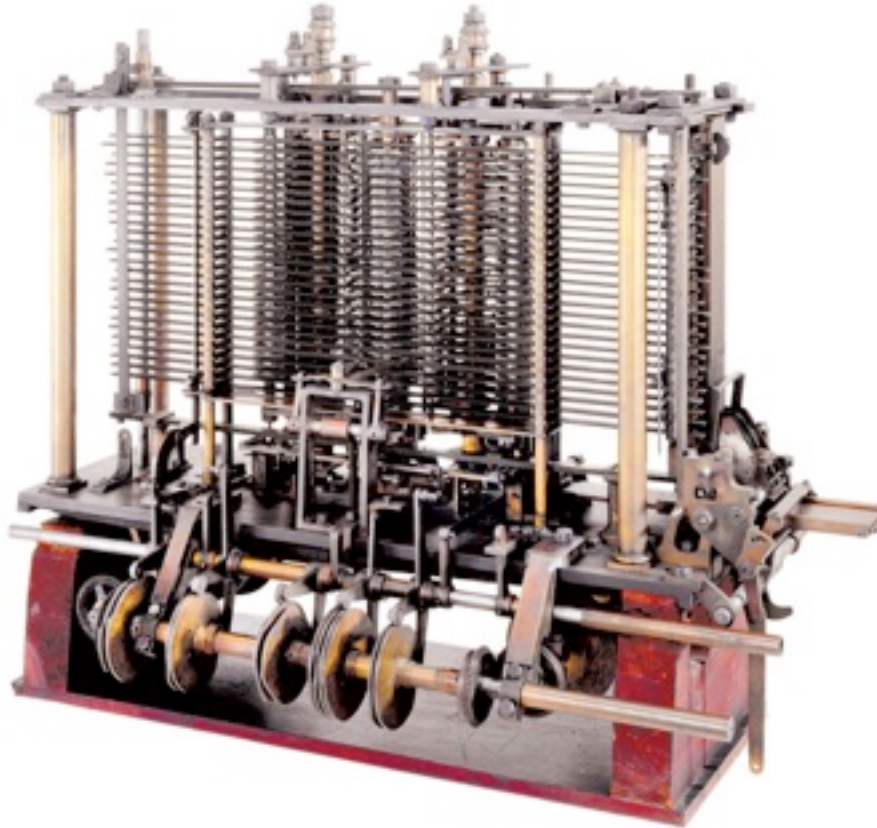




Apple® IIgs

HexCalc 64-bit Calculator Instruction Manual



A Simple Calculator

(Not suitable for the average pocket)

Babbage's Analytical Engine, 1834–1871, the first fully automatic calculating machine
Courtesy Science Museum & Science and Society Picture Library, London

Dedicated to the memories of Joe Kohn 1947-2010
Ryan Suenega 1967-2011

HexCalc is Freeware and Copyright © 2021 Ewen Wannop

HexCalc and its supporting documentation
may not be printed, copied, or distributed for profit.
Distributing and/or archiving is restricted while in an electronic form.
Any “free” distribution must be given permission by Ewen Wannop
in advance -- please contact via email by sending mail to:

spectrumdaddy@speccie.uk

There is no guarantee that the right to redistribute this material
will be granted. The contents of this document may not be
reprinted in part or in whole.

Credits

My thanks go to Enrico Rolfi for suggesting that a conversion application
would be useful, to Kent Dickey and Antoine Vignau, for help with the
LongLong2Dec and Dec2LongLong routines,
and to Chris Vavruska for his help and valuable suggestions.



Preface



Welcome to HexCalc

Background

While troubleshooting a bug report from Enrico Rolfi, he suggested that it would be useful to have an easy way to convert or display numbers in their Hexadecimal, Decimal, or other formats. I have long used the original Mac calculator to convert between Hexadecimal and Decimal numbers, so decided that the answer would be to replicate that calculator as an NDA for the IIgs.

Although the IIgs normally only handles up to 32 bit numbers, it seemed sensible to support the full 64-bit capacity of the Mac calculator.

