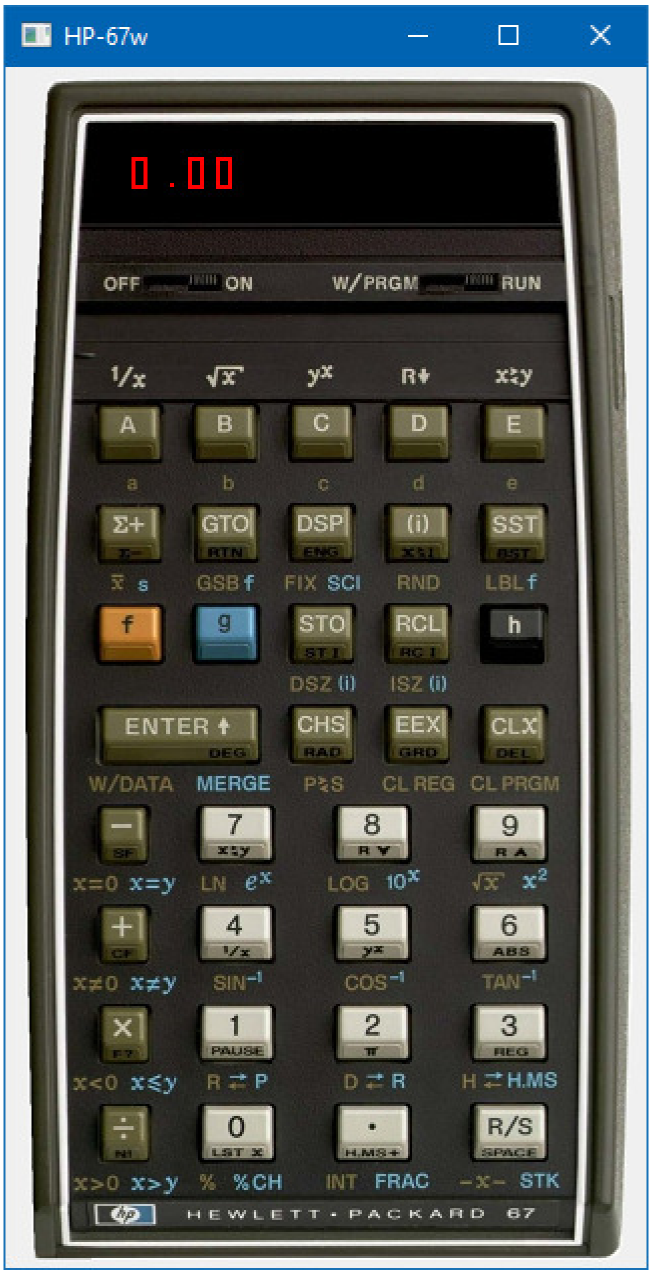
HP67W


HP-67w
0.00
OFF ON
W/PRGM RUN
1/x √x y^x R↓ x⇄y
A B C D E
a b c d e
Σ+ GTO DSP (i) SST
x̄ s GSB f FIX SCI RND LBL f
f g STO RCL h
DSZ (i) ISZ (i)
ENTER↑ CHS EEX CLX
W/DATA MERGE P⇄S CL REG CL PRGM
7 8 9
x=0 x=y LN e^x LOG 10^x √x x²
4 5 6
x≠0 x≠y SIN⁻¹ COS⁻¹ TAN⁻¹
1 2 3
x<0 x≤y R⇄P D⇄R H⇄H.MS
0 • R/S
x>0 x>y % %CH INT FRAC -x- STK
HEWLETT • PACKARD 67

PURPOSE

The HP67W is a Windows-based emulator for a Hewlett-Packard HP67 calculator that was released in 1976. It was HP's successor to the first ever pocket programmable calculator – their HP65 – and it built on lessons learnt from the original. Instead of remembering and replaying 100 keystrokes, memory was increased to 224 steps and it remembered functions instead of each key to get there. HP called this "fully merged keycodes".

The emulator looks like the real thing. It also works like the real thing because internally, it runs the same microcode that ran the real thing. It has the same bugs. It has the same features. It produces the same results to the same precision.

