.REM @

IDENTIFICATION

PRODUCT CODE: AC-E938C-MC

PRODUCT NAME: CXFPBC0 FP11-A,B,C MODULE

PRODUCT DATE: SEPTEMBER 1978

MAINTAINER: DEC/X11 SUPPORT GROUP

SEQ 0002

1. ABSTRACT

FPB IS A BKMOD THAT EXERCISES AN FP11-B OR FP11-C FLOATING POINT PROCESSOR. IT EXERCISES THE PROCESSOR BY CALCULATING THE LEFT AND RIGHT HAND SIDES OF 3 IDENTITIES AND COMPARING THE RESULTS. THE NUMBERS ARE GENERATED BY A RANDOM NUMBER GENERATOR. THE IDENTITIES ARE EVALUATED IN BOTH SINGLE AND DOUBLE PRECISION.

2. REQUIREMENTS

HARDWARE: AN FP11-B OR FP11-C FLOATING POINT PROCESSOR CONFIGURED ON A PDP-11/45, 55, OR 70.

STORAGE:: FPB REQUIRES:
1. DECIMAL WORDS: 1498
2. OCTAL WORDS: 02732
3. OCTAL BYTES: 5664

3. PASS DEFINITION

ONE PASS OF THE FPB MODULE CONSISTS OF 2500 (OCTAL) CYCLES OF THE TEST SEQUENCE.

4. EXECUTION TIME

ONE PASS OF FPB RUNNING ALONE ON A PDP-11/70 TAKES APPROXIMATELY 20 SECONDS.

5. CONFIGURATION REQUIREMENTS

DEFAULT PARAMETERS:

VECTOR: NONE

REQUIRED PARAMETERS:

NONE

6. DEVICE/OPTION SETUP

MAKE SURE A FLOATING POINT PROCESSOR IS INSTALLED.

7. MODULE OPERATION

TEST SEQUENCE:

A. GENERATE 2 SINGLE PRECISION FLOATING POINT NUMBERS.
B. CALCULATE LEFT HAND SIDE IF FIRST IDENTITY.

C. CALCULATE RIGHT HAND SIDE OF FIRST IDENTITY.
D. CALCULATE THE GUARANTEED NUMBER OF IDENTICAL BITS.

PAGE 2

E. TEST THE 2 RESULTS TO BE WITHIN NUMBER OF GUARANTEED BITS.
F. REPEAT STEPS B THRU E FOR SECOND AND THIRD IDENTITIES.
G. TEST LOAD AND STORE CONVERT INSTRUCTIONS.
H. GENERATE 2 DOUBLE PRECISION FLOATING POINT NUMBERS.
I. REPEAT STEPS B THRU G IN DOUBLE PRECISION.

8. OPERATION OPTIONS

NONE

9. NON-STANDARD PRINTOUTS

IF THE LEFT AND RIGHT HAND SIDES OF ANY OF THE IDENTITIES ARE NOT EQUAL WITHIN THE GUARANTEED NUMBER OF BITS AN ERROR CALL IS MADE TO THE DEC/X11 MONITOR. THEN A MESSAGE CALL IS MADE WHICH TYPES THE LEFT AND RIGHT HAND SIDES OF THE EQUATION IN THE FOLLOWING FORMAT:

```
                    DATA1
S EEE AAA BBBBBB CCCCCC DDDDDD XXXXXX
                    DATA2
S EEE AAA BBBBBB CCCCCC DDDDDD XXXXXX
```

WHERE:

```
S       = SIGN
EEE     = EXPONENT (SHOULD ALWAYS BE 200)
AAA     = FRACTION BITS <57:51> (RIGHT JUSTIFIED)
BBBBBB  = FRACTION BITS <50:35>
CCCCCC  = FRACTION BITS <34:19>
DDDDDD  = FRACTION BITS <18:03>
XXXXXX  = EXTENDED EXPONENT (2'S COMPLIMENT NOTATION)
```

THE ABOVE EXAMPLE IS GIVEN FOR A DOUBLE PRECISION FAILURE. IF A SINGLE PRECISION FAILURE OCCURS, WORDS C AND D ARE NOT TYPED.