Thus the format for the NDA was decided, and that it should be able to work either in 64-bit Hexadecimal, Decimal, or Octal, with the Display being echoed as a 64-bit binary number underneath, just as the Mac calculator itself does.

I also wanted to add IPC calls to the NDA, so that an application could make use of the 64-bit calculations if it wished. To check these calls worked as they should, I quickly built a Sampler application to test them. It then dawned on me that although this Sampler did nothing more than the HexCalc NDA itself did, there were some interesting advantages that it had over the NDA. As a result, I fleshed it out into the companion HexCalc Sampler application that you will find bundled along with the NDA itself.

Full details of how to use the NDA, the Sampler, and the IPC calls, are detailed in the following pages.



Reference



The HexCalc NDA

Requirements

To install the HexCalc NDA, drop it into the System:Desk.Accs folder and reboot.

To run the HexCalc Sampler, put it into a folder of your choice. The Sampler requires both the HexCalc NDA, and the Undo Manager to be installed. A copy of the Undo Manager is provided with the archive.

What is the HexCalc NDA

HexCalc is an NDA that lets you calculate or convert up to 64-bit numbers in Hexadecimal, Decimal, or Octal, with an additional Binary display of the result.

The HexCalc Sampler allows you to do the same things as the NDA, but with increased support for Binary numbers, and a quick Conversion routine between the various formats.

Internally, HexCalc works entirely with 64-bit Hexadecimal Integer numbers, so an entry or display in any other format, will first have been converted to a 64-bit number, and then converted back for the result. Because of this limitation, a division calculation will not show the remainder value from the calculation.

Legal

HexCalc and HexCalc Sampler are both Freeware, and may be distributed freely, provided the archive stays intact, no alterations are made to it, and full attribution is always made to the author Ewen Wannop. HexCalc and HexCalc Sampler may not be sold in any form, but may be included in other distributed archives, provided you first seek my permission:

spectrumdaddy@speccie.uk

Using HexCalc



The HexCalc calculator has a traditional layout, with the keypad in the centre, function keys surrounding it, and a Display showing the result of the calculation at the top. It also has a simple Memory Save and Recall function.

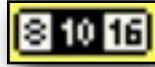
Below the main Display, is the Binary value of the Display.

Depending on the Mode you have selected from the three buttons to the right of the Display, 16 = Hexadecimal, 10 = Decimal, or 8 = Octal, keys in the keypad that cannot be used for that Mode will be hidden.

64-bit Hexadecimal numbers can be up to 16 characters long, and the Display will show a “0x” at the start to define a Hexadecimal number. Decimal numbers can be up to 20 numbers long, with the largest number being “18446744073709551615”. This is the Decimal equivalent of “0xFFFFFFFFFFFFFFFF”. Octal numbers can be 22 numbers long, with the largest being “1777777777777777777777”.

The Keys

You can either type using the number keys on your keyboard itself, or by clicking the keys on the screen with the mouse. The main five function keys can also be typed using “/” for Divide, “-” for Minus, “*” for Multiply, “+” for Add, and “=” for Equals.



To change Mode between Hexadecimal, Decimal, and Octal, use either the OA-H, OA-D, and OA-O keys, or cycle through them with the OA- right or left arrow keys.

Some of the buttons will change colour if they have been set. For instance, in the screenshot above, the Mode has been set to 16-bit Hexadecimal, the display holds a 64-bit Hexadecimal number, the Multiply button has been clicked, and a value has been added to the Memory.

Optionally, you can use Binary number entry instead of the current Mode. To toggle Binary entry, press OA-B, or click on the Binary display,. Only the ‘0’ and ‘1’ keys will be active, and the main display will show the entered number in the current Mode.

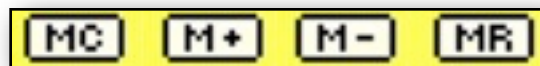
Click the (?) button, or press the “?” key, to show the “About” window.

