ;EXPECTED FPS.
CMP  R4,R5  ;IS FPS CORRECT?
BNE  BBB02  ;IF INCORRECT BRANCH.

MOV  #140204,R4  ;IS FPS CORRECT?
CMP  R4,R5  ;IS FPS CORRECT?
BNE  BBB02  ;IF INCORRECT BRANCH.
```

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

`LPERR 600000`

```assembly
014512 104413
014514 012704 040200

BBB02:  MOV  #40200,R4  ;SET UP THE LOOP ON ERROR ADDRESS.
        MOV  #140204,R4  ;LOAD FPS WITH TRAPS DISABLED.

LDFPS  170104
014520 170104
014522 012704 014542 001236
014530 012700 005114
014534 172410
014536 012700 015104
014542 174410

BBB3:  DIVD  (R0),ACO  ;SET UP ACO OPERAND (NON ZERO).

STFPS  R5  ;GET FPS
STST  R3  ;GET FEC.

MOV  #140200,R4  ;EXPECTED FPS.
CMP  R4,R5  ;IS FPS CORRECT?
```
HOS

DIVD WITH (FSRC=0) AND TRAPS ENABLED.

BB84:
- LATERR
- MOV #200.R4
- LDPS R4

BB85:
- DIVD (RO),ACO
- TEST INSTRUCTION (SHOULD RESULT IN TRAP).

BB86:
- CMP #BB85+2,(SP)
- BEQ 1$:
- JMP #FPSPUR
- IF NOT THEN REPORT AN UNEXPECTED
- FP TRAP.

1$:
- STFSP R5
- STST R3
- CMP (SP)+(SP)+
- RESET THE STACK.

BB87:
- MOV #100200,R4
- IF INCORRECT BRANCH.

BB88:
- MOV #R4,R5
- IF INCORRECT BRANCH.

BB89:
- CMP R2,R3
- IF INCORRECT BRANCH.

BB90:
- JMP #FPBDONE
- OTHERWISE GO TO NEXT TEST.
IOS

259 014724 170205
260 014726 127377 000004 001240
261 014728 010337 001242
262 014730 010537 001244
263 014732 010037 001250
264 014750 012737 140200 001246
265 014756 104017
266 014760 000137 015124
267 014764 010537 001242
268 014766 010437 001244
269 014768 010037 001246
270 015000 010137 001250
271 015004 104020
272 015008 000137 015124
273 015012 010337 001242
274 015016 010237 001240
275 015020 010037 001246
276 015024 010137 001250
277 015028 104021
278 015032 000137 015124
279 015036 010337 001242
280 015040 010237 001240
281 015044 010137 001246
282 015048 010037 001250
283 015052 010137 001252
284 015056 000410
285 015060 000000 000000
286 015064 000000
287 015068 012345
288 015100 010422
289 015104 000410
290 015108 000000
291 015112 000000
292 015116 012345
293 015120 076543
294 015124 104412

BBBDONE:

RSETUP

BBDONE:

:GET FPS.

;EXPECTED FEC.

:REPORT FPS INCORRECT.

BBBER2: MOV R5, 0#TMP4

MOV R4, 0#TMP5

MOV R0, 0#TMP6

MOV R1, 0#TMP7

1$: ERROR 20

JMP BBBDONE

:REPORT FEC INCORRECT.

BBBER3: MOV R3, 0#TMP4

MOV R2, 0#TMP5

MOV R0, 0#TMP6

MOV R1, 0#TMP7

1$: ERROR 21

JMP BBBDONE

:REPORT NO TRAP OCCURRED AFTER TRYING TO DIVIDE

; BY ZERO WITH ALL TRAPS ENABLED.

BBBER4: STFS R3

:GET FEC.

STFS R5

:GET FPS.

MOV R4, 0#TMP4

MOV R3, 0#TMP5

MOV R5, 0#TMP6

MOV R100200, 0#TMP6

MOV R0, 0#TMP7

MOV R1, 0#TMP10

1$: ERROR 22

BR BBBDONE

BBBP1: .WORD 0, 0, 0

BBBP2: .WORD 12345, 54321, 23456, 76543

: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 C?).

:**********************************************************************
**TEST 6**

**DIVF TEST**

**THIS IS A TEST OF THE DIVF INSTRUCTION. NOTE THAT A SUBROUTINE IS**

**USED TO SET UP THE OPERANDS, EXECUTE THE INSTRUCTION AND CHECK THE**

**RESULTS.**

**TEST 6: SCOPE**

**CHECK DIVF WITH (AC=0).**

**SET UP THE LOOP ON ERROR ADDRESS.**

```
LPERR JSR PC,DIVFSUB
1$: .WORD 0,0
2$: .WORD 12345,67012
3$: .WORD 0,0
4$: 0
5$: .WORD 12345,67012
6$: .WORD 23
```

**ERROR 23**

**RESULT BAD.**

**TEST DIVF WITH AC POSITIVE, FSRC POSITIVE AND IN ROUND MODE.**

**SET UP THE LOOP ON ERROR ADDRESS.**

```
LPERR JSR PC,DIVFSUB
1$: .WORD 65535,125252
2$: .WORD 40052,125252
3$: .WORD 30000
4$: 1000
5$: .WORD 140200,0
6$: .ERROR 24
```

**DIV NORMALIZE FAILURE.**

**TEST DIVF WITH AC POSITIVE, FSRC POSITIVE.**

**SET UP THE LOOP ON ERROR ADDRESS.**

```
LPERR JSR PC,DIVFSUB
1$: .WORD 76400,0
2$: .WORD 40200,0
3$: 1000
4$: 1000
5$: .ERROR 25
```

**SIGN BAD.**

**TEST DIVF WITH BOTH OPERANDS POSITIVE.**

**SET UP THE LOOP ON ERROR ADDRESS.**

```
LPERR JSR PC,DIVFSUB
1$: .WORD 56777,177777
2$: .WORD 42777,177777
3$: .WORD 0
4$: 0
5$: .ERROR 2000,2000
```

**ERROR RES.**
K05

WINDEC-11-0FFP-A
PDP 11: 34 FPP DIAGNOSTIC PART 2 MACY11 27(1006) 01-NOV-76 21:12 PAGE 62
OFFPA.P11 01-NOV-76 21:36
T6 DIVF TEST

65: ERROR 23

;TEST THE DIVF INSTRUCTION:
CCCE:
LPERR JSR PC, #DIVFSUB
AC
WORD 13377, 17777
FSCC
WORD 12525, 12525
RES
WORD 0
SENS
WORD -1,-1
ERROR RES.

;TEST DIVIDE ALGORITHM. TEST ROUND CONSTANT.
CCCE:
LPERR JSR PC, #DIVFSUB
AC
WORD 6400, 1
FSCC
WORD 6600, 0
RES
WORD 3600, 1
SENS
WORD 3000, 3000
ERROR RES.

;TEST DIVF.
CCCE:
LPERR JSR PC, #DIVFSUB
AC
WORD 34577, 17777
FSCC
WORD 51377, 17777
RES
WORD 17
SENS
WORD 3400, 3400
ERROR RES.

;DIVF TEST.
CCCB:
LPERR JSR PC, #DIVFSUB
AC
WORD 67652, 12525
FSCC
WORD 56500, 0
RES
WORD 0
SENS
WORD 51543, 107070
ERROR RES.

;DIVF WITH AC NEGATIVE, FSRC NEGATIVE.
CCC9:
LPERR JSR PC, #DIVFSUB
AC
WORD 51543, 107070
FSCC
WORD 162, 162
RES
WORD 51543, 107070
SENS
ERROR RES.

;DIDN'T INCREMENT THE EXPONENT
;AFTER DIVID NORMALIZATION.
DIVF TEST

15: .WORD 140400,0
25: .WORD 140500,0
35: .WORD 040052,125253
45: 0
55: .WORD 140052,125253
65: ERROR 27

DIVF WITH AC NEGATIVE AND FSRC POSITIVE.

CCL10:

15: LPERR
25: JSR PC,2DIVF
35: .WORD 160077,0
45: 0
55: .WORD 60000,0
65: ERROR 27

DIVF WITH AC POSITIVE AND FSRC NEGATIVE.

CCL11:

15: LPERR
25: JSR PC,2DIVF
35: .WORD 140700,0
45: 0
55: .WORD 140052,125253
65: ERROR 27

TEST DIVF BOTH OPERANDS POSITIVE AND TRUNCATE MODE.

CCL12:

15: LPERR
25: JSR PC,2DIVF
35: .WORD 60000,1
45: 0
55: .WORD 60000,1
65: ERROR 30

DIVF WITH POSITIVE OPERANDS AND ROUND MODE.

CCL13:

15: LPERR
25: JSR PC,DIVF
35: .WORD 60000,0
45: 0
55: .WORD 60000,0
65: ERROR 31
MOS

This subroutine, DIVSUB, is called to set up, execute and check the result of a DIV instruction. It is called thus:

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

The operands are set up (using ACO for the AC operand), then FSB is loaded into the FPS. The instruction, DIV is executed. After the execution the result is checked against the expected correct result, RES. If it is correct then the FSB is checked with the expected correct FPS, FSPA. If the FPS was incorrect then it is reported. If the result was incorrect it is compared with ERR to see if the instruction was correct. If the incorrect result matched ERR the control is passed to the error call at ERR. If the incorrect result did not match ERR then the failure is reported in DIVSUB and control is passed to CONT. If no errors are detected then DIVSUB returns control to CONT.

DIVSUB:
```
DIVFSUB: MOV (SP)+, R1 ; GET A POINTER TO THE ARGUMENTS.
          MOV $200, R0 ; SET FD MODE.
          LDFPS R0
          MOV R1, R0 ; LOAD THE AC OPERAND.
          LDD (RO), ACO ; LOAD THE FPS
          MOV 14(R1), R0 ; LOAD THE FSB
          MOVP $16, #TMP2
          MOV $16, #TMP3
          ADD #4, R0 ; ESTABLISH A POINTER TO FSR.
          DIVF (RO), ACO ; TEST INSTRUCTION.
```

Is:
```
DIVF (RO), ACO ; TEST INSTRUCTION.
```

```
DIVFSUB: MOV $200, R0 ; GET THE FPS.
          LDFPS R0
          MOV #DIVFT, R0 ; GET THE RESULT OF THE DIV.
          STD ACO, (RO) ; GET THE DATA IN CASE OF ERROR.
          MOV R1, R2
          MOV R2, #TMP3
          ADD #4, R2
          MOV R2, #TMP4
          ADD #4, R2
          MOV R2, #TMP5
          MOV #DIVFT, #TMP6
```
;DIVF TEST

;IS THE RESULT CORRECT?
CMP R4, 10(R1)
BNE 10$  ;IS FPS CORRECT?
CMP (R0), 20(R1)
BNE 11$  ;DOES THE INCORRECT RESULT
CMP 2(R0), 22(R1)
BNE 11$  ;MATCH THE ANTICIPATED INCORRECT RESULT.
BNE 11$  ;BRANCH IF NO.
JMP (R2)  ;IT MATCHED SO RETURN TO THE ERROR
JMP (R2)  ;REPORT AT THE CALLING ROUTINE.

;REPORT RESULT INCORRECT.
JMP 26(R1)

;REPORT FPS INCORRECT.
JMP 26(R1)

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

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

;DIVD TEST WITH POSITIVE OPERANDS AND IN ROUND MODE.

16:36 000004

;SET UP THE LOOP ON ERROR ADDRESS.
JSR PC, @DIVDSUB

16:46 034777 000000 000000
16:46 034777 000000 000000

;AC
CO6

*MINDEC-11-DFPB-A  PDF 11 34 FPP DIAGNOSTIC PART 2 M 1GII 11 27(1006) 01-NOV-76 21:12 PAGE 67

OFFPBA.P11  01-NOV-76 21:26

77  DIVD TEST

DEC 

016413 000000 6$  ERROR 36
016416 124036

;DIVD TEST.

DEC

016420 016420 104413  LPERR
016422 004737 016630  JSR  PC 2#DIVSUB
016425 100400 000000 000000 15$  .WORD 100400,0,0,0
016428 500,0,0,0
01642B 7647  :FSRC
01642E 125252 125252 5S  .WORD 125252,125252
016433 125252

;SET UP THE LOOP ON ERROR ADDRESS.

DEC

016440 016440 007647
016443 007650

AC

016445 7650  :FPB BEFORE EXECUTION.
016448 7650  :FPB AFTER EXECUTION.
01644B 124033 77777 177777

:ERROR RES.

DEC

016472 104033

:ERROR 33

;DIVD TEST WITH AC POSITIVE AND FSRC NEGATIVE IN ROUND MODE.

DEC

016474 016474 104413
016475 004737 016630

DEC

01647B 004737 016630 15$  .WORD 400,0,0,0
016512 100500 000000 000000
016515 000000 000000
016518 125252 125252 2S  .WORD 125252,125252
01651B 7647  :FSRC
01651E 125252 125252 3S  .WORD 125252,125252
016523 125252

;SET UP THE LOOP ON ERROR ADDRESS.

DEC

016525 007707
016528 007710

DEC

01652B 7710  :FPB BEFORE EXECUTION.
01652E 7710  :FPB AFTER EXECUTION.
016531 177777 177777

:ERROR RES.

DEC

016534 104033

:ERROR 33

;DIVD TEST.

DEC

016550 016550 104413
016552 004737 016630

DEC

016558 170360 170360 15$  .WORD 170360,170360
016562 170360 170360
016565 170360 170360 2S  .WORD 170360,170360
016568 170360 170360
016571 170360 170360 3S  .WORD 170360,170360
016574 170360
016577 402000 000000 000000 000000 000000
01657A 007717 007717 4S  .WORD 402000,0,0,0
01657F 7717  :FSRC
016582 7717  :FPB BEFORE EXECUTION.
016585 7717  :FPB AFTER EXECUTION.
016588 177777 177777 177777 177777 5S  .WORD 177777,177777
0165BB 7717  :FPB BEFORE EXECUTION.
0165BE 7717  :FPB AFTER EXECUTION.
0165C0 7717  :FPB BEFORE EXECUTION.
0165C3 7717  :FPB AFTER EXECUTION.

:ERROR RES.

DEC

0165C6 104033

:ERROR 33

;DIVD TEST.

DEC

0165DA 0165DA 104413
0165DD 004737 016630

DEC

0165E9 170360 170360 15$  .WORD 170360,170360
0165F3 170360 170360
0165F6 170360 170360 2S  .WORD 170360,170360
0165FA 170360 170360
016603 170360 170360 3S  .WORD 170360,170360
016606 170360
016609 7717  :FSRC
01660C 7717  :FPB BEFORE EXECUTION.
01660F 7717  :FPB AFTER EXECUTION.
016612 177777 177777 177777 177777 5S  .WORD 177777,177777
01661F 7717  :FPB BEFORE EXECUTION.
016622 7717  :FPB AFTER EXECUTION.
016625 7717  :FPB BEFORE EXECUTION.
016628 7717  :FPB AFTER EXECUTION.

:ERROR RES.

DEC

01662B 104033

:ERROR 33

;THIS SUBROUTINE. DIVSUB, IS CALLED TO SET UP. EXECUTE
DIV SUB:

; GET A POINTER TO THE ARGUMENTS.
MOV $200,RO
; SET FD MODE.
LDPS RO

; SET UP THE ACO OPERAND.
MOV R1,RO
LDD (RO),ACO
MOV 30(R1),RO
; LOAD THE FPS.
LDPS RO

; ESTABLISH A POINTER TC FSR.
MOV @15,$16,TEMP
ADD @10,RO

; EXECUTE THE TEST INSTRUCTION.
DIVV (RO),ACO

; GET THE FD.
MOV $200,RO
; SET FD MODE.
LDPS RO

; GET THE RESULT.
MOV $DIVDT,RO
STD ACO,(RO)

; SAVE DATA IN CASE OF ERROR.
MOV R1,R2
MOV R2,$16,TEMP
ADD @10,RO
MOV R2,$16,TEMP
ADD @10,RO
MOV R2,$16,TEMP
MOV R4,$16,TEMP
; CHECK THE RESULT.
MOV  R1,R2
ADD  #20,R2
MOV  #DIVOD,R3
MOV  #4,R5
2%:  CMP  (R2)+,(R3)+
     BNE  10%
     SOB  R5.2%
     ; BRANCH IF RESULT INCORRECT.

; IS FPS CORRECT?
CMP  32(R1),R4
BNE  15%
JMP  46(R1)
; BRANCH IF INCORRECT.
; RETURN.
10%:  MOV  R1,R2
ADD  #34,R2
MOV  #DIVOD,R3
MOV  #4,R5
11%:  CMP  (R2)+,(R3)+
     BNE  12%
     SOB  R5.11%
     ; BRANCH IF NO.

; IF THE INCORRECT RESULT WAS
; ANTICIPATED RETURN TO THE
; ERROR REPORT IN THE CALLING
; ROUTINE.
MOV  R1,R2
ADD  #4,R2
JMP  (R2)
; REPORT RESULT INCORRECT.
12%:  ERROR  33
13%:  JMP  46(R1)
; REPORT FPS INCORRECT.

; REPORT FPS INCORRECT.
15%:  ERROR  34
16%:  BR  14%

; 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?).

;.setUp:

; TEST 10 MULF TEST
; THIS IS A TEST OF THE MULF INSTRUCTION. IT MAKES USE OF A SUBROUTINE
; TO SET UP THE OPERANDS, EXECUTE THE MULF INSTRUCTION AND CHECK THE
; RESULTS.

017072 000004

; MULF WITH (FSRC=AC=0)
;MULF BOTH OPERANDS NEGATIVE IN ROUND MODE.

;EEE1:
LPERR PC, @MULFSUB ;SET UP THE LOOP ON ERROR ADDRESS.

;MULF WITH AC POSITIVE AND FSRC NEGATIVE IN TRUNCATE MODE.

;EEE2:
LPERR PC, @MULFSUB ;SET UP THE LOOP ON ERROR ADDRESS.

;MULF WITH AC POSITIVE AND FSRC POSITIVE IN ROUND MODE.

;EEE3:
LPERR PC, @MULFSUB ;SET UP THE LOOP ON ERROR ADDRESS.

;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
    FSRCARG: .WORD X,X
    RES: .WORD X,X
    FPSB: .WORD X
    FPSA: .WORD X

    ;AC OPERAND
    ;FSRC OPERAND
    ;EXPECTED RESULT
    ;FPS BEFORE EXECUTION
    ;FPS AFTER EXECUTION
THE OPERANDS ARE SET UP (USING ACO FOR THE AC OPERAND). THEN
IFPS IS LOADED INTO THE FPS. THE INSTRUCTION, MULF IS EXECUTED.
AFTER THE EXECUTION THE RESULT IS CHECKED AGAINST THE
EXPECTED CORRECT RESULT. 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 ERRRES IN AN ATTEMPT TO ANALYSE THE ERROR. IF
THE INCORRECT RESULT MATCHED ERRRES THEN CONTROL IS PASSED TO
THE ERROR CALL AT ERR. IF THE INCORRECT RESULT DID NOT MATCH ERRRES
THEN THE FAILURE IS REPORTED IN MULF SUB AND CONTROL IS PASSED TO
CONT. IF NO ERRORS ARE DETECTED THEN MULF SUB RETURNS CONTROL
TO CONT.

MULF SUB:

```
            MOV (SP)*, R1  ; SET A POINTER TO THE ARGUMENTS.
            MOV #200,RO  ; SET FD MODE.
            MOV R1,R0  ; LOAD THE AC OPERAND.
            MOV #1,R1,RO  ; LOAD THE FPS
            LDFPS RO  ; LOAD THE FPS
            LDFPS R0  ; LOAD THE FPS
            MOV @1,R1, #252  ; ESTABLISH A POINTER TO FSAC.
            MOV (RO),ACO  ; TEST INSTRUCTION.
            STFPS R4  ; GET THE FPS.
            MOV #200,RO  ; SET FD MODE
            LDFPS RO  ; LOAD THE FPS
            MOV #MULFT,RO  ; GET THE RESULT OF THE MULF.
            STD ACO,(RO)  ; SAVE THE DATA IN CASE OF ERROR.
            MOV R1,R2  ; IS THE RESULT CORRECT?
            MOV R2,#363  ; IF INCORRECT BRANCH.
            MOV R2,#364  ; IF INCORRECT BRANCH.
            MOV R2,#365  ; IF INCORRECT BRANCH.
            MOV R2,#366  ; IF INCORRECT BRANCH.
            MOV R2,#367  ; IF INCORRECT BRANCH.
            MOV 16(R1),#2510  ; IF NO ERRORS OCCURRED RETURN.
```
J06

:DOES THE INCORRECT RESULT MATCH THE ANTICIPATED INCORRECT RESULT.
:BRANCH IF NO.

:IT MATCHED SO RETURN TO THE ERROR REPORT AT THE CALLING ROUTINE.

:REPORT RESULT INCORRECT.

: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 II 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 II SCOPE
::MULD TEST WITH AC POSITIVE AND FSRC POSITIVE.

:SET UP THE LOOP ON ERROR ADDRESS.

:AC

:FSRC

:RES

:ERROR RES.

:ERROR 47
: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.MULDSP

1$: .WORD 65400,0,0,1

AC
1$: .WORD 37577,-1,-1,-2

FSRC
3$: .WORD 64777,-1,-1,-1

RES

2H7:    ; FPS BEFORE EXECUTION.
2H0:    ; FPS AFTER EXECUTION.

5$: .WORD 65000,0,0,0

ERROR RES.

5$: .ERROR 50

; TRUNCATION ERROR.

; MULD TEST WITH BOTH OPERANDS NEGATIVE IN ROUND MODE.

FFF3:    LPERR    ; SET UP THE LOOP ON ERROR ADDRESS.
JSR PC.@MULDSP

1$: .WORD 137577,-1,-1,-2

AC
1$: .WORD 165400,0,0,1

FSRC
3$: .WORD 65000,0,0,0

RES

2H7:    ; FPS BEFORE EXECUTION.
2H0:    ; FPS AFTER EXECUTION.

5$: .WORD 64777,-1,-1,-1

ERROR RES.

5$: .ERROR 51

; ROUND ERROR.

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

FFF4:    LPERR    ; SET UP THE LOOP ON ERROR ADDRESS.
JSR PC.@MULDSP

1$: .WORD 17500,0,0,0

AC
1$: .WORD 123652,125252

FSRC
3$: .WORD 125652,126525

RES

2H0:    ; FPS BEFORE EXECUTION.
2H0:    ; FPS AFTER EXECUTION.

5$: .WORD 103177,-1,-1,-1

ERROR RES.

5$: .ERROR 52

; ROUND ERROR (BAD CONSTANT).

JMP FFDONE

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

JMP  FFDONE

JSR PC.@MULDSP

ACARG: .WORD X,X,X,X

; AC OPERAND
The operands are set up (using AC0 for the AC operand). Then FSB 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 RES. In an attempt to analyse the error, if the incorrect result matched RES then control is passed to the error call at ERR. If the incorrect result did not match RES then the failure is reported in MULSUB and control is passed to CONT. If no errors are detected then MULSUB returns control to CONT.

MULSUB: MOV (SP)+, R1 ; GET A POINTER TO THE ARGUMENTS.
LDPS RO
MOV #200, RO ; SET FD MODE.
MOV R1, RO ; SET UP THE AC0 OPERAND.
LDD (RO), AC0
MOV 3D(R1), RO ; LOAD THE FPS.
LDPS RO
MOV #15, #STMP2 ; ESTABLISH A POINTER TO FSRC.
MOV R1, RO
ADD #10, RO
IS: MULD (RO), AC0 ; EXECUTE THE TEST INSTRUCTION.
STFPS RH ; GET THE FPS.
MOV #200, RO ; SET FD MODE.
LDPS RO
MOV #MULDT, RO ; GET THE RESULT.
STD AC0, (RO) ; AC0, (RO)
MOV R1, R2 ; SAVE DATA IN CASE OF ERROR.
MOV R2, #STMP3
ADD #10, R2
MOV R2, #STMP4
ADD #10, R2
MOV R2, #STMP5
MOV #MULDT, #STMP6
MOV R4, #STMP7
MOV 32(R1), #STMP10
MOV R1, R2 ; CHECK THE RESULT.
ADD #20, R2
MOV #MULDT, R3
MO6

MINDEC-11-0FFF-A PDP 11:34 FPP DIAGNOSTIC PART 2 MACY11 27(1006) 01-NOV-76 21:12 PAGE 77

OffPBA P11 01-NOV-76 21:06 TII MULD TEST

MOV #4, R5
2$: CMP (R2)*,(R3)*
BNE 10$ ;BRANCH IF RESULT INCORRECT.
SOB R5, 2$ ;IS FPS CORRECT?

CMP 32(R1), R4
BNE 15$ ;BRANCH IF INCORRECT.
JMP 46(R1) ;RETURN.

MOV R1, R2
10$: ;WAS INCORRECT RESULT ANTICIPATED?
ADD #12, R2
MOV #MULD1, R3
MOV A4, R5

11$: CMP (R2)*,(R3)*
BNE 12$ ;BRANCH IF NO.
SOB R5, 11$ ;IF THE INCORRECT RESULT WAS
MOV R1, R2 ;ANTICIPATED RETURN TO THE
ADD #44, R2 ;ERROR REPORT IN THE CALLING
JMP (R2) ;ROUTINE.

12$: ;REPORT RESULT INCORRECT.
13$: ERROR 24$ ;REPORT FPS INCORRECT.
14$: JMP 46(R1)

15$: ;REPORT FPS INCORRECT.
16$: ERROR 46 ;REPORT FPS INCORRECT.
BR 14$ ;REPORT FPS INCORRECT.

206020 000000 000000 000000 MULD1: .WORD 0,0,0,0
206026 000000
206030 .
206030 104412 FFFDONE:
206032 000004 RSETUP ;GO INITIATE THE FPS AND STACK; AND
206034 000000 ;SEE IF THE USER HAS EXPRESSED
206034 104413 ;THE DESIRE TO CHANGE THE SOFTWARE
206036 004737 021060 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
206037 020636 000000 ;THE USER TYPED CONTROL Q)?.

;**********************************************************************************************
;TEST 12 UNDER\OVER FLOW, USING MULF WITH TRAPS DISABLED, TEST*
;THIS IS A TEST OF THE OVERFLOW AND UNDERFLOW CONDITIONS USING
;THE MULF INSTRUCTION WITH TRAPS DISABLED. NOTE THAT A SUBROUTINE
;IS USED TO SET UP THE OPERANDS, EXECUTE THE MULF INSTRUCTION AND
;CHECK THE RESULTS.
;**********************************************************************************************
;STATE: SCOPE
;UNDERFLOW, WITH EXPONENT OF RESULT = -129
III: LJPRR PC, @MOVUMNT ;SET UP THE LOOP ON ERROR ADDRESS.
UNDERFLOW, WITH EXPONENT OF RESULT = -193

OVERFLOW, EXPONENT OF RESULT = 128

OVERFLOW, EXPONENT OF RESULT = 130
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 AS SUCH:

ACARG: .WORD X,X :AC OPERAND
FSRCARG: .WORD X,X :FSRC OPERAND
RES: .WORD X,X :EXPECTED RESULT
ERRRES: .WORD X,X :ERROR RESULT
FPSA: .WORD X :FPS BEFORE EXECUTION
FPSB: .WORD X :FPS AFTER EXECUTION
FEC: .WORD X :EXPECTED FEC
FLAG: .WORD X :O/-1, OVER/UNDER FLOW FLAG
ERR1: ERROR X :TRAP ERROR
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 OVLNFTNT RETURNS CONTROL TO THE CALLING ROUTINE AT CONT. IF THE FPS IS BAD OVLNFTNT 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 ERRRES. IF THE FAILURE IN THE RESULTS WAS ANTICIPATED CORRECTLY TO BE ERRRES THEN OVLNFTNT WILL TRANSFER CONTROL TO THE ERROR CALL AT ERR2. OTHERWISE THE RESULT WAS INCORRECT BUT WAS NOT ANTICIPATED AND OVLNFTNT WILL REPORT THE FAILURE AFTER WHICH CONTROL WILL BE PASSED TO CONT.

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

OVLNFTNT:

MOV (SP)+,R1 :GET A POINTER TO THE ARGUMENTS.
MOV B200,RO :SET FD MODE.
LDFPS RO
MOV R1,RO :LOAD ACO, OPERAND.
MOV R1, R2 ;SAVE THE DATA PATTERNS IN CASE OF
MOV R2, @STMP3 ;ERROR.
ADD @R4, R2
MOV R2, @STMP4
ADD R4, R2
MOV @R2, @STMP5
MOV 20(R1), R0 ;LOAD THE FPS.
LDFPS R0
MOV @STMP2
MOV @STMP10
MOV R1, R0 ;SET UP THE FP TRAP VECTOR IN CASE
ADD @R4, R0 ;OF ERROR.
;COMPUTE THE ADDRESS OF FSRC.
MUL % (RO), ACO ;TEST INSTRUCTION.
STEPS R4
STST R5
MOV @STMP3, R0 ;GET FPS.
MOV @R0O, R0 ;GET FE.
MOV @STMP5
MOV @STMP11
;GET THE RESULT.
;CHECK THE RESULT.
CMP (RO)+, (R2)+ ;BRANCH IF INCORRECT.
BNE 15%
SBD R3, 3%
JMP 36(R1) ;RETURN, TEST COMPLETED.
;REPORT INCORRECT FPS.
;WAS THE RESULT OVER OR UNDER FLOA?
;BRANCH IF UNDERFLOW.
;REPORT FPS BAD AFTER OVERFLOW.
;REPORT FPS BAD AFTER UNDERFLOW.
;RESULT INCORRECT.
;SEE IF FAILURE IS ANTICIPATED
MOV R1, R2
ADD #14, R2
MOV R2, A3
CMP (R0)+, (R2)+
BNE 17%
SOB R3, 16%
; BRANCH IF NOT ANTICIPATED.

MOV R1, R2
ADD #34, R2
MOV R2, #0
JMP (R2)

TST 26(R1)
; RESULT WAS NOT ANTICIPATED
; RESULT MUST BE REPORTED HERE.
; FIRST SEE IF ARGUMENTS SHOULD
; HAVE RESULTED IN OVERFLOW OR UNDER
; FLOW BY LOOKING AT THE FLAG.
; BRANCH IF UNDERFLOW EXPECTED.

BNE 19%
BR 4%
; REPORT RESULT INCORRECT, EXPECTING
; OVERFLOW.

18%
ERROR 113
BR 4%
; REPORT RESULT INCORRECT, EXPECTING
; UNDERFLOW.

19%
ERROR 114
BR 4%

20%

; IF AN FP TRAP OCCURS COME HERE.

MOV (SP), R2
CMP #25, R2
BEQ 26%
JMP #FPSPUR
; GET ADDRESS OF TRAP.
; WAS THE TRAP DURING THE MULF INSTRUCTION?
; OTHERWISE GO REPORT A SPURIOUS
; FP TRAP.

CMP (SP)+, (SP)+
STFPS R4
STSP R5
MOV #2000, R0
LDFPS R0
MOV #0VFNT, R0
STD ACO, (R0)
MOV R5, #0
CMP R5, 24(R1)
BNE 27%
; SET FD MODE.
; GET THE RESULT.
; WAS THE FEO ANTICIPATED?
; BRANCH IF NOT ANTICIPATED.

MOV R1, R2
ADD #30, R2
JMP (R2)

TST 26(R1)
; THE ERROR WAS NOT ANTICIPATED SO
; IT MUST BE REPORTED HERE. FIRST SEE IF EXPECTED
; OVERFLOW OR UNDERFLOW.
; BRANCH IF EXPECTING UNDERFLOW

BNE 29%
; REPORT TRAPPED ON OVERFLOW WITH FIV=C

; FAILURE.
EO7

**MINDC-1: DFPK-R**

PDP 11 34 FPP DIAGNOSTIC PART 2  MAC/11 27.1006 01-M0V-76 21:12 PAGE 82

**OFFP8A**

01-M0V-76 21:06

TIE UNDER/OVER FLOW, USING MULF WITH TRAPS DISABLED, TEST

<table>
<thead>
<tr>
<th>Byte</th>
<th>Address</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01472</td>
<td>000004</td>
<td></td>
<td>IIDONE: RSETJP</td>
</tr>
<tr>
<td>01474</td>
<td>000115</td>
<td>104115</td>
<td></td>
</tr>
<tr>
<td>01474</td>
<td>000116</td>
<td>000036</td>
<td></td>
</tr>
<tr>
<td>01474</td>
<td>000117</td>
<td>000000</td>
<td></td>
</tr>
<tr>
<td>01474</td>
<td>104116</td>
<td>000000</td>
<td></td>
</tr>
<tr>
<td>01474</td>
<td>104117</td>
<td>000000</td>
<td></td>
</tr>
<tr>
<td>01474</td>
<td>104118</td>
<td>000000</td>
<td></td>
</tr>
<tr>
<td>01474</td>
<td>104412</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**IIIDONE:**

Go initialize the FPS and STACK: AC
See if the user has expressed
The desire to change the software
Virtual console switch register: Has
The user typed CONTROL Q?

**TEST 13** UNDER/OVER FLOW, USING MULF WITH TRAP DISABLED, TEST

**JJJ1:**

UNDERFLOW, EXponent OF RESULT = -129

LPERR
JSR PC,2#OVLNUN T
1S:
.WORD 20200,0
AC
2S:
.WORD 127272,0
FSC
3S:
.WORD 0,0,0,0
RES
4S:
.WORD 0,0
ERROR RES.
5S:
.WORD 127272,0
FPS BEFORE EXECUTION.
6S:
.WORD 127272,0
FPS AFTER EXECUTION.
7S:
.ERROR 131
ST 331 TO 155 INTO 115 (BUT FUI)
8S:
.ERROR 132
ST 115 (BUT FD)

JJJ2:

LPERR
JSR PC,2#OVLNUN T

**UNDERFLOW, EXponent OF RESULT = -193**
PEND: 11-DFPB-A
PDP 11/34 FPP DIAGNOSTIC PART 2 MACY 1127 (1006) 01-NOV-76 21:12 PAGE 83

FFPB-1, 01-NOV-76 21:56
T13 UNDER/OVER FLOW, USING MULD WITH TRAP DISABLED, "ES"

:OVERFLOW, EXPONENT OF RESULT = 128
JJJ3:

LPERR PC,3MOVUNMT
JSR PC,MOVUNMT

:OVERFLOW, EXPONENT OF RESULT = 130
JJJ4:

LPERR PC,3MOVUNMT
JSR PC,MOVUNMT
G07


UNDERFLOW, USING MULD WITH TRAP DISABLED, TEST

<table>
<thead>
<tr>
<th>BR</th>
<th>8$</th>
<th>ERROR</th>
<th>135</th>
<th>ST 116 (BUF FD)</th>
<th>8$: JMP @JJDDONE</th>
<th>GO TO NEXT TEST</th>
</tr>
</thead>
</table>

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

; AC CC X X X X X
; FSRFLAG = X X X X
; RES = X X X X
; ERRORS: X X X X
; FPS = X X X X
; FOLDER = X X X X
; FEL = X X X X
; FLAG = WORD X X X X
; DATA RESULT ERROR = X X X X
; 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 THEN OVENCLUDT RETURNS CONTROL
; TO THE CALLING ROUTINE AT CONT. IF THE FPS IS BAD OVENCLUDT
; 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 PATTERNS ERRORS. IF THE FAILURE IN
; THE RESULT WAS ANTICIPATED CORRECTLY TO BE ERRORS THEN OVENCLUDT
; WILL TRANSFER CONTROL TO THE ERROR CALL AT ERR2. OTHERWISE THE
; RESULT WAS INCORRECT BUT WAS NOT ANTICIPATED AND OVENCLUDT WILL
; REPORT THE FAILURE AFTER WHICH CONTROL WILL BE PASSED TO CONT.
; IF A TRAP OCCURS (IT SHOULD NOT) THEN OVENCLUDT WILL READ THE FEL.
; SHOULD THE FEL MATCH THE ANTICIPATED FEL OVENCLUDT WILL
; STORE ALL DATA AND TRANSFER CONTROL TO THE ERROR CALL AT ERR1. IF THE
; FEL IS NOT THE SAME AS THE ANTICIPATED FEL OVENCLUDT WILL REPORT
; THE ERROR AND RETURN TO CONT. NOTE THAT OVENCLUDT USES THE FLAG
; TO TELL WHETHER OR NOT THESE PARTICULAR OPERANDS WILL RESULT IN
; UNDERFLOW (FLAG=1) OR OVERFLOW (FLAG=0).

; OVENCLUDT:
; GET A POINTER TO THE ARGUMENTS.
; MOV @200, R0 ; SET FD MODE.
; LDIPS R0
; MOV R1, R0 ; LOAD ACO, OPERAND.
; LDD (R0), ACO
; MOV R1, R2 ; SAVE THE DATA PATTERNS IN CASE OF
; ADD #10, R2 ; ERROR.
; MOV R2, #STMP4
; ADD #10, R2
; MOV R2, #STMP5
; MOV R2, #STMP6
; MOV R2, #STMP7
; MOV R2, #STMP8
; MOV R2, #STMP9
; MOV R2, #STMP10
H07

ANDERC 11-0FPB-A

PDP 11 34 FFP DIAGNOSTIC PART 2 MACY 11 271006 01-Nov-76 21:12 PAGE 65

IFPPB.P: 01-N0V-76 21:06 T13 UNDER-OVER FLOW, USING MULD WITH TRAP DISABLED. TEST

6639 022070 01273 022420 001246 MOV $0VODNTT,$0TMP6
6640 022076 061100 000040 MOV 40(R1),RO ;LOAD THE FPS.
6641 022100 012737 022126 001236 LDPS RD
6642 022112 012737 022312 000044 MOV 1$,$0TMP2
6643 MOV $0VFLECT ;SET UP THE FP TRAP VECTOR IN CASE OF ERROR.
6644 MOV R1,RO ;COMPUTE THE ADDRESS OF FSRC.
6645 ADD #10,RO
6646 022130 00101000 MULD (RO),ACD ;TEST INSTRUCTION.
6647 022140 170904 022420 2$; STFPS RN
6648 022144 070305 STD AC,(RO)
6649 022148 012700 022420 MOV $200,RO
6650 MOV R5,RO ;SET FD MODE.
6651 022154 0010140 MOV F0,RO
6652 022150 0010150 MOV $0VODNTT,RO
6653 MOV R4,RO ;GET THE RESULT.
6654 MOV R5,RO
6655 022160 0010170 MOV $0VODNTT,RO
6656 MOV R1,R2 ;CHECK THE RESULT.
6657 MOV R2,R3
6658 ADD #20,R2
6659 022176 022020 00004 MOV $4,R3
6660 CMP (RO)+(,R2)+ ;BRANCH IF INCORRECT.
6661 022200 0010150 BNE 16%
6662 022202 077303 SOB R3,3%
6663 022204 026104 000042 CMP $42(R1),R4 ;WAS FPS CORRECT?
6664 022210 001002 BNE 10%
6665 022212 000151 000056 MOV $4,R1 ;BRANCH IF FPS IS INCORRECT.
6666 022210 001002 JMP 56(R1) ;RETURN, TEST COMPLETED.
6667 022216 005761 000046 .REPORT INCORRECT FPS.
6668 022222 001002 10% ;WAS THE RESULT OVER OR UNDER FLOW?
6669 022224 104123 TST $46(R1)
6670 022226 000771 BNE 12%
6671 022224 104123 11% ;REPORT FPS BAD AFTER OVERFLOW.
6672 ERROR 123 BR 4%
6673 022220 001012 12% ;REPORT FPS BAD AFTER UNDERFLOW.
6674 BR 4%
6675 022230 001012 13% ;REPORT INCORRECT.
6676 022232 000767 15%; SEE IF FAILURE IS ANTICIPATED
6677 .RESULT INCORRECT.
6678 022234 012700 022420 MOV $0VODNTT,RO ;FAILURE.
6679 022240 010102 022420 MOV R1,R2
6680 ADD #30,R2
6681 022246 012703 000004 MOV $4,R3
6682 022252 022020 CMP (RO)+(,R2)+ ;BRANCH IF NOT ANTICIPATED.
6683 022254 001007 BNE 17%
6684 022256 077303 SOB R3,16%
6685 022260 010102 MOV R1,R2 ;ERROR WAS ANTICIPATED SO RETURN
ADD #54,R2 ;TO THE ERROR REPORT IN THE CALLING
MOV R2,#0;TRAP2 ;ROUTINE.
JMP (R2)

;RESULT WAS NOT ANTICIPATED
;SO ERROR MUST BE REPORTED HERE.
;FIRST SEE IF ARGUMENTS SHOULD
;HAVE RESULTED IN OVERFLOW OR UNDER
;FLOW BY LOOKING AT THE FLAG
;BRANCH IF UNDERFLOW EXPECTED.
TST 46(R1)
BNE 19$

;REPORT RESULT INCORRECT, EXPECTING
;OVERFLOW.
ERROR 125
BR 4$

;REPORT RESULT INCORRECT, EXPECTING
;UNDERFLOW.
ERROR 126
BR 4$

;IF AN FP TRAP OCCURS COME HERE.
MOV (SP),R2 ;GET ADDRESS OF TRAP.
BNE 25$

;GET THE TRAP DURING THE MULF INSTRUCTION'
BEQ 25$
JMP @FPSPUR

;OTHERWISE GO REPORT A SPURIOUS
;FP TRAP.
CMP (SP)+,(SP)+
BNE 25$

;RESET THE STACK.
STFPS R4
STD RS
ST T RS
SPPS R4

;SAVE DATA FOR ERROR REPORT.
GET FP.
GET FEC.
SET FD MODE.

;GET THE RESULT.
MOV #200,RO
MVP 0;OVNNT,RO
STD RO,RD

;HAS THE FEC ANTICIPATED?
BNE 27$

;BRANCH IF NOT ANTICIPATED.
MOV R1,R2
ADD #50,R2
JMP (R2)

;ERROR WAS ANTICIPATED SO
;RETURN TO THE ERROR REPORT OF THE
;CALLING ROUTINE.

;THE ERROR WAS NOT ANTICIPATED SO
;IT MUST BE REPORTED HERE. FIRST SEE IF EXPECTED
;OVERFLOW OR UNDERFLOW.
;BRANCH IF EXPECTING UNDERFLOW
TST 26(R1)
BNE 29$

;REPORT TRAPPED ON OVERFLOW WITH FIV=0
ERROR 132
JMP 56(R1)

;REPORT TRAPPED ON UNDER FLOW WITH FIU=0
ERROR 130
JMP 56(R1)

;WORD 0,0,0,0

**TEST 14** UNDER/OVER FLOW, USING MULD WITH TRAPS ENABLED. TEST

THIS IS A TEST OF THE UNDERFLOW AND OVERFLOW CONDITIONS THAT CAN OCCUR USING THE MULD INSTRUCTION.

A SUBROUTINE IS CALLED TO SET UP THE OPERANDS,

EXECUTE THE MULD INSTRUCTION AND CHECK

THE RESULTS. HERE THE PARTICULAR INTERRUPT

EITHER OVERFLOW OR UNDERFLOW, IS ENABLED SO A TRAP SHOULD OCCUR.

**UNDERFLOW, EXponent OF RESULT = -129**

**UNDERFLOW, EXponent OF THE RESULT = -193**
BR 8B
ERROR 144

; OVERFLOW, EXPONENT OF THE RESULT = 128

LPERR PC, #OVNFT
JSR 022646 004737 02266C
15  : WORD 65252, 125252
29  : WORD 60000, 0
33  : WORD 000052, 125252
53  : WORD -1, -1
69  : WORD -1000
101006  : FPS BEFORE EXECUTION.
9  : 0
0  : FSEC
FLAG
7  : ERROR 147
ST 333 (BUT FIV) NO TRAP
BR 8B
ERROR 143

; OVERFLOW, EXPONENT OF RESULT = 130

LPERR PC, #OVNFT
JSR 022660 004737 022660
15  : WORD 65254, 67654
29  : WORD 60200, 0
33  : WORD 345, 67654
47  : WORD -1, -1
53  : WORD -1015
101002  : FPS BEFORE EXECUTION.
9  : 0
0  : FSEC
FLAG
7  : ERROR 150
ST 133 (BUT FIV) NO TRAP
BR 8B
ERROR 143
8B: JMP KKKDONE

; THIS SUBROUTINE OVNFT 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
FSRCARG:  WORD X,X
RES:  WORD X,X
ERRRES:  WORD X,X
FPF:  WORD X
FPLE:  WORD X
FPF:  WORD X
FSEC:  WORD X
FLAG:  WORD X
ERR:  ERROR X
BR:  CONT
ERROR  X
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 THE THE FPS IS
COMPUTED WITH FOCUS ON THIS TOO IS CORRECT OVUNIT RETURNS CONTROL
TO THE CALLING ROUTINE AT CONT. IF THE FPS IS BAD OVUNIT 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, ERRORS. IF THE FAILURE IN
THE RESULT WAS ANTICIPATED CORRECTLY TO BE ERRORS THEN OVUNIT
WILL TRANSFER CONTROL TO ERROR CALL AT ERR2. OTHERWISE THE
RESULT WAS INCORRECT BUT WAS NOT ANTICIPATED AND OVUNIT WILL-
REPORT THE FAILURE AFTER WHICH CONTROL WILL BE PASSED TO CONT.
IF NO TRAP OCCURS CONTROL IS PASSED TO ERR1.
NOTE THAT OVUNIT USES THE FLAG
TO TELL WHETHER OR NOT THESE PARTICULAR OPERANDS WILL RESULT IN
UNDERFLOW (FLAG=1) OR OVERFLOW (FLAG=0).

OVUNIT: MOV (SP)+, R1 ;GET A POINTER TO THE ARGUMENTS. MOV $300, R0 ;SET FD MODE.
LDFPS R0
MOV R1, R0 ;LOAD ACO, OPERAND.
LDO (R0), ACO
MOV R1, R2 ;SAVE THE DATA PATTERNS IN CASE OF ERROR.
ADD $4, R2
MOV R2, $3, R3
ADD $4, R2
MOV R2, $3, R4
MOV 21(R1), $3, R10
MOV #OVFFY, $3, R6
MOV $20(R1), R0 ;LOAD THE FPS.
LDFPS R0
MOV $1S, $3, R2
MOV #0S5, #3FPVCT ;SET UP THE FP TRAP VECTOR IN CASE OF ERROR.
MOV R1, R0 ;COMPUTE THE ADDRESS OF FSR.
ADD $4, R0
1S: MULF (R0), ACO ;TEST INSTRUCTION. SHOULD CAUSE TRAP.
2S: CFCC JMP #25S ;FAILURE, NO TRAP.
90: 022600 012601
900: 022666 012700 000020
906: 022666 170100
908: 022670 010100
909: 022670 172410
910: 022674 010102
911: 022676 010237 001240
912: 022702 062702 0000004
915: 022706 010237 001242
916: 022712 062702 0000004
917: 022716 010237 001244
918: 022722 016137 000022 001252
919: 022730 012737 023262 001246
91A: 022736 016100 000020
91B: 022742 170100
91C: 022744 012737 022766 001236
91D: 022752 012737 022766 000244
91E: MOV R1, R0 ADD $4, R0
91F: MOV R1, R0 ADD $4, R0
920: 022766 170100
921: 022770 170000
922: 022772 000137 023222
923: 022776 011602
924: MOV (SP), R2 CMP R2, #25 ;TRAP TO HERE AND SEE IF THE PC OF THE
925: CMP R2, #25 ;TRAP WAS THAT OF THE MULF INSTRUCTION.
926: BEQ 5S ;BRANCH IF YES.
927: JMP #2FSPUR ;OTHERWISE REPORT SPURIOUS FP ERROR.
928: 022302 022626
929: 022304 001402
92A: 022306 000137 036114
92B: 022306 000137 036114
92C: 022306 000137 036114
92D: 022306 000137 036114
92E: 022306 000137 036114
92F: 022306 000137 036114
930: 022306 000137 036114
931: 022306 000137 036114
932: 022306 000137 036114
933: MOV (SP)+,(SP)+ ;RESET THE STACK
934: STFPS R0 ;GET FPS
935: MOV R1, R0 ;SET FD MODE.
LDPS RO
MOV #OVFTH,RO
; GET THE RESULT.
MOV ACO,(RO)
STD
MOV R4,#STMP7
MOV R5,#STMP17
MOV R1,R2
MOV R1,R2
ADD #10,R2
MOV #2,R3
CMP (R0)+(R2)+
BNE 15%
; CHECK THE RESULT.
; BRANCH IF INCORRECT.
CMP 22(R1),R4
BNE 10%
; WAS FPS CORRECT?
; BRANCH IF FPS IS INCORRECT.
CMP 24(R1),R5
BNE 5%
; IS FEC CORRECT?
; IF INCORRECT BRANCH.
JMP 36(R1)
; RETURN, TEST COMPLETED.
; REPORT INCORRECT FEC.
TST 26(R1)
BNE 7%
; WAS THE RESULT OVERFLOW OR UNDERFLOW?
; BRANCH IF UNDERFLOW.
ERROR 137
BR 4%
; REPORT BAD FEC ON EXPECTED OVERFLOW.
ERROR 140
BR 4%
; REPORT BAD FEC ON EXPECTED UNDERFLOW.
; REPORT INCORRECT FPS.
TST 26(R1)
BNE 12%
; WAS THE RESULT OVER OR UNDERFLOW?
; BRANCH IF UNDERFLOW.
; REPORT FPS BAD AFTER OVERFLOW.
ERROR 141
BR 4%
; REPORT FPS BAD AFTER UNDERFLOW.
ERROR 142
BR 4%
; RESULT INCORRECT.
MOV #OVFTH,RO
MOV R1,R2
ADD #19,R2
MOV #2,R3
CMP (R0)+(R2)+
BNE 17%
; SEE IF FAILURE IS ANTICIPATED.
; FAILURE.
ROB R3,16%
; BRANCH IF NOT ANTICIPATED.
; ERROR WAS ANTICIPATED SO RETURN
; TO THE ERROR REPORT IN THE CALLING
NO7

**MAINDEC-11-OFPB-A**   PDP :13 FPP DIAGNOSTIC PART 2 MACSY11 27(1006) 01-NOV-76 21:12 PAGE 91

OFFPBA.PI1  01-NOV-76 21:06

T14 UNDERFLOW, USING MULF WITH TRAPS ENABLED, TEST