@

```
                                        .LIST   SEQ,BIN,LOC
                                        .NLIST  MD,MC,CND
                                        .LIST   ME

                    000000      AC0=%0
                    000000      R0=%0
                    000001      AC1=%1
                    000001      R1=%1
                    000002      AC2=%2
                    000002      R2=%2
                    000003      AC3=%3
                    000003      R3=%3
                    000004      R4=%4
                    000005      R5=%5
                    000006      R6=%6
                    000006      SP=R6
                    000007      R7=%7
                    000007      PC=R7


                                        .MACRO  STARS
                                ;;**********************************************************************
                                        .ENDM   STARS
000000'                                 BKMOD   <FPBC >,1,,,,2500,100
000000'                                 MODULE  40020,FPBC  ,1,,,,2500,100
                                        .TITLE  FPBC DEC/X11 SYSTEM EXERCISER MODULE
                                ;       DDXCOM  VERSION  6          23-MAY-78
                                        .LIST   BIN
                                ;**********************************************************************
000000'                         BEGIN:
000000' 050106  041502      040 MODNAM: .ASCII  /FPBC / ;MODULE NAME.
000005'     000                 XFLAC:  .BYTE   OPEN        ;USED TO KEEP TRACK OF WBUFF USAGE
000006' 000001                  ADDR:   1+0                 ;1ST DEVICE ADDR.
000010' 000000                  VECTOR: +0                  ;1ST DEVICE VECTOR.
000012'     000                 BR1:    .BYTE   PRTY+0      ;1ST BR LEVEL.
000013'     000                 BR2:    .BYTE   PRTY+0      ;2ND BR LEVEL.
000014' 000001                  DVID1:  +1                  ;DEVICE INDICATOR 1.
000016' 000000                  SR1:    OPEN                ;SWITCH REGISTER 1
000020' 000000                  SR2:    OPEN                ;SWITCH REGISTER 2
000022' 000000                  SR3:    OPEN                ;SWITCH REGISTER 3
000024' 000000                  SR4:    OPEN                ;SWITCH REGISTER 4
                                ;**********************************************************************
000026' 040020                  STAT:   40020               ;STATUS WORD.
000030' 000224'                 INIT:   START               ;MODULE START ADDR.
000032' 000224'                 SPOINT: MODSP               ;MODULE STACK POINTER.
000034' 000000                  PASCNT: 0                   ;PASS COUNTER.
000036' 002500                  ICONT:  2500                ;# OF ITERATIONS PER PASS=2500
000040' 000000                  ICOUNT: 0                   ;LOC TO COUNT ITERATIONS
000042' 000000                  SOFCNT: 0                   ;LOC TO SAVE TOTAL SOFT ERRORS
000044' 000000                  HRDCNT: 0                   ;LOC TO SAVE TOTAL HARD ERRORS
000046' 000000                  SOFPAS: 0                   ;LOC TO SAVE SOFT ERRORS PER PASS
000050' 000000                  HRDPAS: 0                   ;LOC TO SAVE HARD ERRORS PER PASS
000052' 000000                  SYSCNT: 0                   ;# OF SYS ERRORS ACCUMULATED
000054' 000000                  RANNUM: 0                   ;HOLDS RANDOM # WHEN RAND MACRO IS CALLED
000056'                         CONFIG:
000056' 000000                  RES1:   0                   ;RESERVED FOR MONITOR USE
000060' 000000                  RES2:   0                   ;RESERVED FOR MONITOR USE
000062' 000000                  SVR0:   OPEN                ;LOC TO SAVE R0.
```



```
000064' 000000                  SVR1:   OPEN                ;LOC TO SAVE R1.
000066' 000000                  SVR2:   OPEN                ;LOC TO SAVE R2.
000070' 000000                  SVR3:   OPEN                ;LOC TO SAVE R3.
000072' 000000                  SVR4:   OPEN                ;LOC TO SAVE R4.
000074' 000000                  SVR5:   OPEN                ;LOC TO SAVE R5.
000076' 000000                  SVR6:   OPEN                ;LOC TO SAVE R6.
000100' 000000                  CSRA:   OPEN                ;ADDR OF CURRENT CSR.
000102'                         SBADR:                      ;ADDR OF GOOD DATA, OR
000102' 000000                  ACSR:   OPEN                ;CONTENTS OF CSR.
000104'                         WASADR:                     ;ADDR OF BAD DATA, OR
000104' 000000                  ASTAT:  OPEN                ;STATUS REG CONTENTS.
000106'                         ERRTYP:                     ;TYPE OF ERROR
000106' 000000                  ASB:    OPEN                ;EXPECTED DATA.
000110' 000000                  AWAS:   OPEN                ;ACTUAL DATA.
000112' 000224'                 RSTRT:  RESTRT              ;RESTART ADDRESS AFTER END OF PASS
000114' 000000                  WDTO:   OPEN                ;WORDS TO MEMORY PER ITERATION
000116' 000000                  WDFR:   OPEN                ;WORDS FROM MEMORY PER ITERATION
000120' 000000                  INTR:   OPEN                ;# OF INTERRUPTS PER ITERATION
000122' 000100                  IDNUM:  100                 ;MODULE IDENTIFICATION NUMBER=100
                    001                 .IF NP  <>
                                RBUFVA:                     ;READ BUFFER VIRTUAL ADDRESS
                                RBUFPA: OPEN                ;READ BUFFER PHYSICAL ADDRESS
                                RBUFEA: OPEN                ;READ BUFFER EA BITS
                                RBUFSZ:                     ;SIZE OF THE READ BUFFER
                                WBUFPA: OPEN                ;WRITE BUFFER PHYSICAL ADDRESS
                                WBUFEA: OPEN                ;WRITE BUFFER EA BITS
                                WBUFRQ:                     ;WRITE BUFFER SIZE REQUESTED
                                WBUFSZ: OPEN                ;WRITE BUFFER SIZE AVAILABLE
                                CDERCT: OPEN                ;CDATA/DATCK ERROR COUNT
                                CDWDCT: OPEN                ;CDATA/DATCK WORD COUNT
                                FREE:   OPEN                ;RESERVED FOR FUTURE USE
                                        .NLIST
                                        IMOD%.=0
                                        .LIST
                                        .ENDC
                    000                 .REPT   SPSIZ       ;MODULE STACK STARTS HERE.
                    000040              .NLIST
                                        .WORD   0
                                        .LIST
                                        .ENDR
000124' 000000                          .WORD   0
000126' 000000                          .WORD   0
000130' 000000                          .WORD   0
000132' 000000                          .WORD   0
000134' 000000                          .WORD   0
000136' 000000                          .WORD   0
000140' 000000                          .WORD   0
000142' 000000                          .WORD   0
000144' 000000                          .WORD   0
000146' 000000                          .WORD   0
000150' 000000                          .WORD   0
000152' 000000                          .WORD   0
000154' 000000                          .WORD   0
000156' 000000                          .WORD   0
000160' 000000                          .WORD   0
000162' 000000                          .WORD   0
```

```
  247  000164' 000000                  .WORD   0
  248  000166' 000000                  .WORD   0
  249  000170' 000000                  .WORD   0
  250  000172' 000000                  .WORD   0
  251  000174' 000000                  .WORD   0
  252  000176' 000000                  .WORD   0
  253  000200' 000000                  .WORD   0
  254  000202' 000000                  .WORD   0
  255  000204' 000000                  .WORD   0
  256  000206' 000000                  .WORD   0
  257  000210' 000000                  .WORD   0
  258  000212' 000000                  .WORD   0
  259  000214' 000000                  .WORD   0
  260  000216' 000000                  .WORD   0
  261  000220' 000000                  .WORD   0
  262  000222' 000000                  .WORD   0
  263  000224'                 MODSP:
  264                          ;*************************************************************************
  265                                  .LIST   SEQ
  266                          ;EQUATE STATEMENTS
  267          000000                  OPEN=0
  268          177776                  PS=177776
  269          177776                  PSW=177776
  270
  271          100000                  BIT15=100000
  272          040000                  BIT14=40000
  273          020000                  BIT13=20000
  274          010000                  BIT12=10000
  275          004000                  BIT11=4000
  276          002000                  BIT10=2000
  277          001000                  BIT9=1000
  278          000400                  BIT8=400
  279          000200                  BIT7=200
  280          000100                  BIT6=100
  281          000040                  BIT5=40
  282          000020                  BIT4=20
  283          000010                  BIT3=10
  284          000004                  BIT2=4
  285          000002                  BIT1=2
  286          000001                  BIT0=1
  287          001000                  ADDR22=1000
  288          000000                  R0=%0
  289          000001                  R1=%1
  290          000002                  R2=%2
  291          000003                  R3=%3
  292          000004                  R4=%4
  293          000005                  R5=%5
  294          000006                  R6=%6
  295          000007                  R7=%7
  296          000006                  SP=%6
  297          000007                  PC=%7
  298          000004                  PIRQS=IOT
  299          000340                  PRTY7=340
  300          000300                  PRTY6=300
  301          000240                  PRTY5=240
  302          000200                  PRTY4=200
```



```
  303          000140                  PRTY3=140
  304          000100                  PRTY2=100
  305          000040                  PRTY1=40
  306          000000                  PRTY0=0
  307          000000                  PRTY=0
  308          005746                  PUSH=005746
  309          024646                  PUSH2=024646
  310          005726                  POPSP=005726
  311          022626                  POPSP2=022626
  312          000000                  TRPDFD=0
  313          000000                  NULL=0
  314
  315
  316
  317                          ;DEFFERRED SERVICE:
  318
  319
  320          104400                  EXIT$=TRAP+TRPDFD
  321          104401                  MSG$=TRAP+TRPDFD
  322          104402                  MSGS$=TRAP+TRPDFD
  323          104403                  MSGN$=TRAP+TRPDFD
  324          104404                  DATER$=TRAP+TRPDFD
  325          104405                  HRDER$=TRAP+TRPDFD
  326          104406                  SOFER$=TRAP+TRPDFD
  327          104407                  BREAK$=TRAP+TRPDFD
  328          104410                  END$=TRAP+TRPDFD
  329          104411                  DATCK$=TRAP+TRPDFD
  330          104412                  CDATA$=TRAP+TRPDFD
  331          104413                  ENDIT$=TRAP+TRPDFD
  332
  333
  334                          ;DIRECT SERVICE:
  335
  336
  337          104414                  GWBUF$= TRAP + TRPDFD
  338          104415                  GETPA$= TRAP + TRPDFD
  339          104416                  MAP22$= TRAP + TRPDFD
  340          104417                  RAND$= TRAP + TRPDFD
  341          104420                  OTOA$= TRAP + TRPDFD
  342          104421                  BTOD$= TRAP + TRPDFD
  343                                  .NLIST  MC
  344                          .SBTTL  SINGLE PRECISION
  345                          ;*      THIS TEST TAKES TWO RANDOM NUMBERS (A AND B) AND
  346                          ;*      COMPARES THE RESULTS OF TWO EQUAL CALCULATIONS.
  347                          ;*      EACH SECTION EVALUATES A DIFFERENT EQUATION AS DESCRIBED BELOW:
  348                          ;*      SECT1   (A+B)**2=A**2+2*A*B+B**2
  349                          ;*      SECT2   (A+B)*(A-B)=A**2-B**2
  350                          ;*      SECT3   A/B*B=A
  351  000224'                         STARS
  352                          ;;*************************************************************
  353  000224'                 START:
  354  000224'                 RESTRT:
  355  000224' 004767  004136  LOOP:   JSR     PC,FLTSGL       ;GET RANDOM OPERANDS
  356  000230' 170127  000040          LDFPS   #40             ;INIT FPS
  357  000234' 172567  004446          LDF     $TMP0,AC1       ;LOAD A OPERAND
  358  000240' 172467  004446          LDF     $TMP2,AC0       ;LOAD B OPERAND
```

```
                                        MOV     $REG0,$AC1      ;SETUP EXTENDED
                                        MOV     $REG1,$AC0      ;EXPONENTS
                                        JSR     PC,FLTADD       ;PERFORM THE ADD
                                        STF     AC1,AC0         ;SETUP AC0 TO
                                        MOV     $AC1,$AC0       ;PERFORM THE SQUARE
                                        JSR     PC,FLTMPY       ;DO THE MULTIPLY
                                        STF     AC1,$TMP4       ;SAVE RESULT
                                        MOV     $AC1,$REG2      ;AND SOFTWARE EXP
                                ;
                                ;NOW DO THE RIGHT HAND SIDE OF THE EQUATION
