The HP-67 Simulator for Windows
By
Namir Clement Shammas

This is a short document for using the HP-67 simulator for Windows. I developed the simulator using Visual Basic 6 on a Windows 2000 laptop. The application was tested on Windows 2000 and Windows XP Home Edition. You may need to adjust the screen resolution to properly view the simulator window. Low screen resolutions may fail to display the entire image of the HP-67. The simulator application is not sponsored or affiliated with the Hewlett Packard Company which produced the original HP-67 calculator in the seventies. This simulator does not use the HP-67 ROM code. If you plan to use this simulator for advanced tasks in your work, you will do so at your own risk. Please become very familiar with the simulator by thoroughly testing its features, before you place blind trust on it.

Table of Contents

Basic Features

Using the HP-67 simulator is easy. I have tried to implement as many HP-67 features as possible. I have also taken advantage of the fact that the simulator is a Windows application and used message boxes to communicate with the user.

The simulator application displays the image of the HP-67. The image contains opaque image controls laid over the images of the various buttons, ON/OFF and PRGM/RUN switch labels, as well as the card reader opening. As you move the mouse over the various buttons, the mouse cursor turns into a cross when it is over an opaque image control sitting on top of a button or switch label. This change in shape should tell you that the simulator will execute and operation if you click the mouse button. I have also

included the click sound file, click.wav that plays when you click a button or the label of a switch. You can overwrite that file with other sound files. If you rename that file, the simulator becomes silent. If you click on the area below the LED, the simulator displays an About message box that displays the version number and date.

When you click the OFF switch label, the simulator turns off. If you click the OFF switch label again, the application closes. If you click the ON switch label, after you have clicked the OFF switch label just once, the application simulates the powering up of the calculator. Until you turn on the calculator no other buttons will respond. Like the original HP-67, the simulator does not implement continuous memory. By default, the calculator is on when you run the simulator.

Calculator Operations

While in Run mode, you can click on the various buttons to perform operations that are familiar to veteran HP-67 users. If you are new to the HP-67 then you need to get a copy of the manual of a copy of the PDF from the HP Museum CD set. The museum is at the website www.hpmuseum.com. You will notice that the simulator executes programs much faster that the original HP-67. When you run a program on a real HP-67 the LED display flickers as the calculator executes the various program commands. In the case of the simulator, there is no LED flickering. If the program is relatively short, you can expect to see a result almost immediately. When an operation generates a runtime error, the display shows the word Error in the simulated LED. You have to click on the [CL *x*] button to recover from the error. I have coded the simulator to maintain the values that lead to the error so you can determine why the error occurred.

Entering new numbers involves clicking on the familiar [ENTER] key. I have also coded the simulator to start entering a new number under the following conditions:

1. When you are in the process of entering the digits of an exponent and you click the [EEX] button again. This action makes the simulator enter the previous number in the stack and start on entering a new number.
2. When you are in the process of entering the digits of an exponent and you click the [.] button. This action makes the simulator enter the previous number in the stack and start on entering a new number.
3. When you have clicked on the [.] key to enter a decimal and you click on the [.] key again. This action makes the simulator enter the previous number in the stack and start on entering a new number as a decimal.

The features mentioned above work in both Run and Program modes.

The simulator tracks the numerical range of results. While the PC can handle numbers whose absolute values are in ranges exceeding 1.0E-99 to 9.9999999E+99, I have made the simulator adhere to that range. Thus numbers smaller than 1.0E-99 become zero and numbers 1E+100 and larger trigger an overflow error. The simulator monitors the results mathematical operations and functions (including memory-based arithmetic) to check the above boundaries. Since the precision of the simulator differs from that of the original

HP-67 you need to be careful with programs written for the HP-67 calculator that use tricks based on the precision of the original calculator. While these programs may not be common, stumbling upon them may generate unpredictable results on the simulator.

