LongFloat
Long Precision Floating Point Math Library for the HP48 with ALG48
Version 4.0

Mika Heiskanen Claude-Nicolas Fiechter

April 3, 1997

1 Acknowledgements, copyright & disclaimer of warranty

Special thanks to Joe Horn for his many useful comments, suggestions and detailed bug reports.

2 Overview

LongFloat provides commands for doing basic arithmetic and some transcendental functions to operate with floating point numbers with arbitrary precision.

Arithmetic operations include addition, substraction, multiplication, division, inversion, negation, square root, raising to a power, comparison and taking integer or fractional parts.

Transcendental functions include exponentiation, logarithm, trigonometric and hyperbolic functions and a special command to return π in the currently defined precision.

3 Installation

LongFloat takes approximately 6Kb of memory and works in both HP48G(X) and HP48S(X). LongFloat uses some of the internal subroutines in ALG48, and thus requires it to be installed. See the ALG48 documentation for information on installing ALG48.

Due to the special optimization features used in ALG48, not all storage combinations are allowed. The following ones are possible:

- In SX there are no restrictions.
- In GX if ALG48 is in port 0 or port 1 then LongFloat can be stored in any port.
- In GX if ALG48 is stored in port 2 (or higher) then LongFloat must be stored in the same port, or in port 0.

Installing ALG48 and LongFloat in the same port seems the most natural choice and is highly recommended.

LongFloat is an auto-attaching library (library number 912). To install it on your HP48 download the file `long.lib` onto your (in *binary* mode), put the content of the created variable on the stack, store it in the port of your choice (e.g., `'LONG.LIB' DUP RCL SWAP PURGE 0 STO`) and power-cycle the calculator.

4 Use

4.1 Available Functions

Command	Description	Stack
FADD	Addition	(\$1 \$2 → \$')
FSUB	Substraction	(\$1 \$2 → \$')
FMUL	Multipication	(\$1 \$2 → \$')
FDIV	Division	(\$1 \$2 → \$')
FINV	Inverse	(\$ → \$')
FNEG	Negate	(\$ → \$')
FSQRT	Inverse	(\$ → \$')
FPOW	Power	(\$1 \$2 → \$')
FPI	π	(→ \$')
FEXP	Exponentiation	(\$ → \$')
FLN	Logarithm	(\$ → \$')
FCMP	Comparison	(\$1 \$2 → %)
FSIN	Sine	(\$ → \$')
FCOS	Cosine	(\$ → \$')
FTAN	Tangent	(\$ → \$')
FASIN	Inverse sine	(\$ → \$')
FACOS	Inverse cosine	(\$ → \$')
FATAN	Inverse tangent	(\$ → \$')
FSINH	Hyperbolic sine	(\$ → \$')
FCOSH	Hyperbolic cosine	(\$ → \$')
FTANH	Hyperbolic tangent	(\$ → \$')
FASINH	Inverse hyperbolic sine	(\$ → \$')
FACOSH	Inverse hyperbolic cosine	(\$ → \$')
FATANH	Inverse hyperbolic tangent	(\$ → \$')
FIP	Integer part	(\$ → \$')
FFP	Fractional part	(\$ → \$')

FCMP returns -1,0 or 1, which correspond to succesful $<, =$ or $>$ tests.

4.2 Setting Precision

By default LongFloat library uses a working precision of 20 digits in all calculations. The default precision can be altered by storing the desired precision in real number form to a variable `DIGITS` in the home directory. If the stored object is not a real number, or is one but is not an integer or is outside the range $2 \leq DIGITS \leq 10000$ an error is generated.

In programming one may wish to temporarily alter the working precision, thus can be easily achieved by creating a local variable `DIGITS` containing the desired precision satisfying the limitations given above. A simple example of how to do this is the following program, which returns $\sqrt{2}$ to an accuracy of 100 digits.

```
<<
   100 -> DIGITS
   <<
      "2" FSQRT
   >>
>>
```

The user should note that not all precisions are rational choices for some of the functions. The execution time (HP48GX) for the basic arithmetic operations approaches 1 minute as the precision approaches 1000 digits, square root calculation is already significantly slower and for the transcendental functions using 100 digit precision already takes more time. Also the algorithms used for transcendental functions use suitable scaling formulas to get the arguments into the range of convergence for the respective polynomial approximations. This implies the functions get slower as the absolute value of the exponent increases. In some cases convergence may also be slow if the argument approaches the limit of the region of convergence.

4.3 Representation of Long Precision Floating Point Numbers

To represent long precision floating point numbers LongFloat uses HP48 string type objects. Thus the numbers are easily readable and conversion between regular HP48 real number objects is trivial using the built-in `→STR` and `OBJ→` functions. For example the result for `"10000" FEXP` with the default 20 digit precision will be

`"8.8068182256629216366E4342"`,

a result which would already cause an error in HP48 due to the very large exponent.

5 Contact

Gifts :), bug reports, and constructive comments and suggestions can be sent to either one of the following addresses.

Mika Heiskanen
Jämeräntaival 7 C 355
02150 ESPOO
Finland
mheiskan@cc.hut.fi

Claude-Nicolas Fiechter
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260, U.S.A.
fiechter@cs.pitt.edu