                                ;DO THE A*A FIRST
                                        MOV     $REG0,$AC0      ;GET EXT EXPONENT
                                        LDF     $TMP0,AC0       ;LOAD OPERAND A
                                        MOV     $AC0,$AC1       ;SET OPERAND B EXT EXPONENT
                                        LDF     AC0,AC1         ;LOAD B OPERAND
                                        JSR     PC,FLTMPY       ;EXECUTE THE MULTIPLY
                                        STF     AC1,AC2         ;SAVE RESULT
                                        MOV     $AC1,$AC2
                                ;
                                ;NOW DO THE B*B
                                        LDF     $TMP2,AC0       ;LOAD B OPERAND
                                        LDF     AC0,AC1
                                        MOV     $REG1,$AC0      ;AND EXT EXPONENT
                                        MOV     $AC0,$AC1
                                        JSR     PC,FLTMPY       ;DO THE MULTIPLY
                                        STF     AC1,AC3         ;SAVE THE RESULT
                                        MOV     $AC1,$AC3
                                ;
                                ;NOW DO THE 2*B*A
                                        MOV     #$TMP2,R1
                                        LDF     (R1),AC0        ;LOAD THE B OPERAND
                                        LDF     -(R1),AC1       ;LOAD THE A OPERAND
                                        MOV     $REG1,$AC0      ;AND THE EXT EXPONENTS
                                        MOV     $REG0,$AC1
                                        JSR     PC,FLTMPY       ;DO THE MULTIPLY
                                        LDF     #^D40000,AC0    ;SETUP TO MULTIPLY BY TWO
                                        MOV     #2,$AC0
                                        JSR     PC,FLTMPY       ;DO THE MULTIPLY
                                ;
                                ;NOW SUM THE RESULTS
                                        MOV     $AC3,$AC0
                                        LDF     AC3,AC0         ;GET RESULT OF B*B
                                        JSR     PC,FLTADD       ;ADD THE RESULT
                                        LDF     AC2,AC0         ;GET RESULT OF A*A
                                        MOV     $AC2,$AC0
                                        JSR     PC,FLTADD       ;ADD THIS RESULT
                                        STF     AC1,$TMP6       ;SAVE FINAL RESULT
                                        MOV     $AC1,$REG3
                                ;
                                ;NOW CHECK BOTH SIDES OF THE EQUATION
                                ;CALCULATE THE NUMBER OF CORRECT BITS
                                ;PUT LARGEST EXPONENT OF A**2 OR B**2 IN $AC2
                                        CMP     $AC2,$AC3
                                        BGE     1$              ;BRANCH IF $AC2 ALREADY HAS LARGEST
                                        MOV     $AC3,$AC2       ;$AC3 WAS LARGER
                                1$:     SUB     $AC1,$AC2       ;NOW CALCULATE NUMBER
```



```
                                        SUB     #20.,$AC2       ;OF CORRECT BITS WITHIN 2
                                        NEG     $AC2            ;MAKE RESULT POSITIVE
                                        LDF     $TMP4,AC0       ;LOAD RESULT OF LEFT HAND SIDE
                                        MOV     $REG2,$AC0      ;AND EXTENDED EXPONENT
                                        JSR     PC,FLTSUB       ;SUBTRACT TO SEE HOW CLOSE THEY ARE
                                        SUB     $REG3,$AC1      ;GET DIFFERENCE IN EXT EXPONENTS
                                                                ;ACTUAL EXP'S ARE EQUAL TO 200
                                        BPL     2$              ;ENSURE RESULT IS POSITIVE
                                        NEG     $AC1
                                2$:     CMP     $AC2,$AC1       ;ANSWERS WITHIN ALLOWABLE NUMBER?
                                        BLE     SECT2           ;BRANCH IF YES
                                        MOV     #45,ERRTYP      ;FP RESULT WRONG
                                        HRDER ,<RESULTS WRONG>
                                        ;*****************************************************************
                                        .IF B <>
                                        HRDER$,BEGIN,NULL       ;RESULTS WRONG
                                        .IFF
                                        HRDER$,BEGIN,   ;RESULTS WRONG
                                        .ENDC
                                        ;*****************************************************************
                                        JSR     PC,$FL2O        ;CONVERT FIRST RESULT
                                        $TMP4
                                        $FLBUF
                                        OTOA    $REG2,$OBUFF            ;CONVERT EXTENDED EXPONENT
                                        ;*****************************************************************
                                                                ;CONVERT $REG2 TO ASCII AND
                                        OTOA$,BEGIN,$REG2,$OBUFF        ;STORE AT $OBUFF
                                        ;*****************************************************************
                                        JSR     PC,$FL2O        ;CONVERT OTHER RESULT
                                        $TMP6
                                        $$FLBUF
                                        OTOA    $REG3,$$OBUFF
                                        ;*****************************************************************
                                                                ;CONVERT $REG3 TO ASCII AND
                                        OTOA$,BEGIN,$REG3,$$OBUFF       ;STORE AT $$OBUFF
                                        ;*****************************************************************
                                        MSGN    SP1
                                        MSGN$,BEGIN,SP1 ;ASCII MESSAGE CALL WITH COMMON HEADER
                                        STARS
                                ;;*****************************************************************
                                SECT2:  LDFPS   #40             ;INIT FPS
                                ;DO A+B
                                        LDF     $TMP0,AC1       ;LOAD A OPERAND
                                        LDF     $TMP2,AC0       ;LOAD B OPERAND
                                        MOV     $REG0,$AC1
                                        MOV     $REG1,$AC0
                                        JSR     PC,FLTADD       ;ADD THEM
                                        STF     AC1,AC2         ;SAVE IN AC2
                                        MOV     $AC1,$AC2       ;AND EXT EXPONENT
                                ;NOW DO THE A-B
                                        LDF     $TMP0,AC1       ;LOAD OPERAND A
                                        MOV     $REG0,$AC1      ;AND EXT EXPONENT
```

```
000756' 172467  003730                  LDF     $TMP2,AC0       ;LOAD OPERAND B
000762' 016767  003742  004056          MOV     $REG1,$AC0
000770' 004767  002504                  JSR     PC,FLTSUB       ;SUBTRACT THEM
                                ;NOW DO (A+B)*(A-B)
000774' 172402                          LDF     AC2,AC0         ;GET RESULT OF (A+B)
000776' 016767  004050  004042          MOV     $AC2,$AC0
001004' 004767  002424                  JSR     PC,FLTMPY       ;FORM THE PRODUCT
001010' 174167  003702                  STF     AC1,$TMP4       ;SAVE RESULT
001014' 016767  004030  003710          MOV     $AC1,$REG2      ;AND EXT EXPONENT
                                ;NOW DO THE B*B
001022' 172467  003664                  LDF     $TMP2,AC0       ;LOAD OPERAND B
001026' 016767  003676  004012          MOV     $REG1,$AC0
001034' 172500                          LDF     AC0,AC1         ;B OPERAND IS IN AC0
001036' 016767  004004  004004          MOV     $AC0,$AC1       ;AND EXT EXPONENT
001044' 004767  002364                  JSR     PC,FLTMPY       ;
001050' 174102                          STF     AC1,AC2         ;SAVE RESULT IN AC2
001052' 016767  003772  003772          MOV     $AC1,$AC2
                                ;NOW DO THE A*A
001060' 172467  003622                  LDF     $TMP0,AC0       ;LOAD OPERAND A
001064' 016767  003636  003754          MOV     $REG0,$AC0
001072' 172500                          LDF     AC0,AC1         ;PUT OPR A IN AC1
001074' 016767  003746  003746          MOV     $AC0,$AC1       ;TO PERF THE SQUARE
001102' 004767  002326                  JSR     PC,FLTMPY       ;EXECUTE THE MULTIPLY
001106' 016767  003736  003740          MOV     $AC1,$AC3       ;SAVE EXT EXPO OF A*A
                                ;NOW DO A**2-B**2
001114' 172402                          LDF     AC2,AC0         ;GET B*B
001116' 016767  003730  003722          MOV     $AC2,$AC0       ;A*A IN AC1
001124' 004767  002350                  JSR     PC,FLTSUB
001130' 174167  003566                  STF     AC1,$TMP6       ;SAVE IN MEMORY
001134' 016767  003710  003572          MOV     $AC1,$REG3
                                ;NOW COMPUTE THE RESULTS
                                ;CALCULATE THE NUMBER OF CORRECT BITS
001142' 026767  003704  003704          CMP     $AC2,$AC3       ;DETERMINE WHICH EXP IS LARGER
001150' 002003                          BGE     1$              ;BRANCH IF AC2 LARGER
001152' 016767  003676  003672          MOV     $AC3,$AC2       ;PUT LARGEST IN AC2
001160' 166767  003664  003664  1$:     SUB     $AC1,$AC2       ;CALCUL NO OF CORRECT BITS
001166' 162767  000025  003656          SUB     #21.,$AC2       ;ALLOW 3 BIT ERROR
001174' 005467  003652                  NEG     $AC2
001200' 172467  003512                  LDF     $TMP4,AC0       ;GET LEFT HAND SIDE
001204' 016767  003522  003634          MOV     $REG2,$AC0
001212' 004767  002262                  JSR     PC,FLTSUB       ;SUBTRACT TO SEE HOW CLOSE THEY ARE
001216' 166767  003512  003624          SUB     $REG3,$AC1      ;SUB EXT EXPONENTS
                                                                ;ACTUAL EXPONENTS ARE EQUAL
001224' 100002                          BPL     2$              ;MAKE SURE RESULT IS POSITIVE
001226' 005467  003616                  NEG     $AC1
001232' 026767  003614  003610  2$:     CMP     $AC2,$AC1       ;RESULTS WITHIN RANGE ALLOWED?
001240' 003431                          BLE     $ECT3           ;BRANCH IF YES
001242' 012767  000045  176636          MOV     #45,ERRTYP      ;FP RESULT WRONG
001250'                                 HRDER   <RESULTS WRONG>
                                        ;*****************************************************************
                .IF B <>
001250' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;RESULTS WRONG
                .IFF
                HRDER$,BEGIN,   ;RESULTS WRONG
                .ENDC
                ;*****************************************************************
```



```
001256' 004767  002360                  JSR     PC,$FL2D        ;CONVERT FIRST RESULT
001262' 004716'                         $TMP4
001264' 004736'                         $FLBUF
001266'                                 OTOA    $REG2,$OBUFF            ;CONVERT EXTENDED EXPONENT
                                        ;*************************************************************
                                                                ;CONVERT $REG2 TO ASCII AND
001266' 104420  000000' 004732'         OTOA$,BEGIN,$REG2,$OBUFF;STORE AT $OPUFF
001274' 005062'
                                        ;*************************************************************
001276' 004767  002340                  JSR     PC,$FL2D        ;CONVERT OTHER RESULT
001302' 004722'                         $TMP6
001304' 005002'                         $$FLBUF
001306'                                 OTOA    $REG3,$$OBUFF
                                        ;*************************************************************
                                                                ;CONVERT $REG3 TO ASCII AND
001306' 104420  000000' 004734'         OTOA$,BEGIN,$REG3,$$OBUFF;STORE AT $$OBUFF
001314' 005102'
                                        ;*************************************************************
001316'
001316' 104403  000000' 005144'         MSGN    SP1
                                        MSGN$,BEGIN,SP1 ;ASCII MESSAGE CALL WITH COMMON HEADER
001324'
                                        STARS
                                ;;*************************************************************
001324' 172567  003356          $ECT3:  LDF     $TMP0,AC1       ;LOAD OPERAND A
001330' 172467  003356                  LDF     $TMP2,AC0       ;AND OPERAND B
001334' 016767  003366  003506          MOV     $REG0,$AC1
001342' 016767  003362  003476          MOV     $REG1,$AC0
001350' 004767  002102                  JSR     PC,FLTDIV       ;GO DIVIDE THEM
001354' 004767  002054                  JSR     PC,FLTMPY       ;MULTIPLY RESULT BY B
001360' 174167  003332                  STF     AC1,$TMP4       ;SAVE RESULT
001364' 016767  003460  003340          MOV     $AC1,$REG2
001372' 172467  003310                  LDF     $TMP0,AC0       ;LOAD OPERAND A
001376' 174067  003320                  STF     AC0,$TMP6       ;SAVE INCASE TYPE OUT
001402' 016767  003320  003436          MOV     $REG0,$AC0
001410' 016767  003312  003316          MOV     $REG0,$REG3
001416' 004767  002056                  JSR     PC,FLTSUB       ;SUBTRACT RIGHT AND LEFT HAND SIDES
001422' 166767  003300  003420          SUB     $REG0,$AC1      ;SEE IF RESULT OK