Using the Keyboard

Most calculations require you to enter a number first, then a function key, followed by a second number, and finally the Equals key to display the result. Some functions like ROL and ROR work immediately on the number in the Display, and for the four arithmetic functions, if after the calculation, you press the Equals key without entering a new number, it will repeat the last calculation.

Understanding the Keyboard

The keyboard is divided into five main sections.



The four memory keys at the top:

MC = Clears the memory to zero

M+ = Adds the number in the display to the number in memory

M- = Subtracts the number in the display from the number in memory

MR = Transfers the number currently in memory to the display

If there is a number in memory, the MR key will display as orange.



The six Logic keys are to the left:

AND = Makes a logical AND with the first and second number

OR = Makes a logical OR with the first and second number

NOR = Makes a logical NOR with the first and second number

XOR = Makes a logical XOR (EOR) with the first and second number

ROL = ROLs rotates left the number. The highest bit will rotate to the lowest bit of the 64-bit number

ROR = RORs rotates right the number. The first bit will rotate to the last bit of the 64-bit number

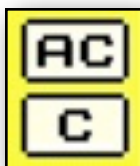
The top four keys will display as green if they have been selected.



The 16 alphanumeric keys 0-F in the center:

Depending on the current Mode you have selected, the display will show either 8, 10 or 16 keys

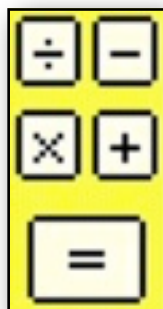
To the bottom left is the (?) About window key



The two Clear keys at top right:

AC = All clear. Clears everything except the Memory

C = Clears the display and any current function



The five Function keys to bottom right:

Divide = Divides the first number by the second

Minus = Subtracts the second number from the first

Multiply = Multiplies the first number by the second

Plus = Adds the two numbers together

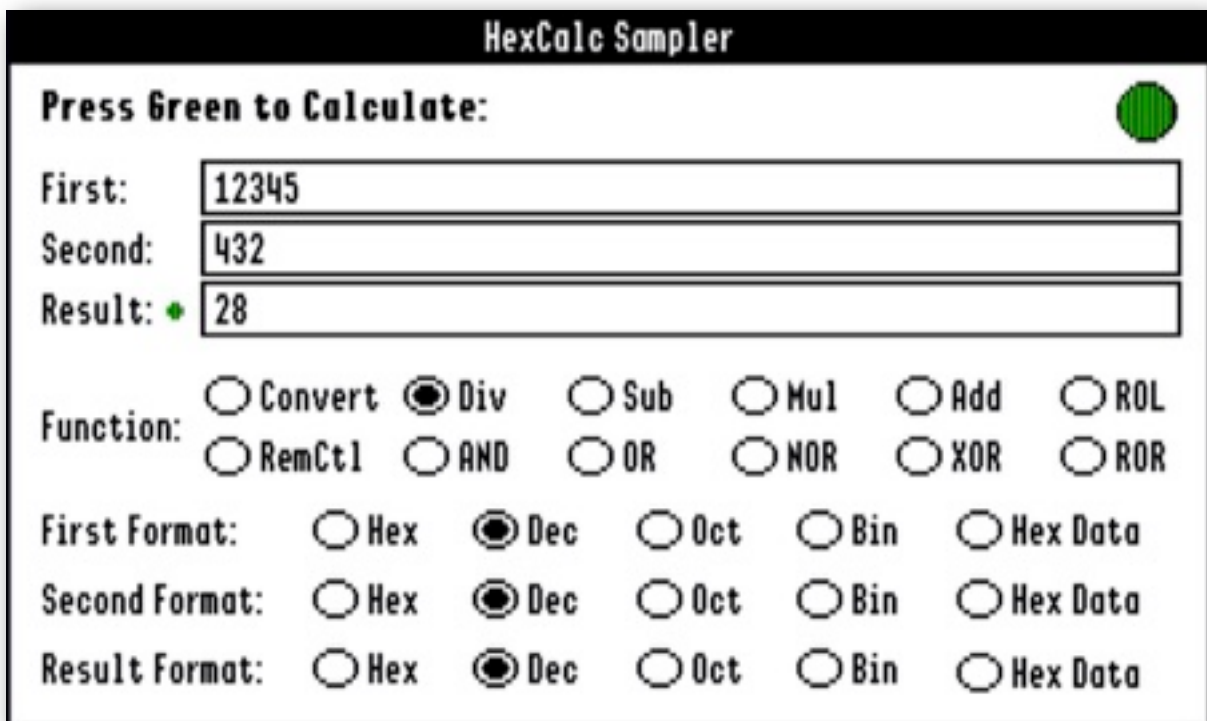
Equals = Completes a calculation, or repeats a calculation