It even matches the display that flickered when running programs and what is flickering matches what was flickering on the real thing.

It supports the weird tricks that were possible with the real thing. This includes "non-normalised numbers" (NNNs) that users figured out ways to enter into the calculator to produce strange timing loops and even word displays that were never intended by the manufacturer but were discovered by users, and which probably led to the alphanumeric display on its successor, the HP41.

The emulator lets you do what you could do with the real thing; but, it also has some additional features that'll probably only interest the die-hard enthusiasts.

If you're so inclined, you can actually see the microcode as it runs in the calculator. You can see where those flickers in the display come from. You can examine what is actually in a storage register, or poke non-standard information directly into one. You no longer need special hardware to trigger special modes. You can just enter those at will from the keyboard. You can also pause, resume or redirect the microcode. You could even, if you felt so inclined, rewrite or replace the microcode. Not that it'd be a pure HP67 anymore, but you could.

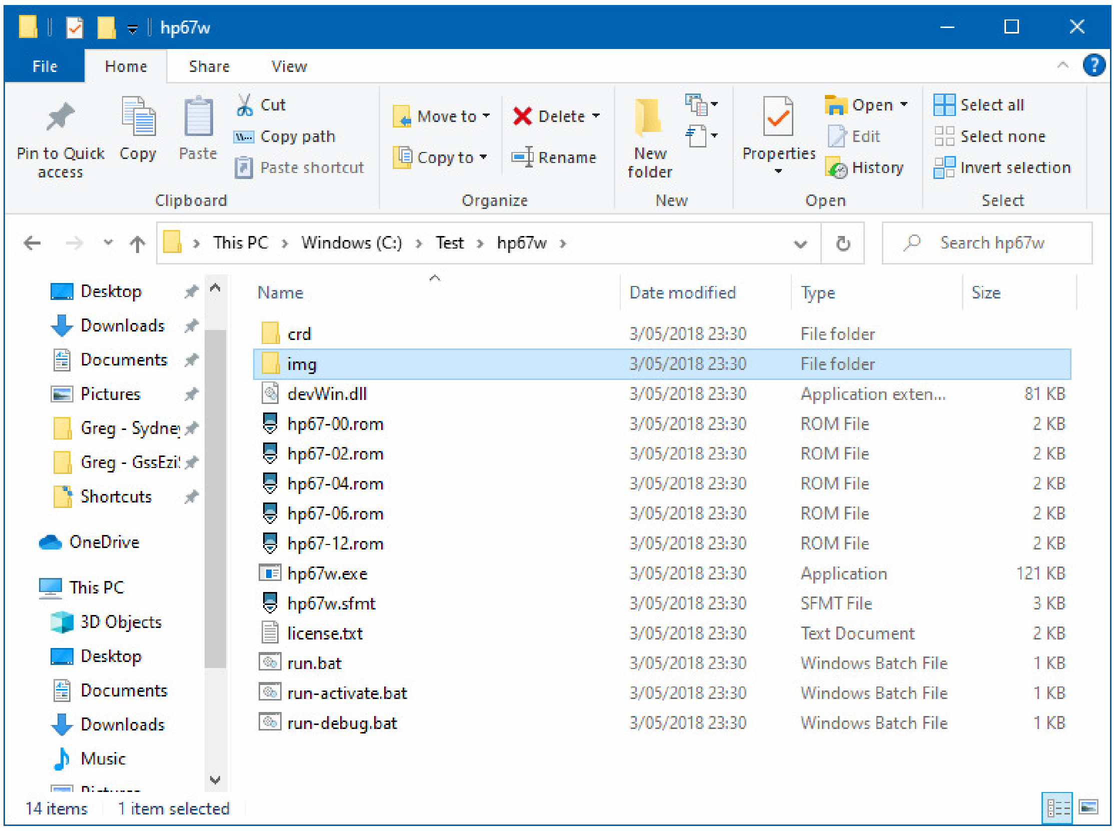
DOWNLOADING

It you're reading this, you've probably already downloaded a copy of the program. If not, and you found this file somewhere, the website is http://www.sydneysmith.com so go there and have a look around. It is currently findable under a menu option for Products / Calculators or Topics / Calculators / HP67.