001430' 100002                          BPL     1$              ;ENSURE DIFFERANCE IS POSITIVE
001432' 005467  003412                  NEG     $AC1
001436' 022767  000026  003404  1$:     CMP     #22.,$AC1       ;RESULTS WITHIN 2 BITS?
001444' 003431                          BLE     $ECT4           ;BRANCH IF YES
001446' 012767  000045  176432          MOV     #45,ERRTYP      ;FP RESULT WRONG
001454'                                 HRDER   <RESULTS WRONG>
                                        ;*************************************************************
                .IF B <>
001454' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;RESULTS WRONG
                .IFF
                HRDER$,BEGIN,   ;RESULTS WRONG
                .ENDC
                ;*************************************************************
001462' 004767  002154                  JSR     PC,$FL2D        ;CONVERT FIRST RESULT
001466' 004716'                         $TMP4
001470' 004736'                         $FLBUF
001472'                                 OTOA    $REG2,$OBUFF            ;CONVERT EXTENDED EXPONENT
```

```
                                        ;*****************************************************************
                                        ;CONVERT $REG2 TO ASCII AND
001472' 104420  000000' 004732'         OTOA$,BEGIN,$REG2,$OBUFF ;STORE AT $OBUFF
001500' 005062'
                                        ;*****************************************************************
001502' 004767  002134                  JSR     PC,$FL2O        ;CONVERT OTHER RESULT
001506' 004722'                         $TMP6
001510' 005002'                         $$FLBUF
001512'                                 OTOA    $REG3,$$OBUFF
                                        ;*****************************************************************
                                        ;CONVERT $REG3 TO ASCII AND
001512' 104420  000000' 004734'         OTOA$,BEGIN,$REG3,$$OBUFF ;STORE AT $$OBUFF
001520' 005102'
                                        ;*****************************************************************
001522'                                 MSGN    SP1
001522' 104403  000000' 005144'         MSGN$,BEGIN,SP1 ;ASCII MESSAGE CALL WITH COMMON HEADER
001530'                                 STARS
                                ;;*****************************************************************
001530' 170011                  SECT4:  SETD
001532' 170403                          CLRD    AC3
001534' 170001                          SETF
001536' 177767  003154                  LDCDF   $TMP4,AC3       ;CONVERT TO SGL PREC
001542' 176367  003154                  STCFD   AC3,$TMP6       ;BRING IT BACK
001546' 170011                          SETD
001550' 173767  003146                  CMPD    $TMP6,AC3       ;ANSWERS OK?
001554' 170000                          CFCC
001556' 001423                          BEQ     SECT5           ;BRANCH IF YES
001560' 012767  000045  176320          MOV     #45,ERRTYP      ;FP RESULT WRONG
001566'                                 HRDER
                                        ;*****************************************************************
                        001             .IF B <>
001566' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;
                                        .IFF
                                        HRDER$,BEGIN,   ;
                        000             .ENDC
                                        ;*****************************************************************
001574' 174367  003106                  STD     AC3,$TMP0       ;GET CONTENTS OF AC3
001600' 004767  002312                  JSR     PC,$FLD2O       ;CONVERT TO ASCIZ
001604' 004706'                         $TMP0
001606' 004736'                         $FLBUF
001610' 004767  002302                  JSR     PC,$FLD2O       ;CONVERT $TMP6 TO ASCIZ
001614' 004722'                         $TMP6
001616' 005002'                         $$FLBUF
001620'                                 MSGN    SP3
001620' 104403  000000' 005214'         MSGN$,BEGIN,SP3 ;ASCII MESSAGE CALL WITH COMMON HEADER
001626'                                 STARS
                                ;;*****************************************************************
001626' 170001                  SECT5:  SETF
001630' 177267  003064                  LDCIF   $TMP4+2,AC2     ;LOAD NUMBER
001634' 175667  003064                  STCFI   AC2,$TMP6+2     ;BRING IT BACK
001640' 026767  003054  003056          CMP     $TMP4+2,$TMP6+2 ;NUMBERS STILL THE SAME?
001646' 001421                          BEQ     TST2            ;BRANCH IF YES
001650' 012767  000045  176230          MOV     #45,ERRTYP      ;FP RESULT WRONG
001656'                                 HRDER ,
```



```
                                        ;*****************************************************************
                        001             .IF B <>
001656' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;
                                        .IFF
                                        HRDER$,BEGIN,   ;
                        000             .ENDC
001664'                                 ;*****************************************************************
                                        OTOA    $TMP4+2,$OBUFF
                                        ;*****************************************************************
                                        ;CONVERT $TMP4+2 TO ASCII AND
001664' 104420  000000' 004720'         OTOA$,BEGIN,$TMP4+2,$OBUFF ;STORE AT $OBUFF
001672' 005062'
001674'                                 ;*****************************************************************
                                        OTOA    $TMP6+2,$$OBUF
                                        ;*****************************************************************
                                        ;CONVERT $TMP6+2 TO ASCII AND
001674' 104420  000000' 004724'         OTOA$,BEGIN,$TMP6+2,$$OBUF ;STORE AT $$OBUF
001702' 005102'
001704'                                 ;*****************************************************************
001704' 104403  000000' 005230'         MSGN    SP4
001712'                                 MSGN$,BEGIN,SP4 ;ASCII MESSAGE CALL WITH COMMON HEADER
                                        STARS
                                ;;*****************************************************************
                                ;.SBTTL  DOUBLE PRECISION
001712' 004767  002442          TST2:   JSR     PC,FLTDBL       ;GET RANDOM OPERANDS
001716' 170127  000340                  LDFPS   #340            ;INIT FPS
001722' 172567  002760                  LDD     $TMP0,AC1       ;LOAD A OPERAND
001726' 172467  002764                  LDD     $TMP4,AC0       ;LOAD B OPERAND
001732' 016767  002770  003110          MOV     $REG0,$AC1      ;SETUP EXTENDED
001740' 016767  002764  003100          MOV     $REG1,$AC0      ;EXPONENTS
001746' 004767  001532                  JSR     PC,FLTADD       ;PERFORM THE ADD
001752' 174100                          STD     AC1,AC0         ;SETUP AC0 TC
001754' 016767  003070  003064          MOV     $AC1,$AC0       ;PERFORM THE SQUARE
001762' 004767  001446                  JSR     PC,FLTMPY       ;DO THE MULTIPLY
001766' 174167  003132                  STD     AC1,FLTMP0      ;SAVE RESULT
001772' 016767  003052  002732          MOV     $AC1,$REG2      ;AND SOFTWARE EXP
                                ;
                                ;NOW DO THE RIGHT HAND SIDE OF THE EQUATION
                                ;DO THE A*A FIRST
002000' 016767  002722  003040          MOV     $REG0,$AC0      ;GET EXT EXPONENT
002006' 172467  002674                  LDD     $TMP0,AC0       ;LOAD OPERAND A
002012' 016767  003030  003030          MOV     $AC0,$AC1       ;SET OPERAND B EXT EXPONENT
002020' 172500                          LDD     AC0,AC1         ;LOAD B OPERAND
002022' 004767  001406                  JSR     PC,FLTMPY       ;EXECUTE THE MULTIPLY
002026' 174102                          STD     AC1,AC2         ;SAVE RESULT
002030' 016767  003014  003014          MOV     $AC1,$AC2
                                ;
                                ;NOW DO THE B*B
002036' 172467  002654                  LDD     $TMP4,AC0       ;LOAD B OPERAND
002042' 172500                          LDD     AC0,AC1
002044' 016767  002660  002774          MOV     $REG1,$AC0      ;AND EXT EXPONENT
002052' 016767  002770  002770          MOV     $AC0,$AC1
002060' 004767  001350                  JSR     PC,FLTMPY       ;DO THE MULTIPLY
002064' 174103                          STD     AC1,AC3         ;SAVE THE RESULT
```

```
                                        MOV     $AC1,$AC3
                                ;
                                ;NOW DO THE 2*B*A
                                        MOV     #$TMP4,R1
                                        LDD     (R1),AC0        ;LOAD THE B OPERAND
                                        LDD     -(R1),AC1       ;LOAD THE A OPERAND
                                        MOV     $REG1,$AC0      ;AND THE EXT EXPONENTS
                                        MOV     $REG0,$AC1
                                        JSR     PC,FLTMPY       ;DO THE MULTIPLY
                                        LDD     #^O40000,AC0    ;SETUP TO MULTIPLY BY TWO
                                        MOV     #2,$AC0
                                        JSR     PC,FLTMPY       ;DO THE MULTIPLY
                                ;
                                ;NOW SUM THE RESULTS
                                        MOV     $AC3,$AC0
                                        LDD     AC3,AC0         ;GET RESULT OF B*B
                                        JSR     PC,FLTADD       ;ADD THE RESULT
                                        LDD     AC2,AC0         ;GET RESULT OF A*A
                                        MOV     $AC2,$AC0
                                        JSR     PC,FLTADD       ;ADD THIS RESULT
                                        STD     AC1,FLTMP1      ;SAVE FINAL RESULT
                                        MOV     $AC1,$REG3
                                ;
                                ;NOW CHECK BOTH SIDES OF THE EQUATION
                                ;CALCULATE THE NUMBER OF CORRECT BITS
                                ;PUT LARGEST EXPONENT OF A**2 OR B**2 IN $AC2
                                        CMP     $AC2,$AC3
                                        BGE     1$              ;BRANCH IF $AC2 ALREADY HAS LARGEST
                                        MOV     $AC3,$AC2       ;$AC3 WAS LARGER
                                1$:     SUB     $AC1,$AC2       ;NOW CALCULATE NUMBER
                                        SUB     #52.,$AC2       ;OF CORRECT BITS WITHIN 2
                                        NEG     $AC2            ;MAKE RESULT POSITIVE
                                        LDD     FLTMP0,AC0      ;LOAD RESULT OF LEFT HAND SIDE
                                        MOV     $REG2,$AC0      ;AND EXTENDED EXPONENT
                                        JSR     PC,FLTSUB       ;SUBTRACT TO SEE HOW CLOSE THEY ARE
                                        SUB     $REG3,$AC1      ;GET DIFFERENCE IN EXT EXPONENTS
                                                                ;ACTUAL EXP'S ARE EQUAL TO 200
                                        BPL     2$              ;ENSURE RESULT IS POSITIVE
                                        NEG     $AC1
                                2$:     CMP     $AC2,$AC1       ;ANSWERS WITHIN ALLOWABLE NUMBER?
                                        BLE     $ECT2D          ;BRANCH IF YES
                                        MOV     #45.,ERRTYP     ;FP RESULT WRONG
                                        HRDER ,<RESULTS WRONG>
                                        ;***************************************************************
                                .IF B <>
                                        HRDER$,BEGIN,NULL       ;RESULTS WRONG
                                .IFF
                                        HRDER$,BEGIN,   ;RESULTS WRONG
                                .ENDC
                                        ;***************************************************************
                                        JSR     PC,$FLD2O       ;CONVERT RESULT 1 TO ASCII
                                FLTMP0
                                $FLBUF
                                        OTOA    $REG2,$OBUFF
                                        ;***************************************************************
                                                                ;CONVERT $REG2 TO ASCII AND
```



```
                                                                ;STORE AT $OBUFF
                                        OTOA$,BEGIN,$REG2,$OBUFF
                                        ;***************************************************************
                                        JSR     PC,$FLD2O       ;CONVERT RESULT 2 TO ASCII
                                        FLTMP1
                                        $$FLBUF
                                        OTOA    $REG3,$$OBUFF
                                        ;***************************************************************
                                                                ;CONVERT $REG3 TO ASCII AND
                                                                ;STORE AT $$OBUFF
                                        OTOA$,BEGIN,$REG3,$$OBUFF
                                        ;***************************************************************
                                        MSGN    SP2
                                        MSGN$,BEGIN,SP2 ;ASCII MESSAGE CALL WITH COMMON HEADER