The four function keys will display as green if they have been selected.

Sampler

The HexCalc Sampler

Using the Sampler



The screenshot shows the HexCalc Sampler application window. At the top, it says "HexCalc Sampler" and "Press Green to Calculate:" with a green circular button. Below this are three input fields: "First:" containing "12345", "Second:" containing "432", and "Result:" containing "28". The "Result:" field has a small green dot next to it. Below the input fields are several radio button options for "Function": Convert, Div (selected), Sub, Mul, Add, ROL, RemCt1, AND, OR, NOR, XOR, and ROR. Below the function options are three rows of radio button options for "First Format:", "Second Format:", and "Result Format:". Each row has five options: Hex, Dec (selected), Oct, Bin, and Hex Data.

The HexCalc Sampler is a useful utility that uses the HexCalc NDA supported IPC calls for its calculations. For many programmers, they may find the Sampler more useful than the NDA itself for quickly converting numbers between the four formats.

Although all three LineEdits can be edited, only the top two are used to supply numbers for the calculations. In the case of the Convert, ROL, and ROR functions, only the top LineEdit is used.

As all three are LineEdits, you must only enter valid characters for the Format that you have selected into the top two, or you will get an error message when you press the Green Calculate button.

Hexadecimal numbers can contain alphanumeric characters from 0-9 and A-F, and optionally can be preceded by “0x”. A Hexadecimal number can be up to 16 characters long, or 18 if “0x” is used. Do not use “\$” to define a Hexadecimal number.

Decimal numbers can contain numbers between 0-9, and can be up to 20 long.

Octal numbers can contain numbers between 0-7, and can be 22 long.

Binary numbers can contain 0’s or 1’s, and optionally spaces between four byte blocks, and disregarding the optional spaces, can be 64 numbers long. With spaces between each byte block, a Binary number can be up to 79 characters long.

Calculating or Converting

To make a calculation, first select the format for the First and Second input numbers, and then the format that you would like the result to be displayed in. Enter the numbers in their appropriate formats, and then either press the Green Calculate button at top right, or press the Return key.

For the Convert, ROL, and ROR, functions, you only need to enter a number in the First LineEdit.

If you select the “Hex Data” format for either of the first two boxes, you must supply a Hexadecimal number. The Sampler app will convert that number locally, and then send a 64-bit value to the HexCalc NDA. If you select “Hex Data” for the Result, then the HexCalc NDA will return a 64-bit Hex value, rather than a formatted cString, and the Sampler will display that as a Hexadecimal representation. In essence, selecting “Hex” or “Hex Data” actually does the same thing, even though the processing is handled differently.

If there is a Remainder from a Division calculation, a small green button will show against the Result. Toggle the green button to show the Remainder, which will display with a red button, and then toggle again to show the Quotient once more.

The three LineEdit boxes support the Undo Manager, so you can Undo, Redo, Cut, Copy, Paste, Clear, and Select All, for any of the text in the LineEdits.

Refer to page 13 for how to construct a command line for the RemCtl remote control function. Enter the string in the First LineEdit, then press the Green Calculate button. Note that the format of the result will be dependant on the format defined in the LineEdit.

The File Menu

As well as being able to OA-Q or Quit the Sampler from the File menu, you can optionally open or close the HexCalc NDA with OA-O or OA-W if required.