INSTALLING

Installation is really easy. The software comes as a ZIP file and you just unZIP the file wherever you'd like. Though, since Windows Vista, I'd suggest avoiding "C:\Program Files" or "C:\Program Files (x86)" as you won't be able to save calculator programs in there. It's a feature of the Operating System these days. I tend to install into a "C:\Test" directory but that's more applicable to me than you. I added a “C:\Progs” directory and install things that run from their own area rather than spreading themselves across the Windows Registry for settings, one area for the program, and a separate area for larger configuration files.

When you’ve installed (unZIPped) the download, you should see something like:



Name	Date modified	Type	Size
crd	3/05/2018 23:30	File folder	
img	3/05/2018 23:30	File folder	
devWin.dll	3/05/2018 23:30	Application exten...	81 KB
hp67-00.rom	3/05/2018 23:30	ROM File	2 KB
hp67-02.rom	3/05/2018 23:30	ROM File	2 KB
hp67-04.rom	3/05/2018 23:30	ROM File	2 KB
hp67-06.rom	3/05/2018 23:30	ROM File	2 KB
hp67-12.rom	3/05/2018 23:30	ROM File	2 KB
hp67w.exe	3/05/2018 23:30	Application	121 KB
hp67w.sfmt	3/05/2018 23:30	SFMT File	3 KB
license.txt	3/05/2018 23:30	Text Document	2 KB
run.bat	3/05/2018 23:30	Windows Batch File	1 KB
run-activate.bat	3/05/2018 23:30	Windows Batch File	1 KB
run-debug.bat	3/05/2018 23:30	Windows Batch File	1 KB

File dates and sizes may vary (if I update the program). There’s also the possibility that future versions will be structured differently but this structure is working well and is likely to remain. Calculator programs get saved in the “crd” directory which is why installing in the standard “C:\Program Files[…]” directories isn’t recommended.

The intention is for this manual to also show up in the above directory. It is likely to be “help.pdf”.

NORMAL USE

Starting. Getting started is easy. Double-click “run.bat” and you’ll see the calculator.

The emulator is free to try for 90 days because I’d rather people got the opportunity to see if they like something before they have to pay for it. The price is currently $5 and that’s an amount to encourage me to do more, rather than a price to start building an empire.

Activating. If you buy a copy you'll get an access code that allows the program to continue running after the 90 days. When you have the code, double-click "run-activate.bat" and enter the code. You'll only need to do it once but I'd suggest keeping the email with the code in case you replace your computer and need to install again on the new computer.

After you've done "run-activate" you can go back to double-clicking "run.bat".

Exiting. This is really easy: just click the standard [X] button at the top right of the emulator window.
Pressing Buttons. Just click the relevant key.

Sliding Switches. Click to the left of the current switch position to slide it left, or to the right to slide it right.

Loading and Saving Programs. This also works like the real thing. If the "W/PRGM – RUN" switch is in the RUN position, it will read a card into the programmable calculator memory. If it is in the W/PRGM position it will write to the card.

Tapping (clicking) the left side of the card slot will read or write from/to side 1 of a card. Tapping (clicking) the right side will use side 2.

Cards are stored in the "crd" sub directory. Side 1 is stored on crd/side1.dat. Side 2 is stored on crd/side2.dat. If you'd like to build a library of cards I suggest creating a set of subdirectories under "crd" and copying side1.dat and sifde2.dat in and out of those. This is similar to the original process of selecting a card from a library pack and putting it in the card slot, or picking a blank card from a pack, putting it in the slot and then labelling it and putting it back in the pack.

If you installed into "C:\Program Files[…]", the operating system will prevent you saving any programs onto cards. It is best to install elsewhere (see earlier).