                                        STARS
                                ;;*************************************************************
                                $ECT2D: LDFPS   #340
                                ;DO A+B
                                        LDD     $TMP0,AC1       ;LOAD A OPERAND
                                        LDD     $TMP4,AC0       ;LOAD B OPERAND
                                        MOV     $REG0,$AC1
                                        MOV     $REG1,$AC0
                                        JSR     PC,FLTADD       ;ADD THEM
                                        STD     AC1,AC2         ;SAVE IN AC2
                                        MOV     $AC1,$AC2       ;AND EXT EXPONENT
                                ;NOW DO THE A-B
                                        LDD     $TMP0,AC1       ;LOAD OPERAND A
                                        MOV     $REG0,$AC1      ;AND EXT EXPONENT
                                        LDD     $TMP4,AC0       ;LOAD OPERAND B
                                        MOV     $REG1,$AC0
                                        JSR     PC,FLTSUB       ;SUBTRACT THEM
                                ;NOW DO (A+B)*(A-B)
                                        LDD     AC2,AC0         ;GET RESULT OF (A+B)
                                        MOV     $AC2,$AC0
                                        JSR     PC,FLTMPY       ;FORM THE PRODUCT
                                        STD     AC1,FLTMP0      ;SAVE RESULT
                                        MOV     $AC1,$REG2      ;AND EXT EXPONENT
                                ;NOW DO THE B*B
                                        LDD     $TMP4,AC0       ;LOAD OPERAND B
                                        MOV     $REG1,$AC0
                                        LDD     AC0,AC1         ;B OPERAND IS IN AC0
                                        MOV     $AC0,$AC1       ;AND EXT EXPONENT
                                        JSR     PC,FLTMPY       ;
                                        STD     AC1,AC2         ;SAVE RESULT IN AC2
                                        MOV     $AC1,$AC2
                                ;NOW DO THE A*A
                                        LDD     $TMP0,AC0       ;LOAD OPERAND A
                                        MOV     $REG0,$AC0
                                        LDD     AC0,AC1
                                        MOV     $AC0,$AC1
                                        JSR     PC,FLTMPY       ;EXECUTE THE MULTIPLY
                                        MOV     $AC1,$AC3       ;SAVE EXT EXPO OF A*A
                                ;NOW DO A**2-B**2
                                        LDD     AC2,AC0         ;GET B*B
```

```
  807  002604' 016767  002242  002234          MOV     $AC2,$AC0       ;A*A IN AC1
  808  002612' 004767  000662                  JSR     PC,FLTSUB
  809  002616' 174167  002312                  STD     AC1,FLTMP1      ;SAVE IN MEMORY
  810  002622' 016767  002222  002104          MOV     $AC1,$REG3
  811                                  ;NOW COMPUTE THE RESULTS
  812                                  ;CALCULATE THE NUMBER OF CORRECT BITS
  813  002630' 026767  002216  002216          CMP     $AC2,$AC3       ;DETERMINE WHICH EXP IS LARGER
  814  002636' 002003                          BGE     1$              ;BRANCH IF AC2 LARGER
  815  002640' 016767  002210  002204          MOV     $AC3,$AC2       ;PUT LARGEST IN AC2
  816  002646' 166767  002176  002176  1$:     SUB     $AC1,$AC2
  817  002654' 162767  000065  002170          SUB     #53.,$AC2
  818  002662' 005467  002164                  NEG     $AC2
  819  002666' 172467  002232                  LDD     FLTMP0,AC0      ;GET LEFT HAND SIDE
  820  002672' 016767  002034  002146          MOV     $REG2,$AC0
  821  002700' 004767  000574                  JSR     PC,FLTSUB       ;SUBTRACT TO SEE HOW CLOSE THEY ARE
  822  002704' 166767  002024  002136          SUB     $REG3,$AC1      ;SUB EXT EXPONENTS
  823                                                                  ;ACTUAL EXPONENTS ARE EQUAL
  824  002712' 100002                          BPL     2$              ;MAKE SURE RESULT IS POSITIVE
  825  002714' 005467  002130                  NEG     $AC1
  826  002720' 026767  002126  002122  2$:     CMP     $AC2,$AC1       ;RESULTS WITHIN RANGE ALLOWED?
  827  002726' 003431                          BLE     $FCT3D          ;BRANCH IF YES
  828  002730' 012767  000045  175150          MOV     #45,ERRTYP      ;FP RESULT WRONG
  829  002736'                                 HRDER ,<RESULTS WRONG>
  830                                          ;*****************************************************************
  831                 001                     .IF B <>
  832  002736' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;RESULTS WRONG
  833                                          .IFF
  834                                          HRDER$,BEGIN,   ;RESULTS WRONG
  835                 000                      .ENDC
  836                                          ;*****************************************************************
  837  002744' 004767  001146                  JSR     PC,$FLD2D       ;CONVERT RESULT 1 TO ASCII
  838  002750' 005124'                          FLTMP0
  839  002752' 004736'                          $FLBUF
  840  002754'                                  OTOA    $REG2,$OBUFF
  841                                          ;*****************************************************************
  842                                                                  ;CONVERT $REG2 TO ASCII AND
  843                                                                  ;STORE AT $OBUFF
  844  002754' 104420  000000' 004732'         OTOA$,BEGIN,$REG2,$OBUFF
  845  002762' 005062'
  846                                          ;*****************************************************************
  847  002764' 004767  001126                  JSR     PC,$FLD2D       ;CONVERT RESULT 2 TO ASCII
  848  002770' 005134'                          FLTMP1
  849  002772' 005002'                          $$FLBUF
  850  002774'                                  OTOA    $REG3,$$OBUFF
  851                                          ;*****************************************************************
  852                                                                  ;CONVERT $REG3 TO ASCII AND
  853                                                                  ;STORE AT $$OBUFF
  854  002774' 104420  000000' 004734'         OTOA$,BEGIN,$REG3,$$OBUFF
  855  003002' 005102'
  856                                          ;*****************************************************************
  857  003004'                                 MSGN    SP2
  858  003004' 104403  000000' 005170'         MSGN$,BEGIN,SP2 ;ASCII MESSAGE CALL WITH COMMON HEADER
  859
  860  003012'                                 STARS
  861                                  ;;*****************************************************************
  862  003012' 172567  001670          $ECT3D: LDD     $TMP0,AC1       ;LOAD OPERAND A
```



```
  863  003016' 172467  001674                  LDD     $TMP4,AC0       ;AND OPERAND B
  864  003022' 016767  001700  002020          MOV     $REG0,$AC1
  865  003030' 016767  001674  002010          MOV     $REG1,$AC0
  866  003036' 004767  000414                  JSR     PC,FLTDIV       ;GO DIVIDE THEM
  867  003042' 004767  000366                  JSR     PC,FLTMPY       ;MULTIPLY RESULT BY B
  868  003046' 174167  002052                  STD     AC1,FLTMP0      ;SAVE RESULT
  869  003052' 016767  001772  001652          MOV     $AC1,$REG2
  870  003060' 172467  001622                  LDD     $TMP0,AC0       ;LOAD OPERAND A
  871  003064' 174067  002044                  STD     AC0,FLTMP1      ;SAVE INCASE TYPE OUT
  872  003070' 016767  001632  001750          MOV     $REG0,$AC0
  873  003076' 016767  001624  001630          MOV     $REG0,$REG3
  874  003104' 004767  000370                  JSR     PC,FLTSUB       ;SUBTRACT RIGHT AND LEFT HAND SIDES
  875  003110' 166767  001612  001732          SUB     $REG0,$AC1      ;SEE IF RESULT OK
  876  003116' 100002                          BPL     1$              ;ENSURE DIFFERANCE IS POSITIVE
  877  003120' 005467  001724                  NEG     $AC1
  878  003124' 022767  000066  001716  1$:     CMP     #54.,$AC1       ;RESULTS WITHIN 2 BITS?
  879  003132' 003431                          BLE     $ECT4D          ;BRANCH IF YES
  880  003134' 012767  000045  174744          MOV     #45,ERRTYP      ;FP RESULT WRONG
  881  003142'                                 HRDER ,<RESULTS WRONG>
  882                                          ;*****************************************************************
  883                 001                     .IF B <>
  884  003142' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;RESULTS WRONG
  885                                          .IFF
  886                                          HRDER$,BEGIN,   ;RESULTS WRONG
  887                 000                      .ENDC
  888                                          ;*****************************************************************
  889  003150' 004767  000742                  JSR     PC,$FLD2D       ;CONVERT RESULT 1 TO ASCII
  890  003154' 005124'                          FLTMP0
  891  003156' 004736'                          $FLBUF
  892  003160'                                  OTOA    $REG2,$OBUFF
  893                                          ;*****************************************************************
  894                                                                  ;CONVERT $REG2 TO ASCII AND
  895                                                                  ;STORE AT $OBUFF
  896  003160' 104420  000000' 004732'         OTOA$,BEGIN,$REG2,$OBUFF
  897  003166' 005062'
  898                                          ;*****************************************************************
  899  003170' 004767  000722                  JSR     PC,$FLD2D       ;CONVERT RESULT 2 TO ASCII
  900  003174' 005134'                          FLTMP1
  901  003176' 005002'                          $$FLBUF
  902  003200'                                  OTOA    $REG3,$$OBUFF
  903                                          ;*****************************************************************
  904                                                                  ;CONVERT $REG3 TO ASCII AND
  905                                                                  ;STORE AT $$OBUFF
  906  003200' 104420  000000' 004734'         OTOA$,BEGIN,$REG3,$$OBUFF
  907  003206' 005102'
  908                                          ;*****************************************************************
  909  003210'                                 MSGN    SP2
  910  003210' 104403  000000' 005170'         MSGN$,BEGIN,SP2 ;ASCII MESSAGE CALL WITH COMMON HEADER
  911  003216'                                 STARS
  912                                  ;;*****************************************************************
  913  003216' 177767  001474          $ECT4D: LDCFD   $TMP4,AC3       ;CHECK CONVERT
  914  003222' 176367  001474                  STCDF   AC3,$TMP6       ;BRING DATA BACK
  915  003226' 172767  001464                  LDD     $TMP4,AC3       ;RELOAD 1ST NUMBER
  916  003232' 170001                          SETF
  917  003234' 173767  001462                  CMPF    $TMP6,AC3       ;NUMBERS THE SAME?
  918  003240' 170000                          CFCC
```

```
 003242' 001421                 BEQ     SECT5D          ;BRANCH IF YES
 003244' 012767  000045  174634 MOV     #45,ERRTYP      ;FP RESULT WRONG
 003252'                        HRDER ,
                                ;*****************************************************************
                                .IF B <>
 003252' 104405  000000' 000000 HRDER$,BEGIN,NULL       ;
                                .IFF
                                HRDER$,BEGIN,   ;
                                .ENDC
                                ;*****************************************************************
 003260' 004767  000356         JSR     PC,$FL2O        ;CONVERT 1ST NO TO ASCIZ
 003264' 004716'                $TMP4
 003266' 004736'                $FLBUF
 003270' 004767  000346         JSR     PC,$FL2O        ;CONVERT 2ND NO TO ASCIZ
 003274' 004722'                $TMP6
 003276' 005002'                $$FLBUF
 003300'                        MSGN    SP5
 003300' 104403  000000' 005244' MSGN$,BEGIN,SP5 ;ASCII MESSAGE CALL WITH COMMON HEADER
 003306'                        STARS
                                ;;*****************************************************************
 003306' 170011                 SECT5D: SETD
 003310' 177267  001402         LDCLD   $TMP4,AC2       ;CHECK INTEGER CONV
 003314' 175667  001402         STCDL   AC2,$TMP6       ;BRING IT BACK
 003320' 026767  001372  001374 CMP     $TMP4,$TMP6     ;FIRST WORDS OK?
 003326' 001401                 BEQ     1$              ;BRANCH IF YES
 003330' 000404                 BR      2$
 003332' 026767  001362  001364 1$:     CMP     $TMP4+2,$TMP6+2 ;2ND WORDS OK?
 003340' 001431                 BEQ     END             ;BRANCH IF YES
 003342' 012767  000045  174536 2$:     MOV     #45,ERRTYP      ;FP RESULT WRONG
 003350'                        HRDER ,
                                ;*****************************************************************
                                .IF B <>
 003350' 104405  000000' 000000 HRDER$,BEGIN,NULL       ;
                                .IFF
                                HRDER$,BEGIN,   ;
                                .ENDC
                                ;*****************************************************************