IPC Calls



The HexCalc IPC Interface

IPC Calls

The HexCalc NDA responds to five IPC calls directed to modName:

```
$8000    askHexCalcAreYouThere
$8001    tellHexCalcProcessNumber
$8002    openHexCalc
$8003    closeHexCalc
$8004    remoteControlHexCalc
```

To see if HexCalc is active and present, call using \$8000, and check modDataOut is not zero:

```
pea      $8000                ; askHexCalcAreYouThere
pea      $8001                ; stop after 1
PushLong #modName
pea      0
pea      0
PushLong #modDataOut
_SendRequest

modDataOut ds    2                ; request count

modName   str    'Speccie~HexCalc~Wannop~'
```

To open the HexCalc NDA from an application, make the same call using \$8002, and to close the HexCalc NDA, call using \$8003.

To use the power of the HexCalc NDA to make 64-bit calculations for your application, call using \$8001 as described below.

You will specify the function you want HexCalc to process, the format of the two cStrings, the format for the result, and then pass data either using cStrings, or a 64-bit value.

To ask HexCalc to make a calculation:

```
    pea    $8001                ; tellHexCalcProcessNumber
    pea    $8001                ; stop after 1
    PushLong #modName
    PushLong #modDataIn
    PushLong #modDataOut
    _SendRequest
```

The input and output blocks are defined thus:

```
modDataIn    anop                182 bytes input
r_function    dc i'$0001'        process required
r_format      dc i'$0101'        format of input data
s_format      dc i'$0001'        format for result
r_value_1     ds 8                64-bit Hex value (Optional)
r_value_2     ds 8                64-bit Hex value (Optional)
r_str_1       ds 80               cString1
r_str_2       ds 80               cString2

modDataOut    anop                100 bytes output
              ds 2                request count
s_error       ds 2                error
s_value_1     ds 8                64-bit Hex Value (Optional)
s_value_2     ds 8                64-bit Hex Value (Optional)
s_str         ds 80               cString
```

Normally, you will need to supply two strings, the first number and the second number, and the function you want the HexCalc NDA to calculate. The result will appear as either a cString in s_str, or alternatively as a 64-bit Hexadecimal value in s_value_1. Any errors are reported in s_error.

Set r_function to the required function:

Bit 0	= Divide	\$0001
Bit 1	= Minus	\$0002
Bit 2	= Multiply	\$0004
Bit 3	= Add	\$0008
Bit 4	= AND	\$0010
Bit 5	= OR	\$0020
Bit 6	= NOR	\$0040
Bit 7	= XOR or EOR	\$0080
Bit 8	= ROL	\$0100
Bit 9	= ROR	\$0200
Bit 10	= Convert	\$0400

Except for ROL, ROR, and Convert, which only require a single string in r_str_1, you must supply both strings in r_str_1 and r_str_2. Specifying the format of the two strings in r_format.

Alternatively, instead of passing cStrings, you can pass 64-bit Hexadecimal values in `r_value_1` and `r_value_2`, by specifying their format as Data.

The two input strings, and the single output string, are all cStrings of a maximum 79 bytes, plus the 0 trailing byte. The input cStrings must contain a valid ASCII representation of the supplied numbers in the format given in `r_format`.

The Formats for `r_str_1` are defined in bits 0 - 4 of `r_format`:

Bit 0	= Hex	\$0001
Bit 1	= Dec	\$0002
Bit 2	= Oct	\$0004
Bit 3	= Bin	\$0008
Bit 4	= Data	\$0010

The Formats for `r_str_2` are defined in bits 8 - 12 of `r_format`:

Bit 8	= Hex	\$0100
Bit 9	= Dec	\$0200
Bit 10	= Oct	\$0400
Bit 11	= Bin	\$0800
Bit 12	= Data	\$1000

Define the Format you would like the returned result in `s_format`:

Bit 0	= Hex	\$0001
Bit 1	= Dec	\$0002
Bit 2	= Oct	\$0004
Bit 3	= Bin	\$0008
Bit 4	= Data	\$0010

The result of the calculation will then be passed either in `s_str` as an ASCII cString, or in `s_value_1` as a 64-bit Hexadecimal Data value.