Note: the format of the information recorded in side1.dat and side2.dat is a textual representation of the information written to a real card. You can open the cards in Notepad.exe (or similar) to see what is written. There is a header, information and a checksum for each side. As stated in the HP67 manual, the card also records the angle mode (DEGrees, RADians or GRaDians), the precision (DSP 0-9), display setting (FIX, SCI, ENG) and the state of the flags. That is all included in the header. As is whether the side is program or data and whether another side needs to be loaded.

The information isn't easily read as it is just the bits recorded on a real card. You can translate to and from the bits using: HP-67 Format Convertor. Select Mag Card on the LHS, paste the contents of side1.dat or side2.dat from Notepad (or similar) into the main LHS box, add two lines at the top "SIDE1" and "38". Or "SIDE2" and "38, then click "Convert". Given the side information already exists in the header, I'll get around to having it work that out for you at some point but, for now, the tool requires those extra two lines.

You can use the same tool to create crd/side1.dat and crd/side2.dat from program listings.

Loading and Saving Data. As per the real thing, you can save data on a mag card. The function to do that is, like the real thing, f W/DATA.

You need to be in RUN mode and to have stored the data you want to save in the calculator memories.

It will ask you to load a card (display says, “Crd”). Tap the LHS to use crd/side1.dat or the RHS to use crd/side2.dat. If you have information stored in the secondary registers, it will ask again for a “Crd” and you should use the other side (typically side 2) for the additional data.

Loading data is also straight forward. While in RUN mode you simply tap the LHS or RHS of the card slot to load crd/side1.dat or crd/side2.dat. The calculator reads the header and loads the information from the card into the primary or secondary data registers (or into program memory if the side contains program information). If a second side is required, you'll see “Crd” in the display and you'll need to tap the other side to load the other side.

Mag cards always supported mixed modes where you could save a short program on one side (up to 112 steps) and data on the other. The emulator also supports that.

ACCESSING THE MICROCODE

Ever wondered how it does what it does? How it gets 0.71 when you key in 45 in DEGrees mode and press f SIN? Or how it knows that 6! Is 720?

It's a long process but you can follow along.

The easiest way I've found is:

a. Start the emulator in debug mode by double-clicking “run-debug.bat”

b. Enter g,167

c. (the calculator starts running, initialises everything and enters the main loop where it waits for you to press a key.)

d. Press a key

e. (this sets a flag internally so that when you continue the main loop, it will see that a key has been pressed and go off and do “things”.)

f. If you want to press more keys before seeing what happens – eg you want to see what [h] [Rv] does and you've pressed [h], press return in the debugging window to go to the next step then enter g,167 to let it process the prefix key. Then press [9] - the key that gives [Rv] when [h] has been pressed.

g. Enter t 10

h. (you'll see the calculator's Woodstock processor step through the next 10 instructions. You'll also see what's in the processor's internal registers and the instructions it is executing.)

i. You may need to enter "t 10" multiple times, or even "t 100" multiple times.

k. If you'd like a more concise display of what's going on, use "ts 10" or "ts 100". The shorter version only lists processor registers that change as a result of the instruction.

l. You can also use "tb 10" or "tb 100" to just show the instructions.

m. When the processor gets back to instruction 167, you're back in the main loop and it has done what it was doing.

The Other Options. The monitor provides a ">>>" prompt. When you see that, you can enter commands that examine or change the calculator internals. The commands are:

a. (Return). **Step**. Do the current step and move to the next. The monitor shows the results of doing the step (the contents of the CPU registers), the PC value and the next instruction.

b. t[b][s] n. **Trace**. Do n steps. This is like Step but it repeats for n times. If you use "tb" (Trace – Brief) it'll only show the instructions. If you use "ts" (Trace – Smart) it'll show you the instructions and any CPU registers that changed.

c. g [address][,brk]*. **Go**. Optionally set the program counter (PC) to address first, Optionally set 1 or more breakpoints. If the PC matches one of the breakpoints, the "Go" stops and you'll see a monitor prompt ">>>" allowing you to enter another command. If it never reaches a breakpoint then the calculator behaves normally and you never see another monitor prompt. So, to get out of debugging mode, just enter "g" at the prompt.

d. l[o][h] [address]. **List**. This shows you the instructions in the calculator's ROM (or after the current one). It can be useful for peeking to see what the calculator is planning next. You can also use "lo" or "lh" to also show the instructions in octal or hexadecimal form.

e. d. **Display**. This has a number of forms:

(1) dr. **Display Registers.** This shows the CPU registers, just like if <return> had been pressed but without executing the next step. It can be used for “where am I?”