```
4975 023176 013237 001236
4976 023202 000112
4977
4978 023204 005761 000026 17$: TST 26(R1) ;RESULT WAS NOT ANTICIPATED.
4979 023202 000112 04975 023176 013237 001236
4980 023204 005761 000026
4981 BNE 19$ ;FIRST SEE IF ARGUMENTS SHOULD HAVE RESULTED IN OVERFLOW OR UNDERFLOW BY LOOKING AT THE FLAG.
4982
4983 023210 001002 ;BRANCH IF UNDERFLOW EXPECTED.
4984
4985 023212 104143 18$: ERROR 143 ;REPORT RESULT INCORRECT, EXPECTING OVERFLOW.
4986 023214 000733
4987
4988 023216 104144 19$: ERROR 144 ;REPORT RESULT INCORRECT, EXPECTING UNDERFLOW.
4989 023220 000731
4990
4991 023222 170204 ;IF NO FP TRAP OCCURS COME HERE.
4992 023224 170305 25$: STFPS R4 ;GET FPS.
4993 023226 012700 000200
4994 023228 170100 ;GET FD MODE.
4995 023222 170020
4996 LDFPS R0
4997 023224 170005
4998 023222 012700 000200
4999 023226 012700 023262
5000 023240 174010
5001 023242 010437 001250
5002 023246 010537 001254
5003 023252 010102
5004 023254 062702 000030
5005 023258
5006 023260 000112
5007 023262 000000 000000 000000
5008 023270 000000
5009
5010 023272 000000
5011 023272 104412
5012 KKKDONE: ;SETUP ;GO INITIALIZE THE FPS AND STACK; AND SEE IF THE USER HAS EXPRESSED THE DESIRE TO CHANGE THE SOFTWARE.
5013
5014
5015
5016
5017
5018
5019
5020
5021
5022
5023
5024
5025
5026
5027
5028
5029
5030
```

;*********************************************************

;TEST IS UNDERFLOW, USING MULF WITH TRAPS ENABLED, TEST

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

;*********************************************************

;ST15: SCOPE

;UNDERFLOW, EXPONENT OF RESULT = -129
B08

TIS UNDER/OVER FLOW, USING MULD WITH TRAPS ENABLED, TEST

LL1:  LPERR PC, 240VUNDI ; SET UP THE LOCP ON ERROPR ADDRESS.

LL2:  LPERR PC, 240VUNDI ; SET UP THE LOCP ON ERROR ADDRESS.

LL3:  LPERR PC, 240VUNDI ; SET UP THE LOCP ON ERROR ADDRESS.
0: ERROR 162 :FLAG
BR 8S: ;ST 333 (BUT FIV: NO TRAP)
ERROR 163 ;ST 700 (BUT FD).

;OVERFLOW, EXPONENT OF THE RESULT " = 130

LLL4:

LPER 1: JSPR PC.240VUNDT ;SET UP THE LOOP ON ERROR ADDRESS.

1S: .WORD 60345,67454 ;AC
2S: .WORD 60200,0,0,0 ;FSRC
3S: .WORD 345,67454 ;RES
4S: .WORD 56756,45676 ;ERROR RES.

5S: 7215 ;FPS BEFORE EXECUTION.
6S: 107202 ;FPS AFTER EXECUTION.
7S: 0 ;FEC
BR 8S: ;FLAG
ERROR 155 ;ST 133 (BUT FIV) NO TRAP

THIS SUBROUTINE, OVUNDT, IS USED TO SET UP THE OPERANDS, EXECUTE
THE MUL 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
ERRE: .WORD X,X,X,X :ERROR RESULT
FPB1: .WORD X :FPS BEFORE EXECUTION
FPB2: .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 AC0 AS THE ACCUMULATOR). THEN
THE MUL Instruction IS EXECUTED. IF THE TRAP OCCURS THEN THE
RESULT IS CHECKED AGAINST RES. IF THE RESULT IS CORRECT THEN THE FPS IS
COMPAWWED 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
MUL IS INCORRECT, THE INCORRECT RESULT IS COMPARED WITH THE
ANTICIPATED FAILING DATA PATTERN, ERRAS. IF THE FAILURE IN
THE RESULT WAS ANTICIPATED CORRECTLY TO BE ERRAS THEN OVUNDT
WILL TRANSFER CONTROL TO THE ERR1 CALL AT ERR2. OTHERWISE THE
BNE 15$ ;BRANCH IF INCORRECT.

CMP 42(R1),R4 
BNE 10$ ;WAS FPS CORRECT?

CMP 44(R1),R5 
BNE 5$ ;IS FEC CORRECT?

JMP 56(R1) ;IF INCORRECT BRANCH.

REPORT INCORRECT FEC. 
TST 46(R1) ;WAS THE RESULT OVERFLOW OR UNDERFLOW?
BNE 7$ ;BRANCH IF UNDERFLOW.

ERROR 151 ;REPORT BAD FEC ON EXPECTED OVERFLOW.
BR 4$ ;REPORT BAD FEC ON EXPECTED UNDERFLOW.

ERROR 152 ;REPORT BAD FEC ON EXPECTED UNDERFLOW.
BR 4$ ;REPORT BAD FEC ON EXPECTED UNDERFLOW.

REPORT INCORRECT FPS.
TST 46(R1) ;WAS THE RESULT OVER OR UNDERFLOW?
BNE 12$ ;BRANCH IF UNDERFLOW.

ERROR 153 ;REPORT FPS BAD AFTER OVERFLOW.
BR 4$ ;REPORT FPS BAD AFTER UNDERFLOW.

ERROR 154 ;REPORT FPS BAD AFTER UNDERFLOW.
BR 4$ ;REPORT FPS BAD AFTER UNDERFLOW.

MOV $0VDTT,RC ;SEE IF FAILURE IS ANTICIPATED
MOV R1,R2 ;FAILURE.
MOV #30,R2
ADD R4,R3
MOV R4,R3
CMP (R0)+,(R2)+ ;BRANCH IF NOT ANTICIPATED.
BNE 17$ 
SOK 3,16$ ;ERROR WAS ANTICIPATED SO RETURN

MOV R1,R2 ;TO THE ERROR REPORT IN THE CALLING
ADD $54,R2 ;ROUTINE.
MOV R2,2*STMP2
JMP (R2) ;RESULT WAS NOT ANTICIPATED

TST 46(R1) ;SO ERROR MUST BE REPORTED HERE.
FIRST SEE IF ARGUMENTS SHOULD ;FIRST SEE IF ARGUMENTS SHOULD
HAVE RESULTED IN OVERFLOW OR UNDER ;HAVE RESULTED IN OVERFLOW OR UNDER
FLOW BY LOOKING AT THE FLAG. ;FLOW BY LOOKING AT THE FLAG.

REPORT RESULT INCORRECT, EXPECTING
FO8

MAINDEC-10: OFFB-A  POP I: 13 FPP DIAGNOSTIC PART 2 MACY11 27(1006) 01-NOV-76 21:12 PAGE 96
OFFBN.P11: 01-NOV-76 21:06

18$: ERROR 155 ;OVERFLOW.
19$: BR 4$ ;OVERFLOW.
20$: ERROR 156 ;UNDERFLOW.

IF NO FP TRAP OCCURS COME HERE.

25$: STFPS R4 ;GET FPS.
     STST RS ;GET FEC.
     MOV #200, R0 ;SET FC MODE.
     LDFPS R0 ;GET THE RESULT.
     MOV #OVDT, R0 ;SET THE RESULT.
     STD ACO,(R0)
     MOV R4,#R8
     MOV R5,#R9
     MOV R1,R2
     ADD #50, R2
     JMP (R2)

LLL DONE: 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 16: MODF TEST **
** THIS IS A TEST OF THE MODF INSTRUCTION, WHICH MAKES USE OF **
** A SUBROUTINE TO SET UP THE OPERANDS, EXECUTE THE MODF INSTRUCTION **
** AND CHECK THE RESULTS. **
** **********************************************************
** 1516: SCOPE **
** MODF WITH (FSRC=AC=0) **
** GGG1: **
LDERR PC,2#MODSUB :SET UP THE LOOP ON ERROR ADDRESS.
JSR AC

1$: .WORD 0,0
2$: .WORD 0,0
3$: .WORD 0,0
4$: .WORD 0,0
5$: .WORD -1,-1
6$: .WORD -1,-1
7$: 13
G08

:MODF TEST, WITH (FSRC=0)

;SET UP THE LOOP ON ERROR ADDRESS.

LPERR  PC, #MODFSUB

JSR  PC

1$:  WORD 123456, 76543
2$:  WORD 0.0
3$:  WORD 0.0
4$:  WORD 0.0
5$:  WORD 0.0
6$:  WORD -1.1
7$:  0
8$:  ERROR 56
9$:  ERROR 57

:MODF TEST WITH (AC=0)

;SET UP THE LOOP ON ERROR ADDRESS.

LPERR  PC, #MODFSUB

JSR  PC

1$:  WORD 0.0
2$:  WDR 76543, 21234
3$:  WORD 0.0
4$:  WORD 0.0
5$:  WORD 0.0
6$:  WORD -1.1
7$:  3
8$:  ERROR 53
9$:  ERROR 57

:MODF TEST WITH EXPONENT OF THE RESULT = 25

;SET UP THE LOOP ON ERROR ADDRESS.

LPERR  PC, #MODFSUB

JSR  PC

1$:  WORD 46252, 125252
2$:  WORD 40300.0
3$:  WORD 0.0
4$:  WORD 46377.0
5$:  WORD 46252, 125252
6$:  WORD 40300.0
7$:  13
8$:  ERROR 53
9$:  ERROR 60
MODF TEST WITH EXPONENT OF THE RESULT = 127

G3G3:   LPERR   PC, R#MODF SUB
        JSR    PC, R#MODF SUB
1$:    WORD   77652, 125252
2$:    WORD   40300, 0
3$:    WORD   0, 0
4$:    WORD   77777, -1
5$:    WORD   77652, 125252
6$:    WORD   40300, 0
7$:    4
8$:    ERROR  53
      BR    9$
9$:    ERROR  60

MODF TEST WITH EXPONENT OF RESULT = 25

G3G3:   LPERR   PC, R#MODF SUB
        JSR    PC, R#MODF SUB
1$:    WORD   46400, 1
2$:    WORD   46390, 0
3$:    WORD   0, 0
4$:    WORD   46390, 1
5$:    WORD   40000, 0
6$:    WORD   -1, -1
7$:    13
8$:    ERROR  61
      BR    9$
9$:    ERROR  54

MODF TEST WITH EXPONENT OF THE RESULT = 24

G3G3:   LPERR   PC, R#MODF SUB
        JSR    PC, R#MODF SUB
1$:    WORD   46000, 1
2$:    WORD   40340, 0
3$:    WORD   0, 0
4$:    WORD   46340, 1
5$:    WORD   40000, 0
6$:    WORD   -1, -1
7$:    0
8$:    ERROR  62
      BR    9$
9$:    ERROR  54

MODF TEST WITH EXPONENT OF THE RESULT = 10

G3G3:   LPERR   PC, R#MODF SUB
        JSR    PC, R#MODF SUB
1$:    WORD   46000, 1
2$:    WORD   40340, 0
3$:    WORD   0, 0
4$:    WORD   46340, 1
5$:    WORD   40000, 0
6$:    WORD   -1, -1
7$:    0
8$:    ERROR  62
      BR    9$
9$:    ERROR  54
MODF TEST

LPERR PC 3 MODFSUB

:SET UP THE LOOP ON ERROR ADDRESS.

JSR 00467E 004677

16: .WORD 425577,-1

AC

26: .WORD 402000.0

FSRC

36: .WORD 40177.17600

FRACTIONAL RES.

46: .WORD 42577.140000

INTEGER RES.

56: .WORD -1,-1

ERROR FRACTIONAL RES.

ERROR INTEGER RES.

76: 0

FPS BEFORE EXECUTION.

FPS AFTER EXECUTION.

86: ERROR 53

BC 96

ERROR 54

96:

:MODF TEST WITH THE EXponent OF THE RESULT = 10

LPERR PC 3 MODFSUB

:SET UP THE LOOP ON ERROR ADDRESS.

JSR 00467E 004677

16: .WORD 425577.140000

AC

26: .WORD 402000.0

FSRC

36: .WORD 40177.17600

FRACTIONAL RES.

46: .WORD 42577.140000

INTEGER RES.

56: .WORD -1,-1

ERROR FRACTIONAL RES.

ERROR INTEGER RES.

76: 0

FPS BEFORE EXECUTION.

FPS AFTER EXECUTION.

86: ERROR 53

BC 96

ST 532 TO .22 INTO NORMALIZE.

96:

:MODF TEST WITH EXponent OF THE RESULT = 9

LPERR PC 3 MODFSUB

:SET UP THE LOOP ON ERROR ADDRESS.

JSR 00467E 004677

16: .WORD 425577.100000

AC

26: .WORD 402000.0

FSRC

36: .WORD 40177.10000

FRACTIONAL RES.

46: .WORD 42577.100000

INTEGER RES.

56: .WORD -1,-1

ERROR FRACTIONAL RES.

ERROR INTEGER RES.

76: 13

FPS BEFORE EXECUTION.

FPS AFTER EXECUTION.

86: ERROR 53

BC 96

ERROR 54

96:

:MODF TEST WITH EXponent OF THE RESULT = 0

LPERR PC 3 MODFSUB

:SET UP THE LOOP ON ERROR ADDRESS.

JSR 00467E 004677

16: .WORD 40177.1

AC

26: .WORD 40200.0

FSRC

36: .WORD 40177.1

FRACTIONAL RES.
; MODF TEST WITH EXPONENT OF THE RESULT = -15
LPPR:  LPERR  JSR  PC,2*MODFSUB
  1$:  .WORD  34377,-1
  2$:  .WORD  40200,0
  3$:  .WORD  34377,-1
  4$:  .WORD  0,0
  5$:  .WORD  0,0
  6$:  .WORD  34377,-1
  7$:  0
  8$:  ERROR 64
  9$:  ERROR 64

; MODF TEST WITH EXPONENT OF RESULT = -64, IN ROUND MODE
LPPR:  LPERR  JSR  PC,2*MODFSUB
  1$:  .WORD  20000,1
  2$:  .WORD  40200,0
  3$:  .WORD  20100,2
  4$:  .WORD  0,0
  5$:  .WORD  20100,1
  6$:  .WORD  0,0
  7$:  0
  8$:  ERROR 65
  9$:  ROUND TRUNK, ST 126 INTO ROUND.

; MODF TEST WITH EXPONENT OF RESULT = 11
LPPR:  LPERR  JSR  PC,2*MODFSUB
  1$:  .WORD  142777,170000
  2$:  .WORD  40200,0
  3$:  .WORD  140000,0
  4$:  .WORD  142777,160000
  5$:  .WORD  0,0
  6$:  .WORD  42777,160000
  7$:  0
  8$:  ERROR 64
  9$:  ERROR 64
K08

; 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
; INTR: .WORD X,X ; INTEGER RESULT
; ERRFR: .WORD X,X ; ERROR FRACTIONAL RESULT
; ERRINT: .WORD X,X ; ERROR INTEGER RESULT
; FPSB: .WORD X ; FPS BEFORE EXECUTION
; FPSA: .WORD X ; FPS AFTER EXECUTION
; ERR1: .WORD X ; FRACTION ERROR
; ERR2: .WORD X ; INTEGER ERROR
; CONT: .WORD X ; RETURN ADDRESS

; The operands are set up (using AC for the AC argument). The MODF
; instruction is executed. Then the results are retrieved.
; The fractional part of the result is compared with FRES. If this is correct
; then the integer part is compared with INTR. If both of these are correct
; then the FPS is compared with FPSA. After execution if no errors occurred
; then MODFSUB will return to CONT. If the FPS was incorrect
; it is reported here. If the fractional result is incorrect it is compared with
; the anticipated and fractional, ERRFR, if this doesn't match
; the true result then the error is reported here. If the anticipated
; failure matches the true result then MODFSUB passes control to the
; error call at ERR2. Likewise if the integer part of the result is
; not correct then it is compared with the anticipated integer
; failure. If this doesn't match then the error is reported here.
; If a match is made however, MODFSUB will return control to the error
; call at ERR2.

MODFSUB: MOV (SP)+,R1 ; Get a pointer to the arguments
       LDFS PS RO ; Set FD mode.
       MOV #200,RO ; Get AC operand
       MOV R1,RO ; Set up AC0
       LDD (RO),AC0
       MOV #MODP1,RO ; Put a background pattern into AC1.
       LDD (RO),AC1
       MOV 30(R1),RO ; Set up the FPS.
       LDFS PS RO
       MOV &IS,2#tmp2
       MOV R1,RO ; Compute the address of the FSRC.
       ADD #4,RO
       MOD (RO),AC0 ; Execute the test instruction.
       STFPS R4 ; Get the FPS.
       MOV #200,RO ; Set FD mode.
LDPS  RO
MOV  @MODFT0,RO
;
GET THE FRACTIONAL RESULT.

MOV  A0, (RO)
STD  A0,(RO)
;
GET THE INTEGER RESULT.

MOV  R1, R2
				;SAVE THE DATA IN CASE OF ERROR.

MOV  R2, #TMP3
ADD  #4, R2
ADD  #4, R2
MOV  R2, @TMP6
MOV  #MODFT0, #TMP7
MOV  #MODFT1, #TMP10
MOV  R4, #TMP11
MOV  32(R1), #TMP12

MOV  010237, 001246
ADD  000004
MOV  #MODFT0, R2
;
CHECK THE FRACTIONAL RESULT.

CMP  #MODFT0, (R2)
BNE  10(R1), R2
;
BRANCH IF INCORRECT.

CMP  10(R1), (R2)
BNE  10S
;
BRANCH IF INCORRECT.

MOV  #MODFT1, R2
;
CHECK THE INTEGER RESULT.

CMP  #MODFT1, (R2)
BNE  15S
;
BRANCH IF INCORRECT.

CMP  14(R1), (R2)
BNE  15
;
BRANCH IF INCORRECT.

CMP  16(R1), (R2)
BNE  15
;
BRANCH IF INCORRECT.

CMP  32(R1), R4
BNE  20S
;
BRANCH IF INCORRECT.

CMP  20(R1), (R2)
BNE  1S
;
BRANCH IF NOT ANTICIPATED.

CMP  32(R1), (R2)
BNE  1S
;
BRANCH IF NOT ANTICIPATED.

MOV  R1, R2
;
THE ERROR WAS ANTICIPATED SO
ADD  #34, R2
;
RETURN TO THE ERROR REPORT AT THE
JMP  (R2)
;
CALLING ROUTINE.

CMP  000122
BNE  11S
;
THE ERROR WAS NOT ANTICIPATED SO
ADD  #34, R2
;
REPORT THE INCORRECT FRACTION HERE.

CMP  00022
BNE  11S
;
THE ERROR WAS NOT ANTICIPATED SO
ADD  #34, R2
;
REPORT THE INCORRECT FRACTION HERE.

CMP  00022
BNE  11S
;
THE ERROR WAS NOT ANTICIPATED SO
ADD  #34, R2
;
REPORT THE INCORRECT FRACTION HERE.

CMP  00022
BNE  11S
;
THE ERROR WAS NOT ANTICIPATED SO
ADD  #34, R2
;
REPORT THE INCORRECT FRACTION HERE.

CMP  00022
BNE  11S
;
THE ERROR WAS NOT ANTICIPATED SO
ADD  #34, R2
;
REPORT THE INCORRECT FRACTION HERE.

CMP  00022
BNE  11S
;
THE ERROR WAS NOT ANTICIPATED SO
ADD  #34, R2
;
REPORT THE INCORRECT FRACTION HERE.
ADD  #40,R2 ;TO THE ERROR REPORT IN THE CALLING
JMP (R2) ;ROUTINE.

15%: ERROR 54
BR 9%

;THE ERROR WAS NOT ANTICIPATED SO PEPOP
;THE INTEGER FAILURE HERE.

;FPS INCORRECT.
20%: MOV R4,2*$TMP11 ;REPORT INCORRECT FPS.
MOV 32(R1),2*$TMP12
21%: ERROR 55
BR 9%

MODFT0: .WORD 0,0,0,0
MODFT1: .WORD 0,0,0,0
MODP1: .WORD -1,-1,-1,-1
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?).

******************
*TEST 17 MODD TEST
*THIS IS A TEST OF THE MODD INSTRUCTION. IT MAKES USE OF A
*SUBROUTINE
*TO SET UP THE ARGUMENTS, EXECUTE THE INSTRUCTION AND CHECK THE
*RESULTS.
*
**************
TST17: SCOPE

MODD WITH (FSRC=AC=0)

LPERR
JSR PC,2*MODDSUB ;SET UP THE LOOP ON ERROR ADDRESS.

1S: .WORD 0,0,0,0
AC
2S: .WORD 0,0,0,0
;FSRC
3S: .WORD 0,0,0,0
;FRACTIONAL RES.
4S: .WORD 0,0,0,0
;INTEGER RES.
5S: .WORD 0,0,0,0
;ERROR FRACTIONAL RES.
MODD TEST WITH EXPONENT OF THE RESULT = 56