If there is a Remainder from a division calculation, it will be returned in `s_value_2` as a 64-bit Hexadecimal Data value. The calling app can then decide how to use that value.

Returned Error values:

\$8001	bad function supplied
\$8002	no input values supplied
\$8003	bad input values
\$8004	the calculation failed
\$8005	HexCalc not open

Run the HexCalc Sampler to see an example of how the IPC calls can be used in an application.

Remote Control of HexCalc

By sending an IPC call of \$8004 to HexCalc, you can remotely control HexCalc as if you were controlling the NDA directly, with the NDA opened or closed. RemCtl or Remote Control of HexCalc may be of limited use, but the function is there if you find it useful to allow user controlled calculations in your applications.

Optionally first open HexCalc with IPC call \$8002, then send a string of commands in the cString rc_str. HexCalc will then process that string, and return the result to you in s_str. Note that the format of the output is normally defined at the start of the input string, but you can change format at any point you like within the string, and the default is for Hexadecimal calculations and results.

If necessary, close HexCalc with IPC call \$8003.

Any errors will be reported in rc_error on return.

```
pea      $8004                ; remoteControlHexCalc
pea      $8004                ; stop after 1
PushLong #modName
PushLong #rcmodDataIn
PushLong #modDataOut        ; see Page 11
_SendRequest

rcmodDataIn anop            80 bytes input
rc_str     ds    80         cString - max 79 characters
```

The rc_str command cString, is built with alphanumeric characters and other commands. The alphanumeric characters represent the main keyboard and the function keys. The other functions are given as \$xx commands.

If you cannot fit sufficient commands into the 79 character limit of the cString, then send more than one string. As calculations can be passed with more than one string, remember to clear the result with \$02 if you are starting a new calculation.

You can use spaces between the commands, each space will add a one second delay.

This simple example: '\$01\$0E 1234+4=' taken step by step:

```
$01  calls 'All Clear' to remove existing numbers or functions
$0E  sets 'Decimal' mode
(space) delays by one second
1234 enters '1234' into the display
+    sets the 'Plus' function
4    enters '4' into the display
=    finalises the calculation with the display showing '1238'
```

The Remote Control Commands:

Command	Key Equivalent	Command	Key Equivalent
0	0	\$01	All Clear
1	1	\$02	Clear
2	2	\$03	AND
3	3	\$04	OR
4	4	\$05	NOR
5	5	\$06	XOR
6	6	\$07	ROL
7	7	\$08	ROR
8	8	\$09	Memory Clear
9	9	\$0A	Memory Plus
A	A	\$0B	Memory Minus
B	B	\$0C	Memory Recall
C	C	\$0D	Octal
D	D	\$0E	Decimal
E	E	\$0F	Hexadecimal
F	F	\$10	Delete Key
+	Add	\$11	Copy
-	Minus	\$12	Shift Copy
x or *	Multiply	\$13	Paste
/	Divide	\$14	Shift Paste
=	Equals		



Extras



Problems

Hopefully you will have none, but if you do, and they cannot be answered by reading these notes, please contact me on:

spectrumdaddy@speccie.uk

Other information

If you have not yet explored my web site, please do. There are a host of useful utilities there, that not only let you access FTP servers, browse the web, access your Mail, or even read and reply to Usenet newsgroups, but also to help you keep your IIgs in order:

<http://speccie.uk>

If you do not already know about Spectrum™, please drop by my home pages and read the PDFs. Apart from all the other wonderful things it does, Spectrum™ offers many useful tools for processing files, such as post processing text files that you have received that may have obstinate formatting.

Spectrum™ is now Freeware, and amongst my other applications, is available from my web site:

<http://speccie.uk/software/spectrum/>

Someone once said to me, 'Spectrum™ does everything!'

HexCalc © 2021 Ewen Wannop