I have implemented the following operations in Run and Program modes as an extra bonus (there are more bonus commands presented later):

1. Clearing the stack by clicking the [g][EEX] keys.
2. Support for RCL+, RCL-, RCL*, RCL/, RCL+ (i), RCL- (i), RCL* (i), and RCL/ (i) operations both in Run and Program modes.
3. Support for additional flags, 4 through 9 that work like flags 0 and 1.
4. Clearing all of the flags (0 through 9) by clicking the [g][CHS] keys.

The operations for the commands –X- and STK emulate the original HP-67 by showing you the X register and the stack registers, respectively, with blinking decimals. In the case of the STK commands the emulator also displays the name of each stack register along with its values. This labeling of stack values allows you to distinguish between neighboring stack registers that have the same values. It's also convenient so you don't have to keep track of which stack register you are looking at. The simulator also supports the PAUSE command but only during program execution. The PAUSE command makes the program pause for about 3 seconds. The simulator supports the REG and uses a custom message box to display information in all 26 registers, using the current display mode and format. One advantage of using a message box to display the values in the memory registers is that you can take your time to examine the values displayed. You can also quickly close the message box by clicking on the [OK] button.

Using the PC Keyboard

You can use the PC keyboard instead of the mouse to access calculator keys. Here is a table that shows you how the PC keyboard is mapped onto the HP-67 simulator keyboard. Note that the PC keys are case sensitive.

PC Key	***HP-67 Key***
-	-
(, I, or)	(i)
*	Multiply key
/	Divide key
+	+
0 through 9	0 through 9
A through E	Keys A through E
c	CLX
d	DSP
Decimal	Decimal
e	ENTER
f	[f] shift key
g	[g] shift key

G	GTO
h	[h] shift key
Q	<sigma>+
r	RCL
R	R/S
s	STO
S	SST
H or the Underscore key	CHS
x	EEX

Card Reader Operations

The simulator mimics card readers operations by working with files instead of magnetic cards. Thus the simulator supports file input and output operations for both programs and data. You can perform all of these operations in Run mode. By contrast, you cannot write a program to a file in while a program is running.

All file input and output operations involve text files. The simulator does not impose any naming convention. My own preference is to use the .hp67 file extension for program files and the .dat file extension for data files. Reading the wrong type of files will most likely result in a run-time or an input error. The simulator displays message boxes to inform you of errors in file related operations

Reading a Program

By clicking on the keys [g][ENTER] you invoke the MERGE command. This command prompts the simulator to display a message box asking you if you want to read a program. If you click the [Yes] button on the message box, the simulator displays a custom file-selection dialog box. Use that dialog box to navigate through drives, directories, and files, to locate the HP-67 listing file you wish to read. To pick a file, double-click on a file in the Files list box. The file you picked appears in the File text box. You can select different files. To confirm the currently selected file and close the dialog box, click the [OK] button. You can click the [Cancel] button to cancel loading a program files. When the simulator loads a program it displays the image of a magnetic card and includes the name of the program file.

The simulator uses text files to write and read HP-67 programs. Using text files allows you to enter and/or edit HP-67 directly using any text editor. The simulator allows the listings to have line numbers. They are optional. The simulator ignores the line number values. You can even include line numbers at will.

When the simulator loads a program from a text file, it checks the listing's commands against a list of valid commands. The simulator will flag invalid commands and report them to you in a message box. The information includes a line number (generated by the simulator) and a command. Keep in mind that the simulator only stores and counts those program lines that contain commands. The simulator ignores blank lines and lines that are comments only. In other words, the simulator turns the proverbial *blind eye* to blank lines

and comment lines. The message box also request that you edit your source file and reload it again (you can do that without stopping the simulator). Moreover, in case of detecting invalid commands, the simulator basically aborts loading the program by resetting the listing to R/S commands. When using # to declare remarks **after** a command, make sure that there is at least one space or a tab character between the end of the command and the # character.

Here are examples of valid HP-67 listings that you can enter using your text editor. The first example shows a short HP-67 program with line numbers:

```
001 LBL A
002 X^2
003 RTN
```

Here is a version of the above program that is void of line numbers:

```
LBL A
X^2
RTN
```

And here is an example with optional line numbers:

```
001 LBL A
X^2
X^2
LOG
005 RTN
```

Since the simulator ignores the line numbers, they need not accurately match the sequence number. You can write a valid listing like the on shown next:

```
1 LBL A
X^2
5 X^2
LOG
1 RTN
```

Of course a listing like the last one can be more confusing to the human reader than to the simulator. Once the simulator reads a program it internally keeps track of the line numbers.

