Chapter 7
Floating Point Arithmetic

Introduction

Floating Point arithmetic is required when the range of numbers exceeds the range of valid integers, or when fractional numbers are needed, eg. 3.14159265. Many applications can get by using *scaled fixed point* arithmetic but this can be cumbersome and has less resolution. This Floating Point system uses an 80 bit floating point stack that is separate from the regular data stack. This is so that we can make it compatible with ANSI Forth standard and because 80 bit numbers won't fit on the normal data stack.

Floating Point Tutorial

First we must load the appropriate Floating Point package. To load the *SANE Floating Point Package*, enter:

```
INCLUDE HSYS:FloatingPoint
```

Now we *must* initialize this system before using it. Enter:

```
FPINIT
```

That will install the proper floating point number conversion routines. Now whenever we enter a number with a decimal point in it, it will be automatically converted to floating point. (Note: numbers with decimal points would normally be converted to double precision 64 bit integers.)

Simple Arithmetic and Output

Let's try entering some numbers and doing some simple arithmetic. Most of the arithmetic operators that Forth has for integers, "* + - / ABS MIN", etc. have their floating point counterparts. To add two floating point numbers together we simply call **F+** .

```
23.5 F.  ( print a fp number )
20.0 7.55 F+ F.  ( add two fp numbers )
17.98 12.345 F/ F.  ( divide two fp numbers )
10.0 PI F* F.
```

We can control the display of our numbers by using **F.R** . If we want to display PI with 4 places after the decimal point in a field 12 characters wide, we can enter:

```
4 PLACES
PI 12 F.R
```

Transcendental Functions

Scientific calculations often require transcendental functions like Sine and Cosine and Logarithm.

```
2.1  5.3 F** F.  ( raise 2.1 to the power of 5.3 )
137.2 FLOG F.    ( log base 10 )
13.72 FLOG F.
```

Let's calculate the hypotenuse of a triangle. Pythagoras' theorem tells us that the length of the hypotenuse of a right triangle is the square root of the sum of the squares of the two sides.

```
: HYPOT  ( a b -f- c , C = SQRT( A**2 + B**2 )
    FDUP F*  ( square B )
    FSWAP FDUP F* F+ ( square A and add )
    FSQRT  ( take square root )
;
```

```
3.0 4.0 HYPOT F.  ( yep, the old 3,4,5 right triangle)
9.8 17.5 HYPOT F.
```

Turnkeying Floating Point Code

Please remember that you must initialize the system before using it. Otherwise it will crash dramatically. Here is an example of a floating point program that will turnkey.

```
: SHOWSINES  ( -- , display sines )
   FPINIT  ( Important!! ) CR
   4 PLACES
   91 0
   DO  I DUP 4 .R  ( show angle )
      FLOAT DEG>RAD ( convert i to radians )
      FSIN   9 F.R CR
   LOOP
   FPTERM
;
```

Floating Point Glossary

This glossary has been organized by function to make it easier to find the right word.

In the stack diagrams, **r** stands for a "Real" number. Real is another name for Floating Point. **-f-** is the floating point version of **--**, that is, it shows the stack (before and after execution).

Floating Point Control

FPINIT (-f- , initialize floating point)

This MUST be called before using any floating point words or you will crash. It opens the appropriate libraries and initializes some variables. The floating point files have AUTO.INIT words that will automatically call FPINIT if they are loaded permanently in a dictionary.

FPTERM (-f- , close libraries and cleanup)

Arithmetic Operators

These operators are similar to the corresponding integer operators and, therefore, don't need much explanation.

F+ (r1 r2 -f- r1+r2)

F- (r1 r2 -f- r1-r2)

F* (r1 r2 -f- r1*r2)

F/ (r1 r2 -f- r1/r2)

FABS (r -f- |r| , take absolute value of R)

FMOD (r1 r2 -f- rem(f1/f2) , calc remainder)

FNEGATE (r -f- -r)

FSQRT (r -f- sqrt(r) , square root)

Transcendental Functions

```
F**          ( r1 r2 -f- r1**r2 , R1 to R2th power )
FLN          ( r -f- ln[r] , natural logarithm )
FLOG   ( r -f- log[r] , base 10 logarithm )
FSIN   ( r.rad -f- sin[r] , take sine of R )
FCOS   ( r.rad -f- cos[r] )
FTAN   ( r.rad -f- tan[r] )
FATAN  ( tan[r] -f- r.rad )
```

Logical Operators

These operators are just like their integer counterparts except they accept floating point numbers. The following words accept two numbers.

```
F=   F<   F>   ( r1 r2 -f- , -- flag )
```

Stack Operators

There is room on the floating point stack for 32 numbers.

```
FOSP   ( ... -f- , clear float stack )
FDUP   ( r -f- r r )
FOVER  ( r1 r2 -f- r1 r2 r1 )
FROT   ( r1 r2 r3 -f- r2 r3 r1 )
FSWAP  ( r1 r2 -f- r2 r1 )
```

Number Storage

These words provide a way of storing floating point numbers in memory and retrieving them.

```
F!           ( r -f- , addr -- , store in memory )
F@           ( addr -- , -f- r )
FCONSTANT ( value <name> -- , declare a big enough constant )
FVARIABLE   ( <name> -- , declare a big enough variable )
```

Number Conversion Operators

```
F>I          ( r -f- , -- n , same as INT )
```

Takes a number from the floating point stack and puts it onto the usual paramater stack.

```
FIX          ( r -f- , -- n , round and convert to integer )
```

Takes a number from the floating point stack, rounds it, and puts it onto the usual paramater stack.

```
FLOAT ( n -- , -f- r , convert integer to float )
```

Takes a number from the parameter stack and moves it onto the floating point stack.

```
I>F         ( n -- , -f- r , same as FLOAT )

INT           ( r -f- , -- n , truncate and convert to integer )
```

Same as F>I .Takes a number from the floating point stack, truncates it, and puts it onto the usual paramater stack.

Display Operators

The single precision display words support 7 significant figures. Thus 1.2345678 F. will display 1.234568 .

```
F>TEXT        ( r -f- addr count , converts fp to text )

F.            ( r -f- , display floating-point in decimal form )

F.R           ( r -f- , width -- , set width of field, print )

PLACES        ( n -- , sets default number of fractional digits )
```

Number Interpreters

```
FNUMBER?      ( $string -- true | false , -f- r )
```

Convert a string to a floating point number. If valid, return TRUE and a float. If invalid, just return a FALSE, no float. For example, used in reading in numbers from a formatted file, or in querying a user to enter input from the computer keyboard.

```
FLITERAL      ( r -f- , compile a floating point number, immediate)
```