 003356'                        OTOA    $TMP4,$OBUFF
                                ;*****************************************************************
                                                        ;CONVERT $TMP4 TO ASCII AND
                                                        ;STORE AT $OBUFF
 003356' 104420  000000' 004716' OTOA$,BEGIN,$TMP4,$OBUFF
 003364' 005062'
                                ;*****************************************************************
 003366'                        OTOA    $TMP4+2,$OBUF1
                                ;*****************************************************************
                                                        ;CONVERT $TMP4+2 TO ASCII AND
                                                        ;STORE AT $OBUF1
 003366' 104420  000000' 004720' OTOA$,BEGIN,$TMP4+2,$OBUF1
 003374' 005072'
                                ;*****************************************************************
 003376'                        OTOA    $TMP6,$$OBUF
                                ;*****************************************************************
                                                        ;CONVERT $TMP6 TO ASCII AND
                                                        ;STORE AT $$OBUF
 003376' 104420  000000' 004722' OTOA$,BEGIN,$TMP6,$$OBUF
```



```
 003404' 005102'
                                ;*****************************************************************
 003406'                        OTOA    $TMP6+2,$$OBU1
                                ;*****************************************************************
                                                        ;CONVERT $TMP6+2 TO ASCII AND
                                                        ;STORE AT $$OBU1
 003406' 104420  000000' 004724' OTOA$,BEGIN,$TMP6+2,$$OBU1
 003414' 005112'
                                ;*****************************************************************
 003416'                        MSGN    SP6
 003416' 104403  000000' 005260' MSGN$,BEGIN,SP6 ;ASCII MESSAGE CALL WITH COMMON HEADER
 003424'                        END:
 003424' 104413  000000'        ENDIT$,BEGIN            ;SIGNAL END OF ITERATION,
                                                        ;MONITOR SHALL TEST END OF PASS
 003430' 000167  174570         JMP     LOOP
 003434'                        STARS
                                ;;*****************************************************************
                                .$BTTL  FLOATING POINT MULTIPLY ROUTINE
                                ;*      THIS ROUTINE MULTIPLIES THE CONTENTS OF AC0 AND AC1
                                ;*      AND LEAVES THE RESULT IN AC1. IT ALSO TAKES CARE OF
                                ;*      THE SOFTWARE EXPONENTS THAT ARE KEPT IN $AC0 AND $AC1.
 003434'                        STARS
                                ;;*****************************************************************
 003434' 066767  001406  001406 FLTMPY: ADD     $AC0,$AC1       ;ADD SOFTWARE EXPONENTS
 003442' 171100                 MULF    AC0,AC1         ;DO THE MULTIPLY
 003444' 012746  100400         MOV     #100400,-(SP)   ;PUT CONTROL WORD ON STACK
 003450' 004767  001044         JSR     PC,EXPEXT       ;CALCULATE EXT EXPONENT
 003454' 000207                 1$:     RTS     PC              ;RETURN
 003456'                        STARS
                                ;;*****************************************************************
                                .$BTTL  FLOATING POINT DIVIDE ROUTINE
                                ;*      THIS ROUTINE DIVIDES THE CONTENTS OF AC1 BY AC0
                                ;*      AND LEAVES THE RESULT IN AC1.
 003456'                        STARS
                                ;;*****************************************************************
 003456' 166767  001364  001364 FLTDIV: SUB     $AC0,$AC1       ;ADJUST SOFTWARE EXPONENTS
 003464' 174500                 DIVF    AC0,AC1         ;EXECUTE THE DIVIDE
 003466' 012746  100400         MOV     #100400,-(SP)   ;PUT CONTROL WORD ON STACK
 003472' 004767  001022         JSR     PC,EXPEXT       ;CALCULATE EXT EXPONENT
 003476' 000207                 1$:     RTS     PC              ;RETURN
 003500'                        STARS
                                ;;*****************************************************************
                                .$BTTL  FLOATING POINT ADD ROUTINE
                                ;*      THIS ROUTINE ADDS THE CONTENTS OF AC0 TO AC1.
                                ;*      THIS CAN ONLY BE DONE IF THE SOFTWARE EXPONENTS
                                ;*      ARE CLOSE ENOUGH TOGETHER SUCH THAT AN ADJUSTMENT
                                ;*      OF THE REAL EXPONENT LEAVES A NON-ZERO NUMBER.
 003500'                        STARS
                                ;;*****************************************************************
 003500' 010667  000134         FLTSUB: MOV     SP,SUBFLG       ;SET SUBTRACT FLAG
 003504' 026767  001336  001336 FLTADD: CMP     $AC0,$AC1       ;CHECK SOFTWARE EXPONENTS
 003512' 003016                 BGT     2$
 003514' 001434                 BEQ     4$
```

```
                        ;ACCUMULATOR 1 IS LARGER THAN ACCUMULATOR 0
003516' 016702  001326          MOV     $AC1,R2 ;GET OPERAND B SOFTWARE EXP
003522' 166702  001340          SUB     $AC0,R2 ;GET DIFFERENCE IN SOFTWARE EXP'S
003526' 020227  000071          CMP     R2,#57.         ;EXP WITHIN DBL PREC RANGE?
003532' 002003                  BGE     1$              ;BRANCH IF ADD NOT REQUIRED
                                                        ;RESULT IS OPERAND B
003534' 005402                  NEG     R2
003536' 176402                  LDEXP   R2,AC0          ;RELOAD THE EXPONENT
003540' 000422                  BR      4$
003542' 176427  177703  1$:     LDEXP   #-75,AC0        ;FAKE EXPONENT SO HARDWARE
003546' 000417                  BR      4$              ;WILL DETECT OUT OF RANGE
                        ;
                        ;ACCUMULATOR 0 IS LARGER THAN ACCUMULATOR 1
003550' 016702  001272  2$:     MOV     $AC0,R2 ;GET SOFTWARE EXP OF OPERAND A
003554' 166702  001270          SUB     $AC1,R2 ;GET DIFFERENCE IN EXP'S
003560' 016767  001262  001262  MOV     $AC0,$AC1       ;MAKE SOFTWARE EXP'S EQUAL
003566' 020227  000071          CMP     R2,#57. ;EXP WITHIN DBL PREC RANGE?
003572' 002003                  BGE     3$              ;BRANCH IF NO
003574' 005402                  NEG     R2
003576' 176502                  LDEXP   R2,AC1          ;RELOAD THE EXPONENT
003600' 000402                  BR      4$
                        ;
                        ;ACCUMULATOR 0 IS MUCH LARGER THAN ACCUMULATOR 1 SO RESULT IS 0
003602' 176527  177703  3$:     LDEXP   #-75,AC1        ;FAKE EXPONENT SO HARDWARE
                                                        ;WILL DETECT OUT OF RANGE
003606' 005767  000026  4$:     TST     SUBFLG          ;ADD OR SUBTRACT?
003612' 001402                  BEQ     5$              ;BRANCH IF ADD
003614' 173100                  SUBF    AC0,AC1
003616' 000401                  BR      6$
003620' 172100          5$:     ADDF    AC0,AC1         ;EXECUTE THE ADD
003622' 013746  100400  6$:     MOV     #100400,-(SP)   ;PUT CONTROL WORD ON STACK
003626' 004767  000666          JSR     PC,EXPEXT       ;CALCULATE EXT EXPONENT
003632' 005067  000002          CLR     SUBFLG          ;INIT SUBTRACT FLAG
003636' 000207                  RTS     PC              ;RETURN
003640' 000000          SUBFLG: .WORD
003642'                 ;       STARS
                        ;;**************************************************************
                        .$BTTL  CONVERT FLOATING BINARY TO OCTAL ASCIZ
                        ;*
                        ;*THIS ROUTINE CONVERTS A 32 BIT FLOATING NUMBER TO AN OCTAL
                        ;*ASCIZ STRING IN THE FOLLOWING FORMAT:
                        ;*
                        ;*              W  XXX  YYY ZZZZZZ
                        ;*      WHERE   W = SIGN BIT
                        ;*              X = 8-BIT EXPONENT (RIGHT JUSTIFIED)
                        ;*              Y = FRACTION BITS <57:51> (RIGHT JUSTIFIED)
                        ;*              Z = FRACTION BITS <50:35>
                        ;*
                        ;*THE ROUTINE IS CALLED AS FOLLOWS:
                        ;*
                        ;*      JSR     PC,$FL2O
                        ;*      NUMBER                  ;ADDRESS OF NUMBER TO CONVERT
003642'                 ;*      BUFFER                  ;ADDRESS OF BUFFER FOR ASCIZ
                        ;       STARS
                        ;;**************************************************************
```



```
003642' 017600  000000  $FL2O:   MOV     @(SP),R0        ;GET ADDRESS OF DATA
003646' 062716  000002          ADD     #2,(SP)         ;ADJUST RETURN PC
003652' 016001  000002          MOV     2(R0),R1        ;PUT SECOND DATA WORD IN R1
003656' 011000                  MOV     (R0),R0         ;PUT FIRST DATA WORD IN R0
003660' 017604  000000          MOV     @(SP),R4        ;GET ADDR OF BUFFER
003664' 062716  000002          ADD     #2,(SP)
003670' 062704  000023          ADD     #23,R4          ;ADJUST TO END OF BUFFER
003674' 112744  000000          MOVB    #0,-(R4)        ;PUT TERMINATOR IN BUFFER
003700' 012705  000005          MOV     #5,R5           ;SET SOB COUNT FOR FRACTION DIGITS
003704' 010103          1$:     MOV     R1,R3           ;GET LSB'S OF FRACTION
003706' 042703  177770          BIC     #^C7,R3         ;SAVE LS 3 BITS
003712' 062703  000060          ADD     #60,R3          ;MAKE THEM ASCII
003716' 110344                  MOVB    R3,-(R4)        ;STORE IN BUFFER
003720' 073027  177775          ASHC    #-3,R0          ;SHIFT NUMBER TO NEXT 3 BITS
003724' 077511                  SOB     R5,1$           ;CONTINUE FOR 7 DIGITS
003726' 010103                  MOV     R1,R3           ;GET NEXT DIGITS
003730' 042703  177776          BIC     #^C1,R3         ;ONLY WANT 1 BIT
003734' 062703  000060          ADD     #60,R3          ;MAKE THEM ASCII
003740' 110344                  MOVB    R3,-(R4)        ;STORE IN BUFFER
003742' 112744  000040          MOVB    #40,-(R4)       ;PUT SPACE IN BUFFER
003746' 073027  177777          ASHC    #-1,R0
003752' 012705  000002          MOV     #2,R5           ;SET SOB COUNT
003756' 010103          2$:     MOV     R1,R3           ;GET LOW WORD
003760' 042703  177770          BIC     #^C7,R3         ;MASK 3 BITS
003764' 062703  000060          ADD     #60,R3          ;MAKE THEM ASCII
003770' 110344                  MOVB    R3,-(R4)        ;PUT IN BUFFER
003772' 073027  177775          ASHC    #-3,R0          ;GET NEXT 3 BITS
003776' 077511                  SOB     R5,2$           ;CONVERT THEM
004000' 010103                  MOV     R1,R3           ;
004002' 042703  177776          BIC     #^C1,R3         ;ONLY WANT 1 BIT
004006' 062703  000060          ADD     #60,R3          ;MAKE IT ASCII
004012' 110344                  MOVB    R3,-(R4)        ;PUT IN BUFFER
004014' 112744  000040          MOVB    #40,-(R4)       ;PUT SPACE IN BUFFER
004020' 112744  000040          MOVB    #40,-(R4)
004024' 072127  177777          ASH     #-1,R1          ;GET FIRST 3 BITS OF EXPONENT
004030' 012705  000002          MOV     #2,R5           ;SET SOB COUNT FOR 2 DIGITS
004034' 010103          3$:     MOV     R1,R3           ;GET LSB'S OF EXPONENT
004036' 042703  177770          BIC     #^C7,R3         ;SAVE 3 BITS
004042' 062703  000060          ADD     #60,R3          ;MAKE THEM ASCII
004046' 110344                  MOVB    R3,-(R4)        ;STORE IN BUFFER
004050' 072127  177775          ASH     #-3,R1          ;GET NEXT 3 BITS
004054' 077511                  SOB     R5,3$           ;CONTINUE
004056' 010103                  MOV     R1,R3           ;GET LAST 2 BITS OF EXPONENT
004060' 042703  177774          BIC     #^C3,R3         ;MAKE SURE ONLY 2 BITS
004064' 062703  000060          ADD     #60,R3          ;MAKE THEM ASCII
004070' 110344                  MOVB    R3,-(R4)        ;STORE IN BUFFER
004072' 112744  000040          MOVB    #40,-(R4)       ;PUT SPACE IN BUFFER
004076' 112744  000040          MOVB    #40,-(R4)
004102' 042700  177776          BIC     #^C1,R0         ;GET SIGN BIT (IT WAS EXTENDED)
004106' 062700  000060          ADD     #60,R0          ;MAKE IT ASCII
004112' 110044                  MOVB    R0,-(R4)        ;PUT IT IN THE BUFFER
004114' 000207                  RTS     PC
004116'                 ;       STARS
                        ;;**************************************************************
                        .$BTTL  CONVERT FLOATING DOUBLE BINARY TO OCTAL ASCIZ
                        ;*
```

```
                        ;*THIS ROUTINE CONVERTS A 64 BIT FLOATING NUMBER TO AN OCTAL
                        ;*ASCIZ STRING IN THE FOLLOWING FORMAT:
                        ;*
                        ;*              U  VVV  WWW XXXXXX YYYYYY ZZZZZZ
                        ;*
                        ;*      WHERE   U = SIGN BIT
                        ;*              V = 8-BIT EXPONENT (RIGHT JUSTIFIED)
                        ;*              W = FRACTION BITS<57:51> (RIGHT JUSTIFIED)
                        ;*              X = FRACTION BITS <50:35>
                        ;*              Y = FRACTION BITS <34:19>
                        ;*              Z = FRACTION BITS <18:03>
                        ;*
                        ;*THE ROUTINE IS CALLED AS FOLLOWS:
                        ;*
                        ;*      JSR     PC,$FLD2O
                        ;*      NUMBER                  ;ADDRESS OF NUMBER TO CONVERT
                        ;*      BUFFER                  ;ADDRESS OF BUFFER FOR ASCIZ
                        ;*      STARS