The simulator also allows you to type comments in listings you create with a text editor. These comments begin with the # character and can occupy entire lines or appear after commands. When the simulator reads a program file it strips the comments that trail commands and ignore lines that contain only a comments. The simulator also ignores blank lines. Here is an example a listing that uses imbedded comments:


```
# My special program
#
# Version 1.0
LBL A
X^2 # square first time
X^2 # square the second time
# Take natural log
LN
RTN
```

It's worth pointing out that the simulator stores the above program as:

```
001 LBL A
002 X^2
003 X^2
004 LN
005 RTN
```

Notice that the fourth line in the original source code appears in step number 001, the fifth line appears as step 002, and so on. Also notice the either line in original source code appears in step number 004 (and not 005) since it comes after an all-comment line. Also notice the comments that following the two X^2 commands have disappeared.

When you write a program and switch to Run mode, the simulator displays the image of a white magnetic card above the custom keys A through E. The card has no program title until you save that program to a file. When you do that, the simulator displays the name of the program file in red on the image of the magnetic card. When you read a program from a file, the simulator also displays the image of the magnetic card and the name of the source program. When you delete a program the image of the magnetic card disappears. When you first start the simulator hides the image of the magnetic card. The image of a title-less magnetic card is a good visual reminder that you have not saved the program in the simulator.

Saving a Program

The HP-67 simulator allows you to save the current program to a text file. When you click on the card reader opening, the simulator displays a message box asking you if you want to write a program, if there is a program in the simulator's memory (otherwise the simulator skips to the prompt to read data). If you click the [Yes] button, the simulator displays another message box asking you if you want to include line numbers with your listing. This message box has the [Yes] and [No] buttons that allow you to include or exclude line numbers from your listing. If you plan to edit the listing with a text editor, I recommend excluding line numbers from your listing as long as your plan to edit them. You can still have the line numbers in your listing to keep track which new line-number-free commands you have added.

The choice of whether or not to use line numbers depends on your style and purpose. Be consistent with whatever valid style you fee more comfortable with.

If you read a listing with the special comments (that use the # character) and attempt to write the current program back to the same source file, the simulator will warn you that you are at risk of overwriting your commented-listing with one that has no comments. The simulator gives you a message box that allows you to either proceed with the operation or back out of it. In the later case, you can later save the current program to a different or new file.

Saving Data

The simulator allows you to write the contents of all the memory registers (R_0 to R_9, R_{s0} to R_{s9}, R_A to R_E, and register I) to a text file. To write the content of the memory registers you click on the [f][ENTER] buttons to invoke the W/DATA command. The simulator displays a custom file selection dialog that allows you to select the output file that stores the values in the memory registers. This operation does not affect the contents of these registers.

Reading Data

The simulator allows you to read the contents of all the memory registers (R_0 to R_9, R_{s0} to R_{s9}, R_A to R_E, and register I) from a text file. To write the content of the memory registers you click on the card reader slot. The simulator first asks you if you want to write a program, if there is a program that you have entered or read. Otherwise you don't see this message box. When you click the [No] button of the first message box, the simulator asks you, using a second message box, if you want to read from a data file. If you click the [Yes] button, the simulator displays a custom file selection dialog that allows you to select the input file that contains the new values loaded into the memory registers.

Programming the Simulator

When you click the W/PRGM switch label, the simulator switches to Program mode. An empty program has R/S commands in each one of the 224 steps. You can insert commands, delete commands, and step through a program just like with an actual HP-67. The main difference with the simulator is that it displays the much-easier to read mnemonic commands (much like the HP-41C calculator) instead of the more cryptic key codes.