```assembly
LPERR  JSR PC 2#MOODSUB ;SET UP THE LOOP ON ERROR ADDRESS.
```

MODD TEST WITH EXPONENT OF THE RESULT = 36

```assembly
LPERR  JSR PC 2#MOODSUB ;SET UP THE LOOP ON ERROR ADDRESS.
```
MODD TEST WITH EXPONENT OF THE RESULT = 30

SET UP THE LOOP ON ERROR ADDRESS.

MODD TEST WITH EXPONENT OF THE RESULT = 31

SET UP THE LOOP ON ERROR ADDRESS.

MODD TEST WITH EXPONENT OF THE RESULT = 0

SET UP THE LOOP ON ERROR ADDRESS.
MODD TEST WITH EXPONENT OF THE RESULT = -115

MODD TEST WITH EXPONENT OF THE RESULT = -63, IN ROUND MODE
THIS SUBROUTINE, MODSUB, IS CALLED TO SETUP THE OPERANDS, EXECUTE THE MODD INSTRUCTION AND CHECK THE RESULTS.

IT IS CALLED THIS:

```
ACARG: .WORD X,X,X,X
FSRCARG: .WORD X,X,X,X
FRES: .WORD X,X,X,X
IFRES: .WORD X,X,X,X
EFRFRES: .WORD X,X,X,X
FINTRES: .WORD X,X,X,X
FPSB: .WORD X
FPSA: .WORD X
ERR1: .WORD X
ERR2: .WORD X
CONT: .WORD X
```

THE OPERANDS ARE SET UP (USING ACARG 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 FPSA IS COMPARED WITH FPSA. IF NO ERRORS OCCURRED, THEN MODSUB 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, EFRFRES. IF THIS DOESN'T MATCH THE TRUE RESULT THEN THE ERROR IS REPORTED HERE. IF THE ANTICIPATED FAILURE MATCHES THE TRUE RESULT THEN MODSUB PASSES CONTROL TO THE ERROR CALL AT ERR1. 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, MODSUB WILL RETURN CONTROL TO THE ERROR CALL AT ERR2.

MODSUB:

```
MODSUB:    MOV  (SP)+,R1 ;GET A POINTER TO THE ARGUMENTS
           MOV  $200,RO ;SET FD MODE.
           LDFPS  RO
           MOV  R1,RO ;SET UP ACO
           LOD  (RO),AC
           MOV  #MODPL,RO ;PUT A BACKGROUND PATTERN INTO AC1.
           LOD  (RO),AC1
           MOV  60(R1),RO ;SET UP THE FPS.
           LDFPS  RO
           MOV  $1,$1,$1,RO
           ADD  $10,RO
```
1$: MODC (RO),ACO ;EXECUTE THE TEST INSTRUCTION.

2$: STFPS R4 ;GET THE FPS.

3$: MOV #200,RO ;SET FD MODE.

4$: LDFPS RO ;GET THE "FRACTIONAL RESULT".

5$: MOV #MODTO,RO ;GET THE INTEGER RESULT.

6$: STD AC0 (RO) STD AC1,(RO) ;SAVE THE DATA IN CASE OF ERROR.

7$: MOV R1,R2 ;CHECK THE FRACTIONAL RESULT.

8$: MOV R3 ;CHECK THE INTEGER RESULT.

9$: CMP (R2),+,(R3) ;BRANCH IF INCORRECT.

10$: BNE 10$ ;CHECK THE FPS.

11$: JMP 72(R1) ;BRANCH IF INCORRECT.

12$: ;FRACTIONAL ERROR.

13$: MOV #MODTO,R2 ;HAS THE FRACTIONAL ERROR ANTICIPATED?

14$: MOV R1,R3

15$: ADD R4,R3

16$: MOV #4,R5

17$: CMP (R2),+,(R3) ;BRANCH IF NOT ANTICIPATED.

18$: BNE 11$ ;THE ERROR WAS ANTICIPATED SO

19$: RETURN TO THE ERROR REPORT AT THE

20$: CALLING ROUTINE.
### 009

**MODD TEST**;

