ExtraFunc49+

Version 0.88, 23rd of May 2006
Steen S. Schmidt

Introduction 1
Installation 2
ExtraFunc49+ 2
ACOT(z) 3
ACOTH(z) 3
ACSC(z) 3
ACSCH(z) 3
ASEC(z) 3
ASECH(z) 3
BELLNUM(n) 4
BETA(p,q) 4
CATALAN(n) 4
COT(z) 4
COTH(z) 4
CSC(z) 4
CSCH(z) 5
DBLFACT(z) 5
DELTA(n) 5
FACTORIAL(z) 5
FIBNUM(n) 6
HEAVISIDE(n) 6
LUCASNUM(n) 6
LUCASSEQU(p,q,n) 6
LUCASSEQV(p,q,n) 6
NCR(n,k) 7
NPR(n,k) 7
PARTFACT(a,b) 7
PELLNUM(n) 7
SEC(z) 7
SECH(z) 8
SINC(z) 8
STIRLINGNUM1(n,m) 8
STIRLINGNUM2(n,m) 8
Revision History 8

Introduction

This library introduces a set of extra functions for the HP49G+. The functions include both common ones (but missing out of the box on the calculator), more specialized ones used in number theory for instance, as well as performance and/or feature enhanced versions of already built-in functions.
To keep this manual reasonable in size, I have refrained from giving elaborate mathematical explanations as to what these functions can be used for, or how they are defined (except where several definitions are commonly used, the one chosen for this specific implementation is stated). For an online mathematical guide I'd advice you to take a look at www.mathworld.com, which is a treasure trove of information.

The source code (Debug4x project) for ExtraFunc will eventually be released, but currently it's not available.

The library is freeware and as such is covered by no warranty whatsoever. I cannot take responsibility for any damage or data loss caused by this program.
If you have any suggestions or find any bugs in the code, you're welcome to contact me on email: sschmidt@nospam.dk or write on the newsgroup comp.sys.hp48 – chances are I might see your post there. Please include details about ROM revision and flag settings if you're reporting a bug.

Installation

Many functions in the library are programmed in C (with HP-GCC), so a common helper library (ELib) is necessary. This means for this version of ExtraFunc49+ to work, you need the following libraries installed on your HP49G+:

- ExtraFunc49+ v0.88
- ELib v0.19 or later
- ARMToolBox v3.12 or later

Consult the documentation for ELib for details on how to install it and what the ARMToolBox is and where to obtain that.

ExtraFunc49+ (library #1132) will work from any port, but port 2 is preferred as this will help keep as much RAM as possible free to work in. The library is 6079.5 bytes in size, and the checksum is #3004h.
Since the C-based functions herein are generally very fast, relatively big calculations are possible (how about finding the 650000th Fibonacci number for instance?). The HP49G+ has a limited amount of RAM at its disposal though – this is what sets the limits as to what numbers can be handled. The maximum amount of RAM available for a C-program is currently around 455 kB, but this entails a completely empty calculator (which is probably rather rare). Keep in mind that when you store something in HOME or in ports 0 and 1, you deduct from the working memory available to C-programs – so, store libraries in port 2 and data on an SD-card if possible.

ExtraFunc49+

Each of the functions below are shown with a stack diagram – the first line in the diagram denotes which arguments go on which level on the stack (in RPN mode), or in case of algebraic mode, which arguments go where in the algebraic expression.
The second line shows which types of arguments are accepted as input – Ñ means all real numbers and Â means all complex numbers. Ù denotes all integers, and can be restricted, for instance, to positive integers like this: Ù$^{+}$. In this document the latter is defined as including zero unless otherwise noted. Symbolic arguments are always accepted unless otherwise noted.

Most of the included functions support automatic list processing (denoted by a {} symbol above the stack diagrams in the function descriptions below). Note, however, that there can be ambiguities between seemingly identical arguments whether they are entered inside a list or entered on the stack. The built-in command π will be an algebraic object (type 9) when entered on the stack for instance, but will remain a command (type 18) when entered directly inside a list. None of the functions in this library support arguments of type 18.