To navigate through a listing you can click at the [SST] button to single step forward through the listing. To step backwards, click the [h][SST] keys. You can jump to various locations by clicking on the [GTO][.] keys. When you click on the latter key sequence, the simulator displays an input box that allows you to enter the following information that directs your jump:

1. Enter a line number (1 to 224) to jump to that line.
2. Enter a decimal (the default input) to jump to line 1.

3. Enter TWO decimals to jump to the end of the program. The simulator defines the end of the program as the line before two consecutive program steps with R/S commands.
4. Enter a dash character followed by a label name (A through E, a through e, and 0 through 9) to jump to that label if it exists. If not, the simulator simply takes you to line 1. For example entering –C allows you to jump to label C in the listing.

The next table shows the programmable commands and their mnemonics. The rule to follow if you type an HP-67 listing with a text editor is this. The command mnemonics are NOT case sensitive. By contrast, the alpha labels (A though E and a through e) are case sensitive. For example the commands GTO A and Gto A, and gto A are all equivalent. However, the jump commands GTO A and GTO a are not equivalent because they refer to different labels!

While on the subject of labels, it is worth mentioning that the simulator regards all labels are alphanumeric. You can of course use the standard labels 0 through 9, A through E, and a through e. You can also use any labels for internal branching. This means you can use *enhanced* labels names made up of multiple digits (such as 12 and 122), multiple characters (such as CALC and DoStat), or a combination of numbers and letters (such as DoStat12 and Calc_2_Biz). Thus the simulator supports HP41C labels (this feature came as an accidental aftermath of the way the simulator handles labels) as long as you observe the following rules in using them:

- Label names are case sensitive. This rule holds true when using standard and/or enhanced labels.
- You can enter the enhanced label names using source code you type with a text editor and then load into the simulator. The simulator itself will only allow you to use standard labels when directly programming it.
- Keep the enhanced label names short so the simulator can display them in the text box that simulates the LED display.
- You can only access the enhanced labels using GTO and GSB statements in a program. The simulator does not support accessing these labels in Run mode.

The simulator supports the commands W/DATA and MERGE to write data and read programs during Program mode. To read data while executing a program you need to insert a R/S command to temporarily halt the program to read the data. When the program halts you read the data (by clicking on the card reader slot image) from a file and then click the [R/S] button to proceed with the program execution.

Command Name	***Mnemonic***
<sigma>-	SUM-
<sigma>+	SUM+
Absolute value	ABS
Add	+
Add HMS	HMS+
Arc cosine	ARCCOS or ACOS

Command Name	***Mnemonic***
Arc sine	ARCSIN or ASIN
Arc tangent	ARCTAN or ATAN
Clear flag	CF followed by 0 though 9 (e.g. CF 2)
Clear registers	CLREG
Common logarithm (base 10)	LOG
Compare the value in stack register X and 0	X=0?, X<>0? (same as X!=0? or X#0?), X<0?, X>0?
Compare the values in the stack registers X and Y	X=Y?, X<>Y? (same as X!=Y? or X#Y?), X<=Y?, X>Y?
Cosine	COS
Decimal	(the dot character)
Decrement and skip if zero (indirect addressing using the I register)	DSZ (i)
Decrement the I register and skip if zero	DSZ
Degrees to Radians	D->R
Digits	0 through 9
Divide	/
Engineering display mode	ENG
Enter a new number in the stack	ENTER
Exponent	EEX
Exponential	E^X
Factorial	N!
Fix display mode	FIX
Fraction part	FRAC
Go to a label	GTO followed by 0 through 9, or A through E, or a through e, or (i). Example is GTO a to jump to label a.
Go to a subroutine	GSB followed by 0 through 9, or A through E, or a through e, or (i). Example is GSB 9 to execute the subroutine at label 9.
HMS to Hours	H<-HMS or ->HMS
Hours to HMS	H->HMS or ->H
Increment and skip if zero (indirect addressing using the I register)	ISZ (i)
Increment the I register and skip if zero	ISZ
Integer part	INT
Label	LBL followed by 0 through 9, or A through E, or a through e.
Last X	LST X
Mean value	MEAN
Multiply	*
Natural logarithm	LN
Pause to display the value of the X register (in program mode the simulator plays the	PAUSE