```
0595 027714 000112
0596 JMP (R2);
1095 027716
1096 18%: ERROR 70
1097 BR 90
6098 027720 000756
6100 027722 012702 030010
6102 027726 010103
6104 027730 062703 000050
6106 027734 012705 000004
6108 027740 022223
6110 027742 001005
6112 027744 077503
6114 027746 010102
6116 027750 062702 000070
6118 027754 000112
6120 027756
6122 027758 014971
6124 027760 003736
6126 027762 010437 001254
6128 MOV R4, A $TMP11
6130 027766 016137 000062 001256
6132 027774 104072
6134 MOV R2(R1), A $TMP12
6136 027776 000727
6138 030000 000000 000000 000000
6140 MODDT0: .WORD 0,0,0,0
6142 030006 000000
6144 030010 000000 000000 000000
6146 MODDT1: .WORD 0,0,0,0
6148 030016 000000
6150 030020 000000
6152 030020 104412
6154 HHHOME:
6156 RSETUP
6158 ;SETUP
6160 ;SETUP
6162 ;SETUP
6164 ;SETUP
6166 ;SETUP
6168 ;SETUP
6170 ;SETUP
6172 ;SETUP
6174 ;SETUP
6176 ;SETUP
6178 ;SETUP
6180 ;SETUP
6182 ;SETUP
6184 ;SETUP
6186 ;SETUP
6188 ;SETUP
6190 ;SETUP
6192 ;SETUP
6194 ;SETUP
6196 ;SETUP
6198 ;SETUP
6200 ;SETUP
6202 ;SETUP
6204 ;SETUP
6206 ;SETUP
6208 ;SETUP
6210 ;SETUP
6212 ;SETUP
6214 ;SETUP
6216 ;SETUP
6218 ;SETUP
6220 ;SETUP
6222 ;SETUP
6224 ;SETUP
6226 ;SETUP
6228 ;SETUP
6230 ;SETUP
6232 ;SETUP
6234 ;SETUP
6236 ;SETUP
6238 ;SETUP
6240 ;SETUP
6242 ;SETUP
6244 ;SETUP
6246 ;SETUP
6248 ;SETUP
6250 ;SETUP
6252 ;SETUP
6254 ;SETUP
6256 ;SETUP
6258 ;SETUP
6260 ;SETUP
6262 ;SETUP
6264 ;SETUP
6266 ;SETUP
6268 ;SETUP
6270 ;SETUP
6272 ;SETUP
6274 ;SETUP
6276 ;SETUP
6278 ;SETUP
6280 ;SETUP
6282 ;SETUP
6284 ;SETUP
6286 ;SETUP
6288 ;SETUP
6290 ;SETUP
6292 ;SETUP
6294 ;SETUP
6296 ;SETUP
6298 ;SETUP
6300 ;SETUP
6302 ;SETUP
6304 030022 000004
```

---

**H09**

- **INDEC-11-OFFP-8**
- **POP 114 FPP DIAGNOSTIC PART 2 MACY11 27:1006 01-NOV-76 21:12 PAGE 131
- **OFFPBA.P11**
- **01-NOV-76 21:06**
- **T17**
- **MODD TEST**

- The error was not anticipated.
  - Report the incorrect fraction here.
- Integer error.
  - Was the integer error anticipated?
- Branch if not anticipated.
  - The error was anticipated so return to the error report in the calling routine.
- The error was not anticipated so report the integer failure here.
- FPS incorrect.
  - Report incorrect FPS.
- The desire to change the software virtual console switch register (has the user typed control C)?

---

*SECONDARY ROUTINE* UNDER-OVERFLOW, 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.

**SECONDARY ROUTINE**

- UNDERFLOW TEST, WITH EXPONENT OF THE RESULT = -129, FIU = 1, FIO = 1
I09

 miềnคือ-1: OFFPRA-D  องค์: 24 กลุ่ม диагนอยส์ ภาค 2 แม่ 171006; 01-NOV-76 21:12 งาน 112.

OFFPRA: พ. 10-NOV-76 21:16

"TO UNDER-OVER FLOW. USING MODF WITH TRAPS DISABLED, "TEST"

MMM1:

LPERR: SET UP THE LOOP ON ERROR ADDRESS.

ISR: PC MODFOV.

JSR: WORD 20123, 45676.

AC:

WORD 20200, 0:

FSRC:

WORD 123, 45676:

FRATIONAL RES.

WORD 0, 0:

INTEGER RES.

ERROR FRATIONAL RES.

ERROR INTEGER RES.

AC:

WORD -1, -1:

FPS BEFORE EXECUTION.

FPS AFTER EXECUTION.

EC:

9%: ERROR 170:

FEC INCORRECT, UNDERFLOW.

BR 9%

ERROR 171:

AC V 1 (2, 3) <= ZEPO, ST :26.

9%:

UNDERFLOW EXP OF RESULT = -193, FIU = 0, FID = 1

MMM2:

LPERR: SET UP THE LOOP ON ERROR ADDRESS.

ISR: PC MODFOV.

JSR: WORD 10000, 0:

AC:

WORD 10000, 0:

FSRC:

WORD 0, 0:

FRATIONAL RES.

WORD 0, 0:

INTEGER RES.

ERROR FRATIONAL RES.

ERROR INTEGER RES.

FPS BEFORE EXECUTION.

FPS AFTER EXECUTION.

EC:

9%: NOP:

BR 9%

ERROR 171:

9%:

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

MMM3:

LPERR: SET UP THE LOOP ON ERROR ADDRESS.

ISR: PC MODFOV.

JSR: WORD 600052, 125252:

AC:

WORD 60200, 0:

FSRC:

WORD 0, 0:

FRATIONAL RES.

WORD 0, 0:

INTEGER RES.

ERROR FRATIONAL RES.

ERROR INTEGER RES.

FPS BEFORE EXECUTION.

FPS AFTER EXECUTION.

EC:

9%: ERROR 172:

BAD FEC ON OVERFLOW.

BR 9%

ERROR 173:

ST 520 TO STORE ZERO TWICE

INTO 162

9%:

OVERFLOW TEST WITH EXPONENT OF THE RESULT = 130, FIU = 0, FID = 1

MMM4:
LPERR  PC, 3, @MDFOV
:SET UP THE LOOP ON ERROR ADDRESS.

JSR  PC, 3, @MDFOV
:AC

WORD  60345, 67654

WORD  60200, 0

FRC

WORD  0, 0

INTEGER RES.

WORD  0, 0

ERROR FRACTIONAL RES.

WORD  0, 0

ERROR INTEGER RES.

WORD  345, 67654

6011

6006

10

FEC

NOP

BR 9%

ST 520 TO 162 INTO STORE ZERO TWICE.

ERROR 174

OVERFLOW TEST WITH EXPONENT OF THE RESULT = 128, RESULT NEGATIVE AND FIV = 1, FIO = 1.

AMNS:

LPERR  PC, 3, @MDFOV
:SET UP THE LOOP ON ERROR ADDRESS.

JSR  PC, 3, @MDFOV
:AC

WORD  160252, 125252

WORD  600000, 0

FRC

WORD  0, 0

INTEGER RES.

WORD  100052, 125252

ERROR FRACTIONAL RES.

WORD  0, 0

ERROR INTEGER RES.

WORD  52, 125252

41006

141006

10

FEC

ERROR 172

BR 9%

ERROR 175

ST 517, BAD SIGN.

JMP  @MDFOV/ERROR
:GO TO THE NEXT TEST.

THIS SUBROUTINE, MDFOV, IS CALLED TO SETUP THE OPERANDS, EXECUTE THE MODF INSTRUCTION AND CHECK THE RESULTS.

IT IS CALLED THIS:

ACARG: .WORD  X, X

AC OPERAND

FRCARG: .WORD  X, X

FRC OPERAND

FRES: .WORD  X, X

FRACTIONAL RESULT

INTRES: .WORD  X, X

INTEGER RESULT

ERFRES: .WORD  X, X

ERROR FRACTIONAL RESULT

ERROR: .WORD  X, X

ERROR INTEGER RESULT

FPB: .WORD  X

FPS BEFORE EXECUTION

FPF: .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 ACQ 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 EPS IS COMPARED WITH FPAS. AFTER EXECUTION IF NO ERRORS OCCURRED
THEN MODFOV WILL RETURN TO CONT. IF THE EPS WAS INCORRECT
IT IS RETURNED HERE. IF THE FRACTION IS INCORRECT IT IS COMPARED WITH
THE ANTICIPATED BAD FRACTION, ERRFES. IF THIS DOESN'T MATCH
THE TRUE RESULT THEN THE ERROR IS REPORTED HERE. IF THE ANTICIPATED
FAILURES MATCHES THE TRUE RESULT THEN MODFOV PASSES CONTROL TO THE
ERROR CALL AT ERR2. 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 ERR00
CALL AT ERR2.

MODFOV: MOV (SP)+, R1 ; GET A POINTER TO THE ARGUMENTS
MOV #2200, R0 ; SET FD MODE.
LDFPS R0
MOV R1, R0
; SET UP ACO
LDD (R0), ACO
MOV #MODP1, R0
; PUT A BACKGROUND PATTERN INTO AC1.
LDD (R0), AC1
MOV 30(R1), R0
; SET UP THE EPS.
LDFPS R0
MOV R1, R0
; COMPUTE THE ADDRESS OF THE FSAE.
ADD #68, R0

IS: MODF (R0), ACO
; EXECUTE THE TEST INSTRUCTION.
STEPS R4
STST R5
MOV #2200, R0
; SET FD MODE.
LDFPS R0
MOV #MODFD0, R0
; GET THE FRACTIONAL RESULT.
STD ACO (R0)
MOV #MODFD1, R0
; GET THE INTEGER RESULT.
STD AC1, (R0)

MOV R1, R2
; SAVE THE DATA IN CASE OF ERROR.
MOV R2, 32'STMP3
ADD #4, R2
MOV R2, 32'STMP4
ADD #4, R2
MOV 32'STMP5
ADD #4, R2
MOV 32'STMP6
ADD #4, R2
MOV 32'STMP7
MOV #MODFD0, 32'STMP7
MOV #MODFD1, 32'STMP10
MOV R4, 32'STMP11
MOV 32'(R1), 32'STMP12
MOV R5, 32'STMP13
MOV 34'(R1), 32'STMP14
MOV 32'R0, R2
; CHECK THE FRACTIONAL RESULT.
CMP 10'(R1), (R2)
BNE 106
CMP 12'(R1), 2(R2)
; BRANCH IF INCORRECT.
L09

MAINDEC-11-OFPB-A PDP :: 34 FPP DIAGNOSTIC PART 2 MACT11 27(10D6) 01-MOV-76 21:12 PAGE 120 UNDER/OVER FLOW, USING MOOF WITH TRAPS DISABLED. 13h

6319 030564 01021 BNE 10%
6320 030566 012702 030736 MOV #MODF01,R2 ;CHECK THE INTEGER RES...
6322 030572 026112 000014 CMP 14(R1),R2. ;BRANCH IF INCORRECT.
6324 030576 001016 BNE 15%
6325 030600 026162 JMP 16(R1),2(R2) ;BRANCH IF INCORRECT.
6326 030606 001012 BNE 15%
6327 030610 026194 000032 CMP 32(R1),R4 ;CHECK THE FPS.
6328 030614 001024 BNE 20% ;BRANCH IF INCORRECT.
6330 030616 026105 000034 CMP 34(R1),R5 ;CHECK THE FEC.
6331 030622 001030 BNE 25% ;BRANCH IF INCORRECT.
6332 030624 000161 000044 9$ JMP 44(R1) ;RETURN.
6334 030626 000165 000044 9$ FRACTIONAL ERROR.
6336 030628 000165 000044 9$ THE ERROR WAS NOT ANTICIPATED 52.
6338 03062a 000074 BR 9$ ;REPORT THE INCORRECT FRACTION HERE.
633a 030630 000165 000044 9$ INTEGER ERROR.
633c 030630 104165 000074 12$ ERROR 165 ;BRANCH IF NOT.
633e 030632 000074 BR 9$ 12$.
6340 030634 026112 000024 CMP 24(R1),(R2) ;THE ERROR WAS ANTICIPATED SO RETURN.
6342 030640 001010 BNE 16% ;ROUTINE.
6344 030642 026162 000026 000002 CMP 26(R1),2(R2)
6346 030650 001004 BNE 16% ;TO THE ERROR REPORT IN THE CALLING
6348 030652 010102 MOV R1,R2 ;ROUTINE.
634a 030654 062702 000042 ADD #42,R2
634c 030660 001112 JMP (R2)
634e 030662 104166 000074 16$ THE ERROR WAS NOT ANTICIPATED 50 REPORT.
6350 030664 000074 BR 9$ ;THE INTEGER FAILURE HERE.
6352 030666 000074 20$ FPS INCORRECT.
6354 030666 010437 001254 MOV R4,2#TMP11 ;REPORT INCORRECT FPS.
6356 030672 016137 000032 001256 MOV 32(R1),2#TMP12
6358 030700 104167 21$ ERROR 167 ;REPORT FEC ERROR.
635a 030702 000750 BR 9$ 21$.
635c 030704 010537 001260 MOV R5,2#TMP13
635e 030710 016137 000034 001262 MOV 34(R1),2#TMP14
6360 030716 010102 MOV R1,R2
6362 030720 062702 000036 ADD #36,R2
6364 030724 001112 JMP (R2)
6366 030726 000000 000000 000000 MODFD0: .WORD 0,0,0,0
6368 030734 000000 000000 000000 MODFD1: .WORD 0,0,0,0
636A 030736 000000 000000 000000 000000
636C 030744 000000 MMMDONE: RSETUP ;GO INITIALIZE THE FPS AND STACK; AND
6370 030746 104412
:SEE IF THE USER HAS EXPRESSED
THE DESIRE TO CHANGE THE SOFTWARE
VIRTUAL CONSOLE SWITCH REGISTER "AS
THE USER TYPED CONTROL G")

:********************************************************************************
:* TEST 21 UNDER\OVER FLOW, USING M O O D W I T H T R A P S D I S A B L E D, TEST *
:* THIS IS A TEST OF THE MODD INS T R U C T I O N S ' O V E R F L O W A N D UNDER F L O W
*: CONDITIONS. A SUBROUTINE IS USED TO SET UP THE OPERANDS, EXECUTE THE
*: MODD INSTRUCTION AND CHECK THE RESULTS.
:* *
*:******************************************************************************
:
:UNDERFLOW TEST WITH EXPONENT OF THE RESULT = -129, FIU = 1, FID = 1

NNN1: LPERR PC, #MODDOV
       ;SET UP THE LOOP ON ERROR ADDRESS.
      JSR PC, #MODDOV
      AC

1$: .WORD 20252, 125252

2$: .WORD 20100, 0, 0, 0

3$: .WORD 177, -1, -1, -1

4$: .WORD 0, 0, 0, 0

5$: .WORD 20252, 125252

6$: .WORD 125252, 125252

7$: .WORD 0, 0, -1, -1

8$: .WORD 0, 0, 0, 0

9$: .WORD 0, 0, 0, 0

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

NNN2: LPERR PC, #MODDOV
       ;SET UP THE LOOP ON ERROR ADDRESS.
      JSR PC, #MODDOV
      AC

1$: .WORD 10000, 0

2$: .WORD 123456, 0

3$: .WORD 0, 0, 0, 0

4$: .WORD 0, 0, 0, 0

5$: .WORD 0, 0, 0, 0

:END OF TEST 21

:END OF PROGRAM.

:END OF MAIN.
NO9

PDP:11 34 FPP DIAGNOSTIC PART 2 MACY11 27(1006) 01-NOV-76 21:12 PAGE 117
OFFPBA P11 01-NOV-76 21:06

TE1 UNDER-OVER FLOW, USING MODD WITH TRAPS DISABLED, TEST

6%: .WORD 0,0

;ERROR INTEGER RES.

7%: .WORD 123456,0

;FPS BEFORE EXECUTION.

8%: NOP

;FPS AFTER EXECUTION.

9%: BR 9%

;ST 047 (BUT FD).

9%: ERROR 203

;ST 047 (BUT FD).

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

N08:

LPERR PC ,#MODDOV

;SET UP THE LOOP ON ERROR ADDRESS.

JSR

;AC

.LOOP

.WORD 60252,125252

;FSRC

.WORD 125252,125252

.WORD 0,0,0,0

;FRACTIONAL RES.

4%: .WORD 177,-1,-1,-1

;INTEGER RES.

5%: .WORD 0,0,0,0

;ERROR FRACTIONAL RES.

6%: .WORD 177,-1

;ERROR INTEGER RES.

7%: .WORD 125252,125252

;FPS BEFORE EXECUTION.

14206

;FPS AFTER EXECUTION.

10

;FEC

8%: ERROR 204

;FEC BAD ON OVERFLOW.

9%: BR 9%

;ST 520 TO 162 INTO 163 (BUT FD).

9%: ERROR 205

;ST 520 TO 162 INTO 163 (BUT FD).

;OVERFLOW TEST WITH EXPONENT OF THE RESULT = 130, FIV = 0, FID = 1

N09:

LPERR PC ,#MODDOV

;SET UP THE LOOP ON ERROR ADDRESS.

JSR

;AC

.LOOP

.WORD 60200,0

;FSRC

.WORD 125252,0

.WORD 0,0,0,0

;FRACTIONAL RES.

4%: .WORD 0,0,0,0

;INTEGER RES.

5%: .WORD 0,0,0,0

;ERROR FRACTIONAL RES.

6%: .WORD 125252,0

;ERROR INTEGER RES.

7%: .WORD 400,0

;FPS BEFORE EXECUTION.

6206

;FPS AFTER EXECUTION.

10

;FEC

8%: NOP

;ST 520 TO 162 INTO STORE ZERO TWICE.
B10

THIS SUBROUTINE, MODDOV, IS CALLED TO SETUP THE OPERANDS, EXECUTE THE MODD INSTRUCTION AND CHECK THE RESULTS.

IT IS CALLED THUS:

ACARG: :WORD X,Y,X,X
FSRCARG: :WORD X,Y,X,X
AC: :WORD X,Y,X,X
FSRC: :WORD X,Y,X,X
FINTRES: :WORD X,Y,X,X
FR: :WORD X,Y,X,X
FERRES: :WORD X,Y,X,X
FERINTRES: :WORD X,Y,X,X
FPS: :WORD X,Y,X,X
FPSA: :WORD X,Y,X,X
ERR1: :ERROR X,Y,X,X
ERR2: :ERROR X,Y,X,X
CONT: :RETURX 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 COMPARSED 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 RETURNED HERE. IF THE FRACTION IS INCORRECT IT IS COMPARED WITH THE ANTICIPATED BAD FRACTION, ERRES. 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.

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

MODDOV:

MOV (SP)+ R1, R0 ;GET A POINTER TO THE ARGUMENTS
LDFPS R0, R0, R0; SET FD MODE.
MOV R1, R0, R0 ;SET UP ACO
LDD (R0), AC0
MOV #MOD1, R0 ;PUT A BACKGROUND PATTERN INTO AC1.
SMV R0, (R1), R0 ;SET UP THE FPS.
MPFS R0, R0, R0
MOV #IS, @STMP2
MOV R1, R0 ;COMPUTE THE ADDRESS OF THE FSRC.
ADD #10, R0

MODDOV (RD), ACO ;EXECUTE THE TEST INSTRUCTION.

SIST R5 ;GET THE FPS.
STFPS R4 ;GET THE FPS.
MOV #200, R0 ;SET FD MODE.
LDFPS R0, R0, R0
MOV #MOD000, R0 ;GET THE FRACTIONAL RESULT.
STD ACO, (RD)
C10

;MOV $MODD1, R0 ;GET THE INTEGER RESULT.
STD AC1,(R0)

;SAVE THE DATA IN CASE OF ERROR.
MOV R1, R2
MOV R2, @$TMP3
ADD @10, R2
MOV R2, @$TMP4
ADD @10, R2
MOV R2, @$TMP5
ADD @10, R2
MOV R2, @$TMP6
MOV $MODDDD, @$TMP7
MOV $MODDDD, @$TMP10
MOV $MODDDD, @$TMP11
MOV @2, @$TMP12
MOV @5, @$TMP13
MOV 64(R1), @$TMP14

;CHECK THE FRACTIONAL RESULT.
MOV $MODDDD, R2
MOV R1, R3
ADD @20, R3
MOV @4, R0

;BRANCH IF INCORRECT.
BNE @10%, (R3)+

;BRANCH IF INCORRECT.
BNE @15%, (R3)+

;BRANCH IF INCORRECT.
JMP 74(R1) ;RETURN.

;FRACTIONAL ERROR.
CMP 62(R1), R4 ;CHECK THE FPS.
BNE 20%, (R3)+ ;BRANCH IF INCORRECT.

;CHECK THE FES.
CMP 64(R1), R5
BNE 25%

;INTEGER ERROR.
MOV $MODD1, R2 ;WAS THE INTEGER ERROR ANTICIPATED?
MOV R1, R3
ADD @50, R3
MOV @4, R5
CMP (R2)+, (R3)+ ;BRANCH IF NOT ANTICIPATED.
**D10**

*3400* 031700 001002
*3400* 031702 062702 000072
*3400* 031706 000012

**T21** UNDER OVER FLOW USING MODE WITH TRAPS DISABLED TEST

MOV R1, R2
ADD #2, R2
JMP (R2)

168: ERROR 177
BR 9%

178: ERROR 177
BR 9%

THE ERROR WAS ANTICIPATED SO RETURN.

THE ERROR WAS NOT ANTICIPATED SO REPORT.

THE INTEGER FAILURE HERE.

FIS INCORRECT.

MOV R4, #STMP11
MOV R6(R1), #STMP12
ERROR 200
BR 9%

FIS INCORRECT FIDS.

MOV R5, #STMP13
MOV R6(R1), #STMP14

REPORT FER ERROR.

MOV R5, R6
ADD #6, R2
JMP (R2)

. WORD 0, 0, 0, 0

.likes:

RSETUP

:GO INITIALIZE THE FPS AND STACK; AND

:SEE IF THE USER HAS EXPRESSED

:THE DESIRE TO CHANGE THE SOFTWARE

:VIRTUAL CONSOLE SWITCH REGISTERS .HAS

:THE USER TYPED CONTROL G?.

:-----------------------------------------------

**TEST 22** INTERRUPT CORRECT FLOWS TEST

:THIS IS A TEST OF THE 'CORRECT' FLOWS. THIS PART OF THE MICRO CODE

:HAS AS ITS PURPOSE INSURING THAT INTERRUPT REQUESTS MADE DURING

:CERTAIN LENGTHY FPP INSTRUCTIONS GET HONORED. THIS IS DONE

:IN A WAY SUCH THAT IF AN INTERRUPT REQUEST OCCURS DURING ONE

:OF THESE INSTRUCTIONS THE STATE OF THAT INSTRUCTION'S

:EXECUTION WILL BE THE SAME AS IF THAT INSTRUCTION HAD NEVER

:BEEN FETCHED AND ITS EXECUTION NEVER STARTED. THE MICRO CODE

:WILL RESTORE ALL REGISTERS, BACK UP THE PC AND LEAVE THE

:FIFS AND ACD THROUGH ACS UNMODIFIED.

:THE INSTRUCTIONS FOR WHICH THIS IS NECESSARY ARE:

:* ADD (OR SUB)

:* DIV

:* MUL

:* MOD

:* (BOTH DOUBLE AND FLOATING)

:* ALL ADDRESSING MODES WILL BE TRIED WITH THE ADDD INSTRUCTION. THEN

:* EACH OF THE OTHER INSTRUCTIONS WILL BE TRIED USING MODE 1.
NOTE THAT THIS TEST NEEDS A SPECIAL INTERRUPT MODULE, WHICH WILL PROBABLY ONLY BE PRESENT IN DEC'S MANUFACTURING ENVIRONMENT. TO RUN, THIS SPECIAL EQUIPMENT IS DESIGNED TO RAISE AN INTERRUPT REQUEST IN THE PROCESSOR IF A BIT IS SET IN ITS STATUS REGISTER AND ONLY WHEN AN FPP INSTRUCTION IS ENCOUNTERED. FOR THIS TEST TO BE RUN CONDITIONALLY (DEPENDENT UPON WHETHER OR NOT THE STATUS REGISTER OF THE TEST EQUIPMENT TIMES OUT WHEN REFERENCED) THIS TEST CAN ALSO BE Deselected BY Turning SWITCH 7 OF THE SWITCH REGISTER (PHYSICAL OR VIRTUAL) ON. THE TEST ASSUMES THAT THE TEST EQUIPMENT'S STATUS REGISTER IS AT A LOCATION COMPARABLE (NOTE THAT ALL REFERENCES TO THIS LOCATION ARE MADE INDIRECT THROUGH THIS PROGRAM'S LOCATION CORINT, SO THAT IF THE USER HAS MODIFIED THE TEST EQUIPMENT'S STATUS REGISTER TO Respond TO A DIFFERENT ADDRESS LOCATION CORINT MUST BE MADE TO CONTAIN THAT STATUS REGISTER'S NEW ADDRESS). THIS PROGRAM ASSUMES THAT THE TRAP VECTOR FOR THE TEST EQUIPMENT IS 110, AGAIN NOTE THAT ALL REFERENCES TO THIS TRAP VECTOR ARE INDIRECT, THROUGH THIS PROGRAM'S LOCATION CORINT (IF THE TEST EQUIPMENT IS MADE TO TRAP TO A DIFFERENT VECTOR LOCATION CORINT MUST CONTAIN "THE ADDRESS OF THIS VECTOR.

********************************************************************

1: ST22: SCOPE

BIT #200,25H: SEE IF THE USER HAS Deselected THIS TEST USING THE SWITCH REGISTER.
BEQ COR3: IF NOT SEE IF TEST EQUIPMENT IS PRESENT.
JMP @COR3: ELSE DO NOT RUN TEST.

COR1: MOV #COR2,#ERRVECT: SEE IF THE TEST EQUIPMENT'S STATUS REGISTER TIMES OUT.
MOV @CORINT,#ERRVECT: IF THE REFERENCE TIMES OUT DO
BR COR3: ;DIDN'T TIME OUT SO START TEST.

COR2: CMP (SP)+,(SP)+: IF THE REFERENCE TIMES OUT DO
MOV @CORINT,#ERRVECT: ;NOT RUN TEST.
JMP @COR3: ;SET UP THE LOOP ON ERROR ADDRESS.

TEST ADDDC MODE 0

COR3:
INC @-1
BNE COR3
TYPE
.WORD CORINES

COR33:
LPERR PC,#COR5
JSR PC,#COR5SUB

:ACO

1$: .WORD 40200,100,200,300

2$: .WORD 123456

3$: .WORD 200

4$: ADDD ACO,ACO

5$: NOP

6$: CLR @ACOR4FLG

7$: BR 252

8$: BR 11
F10

M1000-11-0FFB-A

PDF: 24 FPP DIAGNOSTIC PART 2 MACK12 27/1006: 01-NOV-76 T82 21:12 PAGE 22

OFFFB.AII 01-JW-76 21:06

F10

INTERRUPT CORRECT FLOWS TEST

5$: ERROR 253 ;INCORRECT STATE AT INTERRUPT.

11$: TEST ADD MODE 1

CORS:

LPE: RER
JSR PC, #CORSUB
SET UP THE LOOP ON ERROR ADDRESS.

1$: .WORD 40201, 555, 77007, 111111

3$: 187

4$: ADDA (RO), ACO
TEST INSTRUCTION

2$: .WORD 1

RO

3$: .ADD (RO), ACO
TEST INSTRUCTION

4$: .ADD (RO), ACO
TEST INSTRUCTION

5$: .ADD (RO), ACO
TEST INSTRUCTION

6$: .ADD (RO), ACO
TEST INSTRUCTION

7$: .ADD (RO), ACO
TEST INSTRUCTION

8$: .ADD (RO), ACO
TEST INSTRUCTION

9$: .ADD (RO), ACO
TEST INSTRUCTION

10$: .ADD (RO), ACO
TEST INSTRUCTION

11$: .ADD (RO), ACO
TEST INSTRUCTION

CORS:

LPE: RER
JSR PC, #CORSUB
SET UP THE LOOP ON ERROR ADDRESS.

1$: .WORD 40201, 555, 77007, 111111

3$: 187

4$: ADDA (RO), ACO
TEST INSTRUCTION

2$: .WORD 1

RO

3$: .ADD (RO), ACO
TEST INSTRUCTION

4$: .ADD (RO), ACO
TEST INSTRUCTION

5$: .ADD (RO), ACO
TEST INSTRUCTION

6$: .ADD (RO), ACO
TEST INSTRUCTION

7$: .ADD (RO), ACO
TEST INSTRUCTION

8$: .ADD (RO), ACO
TEST INSTRUCTION

9$: .ADD (RO), ACO
TEST INSTRUCTION

10$: .ADD (RO), ACO
TEST INSTRUCTION

11$: .ADD (RO), ACO
TEST INSTRUCTION

CORS:

LPE: RER
JSR PC, #CORSUB
SET UP THE LOOP ON ERROR ADDRESS.

1$: .WORD 40201, 555, 77007, 111111

3$: 187

4$: ADDA (RO), ACO
TEST INSTRUCTION

2$: .WORD 1

RO

3$: .ADD (RO), ACO
TEST INSTRUCTION

4$: .ADD (RO), ACO
TEST INSTRUCTION

5$: .ADD (RO), ACO
TEST INSTRUCTION

6$: .ADD (RO), ACO
TEST INSTRUCTION

7$: .ADD (RO), ACO
TEST INSTRUCTION

8$: .ADD (RO), ACO
TEST INSTRUCTION

9$: .ADD (RO), ACO
TEST INSTRUCTION

10$: .ADD (RO), ACO
TEST INSTRUCTION

11$: .ADD (RO), ACO
TEST INSTRUCTION

CORS:

LPE: RER
JSR PC, #CORSUB
SET UP THE LOOP ON ERROR ADDRESS.

1$: .WORD 40201, 555, 77007, 111111

3$: 187

4$: ADDA (RO), ACO
TEST INSTRUCTION

2$: .WORD 1

RO

3$: .ADD (RO), ACO
TEST INSTRUCTION

4$: .ADD (RO), ACO
TEST INSTRUCTION

5$: .ADD (RO), ACO
TEST INSTRUCTION

6$: .ADD (RO), ACO
TEST INSTRUCTION

7$: .ADD (RO), ACO
TEST INSTRUCTION

8$: .ADD (RO), ACO
TEST INSTRUCTION

9$: .ADD (RO), ACO
TEST INSTRUCTION

10$: .ADD (RO), ACO
TEST INSTRUCTION

11$: .ADD (RO), ACO
TEST INSTRUCTION

CORS:

LPE: RER
JSR PC, #CORSUB
SET UP THE LOOP ON ERROR ADDRESS.

1$: .WORD 40201, 555, 77007, 111111

3$: 187

4$: ADDA (RO), ACO
TEST INSTRUCTION

2$: .WORD 1

RO

3$: .ADD (RO), ACO
TEST INSTRUCTION

4$: .ADD (RO), ACO
TEST INSTRUCTION

5$: .ADD (RO), ACO
TEST INSTRUCTION

6$: .ADD (RO), ACO
TEST INSTRUCTION

7$: .ADD (RO), ACO
TEST INSTRUCTION

8$: .ADD (RO), ACO
TEST INSTRUCTION

9$: .ADD (RO), ACO
TEST INSTRUCTION

10$: .ADD (RO), ACO
TEST INSTRUCTION

11$: .ADD (RO), ACO
TEST INSTRUCTION

CORS:

LPE: RER
JSR PC, #CORSUB
SET UP THE LOOP ON ERROR ADDRESS.

1$: .WORD 40201, 555, 77007, 111111

3$: 187

4$: ADDA (RO), ACO
TEST INSTRUCTION

2$: .WORD 1

RO

3$: .ADD (RO), ACO
TEST INSTRUCTION

4$: .ADD (RO), ACO
TEST INSTRUCTION

5$: .ADD (RO), ACO
TEST INSTRUCTION

6$: .ADD (RO), ACO
TEST INSTRUCTION

7$: .ADD (RO), ACO
TEST INSTRUCTION

8$: .ADD (RO), ACO
TEST INSTRUCTION

9$: .ADD (RO), ACO
TEST INSTRUCTION

10$: .ADD (RO), ACO
TEST INSTRUCTION

11$: .ADD (RO), ACO
TEST INSTRUCTION

CORS:

LPE: RER
JSR PC, #CORSUB
SET UP THE LOOP ON ERROR ADDRESS.

1$: .WORD 40201, 555, 77007, 111111

3$: 187

4$: ADDA (RO), ACO
TEST INSTRUCTION

2$: .WORD 1

RO

3$: .ADD (RO), ACO
TEST INSTRUCTION

4$: .ADD (RO), ACO
TEST INSTRUCTION

5$: .ADD (RO), ACO
TEST INSTRUCTION

6$: .ADD (RO), ACO
TEST INSTRUCTION

7$: .ADD (RO), ACO
TEST INSTRUCTION

8$: .ADD (RO), ACO
TEST INSTRUCTION

9$: .ADD (RO), ACO
TEST INSTRUCTION

10$: .ADD (RO), ACO
TEST INSTRUCTION

11$: .ADD (RO), ACO
TEST INSTRUCTION

CORS:

LPE: RER
JSR PC, #CORSUB
SET UP THE LOOP ON ERROR ADDRESS.

1$: .WORD 40201, 555, 77007, 111111

3$: 187

4$: ADDA (RO), ACO
TEST INSTRUCTION

2$: .WORD 1

RO

3$: .ADD (RO), ACO
TEST INSTRUCTION

4$: .ADD (RO), ACO
TEST INSTRUCTION

5$: .ADD (RO), ACO
TEST INSTRUCTION

6$: .ADD (RO), ACO
TEST INSTRUCTION

7$: .ADD (RO), ACO
TEST INSTRUCTION

8$: .ADD (RO), ACO
TEST INSTRUCTION

9$: .ADD (RO), ACO
TEST INSTRUCTION

10$: .ADD (RO), ACO
TEST INSTRUCTION

11$: .ADD (RO), ACO
TEST INSTRUCTION

CORS:

LPE: RER
JSR PC, #CORSUB
SET UP THE LOOP ON ERROR ADDRESS.
;TEST DIVD MODE 1
SET UP THE LOOP ON ERROR ADDRESS.

LPERR
JSR PC #CORSUB

;COR11:

033445
056677
1$: .WORD 402211,33445,56677,001122 ;ACO

2$: .WORD 1$
;
2DS
;
4$: DIVD (RO),ACO ;TEST INSTRUCTION

5$: .WORD 1$
;

;TEST MULD MODE 1
SET UP THE LOOP ON ERROR ADDRESS.

LPERR
JSR PC #CORSUB

;COR12:

042512
065411
046252
1$: .WORD 40212,165411,46252,63650 ;ACO

2$: .WORD 1$
;
2DS
;
4$: MULD (RO),ACO ;TEST INSTRUCTION

5$: .WORD 1$
;

;TEST MODD MODE 1
SET UP THE LOOP ON ERROR ADDRESS.

LPERR
JSR PC #CORSUB

;COR13:

04213
045654
054542
1$: .WORD 40213,45654,54542,171623 ;ACO

2$: .WORD 1$
;
2DS
;
4$: MODD (RO),ACO ;TEST INSTRUCTION.

5$: .WORD 1$
;

;TEST DIVE MODE 1
SET UP THE LOOP ON ERROR ADDRESS.

LPERR
JSR PC #CORSUB

;COR1:

104113
032646
1$: .WORD 40211,32646,104113 ;ACO

2$: .WORD 1$
;
3$: .WORD 205
;
4$: .WORD 1$
;

;TEST MULG MODE 1
SET UP THE LOOP ON ERROR ADDRESS.

LPERR
JSR PC #CORSUB

;COR2:

165411
046252
1$: .WORD 40212,165411,46252,63650 ;ACO

2$: .WORD 1$
;
3$: .WORD 210
;
4$: .WORD 1$
;

;TEST MODG MODE 1
SET UP THE LOOP ON ERROR ADDRESS.

LPERR
JSR PC #CORSUB

;COR3:

171623
1$: .WORD 40213,171623 ;ACO

2$: .WORD 1$
;
3$: .WORD 412
;
4$: .WORD 1$
;

;TEST DIVG MODE 1
SET UP THE LOOP ON ERROR ADDRESS.

LPERR
JSR PC #CORSUB

;COR4:

04213
045654
054542
1$: .WORD 40213,45654,54542,171623 ;ACO

2$: .WORD 1$
;
3$: .WORD 412
;
4$: .WORD 1$
;

;TEST MULD MODE 2
SET UP THE LOOP ON ERROR ADDRESS.

LPERR
JSR PC #CORSUB

;COR5:

165411
046252
1$: .WORD 40212,165411,46252,63650 ;ACO

2$: .WORD 1$
;
3$: .WORD 210
;
4$: .WORD 1$
;

;TEST MODD MODE 2
SET UP THE LOOP ON ERROR ADDRESS.

LPERR
JSR PC #CORSUB

;COR6:

171623
1$: .WORD 40213,171623 ;ACO

2$: .WORD 1$
;
3$: .WORD 412
;
4$: .WORD 1$
;

;TEST DIVE MODE 2
SET UP THE LOOP ON ERROR ADDRESS.

LPERR
JSR PC #CORSUB

;COR7:

104113
032646
1$: .WORD 40211,32646,104113 ;ACO

2$: .WORD 1$
;
3$: .WORD 205
;
4$: .WORD 1$
;

;TEST MULG MODE 2
SET UP THE LOOP ON ERROR ADDRESS.

LPERR
JSR PC #CORSUB

;COR8:

165411
046252
1$: .WORD 40212,165411,46252,63650 ;ACO

2$: .WORD 1$
;
3$: .WORD 210
;
4$: .WORD 1$
;

;TEST MODG MODE 2
SET UP THE LOOP ON ERROR ADDRESS.

LPERR
JSR PC #CORSUB

;COR9:

171623
1$: .WORD 40213,171623 ;ACO

2$: .WORD 1$
;
3$: .WORD 412
;
4$: .WORD 1$
;

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

CORSUB: CLR 3#CORFLG ;SET FLAG TO INDICATE NO INTERRUPT 
;PENDING.
MOV (SP)+,R1 ;GET A POINTER TO THE ARGUMENTS.
MOV R1,R2 ;SET ACO.
MOV #200,RO 
LDFPS RO 
LDD (R2),ACO 
MOV 12(R1),RO ;SET UP THE FPS.
LDFPS RO 
MOV 10(R1),RO ;SET UP RD.
MOV R1,R2 
ADD #14,R2 
MOV R2,2#30MP2 ;SAVE ADDRESS OF INSTRUCTION IN CASE 
;OF ERROR.
CLR 3#PSV ;CLEAR THE PRIORITY TO ALLOW INTERRUPTS.
MOV #1,ACORTAP ;SET UP THE INTERRUPT VECTOR.
MOV #1,CORFLG ;SET THE FLAG TO INDICATE 
;AN INTERRUPT IS PENDING.
MOV #1,ACORINT ;ENABLE THE TEST EQUIPMENT'S 
;TRAP FUNCTION AND GO 
JMP 14(R1) ;EXECUTE THE INSTRUCTION.

;TRAP TO HERE WHEN THE INTERRUPT OCCURS.
ACORTAP: CMP 3#CORFLG ;FIRST SEE IF AN INTERRUPT WAS PENDING.
BNE CORTV ;IF NOT GO REPORT AN ERROR.
MOV #0,ACORINT ;MAKE SURE THE TEST EQUIPMENT 
;IS NOT INTERRUPT ENABLED.

CORTV: MOV 32764 005137 033130 ;GET THE FPS.
J10

MOV R200,R2  :GET ACO
LDRPS R2  
MOV #CORTMP,R2  
STD ACO,(R2)  

MOV #CORTMP,2*R$TMP3  
MOV R0,#$TMP5  
MOV R4,#$TMP7  
MOV (SP),#$TMP11  
MOV R1,R2  
MOV R2,#$TMP4  
ADD #10,R2  

MOV (R2)+,#$TMP6  
MOV (R2)+,#$TMP10  
MOV R2,#$TMP12  
CMP (SP),R2  :SEE IF THE TRAP OCCURRED AT THE CORRECT ADDRESS.

BNE CORTVO  :RESET THE STACK.
CMP (SP)+(SP)+  :SEE IF R0 IS CORRECT.
CMP R0,10(R1)  :BR IF NOT CORRECT.
BNE CORTVO  :SEE IF ACO WAS CORRECT.

MOV R1,R2  
MOV #CORTMP,R3  
MOV #R5  

CMP (R2)+(R3)+  :BRANCH IF INCORRECT.
BNE CORTVO  

SDB R5,$  
JMP #32(R1)  :IF EVERYTHING IS CORRECT THEN RETURN.

CORTVO: JMP 30(R1)  :CORRECT FLOWS FAILED SO GO REPORT ERROR.

CORTV: MOV (SP),#$TMP2  
                   :AN INTERRUPT OCCURRED WHEN THE FLAG
                   :CORFLG, DID NOT INDICATE THAT ONE WAS
                   :PENDING SO REPORT SPURIOUS TRAP.

CLR #CORFLG

CORTMP: .WORD 0
CORFLG: .WORD 0,0,0,0

CORINT: .WORD 177774  :THIS IS THE ADDRESS, 177774, OF THE
                   :TEST EQUIPMENT'S STATUS REGISTER.
                   :THE CONTENTS OF CORINT CAN BE MODIFIED
                   :IF THIS STATUS REGISTER'S ADDRESS IS
                   :CHANGED.
                   :THIS IS THE ADDRESS OF THE TEST EQUIPMENTS
                   :TRAP VECTOR. LIKE THE STATUS REGISTER'S ADDRESS
                   :DESCRIBED IMMEDIATELY ABOVE
                   :THIS VECTOR CAN BE CHANGED, BUT THE
                   :CONTENTS OF CORTRP MUST INDICATE THE
                   :CHANGE.

CORTRP: .WORD 110

CORDONE: RST  
          :GO INITIALIZE THE FPS AND STACK; AND
REPEATED X TIMES

SEOH:

END OF PASS ROUTINE

INCREASE THE PASS NUMBER ($PASS)
INDICATE END-OF-PROGRAM AFTER 1 PASSES THRUI THE PROGRAM
IF SWI=1 INHIBIT TRACE TRAP
IF THERE'S A MONITOR GO TO IT
IF THERE ISN'T JUMP TO LOOP

SEOPC:

WORD 1

BCT $DOAGH

YES

RESTORE COUNTER

TYPE .65$

GET OVER THE ASCIZ

TYPE ASCIZ STRING

BR .65$

GET OVER THE ASCIZ

MOV $PASS,-(SP)

SAVE $PASS FOR TYPEOUT

Type Pass Number in Octal

Type 6 Digits

Type ASCIZ String

Supress Leading Zeros

Type ASCIZ String

BR .65$

Supress Leading Zeros

BR .65$

MOV $ERTTL,-(SP)

Save $ERTTL for TYPEOUT

Total Number of Errors in Octal

Go Type-Octal ASCIZ

Supress Leading Zeros

Type Carriage Return, Line Feed

Clear Error-Total

Get Monitor Address

Branch If No Monitor

Set Up For An ATI Or RTT
BR $RTRN : GO DO AN RTI OR RTI TO LOAD THE PSW
WITH A CLEARED "T" BIT

$CLR.T: MOV 2442,RO : INSURE RO CONTAINS THE MONITORS
REQ $DOAGH : RETURN ADDRESS
RESET

$ENDAD: JSR PC,RO : CLEAR THE WORLD
NOP NO : GO TO MONITOR
NOP FOR 0005 : SAVE ROOM
NOP
ACT11

$DOAGH: TRAP : PUSH OLD PSW AND PC ON STACK
BIC #0,(SP) : CLEAR THE "T" BIT
BIT #BIT1,5,SWR : RUN WITH TRACE TRAP?
BNE 1: : BR IF NO
BNE 1: IS IT TIME FOR TRACE TRAP
BNE 1: : BR IF NO
BNE 1: SET TRACE TRAP
BNE 1: JUMP TO START OF TEST
$RTYN: RTI : RETURN--THIS IS CHANGED TO
$LOOP: IMP 2(PC)+ : AN "RTI" IF "RTT" IS A LEGAL
$RTNAD: :INUCTION
$RETRN: 377 : "T" BIT STATE INDICATOR
$RETN: 377 377 000 : NULL Character String

.SBTL SCOPE HANDLER ROUTINE

;*********************************************************************
; THIS ROUTINE CONTROLS THE LOADING OF SUBTESTS. IT WILL INCREMENT
; AND LOAD THE TEST NUMBER($STNM) INTO THE DISPLAY REG. (DISPLAY<7:0>)
; AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:0>
; THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
; #SW1=1 LOOP ON TEST
; #SW1=1 INHIBIT ITERATIONS
; #SW2=1 LOOP ON ERROR
; #SW3=1 LOOP ON TEST IN SWR<7:0>
; CALL $SCOPE ; ;SCOPE=10T
; $SCOPE: CKSWR : TEST FOR CHANGE IN SOFT-SWR
1%: BIT #BIT14,2,SWR : LOOP ON PRESENT TEST?
BNE $OVER : YES IF SW1=1

; START OF CODE FOR THE XOR TESTER
$XTSR: BR 6$ : IF RUNNING ON THE "XOR" TESTER CHANGE
$X: IF THIS INSTRUCTION TO A "NOP" (NOP=24D)
MOV $ERRVEC,-(SP) : SAVE THE CONTENTS OF THE ERROR VECTOR
MOV 5$,$ERRVEC : SET FOR TIMEOUT
TST #1706D : TIME OUT ON XOR*
MOV (SP)+,$ERRVEC : RESTORE THE ERROR VECTOR
BR $SVLAD : GO TO THE NEXT TEST

5%: CMP (SP)+,(SP)+ : CLEAR THE STACK AFTER A TIME OUT
***ERROR HANDLER ROUTINE***

; THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
; SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL,
; AND GO TO ERROR ON ERROR
; THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
; $SWIS=1  HALT ON ERROR
; $SWIS=3  INHIBIT ERROR TYPEOUTS
; $SWI0=1  BELL ON ERROR
; $SWI0=1  LOOP ON ERROR
; CALL ERROR N ;ERROR=EMT AND N=ERROR ITEM NUMBER

$ERROR:
  CKSWR $ERFLG ;TEST FOR CHANGE IN SOFT-SWR
  INCB $ERFLG ;SET THE ERROR FLAG
  BEQ $%  ;DON'T LET THE FLAG GO TO ZERO
N10