004116'                 ;;*****************************************************************
004116' 017667 000000 000016 $FLD2O: MOV     @(SP),1$        ;GET ADDRESS OF DATA TO CONVERT
004122' 062716 000002           ADD     #2,(SP)         ;ADJUST RETURN PC
004126' 017667 000000 000006    MOV     @(SP),2$        ;GET ADDR OF BUFFER
004136' 004767 177500           JSR     PC,$FL2O        ;CONVERT MS 32 BITS
004142' 000000          1$:     .WORD
004144' 000000          2$:     .WORD
004146' 016700 177772           MOV     2$,R0           ;PUT ADR OF BUFFER IN R0
004152' 062716 000002           ADD     #2,(SP)
004156' 062700 000041           ADD     #41,R0          ;ADJUST TO END OF BUFFER
004162' 105040                  CLRB    -(R0)           ;PUT TERMINATOR IN BUFFER
004164' 016701 177752           MOV     1$,R1           ;GET ADDRESS OF DATA TO CONVERT
004170' 062701 000004           ADD     #4,R1           ;ADJUST TO LOWER 32 BITS
004174' 012102                  MOV     (R1)+,R2        ;SAVE THE DATA
004176' 012103                  MOV     (R1)+,R3
004200' 012701 000002           MOV     #2,R1           ;SET LOOP COUNT
004204' 012704 000005   3$:     MOV     #5,R4           ;SET LOOP COUNT
004210' 010305          4$:     MOV     R3,R5           ;GET LS 32 BITS OF DATA
004212' 042705 177770           BIC     #^C7,R5         ;MASK 3 BITS
004216' 062705 000060           ADD     #60,R5          ;MAKE THEM ASCII
004222' 110540                  MOVB    R5,-(R0)        ;PUT IN BUFFER
004224' 073227 177775           ASHC    #-3,R2          ;GET NEXT 3 BITS
004230' 077411                  SOB     R4,4$           ;CONTINUE
004232' 010305                  MOV     R3,R5           ;GET LS 32 BITS
004234' 042705 177776           BIC     #^C1,R5         ;ONLY WANT 1 BIT
004240' 062705 000060           ADD     #60,R5          ;MAKE IT ASCII
004244' 110540                  MOVB    R5,-(R0)        ;PUT IN TABLE
004246' 112740 000040           MOVB    #40,-(R0)       ;PUT SPACE IN TABLE
004252' 073227 177777           ASHC    #-1,R2
004256' 077126                  SOB     R1,3$           ;CONVERT NEXT 16 BITS
004260' 000207                  RTS     PC
004262'                 STARS
                        ;;*****************************************************************

                        .SBTTL  RANDOM NUMBER GENERATOR ROUTINE

                        ;*THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
                        ;*WITH A RANGE OF 0 TO 2(+33)-1.
```



```
                        ;*CALL:
                        ;*      JSR     PC,$RAND        ;;CALL THE ROUTINE
                        ;*      RETURN                  ;;RETURN HERE THE RANDOM
                        ;*                              ;;NUMBER WILL BE IN
                        ;*                              ;;$HINUM,$LONUM
004262'                 STARS
                        ;;*****************************************************************
004262' 010046          $RAND:  MOV     R0,-(SP)
004264' 010146                  MOV     R1,-(SP)
004266' 010246                  MOV     R2,-(SP)
004270' 016700 000562           MOV     $LONUM,R0       ;;SET R0 WITH LOW
004274' 016701 000560           MOV     $HINUM,R1       ;;SET R1 WITH HIGH
004300' 012702 177771           MOV     #-7,R2          ;;SET SHIFT COUNT
004304' 006300          1$:     ASL     R0              ;;SHIFT R0 LEFT AND
004306' 006101                  ROL     R1              ;;ROTATE CARRY INTO R1 AND
004310' 005202                  INC     R2              ;;CHECK FOR DONE
004312' 001374                  BNE     1$              ;;CONTINUE SHIFT LOOP
004314' 066700 000536           ADD     $LONUM,R0       ;;ADD NUMBER TO MAKE X 129
004320' 005501                  ADC     R1              ;;PROPOGATE CARRY
004322' 066701 000532           ADD     $HINUM,R1       ;;ADD NUMBER TO MAKE X 129
004326' 062700 001057           ADD     #1057,R0        ;;ADD LOW CONSTANT
004332' 005501                  ADC     R1              ;;PROPOGATE CARRY
004334' 062701 047401           ADD     #47401,R1       ;;ADD HIGH CONSTANT
004340' 010067 000512           MOV     R0,$LONUM       ;;SAVE R0
004344' 010167 000510           MOV     R1,$HINUM       ;;SAVE R1
004350' 012602                  MOV     (SP)+,R2
004352' 012601                  MOV     (SP)+,R1
004354' 012600                  MOV     (SP)+,R0
004356' 000207                  RTS     PC              ;;RETURN
004360'                 STARS
                        ;;*****************************************************************
                        .SBTTL  FLOATING POINT NUMBER GENERATOR
                        ;*      THIS ROUTINE GENERATES TWO RANDOM FLOATING POINT NUMBERS
                        ;*      IN EITHER SINGLE OR DOUBLE PRECISION. FOR SINGLE PRECISION
                        ;*      THE NUMBERS ARE STORED IN $TMP0 AND $TMP2. DOUBLE PRECISION
                        ;*      NUMBERS ARE STORED IN $TMP0 AND $TMP4.
                        ;*      IN EITHER SINGLE OR DOUBLE THE EXTENDED EXPONENT IS STORED
                        ;*      IN $REG0 AND $REG1.
004360'                 STARS
                        ;;*****************************************************************
004360' 012767 000002 000130 FLTDBL: MOV     #2,SOBDBL      ;SET LOOP FOR 2, FOUR WORD NUMBERS
004366' 016700 000124 FLTSGL: MOV     SOBDBL,R0       ;SET WORD LENGTH LOOP
004372' 012702 004706'          MOV     #$TMP0,R2       ;GET ADDRESS TO STORE WORDS IN
004376' 012701 000002           MOV     #2,R1           ;SET NUMBER OF WORDS TO 2
004402' 004767 177654   2$:     JSR     PC,$RAND        ;GET RANDOM NUMBER
004406' 022701 000002           CMP     #2,R1           ;FIRST TIME?
004412' 001404                  BEQ     3$              ;BRANCH IF YES
004414' 022767 000002 000074    CMP     #2,SOBDBL       ;DOUBLE PRECISION?
004422' 001407                  BEQ     4$              ;BRANCH IF YES
004424' 016703 000430   3$:     MOV     $HINUM,R3       ;GET EXPONENT PART
004430' 042703 000177           BIC     #177,R3         ;CHECK FOR MINUS ZERO
004434' 022703 100000           CMP     #81115,R3
004440' 001760                  BEQ     2$              ;BRANCH IF MINUS ZERO
004442' 016722 000412   4$:     MOV     $HINUM,(R2)+    ;SAVE HINUM
004446' 016722 000404           MOV     $LONUM,(R2)+    ;SAVE LONUM
```

```
004452' 077125                  SOB     R1,2$           ;CONTINUE
004454' 077030                  SOB     R0,1$           ;CONTINUE FOR DOUBLE PREC
004456' 012746  004706'         MOV     #$TMP0,-(SP)    ;PUT ADDRESS OF NUMBER ON STACK
004462' 012746  001002          MOV     #1002,-(SP)     ;PUT CONTROL WORD ON STACK
004466' 022767  000002  000022  CMP     #2,SOBDBL       ;DOUBLE PREC?
004474' 001002                  BNE     5$              ;BRANCH IF NO
004476' 012716  001004          MOV     #1004,(SP)      ;CHANGE CONTROL WORD
004502' 004767  000012  5$:     JSR     PC,EXPEXT       ;CALCULATE EXT EXPONENTS
004506' 012767  000001  000002  MOV     #1,SOBDBL       ;INIT SOBDBL FOR SINGLE PREC
004514' 000207                  RTS     PC              ;RETURN
004516' 000001  SOBDBL: .WORD   1
004520'                 STARS
                        ;;*****************************************************************
                        .SBTTL  FLOATING POINT EXPONENT EXTENSION
                        ;*      THIS ROUTINE CONVERTS THE ACTUAL EXPONENT OF A FLOATING POINT
                        ;*      NUMBER INTO AN ACTUAL EXPONENT OF 200 AND AN EXTENDED
                        ;*      EXPONENT EQUAL TO THE DIFFERANCE BETWEEN THE ORIGINAL
                        ;*      ACTUAL EXPONENT AND 200.
                        ;*
                        ;*      THE ROUTINE IS ENTERED WITH A CONTROL WORD ON THE STACK.
                        ;*      BIT 15 OF THE CONTROL WORD INDICATES WHETHER THE NUMBER
                        ;*      IS IN MEMORY (<15>=0) OR IN AN ACCUMULATOR (<15>=1).
                        ;*      IF THE NUMBER IS IN AN ACCUMULATOR, BITS <9:8> INDICATE
                        ;*      THE ACCUMULATOR NUMBER. IF THE NUMBER(S) IS IN MEMORY,
                        ;*      BITS <9:8> INDICATE THE NUMBER OF NUMBERS TO CONVERT AND
                        ;*      BITS <2:0> INDICATE THE WORD LENGTH OF THE NUMBER(S).
                        ;*      IN THE CASE OF A MEMORY CONVERSION, THE ADDRESS OF THE
                        ;*      FIRST WORD TO CONVERT IS ALSO ON THE STACK (PRECEDING
                        ;*      THE CONTROL WORD).
004520'                 STARS
                        ;;*****************************************************************
004520' 012605          EXPEXT: MOV     (SP)+,R5        ;SAVE RETURN PC
004522' 012600                  MOV     (SP)+,R0        ;GET CONTROL WORD
004524' 100437                  BMI     2$              ;BRANCH IF ACC CONVERSION
004526' 012601                  MOV     (SP)+,R1        ;GET START ADDRESS
004530' 162700  000400          SUB     #400,R0
004534' 013702  004706'         MOV     #$TMP0,R2       ;GET OFFSET FROM $TMP0
004540' 160102                  SUB     R1,R2
004542' 005402                  NEG     R2
004544' 006202                  ASR     R2
004546' 062702  004726'         ADD     #$REG0,R2       ;GEN ADDRESS OF EXT WORD
004552' 011103          1$:     MOV     (R1),R3         ;GET DATA
004554' 042703  100177          BIC     #100177,R3      ;GET EXPONENT
004560' 072327  177771          ASH     #-7,R3          ;RIGHT JUSTIFY EXPONENT
004564' 162703  000200          SUB     #200,R3         ;CONVERT TO 2'S COMPLIMENT
004570' 010312                  MOV     R3,(R2)         ;ADD TO EXTENDED EXPONENT
004572' 042711  077600          BIC     #77600,(R1)     ;MAKE ACTUAL
004576' 052711  040000          BIS     #BIT14,(R1)     ;EXPONENT 200
004602' 162700  000400          SUB     #400,R0         ;ANY MORE WORDS?
004606' 100435                  BMI     5$              ;BRANCH IF NO
004610' 110003                  MOVB    R0,R3           ;GET WORD LENGTH
004612' 006303                  ASL     R3
004614' 060301                  ADD     R3,R1           ;SELECT NEXT NUMBER ADDRESS
004616' 062702  000002          ADD     #2,R2           ;SELECT NEXT EXTENDED ADDRESS
004622' 000753                  BR      1$              ;CONTINUE
                        ;ACCUMULATOR CONVERSION
```



```
004624' 072027  177776  2$:     ASH     #-2,R0          ;GET ACCUMULATOR NUMBER
004630' 042700  177477          BIC     #177477,R0      ;
004634' 010002                  MOV     R0,R2           ;GENERATE
004636' 072227  177773          ASH     #-5,R2          ;ADDRESS OF
004642' 062702  005046'         ADD     #$AC0,R2        ;EXTENDED EXPONENT
004646' 042767  000300  000004  BIC     #300,3$         ;GENERATE INSTRUCTION
004654' 050067  000000          BIS     R0,3$           ;TO GET EXPONENT
004660' 175003          3$:     STEXP   AC0,R3          ;GET EXPONENT
004662' 060312                  ADD     R3,(R2)         ;ADD TO EXTENDED EXPONENT
004664' 005003                  CLR     R3
004666' 042767  000300  000004  BIC     #300,4$         ;GENERATE INSTRUCTION
004674' 050067  000000          BIS     R0,4$           ;TO LOAD EXPONENT BACK TO ACC
004700' 176403          4$:     LDEXP   R3,AC0          ;LOAD EXPONENT OF 200
004702' 010546          5$:     MOV     R5,-(SP)        ;RESTORE RETURN PC
004704' 000207                  RTS     PC              ;RETURN

004706' 000002  $TMP0:  .BLKW   2
004712' 000002  $TMP2:  .BLKW   2
004716' 000002  $TMP4:  .BLKW   2
004722' 000002  $TMP6:  .BLKW   2
004726' 000000  $REG0:  .WORD
004730' 000000  $REG1:  .WORD
004732' 000000  $REG2:  .WORD
004734' 000000  $REG3:  .WORD
004736' 000044  $FLBUFF:.BLKB   44
005002' 000044  $$FLBUF:.BLKB   44
005046' 000000  $AC0:   .WORD
005050' 000000  $AC1:   .WORD
005052' 000000  $AC2:   .WORD
005054' 000000  $AC3:   .WORD
005056' 123456  $LONUM: .WORD   123456
005060' 065432  $HINUM: .WORD   65432
005062' 000000  $OBUFF: .WORD
005064' 000000          .WORD
005066' 000000          .WORD
005070' 000000          .WORD
005072' 000000  $OBUF1: .WORD
005074' 000000          .WORD
005076' 000000          .WORD
005100' 000000          .WORD
005102' 000000  $$OBUF: .WORD
005104' 000000          .WORD
005106' 000000          .WORD
005110' 000000          .WORD
005112' 000000  $$OBU1: .WORD
005114' 000000          .WORD
005116' 000000          .WORD
005120' 000000          .WORD
005122' 000000  $CNT:   .WORD
005124' 000004  FLTMP0:.BLKW    4
005134' 000004  FLTMP1:.BLKW    4

005144' 005304'  SP1:    MSG1
005146' 004736'          $FLBUFF
005150' 005361'          MSG2
```

```
1367  005152' 005062'                        $OBUFF
1368  005154' 005361'                        MSG2
1369  005156' 005002'                        $$FLBUFF
1370  005160' 005361'                        MSG2
1371  005162' 005102'                        $$OBUFF
1372  005164' 005364'                        MSG3
1373  005166' 177777                         177777
1374  005170' 005540'                SP2:    MSG7
1375  005172' 004736'                        $FLBUF
1376  005174' 005361'                        MSG2
1377  005176' 005062'                        $OBUFF
1378  005200' 005567'                        MSG7A
1379  005202' 005002'                        $$FLBUF
1380  005204' 005361'                        MSG2
1381  005206' 005102'                        $$OBUF
1382  005210' 005364'                        MSG3
1383  005212' 177777                         177777
1384  005214' 005366'                SP3:    MSG4
1385  005216' 004736'                        $FLBUF
1386  005220' 005361'                        MSG2
1387  005222' 005002'                        $$FLBUF
1388  005224' 005364'                        MSG3
1389  005226' 177777                         177777
1390  005230' 005463'                SP4:    MSG5
1391  005232' 005062'                        $OBUFF