Command Name	***Mnemonic***
key click sound so you can distinguish between successive PAUSE commands that display the same value)	
Percent change	%CH
Percentage	%
Polar to rectangular	R<-P
Power of ten	10^X
Radians to degrees	D<-R
Raise to power	Y^X
Read a program from a file	MERGE
Recall	RCL followed by register 0 through 9, or (i) to use the register I for indirection.
Recall register I	RC I, RCL I, or RC
Reciprocal	1/X
Rectangular to polar	R->P
Roll stack down	Rv or ROLL DN
Roll stack up	R^ or ROLL UP
Round a number	RND
Run/Stop	R/S or STOP
Scientific display mode	SCI
Set flag	SF followed by 0 though 9 (e.g. SF 1)
Set the number of displayed digits	DSP followed by 0 though 9, or (i)
Show memory registers	REG
Show stack registers	STK or PRTSTK
Show X register (in program mode the simulator plays the key click sound so you can distinguish between successive –X- commands that display the same value)	-X- or PRTX
Sine	SIN
Square	X^2
Square root	SQRT
Standard deviation	SDEV
Storage arithmetic	STO+, STO-, STO*, and STO/ followed by register 0 through 9, or (i) to use the register I for indirection. Example is STO+ 1.
Store	STO followed by register 0 through 9, or (i) to use the register I for indirection
Store in register I	ST I, STO I, or STATEMENT
Subtract	-
Swap primary and secondary registers	P<>S
Swap X and I registers	X<>I
Swap X and Y stack registers	X<>Y
Tangent	TAN

Command Name	*Mnemonic*
Test flag	F? followed by 0 though 9 (e.g. F? 0). You can use also the commands F0? through F9? to test flags 0 through 9. The simulator will replace these commands with the commands F? 0 through F? 9.
Write data to a file	W/DATA

Note that in the above table, the simulator substitutes the commands that appear in red (which you can type in a source file using a text editor) with similar commands. For example, the simulator replaces the command ASIN with ARCSIN.

The simulator supports unique labels in a program. This approach allows the simulator to quickly compile the locations of all program labels when you load a program or when you switch from Program mode to Run mode. By compiling the locations of the program's labels, the simulator can execute quick jumps to labels. This approach avoids the less efficient line-by-line search for labels. So if you are porting programs to the simulator please edit duplicate labels. To deal with duplicate labels especially labels A through E and a through e, use the extra labels shown in the next table. So if you have a listing that uses label D twice, you need to convert the second label D into label gD (or hD). This way you can invoke the tasks associated with the original duplicate label D by clicking on [D] and [g][D]. You may need to edit any GTO command that jumps to the second label D into a GTO gD command. Testing such programs is imperative to make sure they work properly.

Label	*Accessed Using*
gA	[g][A]
hA	[h][A]
gB	[g][B]
hB	[h][B]
gC	[g][C]
gC	[h][C]
gD	[g][D]
hD	[h][D]
gE	[g][E]
hE	[h][E]

You must enter the above labels in a program source file using a text editor.

Additional Commands

The next table shows additional commands that the HP-67 simulator supports. Please use these additional commands only if you are not going to run the simulator programs on an actual HP-67 or HP-97. You can only enter and edit the additional commands and enhanced labels using a text editor to insert or edit them in a program listing.