```
*MAINDC-11-DFPB-A  PDP-11 34 FFP DIAGNOSTIC PART 2  MACYII 27/1006  G1-NOV-76  21:12  PAGE 130

DFPB4.P11  01-NOV-76  21:08  ERROR HANDLER ROUTINE

7159  038312  016777  145144  145202  MOV  $TSTNM,DISPLAY  ;DISPLAY TEST NUMBER AND ERROR FLAG
7160  038370  027777  002000  145172  BIT  $BIT00,3SWR  ;BELL ON ERROR?
7161  038342  011402  013036  37:  BEQ  1$  ;NO - SKIP
7162  038330  010401  013036  TYPE  $R$E$LT$L  ;RING BELL
7163  038360  005267  145132  1$:  INC  $SERTL  ;COUNT THE NUMBER OF ERRORS
7164  038380  011667  145132  MOV  (SP),$ERRPC  ;GET ADDRESS OF ERROR INSTRUCTION
7165  038330  002677  000002  145124  SUB  $2,3$ERRPC
7166  038372  117767  145120  145114  MOV#  3$ERRPC,$ITEMB  ;SKIP TYPEOUT IF SET?
7167  038400  038777  020000  145132  BNE  1$BIT13,3$SWR  ;SKIP TYPEOUTS
7168  038406  001004  002124  BNE  20$  ;GO TO USER ERROR ROUTINE
7169  038410  004767  002124  JSR  PC,ER$Y$TYPE
7170  038414  104401  001313  TYPE  ,$CR$LF
7171  038420  122757  000001  145310  CMPLB  $A$PTENV,$ENV  ;RUNNING IN APT MODE
7172  038420  010007  000004  BNE  2$  ;NO,SKIP APT ERROR REPORT
7173  038430  116760  145060  JSR  PC,$SAT4  ;SET ITEM NUMBER AS ERROR NUMBER
7174  038436  007407  000004  BNE  3$  ;REPORT FATAL ERROR TO APT
7175  038440  007777  000004  BNE  22$  ;HALT ON ERROR
7176  038450  145066  145066  BPL  3$  ;SKIP IF CONTINUE
7177  038452  010000  000000  HALT  4$  ;HALT ON ERROR!
7178  038466  014002  010000  OR$SWR  5$:  CMP  $ITEMB,2$  ;TEST FOR CHANGE IN SOFT-SWR
7179  038466  001000  001000  145052  BIT$W  $BIT09,3$SWR  ; Loop ON ERROR SWITCH SET?
7180  038466  014002  010000  BEQ  4$  ;BR IF NO
7181  038470  016767  145014  MOV  $LP$ERR,(SP)  ;FUDGE RETURN FOR LOOPING
7182  038474  005767  145204  JSR  PC,ESCAPE  ;CHECK FOR AN ESCAPE ADDRESS
7183  038474  011402  013036  BNE  5$:  BNE  5$  ;BR IF NONE
7184  038470  016716  145176  MOV  $ESCAPE,(SP)  ;FUDGE RETURN ADDRESS FOR ESCAPE
7185  038470  002737  000042  3$:  CMP  $SENDAD,3$W  ;ACT-11 AUTO-ACCEPT?
7186  038474  001001  001001  BNE  6$:  BNE  6$  ;BRANCH IF NO
7187  038474  000000  000000  HALT  7$:  BNE  7$  ;YES
7188  038410  022737  003362  000042  CMP  8$:  BNE  8$  ;SEE IF ERROR #377
7189  038410  001001  001001  HALT  9$:  BNE  9$  ;SEE IF ERROR #377
7190  038410  032777  010000  145012  BIT  $BIT09,3$SWR
7191  038414  001001  001001  BNE  6$:  BNE  6$  ;SEE IF ERROR #377
7192  038414  000000  000000  HALT  6$:  BNE  6$  ;SEE IF ERROR #377
7193  038412  027777  010000  145012  BIT  $BIT09,3$SWR
7194  038410  001001  001001  BNE  7$:  BNE  7$  ;SEE IF ERROR #377
7195  038412  001001  001001  BNE  7$:  BNE  7$  ;SEE IF ERROR #377
7196  038410  001001  001001  BNE  8$:  BNE  8$  ;SEE IF ERROR #377
7197  038414  000000  000000  HALT  9$:  BNE  9$  ;SEE IF ERROR #377
7198  038412  027777  003377  145012  CMPB  $377,3$REGO
7199  038410  001002  000002  BNE  1$:  BNE  1$  ;SEE IF ERROR #377
7200  038412  027777  003377  145012  CMPB  $377,3$REGO
7201  038410  001002  000002  BNE  2$:  BNE  2$  ;SEE IF ERROR #377
7202  038410  001000  001000  BNE  3$:  BNE  3$  ;SEE IF ERROR #377
7203  038410  001001  001001  BNE  4$:  BNE  4$  ;SEE IF ERROR #377
7204  038410  001001  001001  BNE  5$:  BNE  5$  ;SEE IF ERROR #377
7205  038410  001001  001001  BNE  6$:  BNE  6$  ;SEE IF ERROR #377
7206  038410  001001  001001  BNE  7$:  BNE  7$  ;SEE IF ERROR #377
7207  038410  001001  001001  BNE  8$:  BNE  8$  ;SEE IF ERROR #377
7208  038410  001001  001001  BNE  9$:  BNE  9$  ;SEE IF ERROR #377
7209  038410  001001  001001  BNE  0$:  BNE  0$  ;SEE IF ERROR #377
7210  038410  001001  001001  BNE  1$:  BNE  1$  ;SEE IF ERROR #377
7211  038410  001001  001001  BNE  2$:  BNE  2$  ;SEE IF ERROR #377
7212  038410  001001  001001  BNE  3$:  BNE  3$  ;SEE IF ERROR #377
7213  038410  001001  001001  BNE  4$:  BNE  4$  ;SEE IF ERROR #377
7214
```

.SBTTL  SAVE AND RESTORE RO-RS ROUTINES

;******************************************************************
;*SAVE RO-RS
;*CALL:
;*  SAVREG
;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
;*
;*+TOP----(+16)
;*+2----(+18)
;*+4----(+18)
```
SAVE AND RESTORE RO-RS ROUTINES