ACOT(z)
{}

Level 1		Level 1
z	→	ACOT(z)
Ñ, Â	→	Ñ, Â

Arc cotangent. Equivalent to ATAN(1/z).

ACOTH(z)
{}

Level 1		Level 1
z	→	ACOTH(z)
Ñ, Â	→	Ñ, Â

Hyperbolic arc cotangent. Equivalent to ATANH(1/z).

ACSC(z)
{}

Level 1		Level 1
z	→	ACSC(z)
Ñ, Â	→	Ñ, Â

Arc cosecant. Equivalent to ASIN(1/z).

ACSCH(z)
{}

Level 1		Level 1
z	→	ACSCH(z)
Ñ, Â	→	Ñ, Â

Hyperbolic arc cosecant. Equivalent to ASINH(1/z).

ASEC(z)
{}

Level 1		Level 1
z	→	ASEC(z)
Ñ, Â	→	Ñ, Â

Arc secant. Equivalent to ACOS(1/z).

ASECH(z)
{}

Level 1		Level 1
z	→	ASECH(z)
Ñ, Â	→	Ñ, Â

Hyperbolic arc secant. Equivalent to ACOSH(1/z).

BELLNUM(n)

{}

Level 1		Level 1
n	→	nth Bell number
Ù$^{+}$	→	Ù$^{+}$

The nth Bell number.

BETA(p,q)

{}

Level 2	Level 1		Level 1
p	q	→	beta(p,q)
Ñ, Â	Ñ, Â	→	Ñ, Â

The beta function, also called the beta integral or the Eulerian integral of the first kind.

CATALAN(n)

{}

Level 1		Level 1
n	→	nth Catalan number
Ù$^{+}$	→	Ù$^{+}$

The nth Catalan number.

COT(z)

{}

Level 1		Level 1
z	→	COT(z)
Ñ, Â	→	Ñ, Â

Cotangent. Equivalent to 1/TAN(z).

COTH(z)

{}

Level 1		Level 1
z	→	COTH(z)
Ñ, Â	→	Ñ, Â

Hyperbolic cotangent. Equivalent to 1/TANH(z).

CSC(z)

{}

Level 1		Level 1
z	→	CSC(z)
Ñ, Â	→	Ñ, Â

Cosecant. Equivalent to 1/SIN(z).

CSCH(z)

{}

Level 1		Level 1
z	→	CSCH(z)
Ñ, Â	→	Ñ, Â

Hyperbolic cosecant. Equivalent to 1/SINH(z).

DBLFACT(z)

{}

Level 1		Level 1
z	→	z!!
Ñ, Â	→	Ñ, Â

The double factorial, which is a special case of the multifactorial. For complex arguments I use this definition:

$$z!! = \Gamma\left(1+\frac{z}{2}\right) 2^{\frac{1+2z-\cos(\pi z)}{4}} \pi^{\frac{\cos(\pi z)-1}{4}}$$

For complex arguments the double factorial can easily overflow, so it is advicable to have system flags -20 and -21 set when using this function.

DELTA(n)

{}

Level 1		Level 1
n	→	δ(n)
Ñ	→	Ñ

Dirac’s delta function, also known as the impulse symbol. The definition used in this function is δ(0)=1, otherwise 0.

FACTORIAL(z)

{}

Level 1		Level 1
z	→	z!
Ñ, Â	→	Ñ, Â

The factorial is already built into the HP49G+ as the FACT and ! functions, but this implementation is much faster (100 times or more – 2000! Takes well over 2 minutes with the built-in functions, and only around 1 second with this FACTORIAL function).
Also, this function support complex arguments which the built-in ones do not, and it supports much larger input than the built-in ones’ limit of 9999.