Command Name	***Mnemonic***
Bessel Function $J_0(x)$	BESSJ0
Bessel Function $J_1(x)$	BESSJ1
Bessel Function $J_n(x)$ (the values for n and x are taken from the stack registers Y and X, respectively)	BESSJN
Beep	BEEP
Bessel Function $Y_0(x)$	BESSY0
Bessel Function $Y_1(x)$	BESSY1
Bessel Function $Y_n(x)$ (the values for n and x are taken from the stack registers Y and X, respectively)	BESSYN
Beta function (the arguments m and n are taken from the stack registers Y and X, respectively)	BETA
Ceiling function	CEIL
Chebyshev function (the values for order n and x are taken from the stack registers Y and X, respectively)	CHEB
Clear the flags (0 through 9)	CLRFLG
Clear the stack	CLSTK
Combination ${}_mC_n$ (the values for m and n are taken from the Y and X stack registers, respectively)	COMB
Complimentary error function	ERFC
Cosine Integral	CI
Decrement and skip if zero (indirect addressing using the any register	DSZ IND followed by registers A through E, 0 through 25.
Decrement any numeric register and skip if zero	DSZ followed by registers 0 through 25.
Error function	ERF
Exponential Integral	EXPINT
Floor function	FLOOR
Fresnel cosine integral	FCOS
Fresnel sine integral	FSIN
Gamma function	GAMMA
Hermite function (the values for order n and x are taken from the stack registers Y and X, respectively)	HERM
Hyperbolic cosine	COSH
Hyperbolic sine	SINH
Hyperbolic tangent	TANH
Inverse hyperbolic cosine	ACOSH
Inverse hyperbolic sine	ASINH
Inverse hyperbolic tangent	ATANH

Command Name	***Mnemonic***
Laguerre function (the values for order n and x are taken from the stack registers Y and X, respectively)	LAGR
Legendre function (the values for order n and x are taken from the stack registers Y and X, respectively)	LEGN
Linear Regression (this command pushes r^2, slope, and intercept in the stack in that order). The data for this command must have been entered using the <sigma>+ command.	LR
Linear Regression (this command stores the intercept and slope in registers A and B, respectively. Use with the commands XHAT and YHAT). The data for these commands must have been entered using the <sigma>+ command.	LR2
Permutation ${}_mP_n$ (the values for m and n are taken from the Y and X stack registers, respectively)	PERM
Project X on Y (LR2 command must be used before using this command to first store the intercept and slope in registers A and B, respectively.)	YHAT
Project Y on X (LR2 command must be used before using this command to first store the intercept and slope in registers A and B, respectively.)	XHAT
Random number (between 0 and 1)	RAN#
Recall values using indirection that involves any memory register.	RCL IND followed by A through E, or 0 to 24. Example is RCL IND 7 to use register 7 for indirection.
Recall-register arithmetic.	RCL+, RCL-, RCL*, and RCL/ followed by register 0 through 9, or (i) to use register I for indirection. Example is RCL+ 1.
Sine Integral	SI
Storage using indirection that involves any memory register.	STO IND followed by A through E, or 0 to 24. Example is STO IND 7 to use register 7 for indirection.

The above table contains two commands that handle linear regression. The command LR calculates the correlation coefficient, slope, and intercept, and then pushes these results in the stack. The command LR2 calculates the regression intercept and slope and stores them in registers A and B, respectively. You can use these values to calculated projected Y values with the YHAT command. Likewise, you can calculate the projected X values

using the XHAT command. Here is a short programming example that shows how the commands LR2, YHAT, and XHAT work together.

```
# Initialize
LBL e
CLREG
P<>S
RTN
# Enter Data
LBL A
# swap X and Y since user enters (X,Y) data
X<>Y
Sum+
RTN
# Remove data
Lbl a
X<>Y
Sum-
RTN
# Perform the linear regression
LBL C
LR2
RTN
# Project X onto Y
LBL D
YHAT
RTN
# Project Y onto X
LBL E
XHAT
RTN
```

The label C has the command LR2 which calculates the regression intercept and slope and stores them in registers A and B. The label D allows the user to calculate Y based on a value of X (in the X stack register). Likewise, the label E permits the user to calculate X based on a value of Y in the stack (in the X stack register). The labels A and a allow you to enter observations and remove incorrect or unwanted ones, respectively. The label e initializes the statistical registers.

Executing a Program

The simulator allows you to execute a program at full speed or single-step through it. To execute a program at full speed you can:

1. Click the labels A through E, and [f][A] through [f][E] to run the program starting with the labels A through E and a through e, respectively.

2. Click [f][GSB] followed by clicking on a label A through E, or a numeric label 0 through 9.
3. Click [g][GSB] followed by clicking on a label a through e.
4. Click [f][GTO], followed by clicking on an alphanumeric or numeric label and then clicking the [R/S] keys.