(2) dp. **Display Program** counter and step. This is the second part of “where am I?” It shows the current PC value and instruction.

(3) ddN. **Display Data** register bank N (0-3,9). The HP67 has four banks of 16 data registers. These are banks 0, 1, 2 and 3.

NOTE 1. The first program step you enter into the calculator get stored at the top of bank 2 (in register 47). As you enter more program steps they get stored in progressively lower registers. After 112 program steps have been entered all of bank 2 (registers 32-47) will be full. There are seven program steps per register and 16 registers time 7 steps per register gives 112 program steps. If you add another program step, it goes into the top of bank 1 (register 31) and the process continues to the start of bank 1. You can only enter 224 program steps. Bank 0 is the primary storage registers. This is memories 0 through 9, A-E and I. Bank 3 is the secondary storage registers (s0-s9 or 10-19). That leaves six registers seemingly unused in bank 3. Some of those are used internally by the microcode.

NOTE 2. Bank 9 is a side-effect of the card reader. It has it’s own, limited, memories that get used during read and write operations. They show up where a memory bank 9 would be. You can display the contents of Data register bank 9 to see those.

(4) di. **Display** cpu **Internals.** There are a number of internal settings in the Woodstock CPU that aren’t normally considered “registers”. They really are more settings than values. Again, the card reader also adds some settings or flags for things like “ready”, “card inserted”, and so on. This command shows the internal settings.

f. s. **Set.** Like Display, this has a number of forms. These are:

(1) sdNN val. **Set Data** register NN(10) to value(14BCD). NN is a register number, normally from 0 to 63 to span the four banks of 16 registers each. As covered above, 0-15 are primary memories (0-9,A-E,I); registers 16-31 are program steps 113-224, registers 32-47 are program steps 001-112, and registers 48-63 are secondary registers s0-s9 (or 10-19) plus six for internal use.

NOTE 1. 14BCD is a string of up to 14 characters in the following sequence: mantissa sign, most significant digit, …, least significant digit, exponent sign, exponent tens, exponent ones. This applies mainly to primary and secondary storage registers which (almost always) store numbers. The calculator works with 10 digit precision internally so the mantissa is always 10 digits. A “0” in a sign position is positive. A “9” in a sign position is negative. An exponent of “999” is e-1. “998” is e-2. A mantissa of “01…” is +1…. “91…” is -1…. They work slightly differently for mantissa and exponent. A mantissa is always either 0 or between 1 and 9.999999999 ie the decimal point is after the first digit if the exponent is “000”.

NOTE 2. Whilst the normal storage registers almost always contain normal numbers (and were always intended to), you can poke other values into the positions. One example of this is sign digits that aren’t “0” or “9”. Another example is putting a hex digit (A-F) in one or more of the positions. This allows the calculator to display words and for non-normalised numbers to be used in calculations to create long delay loops, as was done in the day on real HP67 calculators.

NOTE 3. Despite the two digits implied by NN, you can use register numbers up to 159. This allows setting values in “bank 9”.

(2) sp addr val. **Set Program.** This modifies the Microcode. It changes microcode address addr (an octal number) to contain instruction “val” (another octal number). The change is only for the session you’re in. If you close and open the emulator again, it will use the original microcode.

NOTE. If you do want to make a permanent change, you can (backup then) edit the “hp67-XX.rom” files in the directory. The files are as follows:

00. The first ROM.	Addresses 00000 to 01777
02. Second ROM.	Addresses 02000 to 03777
04. Third ROM.	Addresses 04000 to 05777
06. Fourth ROM.	Addresses 06000 to 07777
12. Fifth ROM.	Addresses 12000 to 13777

There is a gap in ROM space between 10000 and 11777. That’s the way it was made. Perhaps they saved a few instructions and could remove a ROM.