1392  005234' 005361'                        MSG2
1393  005236' 005102'                        $$OBUF
1394  005240' 005364'                        MSG3
1395  005242' 177777                         177777
1396  005244' 005617'                SP5:    MSG8
1397  005246' 004736'                        $FLBUF
1398  005250' 005361'                        MSG2
1399  005252' 005002'                        $$FLBUF
1400  005254' 005364'                        MSG3
1401  005256' 177777                         177777
1402  005260' 005504'                SP6:    MSG6
1403  005262' 005062'                        $OBUFF
1404  005264' 005361'                        MSG2
1405  005266' 005072'                        $OBUF1
1406  005270' 005361'                        MSG2
1407  005272' 005102'                        $$OBUF
1408  005274' 005361'                        MSG2
1409  005276' 005112'                        $$OBU1
1410  005300' 005364'                        MSG3
1411  005302' 177777                         177777
1412
1413
1414  005304' 020045  020040  020040  MSG1:   .ASCIZ  '%              DATA1                   DATA2%'
1415  005312' 020040  020040  020040
1416  005320' 020040  042040  052101
1417  005326' 030501  020040  020040
1418  005334' 020040  020040  020040
1419  005342' 020040  020040  020040
1420  005350' 020040  040504  040524
1421  005356' 022462      000
1422  005361'     040  000040          MSG2:   .ASCIZ  ' '
```



```
1423  005364' 000045                  MSG3:   .ASCIZ  '%'
1424  005365' 020045  020040  020040  MSG4:   .ASCIZ  '%                      DATA1                           DATA2%'
1425  005374' 020040  020040  020040
1426  005402' 020040  020040  020040
1427  005410' 020040  020040  040504
1428  005416' 040524  020061  020040
1429  005424' 020040  020040  020040
1430  005432' 020040  020040  020040
1431  005440' 020040  020040  020040
1432  005446' 020040  020040  020040
1433  005454' 040504  040524  022462
1434  005462'     000
1435  005463'     045  042040  052101  MSG5:   .ASCIZ  '% DATA1   DATA2%'
1436  005470' 030501  020040  042040
1437  005476' 052101  031101  000045
1438  005504' 020045  020040  042040  MSG6:   .ASCIZ  '%    DATA1           DATA2%'
1439  005512' 052101  030501  020040
1440  005520' 020040  020040  020040
1441  005526' 020040  042040  052101
1442  005534' 031101  000045
```

```
  1443  005540'  020045  020040  020040  MSG7:   .ASCIZ  '%              DATA1%'
  1444  005546'  020040  020040  020040
  1445  005554'  020040  020040  040504
  1446  005562'  040524  022461     000
  1447  005567'     045  020040  020040  MSG7A:  .ASCIZ  '%              DATA2%'
  1448  005574'  020040  020040  020040
  1449  005602'  020040  020040  020040
  1450  005610'  040504  040524  022462
  1451  005616'     000
  1452  005617'     045  020040  020040  MSG8:   .ASCIZ  '%      DATA1                   DATA2%'
  1453  005624'  020040  040504  040524
  1454  005632'  020061  020040  020040
  1455  005640'  020040  020040  020040
  1456  005646'  020040  020040  020040
  1457  005654'  042040  052101  031101
  1458  005662'  000045
  1459           000001                  .END
```



```
ACSR     000102R          199#
ADDR     000006R          165#
ADDR22=  001000           287#
ASB      000106R          203#
ASTAT    000104R          201#
AWAS     000110R          204#
BEGIN    000000R          162#   430   442   452   456   522   534   544   548   574   586   596   600
                          616    629   641   650   657   661   740   752   762   766   832   844   854
                          858    884   896   906   910   924   936   951   960   967   974   981   985
                          987
BIT0  =  000001           286#
BIT1     000002           285#
BIT10 =  002000           276#
BIT11 =  004000           275#
BIT12 =  010000           274#
BIT13 =  020000           273#
BIT14 =  040000           272#  1302
BIT15 =  100000           271#  1251
BIT2     000004           284#
BIT3     000010           283#
BIT4     000020           282#
BIT5     000040           281#
BIT6     000100           280#
BIT7     000200           279#
BIT8     000400           278#
BIT9     001000           277#
BREAK$=  104407           327#
BR1      000012R          167#
BR2      000013R          168#
BTOD$ =  104421           342#
CDATA$=  104412           330#
CONFIG   000056R          187#
CSRA     000100R          197#
DATCK$=  104411           329#
DATER$=  104404           324#
DVID1    000014R          169#
END      003424R          946    986#
ENDIT$=  104413           331#   987
ENDS  =  104410           328#
ERRTYP   000106R          202#   426*  518*  570*  612*  637*  736*  828*  880*  920*  947*
EXIT$ =  104400           320#
EXPEXT   004520R         1002   1015  1062  1262  1286#
FLTADD   003504R          361    401   404   465   671   711   714   775  1028#
FLTDBL   004360R          665   1240#
FLTDIV   003456R          556    866  1012#
FLTMPY   003434R          364    374   383   393   396   477   485   493   557   674   684   693   703
                          706    787   795   803   867   999#
FLTMP0   005124R          675*   727   746   788*  819   838   868*  890  1361#
FLTMP1   005134R          715*   756   809*  848   871*  900  1362#
FLTSGL   004366R          355   1241#
FLTSUB   003500R          419    473   498   511   564   729   783   808   821   874  1027#
GETPA$=  104415           339#
GWBUF$=  104414           337#
HRDCNT   000044R          182#
HRDER$=  104405           325#   430   522   574   616   641   740   832   884   924   951
HRDPAS   000050R          184#
```

```
ICONT   000036R          179#
ICOUNT  000040R          180#
IDNUM   000122R          209#
INIT    000030R          176#
INTR    000120R          208#
LOOP    000224R          355#    989
MAP22$= 104416           339#
MODNAM  000000R          163#
MODSP   000224R          177     263#
MSCN$   = 104403         323#    456     548     600     629     661     766     858     910     936     985
MSG$    = 104402         322#
MSG$    = 104401         321#
MSG1    005304R          1364#   1414#
MSG2    005361R          1366    1370    1376    1380    1386    1392    1398    1404    1406    1408    1422#
MSG3    005364R          1372    1382    1388    1394    1400    1410    1423#
MSG4    005366R          1384    1424#
MSG5    005463R          1390    1435#
MSG6    005504R          1402    1438#
MSG7    005540R          1404    1443#
MSG7A   005567R          1376    1447#
MSG8    005617R          1396    1452#
NULL    = 000000         313#    430     522     574     616     641     740     832     884     924     951
OPEN    = 000000         164     170     171     172     173     190     191     192     193     194     195     196     197
                         199     201     203     204     206     207     208     267#
OTOA$ = 104420           341#    442     452     534     544     586     596     650     657     752     762     844     854
                         896     906     960     967     974     981
PASCNT  000034R          178#
PIRQS = 000004           298#
POPSP = 005726           310#
POPSP2= 022626           311#
PRTY    000000           167     168     307#
PRTY0 = 000000           306#
PRTY1 = 000040           305#
PRTY2 = 000100           304#
PRTY3 = 000140           303#
PRTY4 = 000200           302#
PRTY5 = 000240           301#
PRTY6 = 000300           300#
PRTY7 = 000340           299#
PS    = 177776           268#
PSW   = 177776           269#
PUSH  = 005746           308#
PUSH2 = 024646           309#
RAND$ = 104417           340#
RANNUM  000054R          186#
RESTRT  000244R          205     354#
RES1    000056R          188#
RES2    000060R          189#
RSTRT   000112R          205#
SBADR   000102R          198#
SECT2   000700R          425     459#
SECT2D  002366R          735     769#
SECT3   001324R          517     552#
SECT3D  002012R          827     862#
SECT4   001530R          569     603#
SECT4D  003216R          879     913#
```



```
SECT5   001626R          611     632#
SECT5D  003306R          919     939#
SOBDBL  004516R          1240*   1241    1247    1259    1263*   1265#
SOFCNT  000042R          181#
SOFERS= 104406           326#
SOFPAS  000046R          183#
SPOINT  000032R          177#
SPSIZ = 000040           1#      226
SP1     005144R          456     548     600     1364#
SP2     005170R          766     858     910     1374#
SP3     005214R          629     1384#
SP4     005230R          661     1390#
SP5     005244R          936     1396#
SP6     005260R          985     1402#
SR1     000016R          170#
SR2     000020R          171#
SR3     000022R          172#
SR4     000024R          173#
START   000224R          176     353#
STAT    000026R          175#
SUBFLG  003640R          1027*   1056    1063*   1065#
SVR0    000062R          190#
SVR1    000064R          191#
SVR2    000066R          192#
SVR3    000070R          193#
SVR4    000072R          194#
SVR5    000074R          195#
SVR6    000076R          196#
SYSCNT  000052R          185#
TRPDFD= 000022           312#    320     321#    322#    323#    324#    325#    326#    327#    328#    329#    330#    331#
                         332#    337     338#    339#    340#    341#    342#    343#
TST2    001712R          636     665#
VECTOR  000010R          166#
WASADR  000104R          200#
WDFR    000116R          207#
WDTO    000114R          206#
XFLAG   000005R          164#
$AC0    005046R          360*    363*    370*    372     381*    382     391*    395*    399*    403*    418*    464*    472*
                         476*    482*    484     490*    492     497*    510*    555*    562*    670*    673*    680*    682
                         691*    692     701*    705*    709*    713*    728*    774*    782*    786*    792*    794     800*
                         802     807*    820*    865*    872*    999     1012    1028    1033    1044    1046    1315    1338#
$AC1    005050R          359*    363     366     372*    376     382*    385     392*    406     414     420*    423*    424
                         463*    467     470*    478     484*    487     492*    494     500     506     512*    515*    516
                         554*    559     565*    567*    568     669*    673     676     682*    686     692*    695     702*
                         716     724     730*    733*    734     773*    777     780*    789     794*    797     804*    804
                         810     816     822*    825*    826     864*    869     875*    877*    878     999*    1017*   1026
                         1032    1045    1046*   1339#
$AC2    005052R          376*    403     411     413#    414*    415*    416*    424     467*    476     487*    497     503
                         505*    506*    507*    508*    516     686*    713     721     723*    724*    725*    726*    734
                         785*    786     797*    807     813     815*    816*    817*    818*    826     1340#
$AC3    005054R          385*    399     411     413     494*    503     505     695*    709     721     723     804*    813
                         815     1341#
$CNT    005122R          1360#
$FLBUF  004736R          437     529     581     624     747     839     891     931     1336#   1365    1375    1385    1397
$FLD2D  004116R          622     625     745     755     837     847     889     899     1162#
$FL2D   003642R          435     445     527     537     579     589     929     932     1087#   1165
```

```
$HINUM  005060R    1211    1219    1224*   1249    1253    1343#
$LONUM  005056R    1210    1217    1223*   1254    1342#
$OBUFF  005062R     442     534     586     650     752     844     896     960    1344#   1367    1377    1391    1403
$OBUF1  005072R     967    1348#   1405
$RAND   004262R    1207#   1244
$REG0   004726R     359     370     392     463     470     490     554     562     563     565     669     680     702
                    773     780     800     864     872     873     875    1295    1332#
$REG1   004730R     360     381     391     464     472     482     555     670     691     701     774     782     792
                    965    1333#
$REG2   004732R     366*    418     442     479*    510     534     559*    586     676*    728     752     789*    820
                    844     869*    896    1334#
$REG3   004734R     406*    420     452     500*    512     544     563*    596     716*    730     762     810*    822
                    854     873*    906    1335#
$TMP0   004706R     357     371     461     469     489     552     560     621*    623     667     681     771     779
                    799     862     870    1242    1257    1291    1328#
$TMP2   004712R     358     379     388     462     471     481     553    1329#
$TMP4   004716R     365*    417     436     478*    509     528     558*    580     606     633     635     650     668
                    689     698     772     781     791     863     913     915     930     940     942     945     960
                    967    1330#
$TMP6   004722R     405*    446     499*    538     561*    590     607*    609     626    634*     635     657     914*
                    917     933     941*    942     945     974     981    1331#
$$FLBU  005002R     447     539     591     627     757     849     901     934    1337#   1369    1379    1387    1399
$$OBUF  005102R     452     544     596     657     762     854     906     974    1352#   1371    1381    1393    1407
$$OBU1  005112R     981    1356#   1409
.       = 005664R  1328#   1329#   1330#   1331#   1336#   1337#   1361#   1362#

. ABS.  000000  000
        005664  001

ERRORS DETECTED:  0
DEFAULT GLOBALS GENERATED:  0

XFPBC0,XFPBC0/SQL/CRF:SYM=DDXCOM,XFPBC0
RUN-TIME:  2.3  .4 SECONDS
RUN-TIME RATIO: 33/6=4.8
CORE USED:  7K  (13 PAGES)
```