Clicking the [h][GTO] (which is the RTN command) resets the program location counter to line 1 and resets the program state.

While you can end a program with an RTN or R/S command, I suggest that you use the RTM command. This approach allows single-stepping to display a message box telling you when the program has ended. By using the R/S instead, single-stepping will attempt to systematically execute the next statement, even after the *logical* end of the program. When the simulator ends a program it plays a beep.

If the simulator encounters a run-time error, it displays a custom message box that shows the offending line and command.

You can also single-step though programs by using the GTO command in run mode to first direct the program pointer to a label. You can then click the [SST] button to single-step though the program. During this process, the simulator displays the current step in the LED text box for about 2 seconds. You CAN back up a step by clicking the [h][SST] keys. Doing so can corrupt the program execution (especially if you back out of a called subroutine) and give erroneous results. Use this feature with great care!! While in single-step mode, the simulator displays a message box letting you know that the program execution has ended. The simulator also beeps to give you an audible signal. Remember to use the RTN command to reset the program counter if you have problems with back-stepping in a program.

You can switch from single-stepping through a program execution to full speed execution by clicking the [R/S] key.

Saving and Recalling the State of the Calculator

The simulator can save and load the following information:

- The program
- The values in the memory registers
- The values in the stack registers
- The state of flags 0 through 9. The states of flags 2 and 3 are always clear when written to file.
- The number of displayed digits
- The display mode

To save the current state of the simulator, move the mouse to the HP logo at the bottom of the HP-67 image. The mouse cursor becomes an I-beam. When you click the mouse, the simulator displays a custom file selection dialog. You can specify a file to store the

state of the simulator and click the [OK] button to save the simulator state to that file. If you click the [Cancel] button you save the simulator state to the default file hpstate.txt.

To load a previously saved state of the simulator, move the mouse to the 67 logo at the bottom of the HP-67 image. The mouse cursor becomes an I-beam. When you click the mouse, the simulator displays a custom file selection dialog. You can specify a file to load the state of the simulator from and click the [OK] button to load the simulator state from that file. If you click the [Cancel] button you will load the simulator state from the default file hpstate.txt.

In Closing

I hope you enjoy the HP-67 simulator and its speed of program execution. This simulator is a labor of love. Writing the simulator has given me great respect for the folks at HP who created the original HP-67. Creating a simulator requires attention to many details and features, and the ability to orchestrate many things together. My hope is that the simulator provides you with an electronic version of the HP-67 that can live on your PC. This version will not have problems related to aging batteries, card reader, LED segment, or keyboard.

You can now load HP-67 programs from listings published on the Internet on various web sites. Please take the time and examine the listing to make sure that they conform to what the simulator expects. I have found some listings with the following listing features:

- They use of the x character to mean multiply.
- They use of the divide symbol instead of the / character that the simulator uses for division.
- They use of superscript characters used with raising powers (E^X, Y^X and X^2).
- They use of comments that do not come after the # character (which is a convention that has gained some momentum).
- They use the same label more than once. This type of program needs some care in converting the duplicate labels into unique labels. Some of these programs include HP-67 Pacs!

While these listings read a bit better, they need to be edited before you can run them in the simulator. So don't load these listings blindly. Despite the editing, you still save a considerable amount of typing time.

Many thanks go to Katie Wasserman for the picture of one of her HP-67 calculators. Also many thanks for the brave and patient beta testers. Finally, many thanks go to the team at Hewlett Packard that designed the original HP-67 and HP-97 calculators that many owners have enjoyed and continue to do so. I welcome your feedback and comments and strive to respond to them as time permits.

Document Control Information

The revision history for this document is:

- Created version 0.9 on October 20, 2003.
- Edited version 0.9 between October 21, 2003 and November 6, 2003
- Released version 1.0 on November 7, 2003.
- Version 1.01 on November 16, 2003. Removed Greek Sigma symbol and up-arrow symbol since they did not appear properly in PDF file.

Software Version

The version for the simulator with this release is version 1A.