(3) srNAME val. **Set Register.** This modifies the cpu registers in the Woodstock processor. NAME is one of: a b c d e f m1 m2 s pc p. Most of the registers are in 14BCD format so val for most is also in 14BCD format. The exceptions are:

s (which is a set of 16 flags) whose val looks like “.12.4.6……”. This is the same format as it displays as. A “.” Indicates clear (not set) and a number or letter indicates it is set.

pc (which is a 5 digit octal number). Val is an octal number.

p (which is a value from 0 to 13 that is mainly used to point to a digit within a 14BCD register. Val is a number between 0 and 13, as shown in the display, trace or step commands.

NOTE. Entering sr without a register name will provide a list of register names.

(4) st num. **Set Talk.** This turns on some additional information when some instructions are executed. Num is a combination of: 1=GEN, 2=MEM, 4=CRD. So st 7 turns all of the additional information on. It can be useful when you are trying to see what is happening in some parts of the microcode. It can be distracting or just fill up additional lines at other times. Use as desired.

(5) There isn't a direct way to set internal flags in the processor. However, if you need to do so, you can always use "sp …" to create an instruction in an unused memory location (eg 14000), "srpc …" to go to that instruction, and then Step through it to create the desired oucome.

g. q. **Quit.** This exits the emulator.

Examples of some commands are:

```
>>> t 10
A=0000FFFFFFF100 D=00000000000000 M1=00000000000000 P=11
B=03000000000022 E=00000000000000 M2=00000000000000
C=00000000000000 F=00000000000000 S =.........9.....F
00167: 0 -> s 3

A=0000FFFFFFF100 D=00000000000000 M1=00000000000000 P=11
B=03000000000022 E=00000000000000 M2=00000000000000
C=00000000000000 F=00000000000000 S =.........9.....F
00170: crc 1500; side2?=0

A=0000FFFFFFF100 D=00000000000000 M1=00000000000000 P=11
B=03000000000022 E=00000000000000 M2=00000000000000
C=00000000000000 F=00000000000000 S =.........9.....F
00171: if 1 = s 3 then goto 00263

A=0000FFFFFFF100 D=00000000000000 M1=00000000000000 P=11
B=03000000000022 E=00000000000000 M2=00000000000000
C=00000000000000 F=00000000000000 S =.........9.....F
00173: 0 -> s 1

A=0000FFFFFFF100 D=00000000000000 M1=00000000000000 P=11
B=03000000000022 E=00000000000000 M2=00000000000000
C=00000000000000 F=00000000000000 S =.........9.....F
00174: crc  300; wpgm?=0
```

```
A=0000FFFFFFF100 D=00000000000000 M1=00000000000000 P=11
B=03000000000022 E=00000000000000 M2=00000000000000
C=00000000000000 F=00000000000000 S =.........9.....F
00175: if 1 = s 3 then goto 00204

A=0000FFFFFFF100 D=00000000000000 M1=00000000000000 P=11
B=03000000000022 E=00000000000000 M2=00000000000000
C=00000000000000 F=00000000000000 S =.........9.....F
00177: if 0 = s 11 then goto 00206

A=0000FFFFFFF100 D=00000000000000 M1=00000000000000 P=11
B=03000000000022 E=00000000000000 M2=00000000000000
C=00000000000000 F=00000000000000 S =.........9.....F
00206: 0 -> s 3

A=0000FFFFFFF100 D=00000000000000 M1=00000000000000 P=11
B=03000000000022 E=00000000000000 M2=00000000000000
C=00000000000000 F=00000000000000 S =.........9.....F
00207: crc  560; card?=0

A=0000FFFFFFF100 D=00000000000000 M1=00000000000000 P=11
B=03000000000022 E=00000000000000 M2=00000000000000
C=00000000000000 F=00000000000000 S =.........9.....F
00210: if 1 = s 3 then goto 01324

>>> tb 10
00212: if 0 = s 15 then goto 00167
00214: display off
00215: b exchange c[w]
00216: crc  400; aKey=1
00217: keys to a
00220: 0 -> c[x]
00221: a exchange c[xs]