$SAVREG:

R:

* MOV RO,-(SP) : PUSH RO ON STACK
* MOV R1,-(SP) : PUSH R1 ON STACK
* MOV R2,-(SP) : PUSH R2 ON STACK
* MOV R3,-(SP) : PUSH R3 ON STACK
* MOV R4,-(SP) : PUSH R4 ON STACK
* MOV R5,-(SP) : PUSH R5 ON STACK
* MOV R6,-(SP) : SAVE R6 OF MAIN FLOW
* MOV R7,-(SP) : SAVE R7 OF MAIN FLOW
MOV 22(SP),-(SP) : SAVE PC OF CALL
RTI

$RESREG:

R:

* MOV (SP)+,22(SP) : RESTORE PC OF CALL
* MOV (SP)+,15(SP) : RESTORE PS OF CALL
* MOV (SP)+,12(SP) : RESTORE PS OF MAIN FLOW
* MOV (SP)+,9(SP) : RESTORE PS OF MAIN FLOW
* MOV (SP)+,6(SP) : POP STACK INTO R5
MOV (SP)+,4(SP) : POP STACK INTO R4
MOV (SP)+,3(SP) : POP STACK INTO R3
MOV (SP)+,2(SP) : POP STACK INTO R2
MOV (SP)+,1(SP) : POP STACK INTO R1
MOV (SP)+,RO : POP STACK INTO RO
RTI

.SBTIL TYPE ROUTINE

:: RUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
:: THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
:: NOTE1: NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
:: NOTE2: $FILL CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
:: NOTE3: $FILL CONTAINS THE Character TO FILL AFTER.
::
:: CALL:
:: #1: USING A TRAP INSTRUCTION
:: TYPE ,MESADR :: MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
:: OR
:: TYPE
:: MESSADR

$TYPE: TSTB STPLG :: IS THERE A TERMINAL
BPL 1S :: BR IF YES
HALT :: HALT HERE IF NO TERMINAL
BR 3S :: LEAVE
D11

$TYPES: RTS PC

.SBTL "BINARY TO OCTAL (ASCII) AND TYPE"

**THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT OCTAL (ASCII) NUMBER AND TYPE IT.**

*STYP0S--ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE*

**CALL:**
* MOV NUM,-(SP) ; NUMBER TO BE TYPED*
* TYPOS ; CALL FOR TYPEDOUT*
* .BYTE N ; N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE*
* .BYTE M ; M=1 OR 0
  ; = TYPE LEADING ZEROS
  ; = SUPPRESS LEADING ZEROS*

*STYP0C---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS "HE LAST"* STYP0S OR STYP0C

**CALL:**
* MOV NUM,-(SP) ; NUMBER TO BE TYPED*
* TYPOC ; CALL FOR TYPEDOUT*

*STYP0C----ENTER HERE FOR TYPEDOUT OF A 16 BIT NUMBER*

**CALL:**
* MOV NUM,-(SP) ; NUMBER TO BE TYPED*
* TYPOC ; CALL FOR TYPEDOUT*

034636 317646 000000 $STYP0S: MOV A,(SP),-(SP) ; PICKUP THE MODE
034642 116667 000001 000211 MOV B,1,(SP),$OFILL ; LOAD ZERO FILL SWITCH
03464F 116667 000207 MOV B,(SP)+,$Mode+$1 ; NUMBER OF DIGITS TO TYPE
034652 062716 000002 ADD $2,(SP) ; ADJUST RETURN ADDRESS
034656 000406 BR $STYP0C
034662 117677 000001 000171 $STYP0C: MOV B,0,$FOFF ; SET THE ZERO FILL SWITCH
034669 117677 000006 000165 MOV B,$0,$Mode+$1 ; GET FOR SIX(6) DIGITS
03466F 117677 000005 000154 $STYP0C: MOV B,$5,$OCTN ; SET THE ITERATION COUNT
034676 062704 000002 MOV R3,-(SP) ; SAVE R3
03467E 010346 MOV R4,-(SP) ; SAVE R4
034682 010446 MOV R5,-(SP) ; SAVE R5
034686 116704 000145 MOV $0,$Mode+$1,R4 ; GET THE NUMBER OF DIGITS TO TYPE
03468D 005404 NEG R4
034690 062704 000006 ADD $6,R4 ; SUBTRACT IT FOR MAX. ALLOWED
034696 110467 000132 MOV B,R4,$MODE ; SAVE IT FOR USE
03469D 016704 001235 MOV B,$0FILL,R4 ; GET THE ZERO FILL SWITCH
0346A2 000012 CLR R3 ; CLEAR THE OUTPUT WORD
0346A8 005003 1$: ROL R5 ; ROTATE MSB INTO "C"
0346AA 000401 3$: BR 3$ ; GO DO MSB
0346AD 000105 2$: ROL R5, R5 ; FORM THIS_DIGIT
0346B2 000105 2$: ROL R5, R5
0346BA 000105 2$: ROL R5, R5
0346BE 001050 3$: ROL R5, R3 ; GET LSB OF THIS_DIGIT
0346C4 105367 000076 DECB $MODE ; TYPE THIS_DIGIT"
0346C7 100016 BPL 7$ ; BR IF NO
0346CD 042703 177770 BIC $177770,R3 ; GET RID OF JUNK
G11

 THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY

 CALL: RDCHR: INPUT A SINGLE CHARACTER FROM THE "":
 RETURN: CHARACTER IS ON THE STACK

 PUSH DOWN THE PC
 SAVE THE PS
 WAIT FOR A CHARACTER
H11

MOV 35724 4(SP),#104401

.SBTTL TRAP TABLE

::THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE Routines CALLED
::BY THE "TRAP" INSTRUCTION.

::ROUTINE

::STRPAD: .WORD STRAP2
::STRAP: .TYPE CALL=TYPE

::SAVE RO
::GET Trap Address
::Backup by 2
::Get right byte of Trap
::Position for Indexing
::Index to Table
::Go to Routine

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

::MOVE THE PC DOWN
::Move the PSW down
::RESTORE THE PSW

::This routine will pick up 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.

::SAVE RO
::GET Trap Address
::Backup by 2
::Get right byte of Trap
::Position for Indexing
::Index to Table
::Go to Routine

::Go back to User
::Control "O"
.SBTTL  POWER DOWN AND UP ROUTINES

;POWER DOWN ROUTINE

$PWRDN:   MOV  #$ILLUP, &PWRECV ; SET FOR FAST UP
MOV  #130, &PWRECV+2 ; PRI0:7
MOV  R0, - (SP)
PUSH R0 ON STACK
MOV  A1, - (SP)
PUSH A1 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, - (IN)
PUSH R5 ON STACK
MOV  $SMR, - (SP)
PUSH $SMR ON STACK
MOV  SP, $SAVR6 ; SAVE SP
MOV  $SAVR6 ; SET UP VECTOR
HALT
BR   -.2 ; HANG UP

;POWER UP ROUTINE

$PWRUP:   MOV  #$ILLUP, &PWRECV ; SET FOR FAST DOWN
MOV  $SAVR6, SP ; GET SP
CLR  $SAVR6 ; WAIT LOOP FOR THE TTY

1$:  INC  $SAVR6 ; WAIT FOR THE INC
BNE  1$  ; OF WORD
MOV  (SP)+, $SMR ; POP STACK INTO $SMR
MOV  (SP)+, R5 ; POP STACK INTO R5
MOV  (SP)+, R4 ; POP STACK INTO R4
MOV  (SP)+, R3 ; POP STACK INTO R3
MOV  (SP)+, R2 ; POP STACK INTO R2
MOV  (SP)+, R1 ; POP STACK INTO R1
MOV  (SP)+, R0 ; POP STACK INTO R0
MOV  #137, &PWRECV ; SET UP THE POWER DOWN VECTOR
MOV  #130, &PWRECV+2 ; PRI0:7

$SPR:    MOV  $CW, PCER ; REPORT THE POWER FAILURE
MOV  $FAIL, MSGPTR ; POWER FAIL MESSAGE POINTER
MOV  (PC)+, (SP) ; RESTART AT START

$SPRAG:  MOV  $START, Addr ; MY START ADDRESS
CLR  #20, (SP)  ; CLEAR "I" BIT
CLR  $B8 ; CLEAR THE "I" BIT FLAG
ATI

$ILLUP:  HALT  ; THE POWER UP SEQUENCE WAS STARTED
MAINDL: 12-OFFP-A
PDP: 34 FPP DIAGNOSTIC PART 2 MACY11 27(1006) 01-NOV-76 21:26 PAGE : 0C

ERROR TYPE OLT ROUTINE

7703 036314 001002
BNE 6
JMPSMERT4

7723 036316 000137 036576

7726 036322 011000
6%

7727 036324 105710
EFT1:
TSTB (R0)
BNE 7%

7728 036326 001004

7730 036330 013446
MOV (R1)+,-(SP)

7731 036332 104402
TYPOC (R0)
JMP 3MERT2

7732 036334 000137 036560

7733 036340 7%

7734 036340 122710 000002
8%

7735 036344 001011

7736 036344 013102
MOV (R1)+,R2

7737 036346 012246
MOV (R2)+,-(SP)

7738 036350 104401
TYPOC (R0)

7740 036352 104401

7741 036354 037153

7742 036356 037153

7743 036356 011246
MOV (R2),-(SP)

7744 036362 104402
TYPOC (R0)

7745 036364 000137 036560

7746 036364 9%

7747 036370 122710 000003
9%

7748 036374 001021

7749 036374 013102
MOV (R1)+,R2

7750 036376 012246
MOV (R2)+,-(SP)

7751 036380 104401
TYPOC (R0)

7753 036401 037153

7755 036410 012246

7756 036412 104401
TYPOC (R0)

7757 036414 104401

7758 036416 037153

7759 036420 012246

7760 036422 104401
TYPOC (R0)

7761 036424 104401

7762 036426 037153

7763 036430 011246
MOV (R2),-(SP)

7764 036432 104402
TYPOC (R0)

7765 036434 000137 036560

7766 036434 10%

7767 036440 122710 000004
10%

7768 036444 001005

7769 036444 013146

7770 036446 104403

7771 036450 016

7772 036452 0

7773 036454 000

7774 036454 000137 036560

; GET THE ADDRESS OF THE DATA TABLE.
; RETURN IF NO DATA.
; GET A POINTER TO THE DATA TABLE.
; FORMAT ZERO.
; FORMAT TWO.
; FORMAT TWO SO TYPE TWO OCTAL NUMBERS.
; FORMAT THREE.
; FORMAT THREE SO TYPE FOUR OCTAL NUMBERS.
; FORMAT FOUR.
; FORMAT FOUR SO TYPE AN OCTAL NUMBER SUPPRESSING LEADING ZEROS.
```
; ************ FPP SPURIOUS TRAP TO 244 HANDLER
; ************************************************************
; ************************************************************
; THIS ROUTINE HANDLES UNEXPECTED TRAPS TO THE FPP TRAP VECTOR AT 244.
```
:THE LAST FPP INSTRUCTION EXECUTED AND ITS ADDRESS HAS BEEN RECORDED  
:THESE ALONG WITH THE FEC, FPS AND PC OF TRAP ARE REPORTED.

;FPPSPUR: MOV (SP),2#TMP2 ;SAVE PC OF TRAP.  
CMP (SP)+,(SP)+ ;RESTORE SP.  
STFPS RO ;GET FPS  
MOV RO,2#TMP3  
ST T RO  
MOV RO,2#TMP4 ;GET FEC

1$: ERROR 247  
RSETUP ;GO INITIALIZE THE FPS AND STACK; AND SEE IF THE USER HAS EXPRESSED THE DESIRE TO CHANGE THE SOFTWARE; VIRTUAL CONSOLE SWITCH REGISTER (HAS THE USER TYPED CONTROL G?).

JMP @SEOP

.SBTTL CPU SPURIOUS TRAP TO 4 HANDLER

;******************************************************************
;** THIS ROUTINE REPORTS UNEXPECTED CPU TRAPS TO VECTOR 4. **
;******************************************************************

;CSPUR: MOV (SP),2#TMP2 ;SAVE PC OF TRAP.  
CMP (SP)+,(SP)+  

1$: ERROR 250 ;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 @SEOP

.SBTTL CPU SPURIOUS TRAP TO 10 HANDLER

;******************************************************************
;** THIS ROUTINE REPORTS UNEXPECTED CPU TRAPS TO VECTOR 10. **
;******************************************************************

;CPTWO: MOV (SP),2#TMP2 ;SAVE PC OF TRAP.  
CMP (SP)+,(SP)+  

1$: ERROR 251 ;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 @SEOP

.SBTTL SET LOOP ON ERROR ADDRESS ROUTINE

;******************************************************************
SET LOOP ON ERROR ADDRESS ROUTINE

; SBTL TL 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.
; **********

; DO:
CMP a$SWR, 177570
BNE 1$ ; SEE IF THERE IS A PHYSICAL

; CONSOLE SWITCH REGISTER.

; BRANCH IF NO.

; OTHERWISE TYPE THE CONTENTS

; OF THE PROGRAM VIRTUAL SWITCH REGISTER

; AND GIVE THE USER A CHANCE TO

; MODIFY IT.

; E:

MOV #FPSPUR, &FPVEXT
MOV #CPSPUR, &ERRVEXT
MOV #CPSPUR, &ERRVEXT
MOV #CPSPUR, &ERRVEXT
MOV (SP), RO
MOV &STACK, SP
CLR R4
LDPS R4
JMP (RO)

; THESE ARE SPECIAL MESSAGES:

MSA1: .ASCIIZ 'TRAPPED AT:' <TAB> <TAB>
MSA2: .ASCIIZ 'EXPECTED TRAP AT:' <TAB>
MSA3: .ASCIIZ 'GOT RN:' <TAB>
MSA4: .ASCIIZ 'EXPECTED RN:' <TAB>
MSA5: .ASCIIZ 'GOT ACD:' <TAB>
MSA6: .ASCIIZ 'EXPECTED ACD:' <TAB>

POWER: .ASCIIZ '<CRLF>' POWER FAILURE. PROGRAM RESTARTING.' <CRLF>

$TAB: .ASCIIZ '<TAB>'

SPACE: .ASCIIZ '<SPACE>'

AC OPERAND: .ASCIIZ 'AC OPERAND:' <TAB>
FRAC OPERAND: .ASCIIZ 'FRAC OPERAND:' <TAB>
ACO BEFORE EXECUTION: .ASCIIZ 'ACO BEFORE EXECUTION:' <TAB>
ACO AFTER EXECUTION: .ASCIIZ 'ACO AFTER EXECUTION:' <TAB>
EXPECTED RESULT: .ASCIIZ 'EXPECTED RESULT:' <TAB>
GOT RESULT: .ASCIIZ 'GOT RESULT:' <TAB>
FRACTIONAL RESULT: .ASCIIZ 'FRACTIONAL RESULT:' <TAB>
INTEGER RESULT: .ASCIIZ 'INTEGER RESULT:' <TAB>
EXPECTED FRACTION: .ASCIIZ 'EXPECTED FRACTION:' <TAB>
EXPECTED INTEGER: .ASCIIZ 'EXPECTED INTEGER:' <TAB>
B12

THESE ARE ERROR MESSAGES:

EM1. ASIZ: 'FPS BAD AFTER CMPD (R), A.'
EM2. ASIZ: 'ACD MODIFIED BY CMPD (R), A.'
EM3. ASIZ: 'FPS BAD AFTER CMPD.'
EM4. ASIZ: '(BUT ENMT) STATE 22S WENT TO 475 INSTEAD OF 075.'
EM5. ASIZ: 'FPS BAD AFTER CMPD.'
EM6. ASIZ: '(BUT ENMT) STATE 03S WENT TO 075 INSTEAD OF 475.'
EM7. ASIZ: 'FPS BAD AFTER CMPD.'
EM8. ASIZ: '(BUT ENMT) STATE 03S WENT TO 075 INSTEAD OF 475.'
EM9. ASIZ: 'FPS BAD AFTER CMPD.'
EM10. ASIZ: '(BUT ENMT YB) STATE 777 SHOULD HAVE GONE TO 007.'
EM11. ASIZ: 'FPS BAD AFTER CMPD.'
EM12. ASIZ: '(BUT ENMT ZB) STATE 456 SHOULD HAVE GONE TO 010.'
EM13. ASIZ: 'FPS BAD AFTER CMPD.'
EM14. ASIZ: '(BUT XNBT Z2BT) STATE 456 TO 012, TO 363 TO 120.'
EM15. ASIZ: 'FPS BAD AFTER CMPD.'
EM16. ASIZ: 'FPS BAD AFTER CMPD.'
FLAG RESET AND CONSOLE TEST ROUTINE

ASCII 'DIVF (R), A FAILED.'
ASCII 'BUT Y61 WENT TO STATE 006 INSTEAD OF 206.'
ASCII 'XOR OF SIGN BITS FAILED STATE 006.'
ASCII 'DIVF (R), A FAILED.'
ASCII '(BUT Y61 WENT TO STATE 206 INSTEAD OF 006.'
ASCII 'DIVF (R), A FAILED.'
ASCII 'TRUNCATION ERROR. FT=1.'
ASCII 'DIVF (R), A FAILED.'
ASCII 'ROUND ERROR. FT=0.'
ASCII 'DIVD (R), A FAILED.'
ASCII 'FPS BAD AFTER DIVD (R), A.'
ASCII 'DIVD (R), A FAILED.'
ASCII 'TRUNCATION ERROR. FT=1.'
ASCII 'DIVD (R), A FAILED.'
ASCII 'ROUND ERROR. FT=0.'
ASCII 'MULF (R), A FAILED.'
ASCII 'FPS BAD AFTER MULF (R), A.'
ASCII 'MULF (R), A FAILED.'
ASCII 'SIGN BIT BAD STATE 511.'
ASCII 'MULF (R), A FAILED.'
ASCII 'NORMALIZATION FAILED.'
ASCII 'MULF (R), A FAILED.'
ASCII 'NORMALIZATION FAILED.'
ASCII 'MULF (R), A FAILED.'
ASCII 'ROUND ERROR. FT=0.'
ASCII 'MULF (R), A FAILED.'
ASCII 'TRUNCATION ERROR. FT=1.'
ASCII 'MULD (R), A FAILED.'
ASCII 'BAD CONSTANT USED IN THE MUL ALGORITHM.'
ASCII 'MULD (R), A FAILED.'
ASCII 'TRUNCATION ERROR. FT=1.'
ASCII 'MULD (R), A FAILED.'
ASCII 'ROUND ERROR. FT=0.'
FLAG RESET AND CONSOLE TEST Routines

D12
F12

FLAG RESET AND CONSOLE TEST ROUTINE

ASCII 'MODF (R), A INTEGER BAD.' (CRLF)

ASCII 'ACI DID NOT GET 0 IN STATE 142.' (CRLF)

ASCII 'MODF (R), A INTEGER BAD.' (CRLF)

ASCII 'ACI DID NOT GET THE INTEGER IN STATE 134.' (CRLF)

ASCII 'MODF (R), A FRACTION BAD.' (CRLF)

ASCII 'A BAD CONSTANT WAS USED (NOT 24) IN STATE 046.' (CRLF)

ASCII 'CRLF' 'OR (BUT NBIT) STATE 525 WENT TO 050 INSTEAD OF 150.' (CRLF)

ASCII 'MODF (R), A FRACTION BAD.' (CRLF)

ASCII 'A BAD CONSTANT WAS USED (NOT 24) IN STATE 046.' (CRLF)

ASCII 'CRLF' 'OR (BUT NBIT) STATE 525 WENT TO 150 INSTEAD OF 250.' (CRLF)

ASCII 'MODF (R), A FRACTION BAD.' (CRLF)

ASCII 'A BAD CONSTANT WAS USED (NOT 24) IN STATE 046.' (CRLF)

ASCII 'CRLF' 'OR (BUT NBIT) STATE 525 WENT TO 150 INSTEAD OF 250.' (CRLF)

ASCII 'MODF (R), A FRACTION BAD.' (CRLF)

ASCII 'A BAD CONSTANT WAS USED (NOT 24) IN STATE 046.' (CRLF)

ASCII 'CRLF' 'OR (BUT NBIT) STATE 525 WENT TO 150 INSTEAD OF 250.' (CRLF)

ASCII 'MODF (R), A FRACTION BAD.' (CRLF)

ASCII 'A BAD CONSTANT WAS USED (NOT 24) IN STATE 046.' (CRLF)

ASCII 'CRLF' 'OR (BUT NBIT) STATE 525 WENT TO 150 INSTEAD OF 250.' (CRLF)

ASCII 'MODF (R), A FRACTION BAD.' (CRLF)

ASCII 'A BAD CONSTANT WAS USED (NOT 24) IN STATE 046.' (CRLF)

ASCII 'CRLF' 'OR (BUT NBIT) STATE 525 WENT TO 150 INSTEAD OF 250.' (CRLF)

ASCII 'MODF (R), A FRACTION BAD.' (CRLF)

ASCII 'A BAD CONSTANT WAS USED (NOT 24) IN STATE 046.' (CRLF)

ASCII 'CRLF' 'OR (BUT NBIT) STATE 525 WENT TO 150 INSTEAD OF 250.' (CRLF)

ASCII 'MODF (R), A FRACTION BAD.' (CRLF)

ASCII 'A BAD CONSTANT WAS USED (NOT 24) IN STATE 046.' (CRLF)

ASCII 'CRLF' 'OR (BUT NBIT) STATE 525 WENT TO 150 INSTEAD OF 250.' (CRLF)

ASCII 'MODF (R), A FRACTION BAD.' (CRLF)

ASCII 'A BAD CONSTANT WAS USED (NOT 24) IN STATE 046.' (CRLF)

ASCII 'CRLF' 'OR (BUT NBIT) STATE 525 WENT TO 150 INSTEAD OF 250.' (CRLF)

ASCII 'MODF (R), A FRACTION BAD.' (CRLF)

ASCII 'A BAD CONSTANT WAS USED (NOT 24) IN STATE 046.' (CRLF)

ASCII 'CRLF' 'OR (BUT NBIT) STATE 525 WENT TO 150 INSTEAD OF 250.' (CRLF)

ASCII 'MODF (R), A FRACTION BAD.' (CRLF)

ASCII 'A BAD CONSTANT WAS USED (NOT 24) IN STATE 046.' (CRLF)

ASCII 'CRLF' 'OR (BUT NBIT) STATE 525 WENT TO 150 INSTEAD OF 250.' (CRLF)

ASCII 'MODF (R), A FRACTION BAD.' (CRLF)

ASCII 'A BAD CONSTANT WAS USED (NOT 24) IN STATE 046.' (CRLF)

ASCII 'CRLF' 'OR (BUT NBIT) STATE 525 WENT TO 150 INSTEAD OF 250.' (CRLF)

ASCII 'MODF (R), A FRACTION BAD.' (CRLF)

ASCII 'A BAD CONSTANT WAS USED (NOT 24) IN STATE 046.' (CRLF)

ASCII 'CRLF' 'OR (BUT NBIT) STATE 525 WENT TO 150 INSTEAD OF 250.' (CRLF)

ASCII 'MODF (R), A FRACTION BAD.' (CRLF)

ASCII 'A BAD CONSTANT WAS USED (NOT 24) IN STATE 046.' (CRLF)

ASCII 'CRLF' 'OR (BUT NBIT) STATE 525 WENT TO 150 INSTEAD OF 250.' (CRLF)

ASCII 'MODF (R), A FRACTION BAD.' (CRLF)

ASCII 'A BAD CONSTANT WAS USED (NOT 24) IN STATE 046.' (CRLF)

ASCII 'CRLF' 'OR (BUT NBIT) STATE 525 WENT TO 150 INSTEAD OF 250.' (CRLF)
H12

FLAG RESET AND CONSOLE TEST ROUTINE

EM202: .ASCIZ "(CRLF) "EXPECTING UNDERFLOW, FEC=12.'