FIBNUM(n)

{}

Level 1		Level 1
n	→	n^{th} Fibonacci number
Ù$^{+}$	→	Ù$^{+}$

The n^{th} Fibonacci number. The 10000^{th} Fibonacci number is calculated in around 1 second.

HEAVISIDE(n)

{}

Level 1		Level 1
n	→	H(n)
Ñ	→	Ñ

Heaviside, or unit, step function. The definition used in this function is H(<0)=0, otherwise 1.

LUCASNUM(n)

{}

Level 1		Level 1
n	→	n^{th} Lucas number
Ù$^{+}$	→	Ù$^{+}$

The n^{th} Lucas number. The 10000^{th} Lucas number is calculated in around 1 second.

LUCASSEQU(p,q,n)

Level 3	Level 2	Level 1		Level 1
p	q	n	→	$U_n(p,q)$
Ù$^{+}$	Ù$^{-}$	Ù$^{+}$	→	Ù$^{+}$

The n^{th} term in the Lucas sequence $U_n(p,q)$. The following limitations apply to the arguments:

- p may not be negative.
- q may not be positive.
- $(p-q)\cdot 99999999$ may not exceed $2^{32}-1$.

LUCASSEQV(p,q,n)

Level 3	Level 2	Level 1		Level 1
p	q	n	→	$V_n(p,q)$
Ù$^{+}$	Ù$^{-}$	Ù$^{+}$	→	Ù$^{+}$

The n^{th} term in the Lucas sequence $V_n(p,q)$. The following limitations apply to the arguments:

- p may not be negative.
- q may not be positive.
- $(p-q)\cdot 99999999$ may not exceed $2^{32}-1$.

NCR(n,k)

{}

Level 2	Level 1		Level 1
n	k	→	$_nC_k$
Ù$^+$	Ù$^+$	→	Ù$^+$

The binomial coefficient, or the number of ways of picking k unordered outcomes from n possibilities. Already built into the HP49G+ as COMB, but this implementation is up to 10 times faster.

NPR(n,k)

{}

Level 2	Level 1		Level 1
n	k	→	$_nP_k$
Ù$^+$	Ù$^+$	→	Ù$^+$

Permutations, or the number of ways of picking k ordered outcomes from n possibilities. Already built into the HP49G+ as PERM, but this implementation is up to a 100 times faster.

PARTFACT(a,b)

{}

Level 2	Level 1		Level 1
a	b	→	PARTFACT(a,b)
Ù	Ù	→	Ù

A partial factorial, defined like this:

a H**a+1**L **...** H**b−1**L **b**

PARTFACT(7,11) is equivalent to 7·8·9·10·11, or 55440.

PELLNUM(n)

{}

Level 1		Level 1
n	→	nth Pell number
Ù$^+$	→	Ù$^+$

The nth Pell number. The 10000th Pell number is calculated in around 2 seconds.

SEC(z)

{}

Level 1		Level 1
z	→	SEC(z)
Ñ, Â	→	Ñ, Â

Secant. Equivalent to 1/COS(z).

SECH(z)

{}

Level 1		Level 1
z	→	SECH(z)
Ñ, Â	→	Ñ, Â

Hyperbolic secant. Equivalent to 1/COSH(z).

SINC(z)

{}

Level 1		Level 1
z	→	SECH(z)
Ñ, Â	→	Ñ, Â

Sine cardinal. The definition used in this implementation is SIN(z)/z as opposed to SIN(π·z)/(π·z).

STIRLINGNUM1(n,m)

{}

Level 2	Level 1		Level 1
n	m	→	*s*(n,m)
Ù$^{+}$	Ù$^{+}$	→	Ù

Signed Stirling number of the first kind. Currently unstable!

STIRLINGNUM2(n,m)

{}

Level 2	Level 1		Level 1
n	m	→	*S*(n,m)
Ù$^{+}$	Ù$^{+}$	→	Ù

Stirling number of the second kind. Currently unstable!

Revision History

v0.88: Initial public beta.