00222: shift right a[x]
00223: p <- 0
00224: load constant 5
>>> ts 10
00225: a + c -> a[x]                ; A=0000FFFFFFF105
00226: c - 1 -> c[xs]               ; C=03000000000005
00227: if n/c goto 0225             ;
00225: a + c -> a[x]                ; A=0000FFFFFFF10A
00226: c - 1 -> c[xs]               ; C=03000000000F05
00227: if n/c goto 0225             ;
00230: a - c -> a[x]                ; A=0000FFFFFFF205
00231: shift left a[x]              ; A=0000FFFFFFF050
00232: 0 -> c[x]                    ; C=03000000000000
00233: p <- 2                       ; P= 2
>>>
A=0000FFFFFFF050 D=00000000000000 M1=00000000000000 P= 2
B=00000000000000 E=00000000000000 M2=00000000000000
C=03000000000000 F=00000000000000 S =.........9.....F
```

```
00234: load constant 4

>>> l
00235: 0 -> s 3
00236: if 0 = s 4 then goto 00254
00240: if 0 = s 6 then goto 00254
00242: a + c -> a[xs]
00243: if 1 = s 8 then goto 00251
00245: a + c -> a[xs]
00246: if 1 = s 7 then goto 00251
00250: a + c -> a[xs]
00251: m1 exch c
00252: delayed rom 1
00253: a -> rom address
00254: m1 exch c
>>>
A=0000FFFFFFF050 D=00000000000000 M1=00000000000000 P= 1
B=00000000000000 E=00000000000000 M2=00000000000000
C=03000000000400 F=00000000000000 S =.........9.....F
00235: 0 -> s 3

>>> l 0
00000: nop
00001: if n/c goto 0370
00002: delayed rom 2
00003: if n/c goto 0043
00004: p <- 1
00005: load constant 3
00006: c -> addr
00007: data register -> c 13
00010: return
00011: c -> a[x]
00012: c -> addr
00013: data register -> c 0
>>> di
Display           : off (15 digits)
Arithmetic base   : 16
Ram address       : 62
f                 : 0
bank/dlyrom       : 0/-1
sp, stack         : 0 [00161,01163]
lastkey           : 10
CRC R             : ..2.4.......
>>> dr
A=0000FFFFFFF050 D=00000000000000 M1=00000000000000 P= 1
B=00000000000000 E=00000000000000 M2=00000000000000
C=03000000000400 F=00000000000000 S =.........9.....F
>>> dd3
data[48]=00000000000000
data[49]=00000000000000
data[50]=00000000000000
```

```
data[51]=00000000000000
data[52]=00000000000000
data[53]=00000000000000
data[54]=00000000000000
data[55]=00000000000000
data[56]=00000000000000
data[57]=00000000000000
data[58]=00000000000000
data[59]=00000000000000
data[60]=00000000000000
data[61]=00000000000000
data[62]=00000012220000
data[63]=00000000000000
>>> dp
00236: if 0 = s 4 then goto 00254
>>> sr
register names are: a b c d e f m1 m2 s pc p
>>>
```

FREQUENTLY ASKED QUESTIONS

Q. I can’t save any programs.
A. My best guess is you’ve installed in the usual spot, “C:\Program Files[…]” which is blocked from file writing. See “Installing” earlier in this manual.

Q. I don’t like the blue color used on the keyboard.
A1. The blue is a match to the real thing.
A2. If you really, really want to change it; the “img” subdirectory contains a file “face2.bmp”. If you change the color in that (perhaps by selecting individual areas that contain blue lettering and then adjusting the area to be more or less blue) the result will display a different blue.
A3. You can also replace the face2.bmp file with a different one such as a drawn HP67 which would be a lot easier to replace colors on. It will need to be the same pixel width and height to work with the program.

MORE QUESTIONS

The website has a reasonable amount of information on it and it may have additional FAQ items. It also has a lot of examples of tracing what the various calculators do. If all else fails, the contact page is:

http://www.sydneysmith.com/wordpress/contact/

I can’t promise a speedy response but I will try to address what you need.