EM203: .ASCII "(CRLF) "MODD (R), A INTEGER BAD, RESULT OVER OR UNDERFLOW," STATE 241 WENT TO 126 INSTEAD OF 137.'

EM204: .ASCII "(CRLF) "MODD (R), A INTEGER BAD, RESULT OVER OR UNDERFLOW," STATE 047 WENT TO 126 INSTEAD OF 137.'

EM205: .ASCII "(CRLF) "FEV BAD AFTER MODD (R), A,'.

EM206: .ASCII "(CRLF) "EXPECTING OVERFLOW, FEC=10.'

EM207: .ASCII "(CRLF) "ADD (R), A INTEGER BAD, RESULT OVER OR UNDERFLOW," STATE 520 WENT TO 162 INSTEAD OF 163.'

EM208: .ASCII "(CRLF) "ADD (R), A INTEGER BAD, RESULT OVER OR UNDERFLOW," STATE 520 WENT TO 162 INSTEAD OF 163.'

EM209: .ASCII "(CRLF) "ADD (R), A INTEGER BAD, RESULT OVER OR UNDERFLOW," STATE 520 WENT TO 162 INSTEAD OF 163.'

EM210: .ASCII "(CRLF) "ADD (R), A INTEGER BAD, RESULT OVER OR UNDERFLOW," STATE 520 WENT TO 162 INSTEAD OF 163.'

EM211: .ASCII "(CRLF) "ADD (R), A INTEGER BAD, RESULT OVER OR UNDERFLOW," STATE 520 WENT TO 162 INSTEAD OF 163.'

EM212: .ASCII "(CRLF) "ADD (R), A INTEGER BAD, RESULT OVER OR UNDERFLOW," STATE 520 WENT TO 162 INSTEAD OF 163.'

EM213: .ASCII "(CRLF) "ADD (R), A INTEGER BAD, RESULT OVER OR UNDERFLOW," STATE 520 WENT TO 162 INSTEAD OF 163.'

EM214: .ASCII "(CRLF) "ADD (R), A INTEGER BAD, RESULT OVER OR UNDERFLOW," STATE 520 WENT TO 162 INSTEAD OF 163.'

EM215: .ASCII "(CRLF) "ADD (R), A INTEGER BAD, RESULT OVER OR UNDERFLOW," STATE 520 WENT TO 162 INSTEAD OF 163.'

EM216: .ASCII "(CRLF) "ADD (R), A INTEGER BAD, RESULT OVER OR UNDERFLOW," STATE 520 WENT TO 162 INSTEAD OF 163.'

EM217: .ASCII "(CRLF) "ADD (R), A INTEGER BAD, RESULT OVER OR UNDERFLOW," STATE 520 WENT TO 162 INSTEAD OF 163.'

EM218: .ASCII "(CRLF) "ADD (R), A INTEGER BAD, RESULT OVER OR UNDERFLOW," STATE 520 WENT TO 162 INSTEAD OF 163.'

EM219: .ASCII "(CRLF) "ADD (R), A INTEGER BAD, RESULT OVER OR UNDERFLOW," STATE 520 WENT TO 162 INSTEAD OF 163.'

EM220: .ASCII "(CRLF) "ADD (R), A INTEGER BAD, RESULT OVER OR UNDERFLOW," STATE 520 WENT TO 162 INSTEAD OF 163.'

EM221: .ASCII "(CRLF) "ADD (R), A INTEGER BAD, RESULT OVER OR UNDERFLOW," STATE 520 WENT TO 162 INSTEAD OF 163.'
I12

Flag reset and console test routine

EM222: .ASCII "ADD (R), A TRAPPED TO 244. (CRLF)
TH~ Result was an underflow condition but FIU= 0."
EM223: .ASCII "ADD (R), A FAILED TO TRAP TO 244. (CRLF)
TH~ Result was an underflow condition and FIU=1. (CRLF)
EM224: .ASCII "ADD (R), A TRAPPED TO 244. (CRLF)
TH~ Result was an underflow condition but FIU= 0."
EM225: .ASCII "ADD (R), A FAILED TO TRAP TO 244. (CRLF)
TH~ Result was an underflow condition and FIU=1. (CRLF)
EM226: .ASCII "ADD (R), A TRAPPED TO 244. (CRLF)
TH~ Because of an expected overflow condition."
EM227: .ASCII "ADD (R), A TRAPPED TO 244. (CRLF)
TH~ Because of an expected overflow condition."
EM230: .ASCII "ADD (R), A TRAPPED TO 244. (CRLF)
TH~ Because of an expected underflow condition."
EM231: .ASCII "ADD (R), A TRAPPED TO 244. (CRLF)
TH~ Because of an expected underflow condition."
EM232-EM210
J12

PDP 11 34 FPP DIAGNOSTIC PART 2 MACYII 27(1006) 01-NOV-76 21:12 PAGE 152

FLAG RESET AND CONSOLE TEST ROUTINE

063754  041520  041040  042101  .ASCIZ \PC BAD AFTER LDCFD #NUM,A. TRAP TO 4.
064092  041520  041040  042101  EM236: .ASCIZ \PC BAD AFTER LDCFD #NUM,A.
064095  041520  041040  042101  EM237: .ASCIZ \RO BAD AFTER LDCFD (R)+,A.
064055  041060  040000  040502  .ASCI "\CRLF" A BAD CONSTANT WAS USED.
064107  0200  020101  040502  EM240: .ASCIZ \FPS BAD AFTER LDCFD (R)+,A."
064141  0106  051520  041040  EM241: .ASCIZ \FPS BAD AFTER LDCFD (R)+,A."
064175  0114  041504  042106  .ASCI "\LDCFD (R)+,A FAILED."
064221  0200  044124  020105  .ASCIZ \CRLF" THE FD "
064231  0102  052111  053440  .ASCI "BIT WAS NOT COMPLIMENTED ."
064262  047111  051440  04524  .ASCIZ \IN STATE 017.
064300  050106  020230  040502  EM242: .ASCIZ \FPS BAD AFTER LDCFD (R)+,A."
064334  042114  042103  020106  EM243: .ASCIZ \LDCFD (R)+,A FAILED."
064376  042100  042100  043040  .ASCIZ \CRLF" THE FD "
064370  044502  020124  040527  .ASCI "BIT WAS NOT COMPLIMENTED ."
064421  0111  020116  052123  .ASCIZ \IN STATE 017." EM244: .ASCIZ \LDCFD (R)+,A RESULT INCORRECT."
064437  0114  041504  043104  EM245: .ASCIZ \LDCFD (R)+,A FAILED."
064521  0200  042523  020124  .ASCIZ \CRLF" SET SIGN FAILED ."
064526  047111  051440  04524  .ASCIZ \IN STATE 512 ." EM246: .ASCIZ \UNEXPECTED CPU TRAP TO 4 ."
064550  047125  051405  045260  EM247: .ASCIZ \UNEXPECTED CPU TRAP TO 4 ."
064614  047110  051426  045280  EM251: .ASCIZ \UNEXPECTED CPU TRAP TO 10 ."
064701  0103  051117  042522  EM252: .ASCIZ \CORRECT FLOWS INTERRUPT TEST MODULE FAILED TO INTERRUPT ."

064772  042101  042104  040440  EM253: .ASCIZ \ADDD ACD,ACD FAILED IN THE INTERRUPT CORRECT FLOWS."
064772  042101  042104  040440  EM254: .ASCIZ \ADDD (RO),ACD FAILED IN THE INTERRUPT CORRECT FLOWS ."
These are the data format specifiers for the data table:

DF1: .BYTE 4,0,5,0,5,0,5,5,3,5,5,3
DF2: .BYTE 4,0,5,0,5,5,3,5,5,3
DF3=DF1
DF4=DF1
DF5=DF1
DF6=DF1
DF7=DF1
DF8=DF1
DF9=DF1
DF10=DF1
DF11=DF1
DF12=DF1
DF13=DF1
DF14=DF1
DF15=DF1
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>046216</td>
</tr>
<tr>
<td>B</td>
<td>046216</td>
</tr>
<tr>
<td>C</td>
<td>046216</td>
</tr>
<tr>
<td>D</td>
<td>046216</td>
</tr>
<tr>
<td>E</td>
<td>046216</td>
</tr>
<tr>
<td>F</td>
<td>046216</td>
</tr>
<tr>
<td>G</td>
<td>046216</td>
</tr>
<tr>
<td>H</td>
<td>046216</td>
</tr>
<tr>
<td>I</td>
<td>046216</td>
</tr>
<tr>
<td>J</td>
<td>046216</td>
</tr>
<tr>
<td>K</td>
<td>046216</td>
</tr>
<tr>
<td>L</td>
<td>046216</td>
</tr>
<tr>
<td>M</td>
<td>046216</td>
</tr>
<tr>
<td>N</td>
<td>046216</td>
</tr>
<tr>
<td>O</td>
<td>046216</td>
</tr>
<tr>
<td>P</td>
<td>046216</td>
</tr>
<tr>
<td>Q</td>
<td>046216</td>
</tr>
<tr>
<td>R</td>
<td>046216</td>
</tr>
<tr>
<td>S</td>
<td>046216</td>
</tr>
<tr>
<td>T</td>
<td>046216</td>
</tr>
<tr>
<td>U</td>
<td>046216</td>
</tr>
<tr>
<td>V</td>
<td>046216</td>
</tr>
<tr>
<td>W</td>
<td>046216</td>
</tr>
<tr>
<td>X</td>
<td>046216</td>
</tr>
<tr>
<td>Y</td>
<td>046216</td>
</tr>
<tr>
<td>Z</td>
<td>046216</td>
</tr>
</tbody>
</table>

**Note:** The table seems to be a symbolic representation, possibly for a technical or scientific context. Without more context, it's challenging to interpret the values accurately.
<table>
<thead>
<tr>
<th>EM10</th>
<th>057625</th>
<th>EM24</th>
<th>041756</th>
<th>EM37</th>
<th>051643</th>
<th>GGER9</th>
<th>010402</th>
<th>GG7</th>
<th>007144</th>
</tr>
</thead>
<tbody>
<tr>
<td>EM11</td>
<td>057722</td>
<td>EM25</td>
<td>041941</td>
<td>EM38</td>
<td>051646</td>
<td>GGER7</td>
<td>010450</td>
<td>GG8</td>
<td>007142</td>
</tr>
<tr>
<td>EM12</td>
<td>057127</td>
<td>EM26</td>
<td>041975</td>
<td>EM39</td>
<td>051733</td>
<td>GGER8</td>
<td>010546</td>
<td>GG9</td>
<td>007222</td>
</tr>
<tr>
<td>EM13</td>
<td>057224</td>
<td>EM27</td>
<td>043000</td>
<td>EM40</td>
<td>052035</td>
<td>GGER9</td>
<td>010614</td>
<td>G'64R = 104403</td>
<td></td>
</tr>
<tr>
<td>EM14</td>
<td>057329</td>
<td>EM28</td>
<td>043134</td>
<td>EM41</td>
<td>052231</td>
<td>GGDON</td>
<td>025712</td>
<td>HGDA70</td>
<td>006696</td>
</tr>
<tr>
<td>EM15</td>
<td>057426</td>
<td>EM29</td>
<td>043437</td>
<td>EM42</td>
<td>052436</td>
<td>GGE1</td>
<td>025846</td>
<td>HGDA80</td>
<td>006696</td>
</tr>
<tr>
<td>EM16</td>
<td>057539</td>
<td>EM30</td>
<td>044726</td>
<td>EM43</td>
<td>052655</td>
<td>GGE10</td>
<td>025910</td>
<td>HGDA90</td>
<td>006696</td>
</tr>
<tr>
<td>EM17</td>
<td>055417</td>
<td>EM31</td>
<td>044916</td>
<td>EM44</td>
<td>052700</td>
<td>GGE11</td>
<td>025960</td>
<td>HGDA00</td>
<td>006696</td>
</tr>
<tr>
<td>EM18</td>
<td>055417</td>
<td>EM32</td>
<td>045314</td>
<td>EM45</td>
<td>053033</td>
<td>GGE12</td>
<td>026130</td>
<td>HGDA10</td>
<td>006696</td>
</tr>
<tr>
<td>EM19</td>
<td>055417</td>
<td>EM33</td>
<td>045610</td>
<td>EM46</td>
<td>053233</td>
<td>GGE13</td>
<td>026200</td>
<td>HGDA20</td>
<td>006696</td>
</tr>
<tr>
<td>EM20</td>
<td>055417</td>
<td>EM34</td>
<td>045641</td>
<td>EM47</td>
<td>053435</td>
<td>GGE14</td>
<td>026200</td>
<td>HGDA30</td>
<td>006696</td>
</tr>
<tr>
<td>EM21</td>
<td>055417</td>
<td>EM35</td>
<td>045641</td>
<td>EM48</td>
<td>053635</td>
<td>GGE15</td>
<td>026200</td>
<td>HGDA40</td>
<td>006696</td>
</tr>
<tr>
<td>EM22</td>
<td>055417</td>
<td>EM36</td>
<td>045641</td>
<td>EM49</td>
<td>053635</td>
<td>GGE16</td>
<td>026200</td>
<td>HGDA50</td>
<td>006696</td>
</tr>
<tr>
<td>EM23</td>
<td>055417</td>
<td>EM37</td>
<td>045641</td>
<td>EM50</td>
<td>053635</td>
<td>GGE17</td>
<td>026200</td>
<td>HGDA60</td>
<td>006696</td>
</tr>
<tr>
<td>EM24</td>
<td>055417</td>
<td>EM38</td>
<td>045641</td>
<td>EM51</td>
<td>053635</td>
<td>GGE18</td>
<td>026200</td>
<td>HGDA70</td>
<td>006696</td>
</tr>
<tr>
<td>EM25</td>
<td>055417</td>
<td>EM39</td>
<td>045641</td>
<td>EM52</td>
<td>053635</td>
<td>GGE19</td>
<td>026200</td>
<td>HGDA80</td>
<td>006696</td>
</tr>
<tr>
<td>EM26</td>
<td>055417</td>
<td>EM40</td>
<td>045641</td>
<td>EM53</td>
<td>053635</td>
<td>GGE20</td>
<td>026200</td>
<td>HGDA90</td>
<td>006696</td>
</tr>
<tr>
<td>EM27</td>
<td>055417</td>
<td>EM41</td>
<td>045641</td>
<td>EM54</td>
<td>053635</td>
<td>GGE21</td>
<td>026200</td>
<td>HGDA00</td>
<td>006696</td>
</tr>
<tr>
<td>EM28</td>
<td>055417</td>
<td>EM42</td>
<td>045641</td>
<td>EM55</td>
<td>053635</td>
<td>GGE22</td>
<td>026200</td>
<td>HGDA10</td>
<td>006696</td>
</tr>
<tr>
<td>EM29</td>
<td>055417</td>
<td>EM43</td>
<td>045641</td>
<td>EM56</td>
<td>053635</td>
<td>GGE23</td>
<td>026200</td>
<td>HGDA20</td>
<td>006696</td>
</tr>
<tr>
<td>EM30</td>
<td>055417</td>
<td>EM44</td>
<td>045641</td>
<td>EM57</td>
<td>053635</td>
<td>GGE24</td>
<td>026200</td>
<td>HGDA30</td>
<td>006696</td>
</tr>
<tr>
<td>EM31</td>
<td>055417</td>
<td>EM45</td>
<td>045641</td>
<td>EM58</td>
<td>053635</td>
<td>GGE25</td>
<td>026200</td>
<td>HGDA40</td>
<td>006696</td>
</tr>
<tr>
<td>EM32</td>
<td>055417</td>
<td>EM46</td>
<td>045641</td>
<td>EM59</td>
<td>053635</td>
<td>GGE26</td>
<td>026200</td>
<td>HGDA50</td>
<td>006696</td>
</tr>
<tr>
<td>EM33</td>
<td>055417</td>
<td>EM47</td>
<td>045641</td>
<td>EM60</td>
<td>053635</td>
<td>GGE27</td>
<td>026200</td>
<td>HGDA60</td>
<td>006696</td>
</tr>
<tr>
<td>EM34</td>
<td>055417</td>
<td>EM48</td>
<td>045641</td>
<td>EM61</td>
<td>053635</td>
<td>GGE28</td>
<td>026200</td>
<td>HGDA70</td>
<td>006696</td>
</tr>
<tr>
<td>EM35</td>
<td>055417</td>
<td>EM49</td>
<td>045641</td>
<td>EM62</td>
<td>053635</td>
<td>GGE29</td>
<td>026200</td>
<td>HGDA80</td>
<td>006696</td>
</tr>
<tr>
<td>EM36</td>
<td>055417</td>
<td>EM50</td>
<td>045641</td>
<td>EM63</td>
<td>053635</td>
<td>GGE30</td>
<td>026200</td>
<td>HGDA90</td>
<td>006696</td>
</tr>
<tr>
<td>EM37</td>
<td>055417</td>
<td>EM51</td>
<td>045641</td>
<td>EM64</td>
<td>053635</td>
<td>GGE31</td>
<td>026200</td>
<td>HGDA00</td>
<td>006696</td>
</tr>
<tr>
<td>EM38</td>
<td>055417</td>
<td>EM52</td>
<td>045641</td>
<td>EM65</td>
<td>053635</td>
<td>GGE32</td>
<td>026200</td>
<td>HGDA10</td>
<td>006696</td>
</tr>
<tr>
<td>EM39</td>
<td>055417</td>
<td>EM53</td>
<td>045641</td>
<td>EM66</td>
<td>053635</td>
<td>GGE33</td>
<td>026200</td>
<td>HGDA20</td>
<td>006696</td>
</tr>
<tr>
<td>EM40</td>
<td>055417</td>
<td>EM54</td>
<td>045641</td>
<td>EM67</td>
<td>053635</td>
<td>GGE34</td>
<td>026200</td>
<td>HGDA30</td>
<td>006696</td>
</tr>
<tr>
<td>EM41</td>
<td>055417</td>
<td>EM55</td>
<td>045641</td>
<td>EM68</td>
<td>053635</td>
<td>GGE35</td>
<td>026200</td>
<td>HGDA40</td>
<td>006696</td>
</tr>
<tr>
<td>EM42</td>
<td>055417</td>
<td>EM56</td>
<td>045641</td>
<td>EM69</td>
<td>053635</td>
<td>GGE36</td>
<td>026200</td>
<td>HGDA50</td>
<td>006696</td>
</tr>
</tbody>
</table>

For more information, please refer to "symbol table."