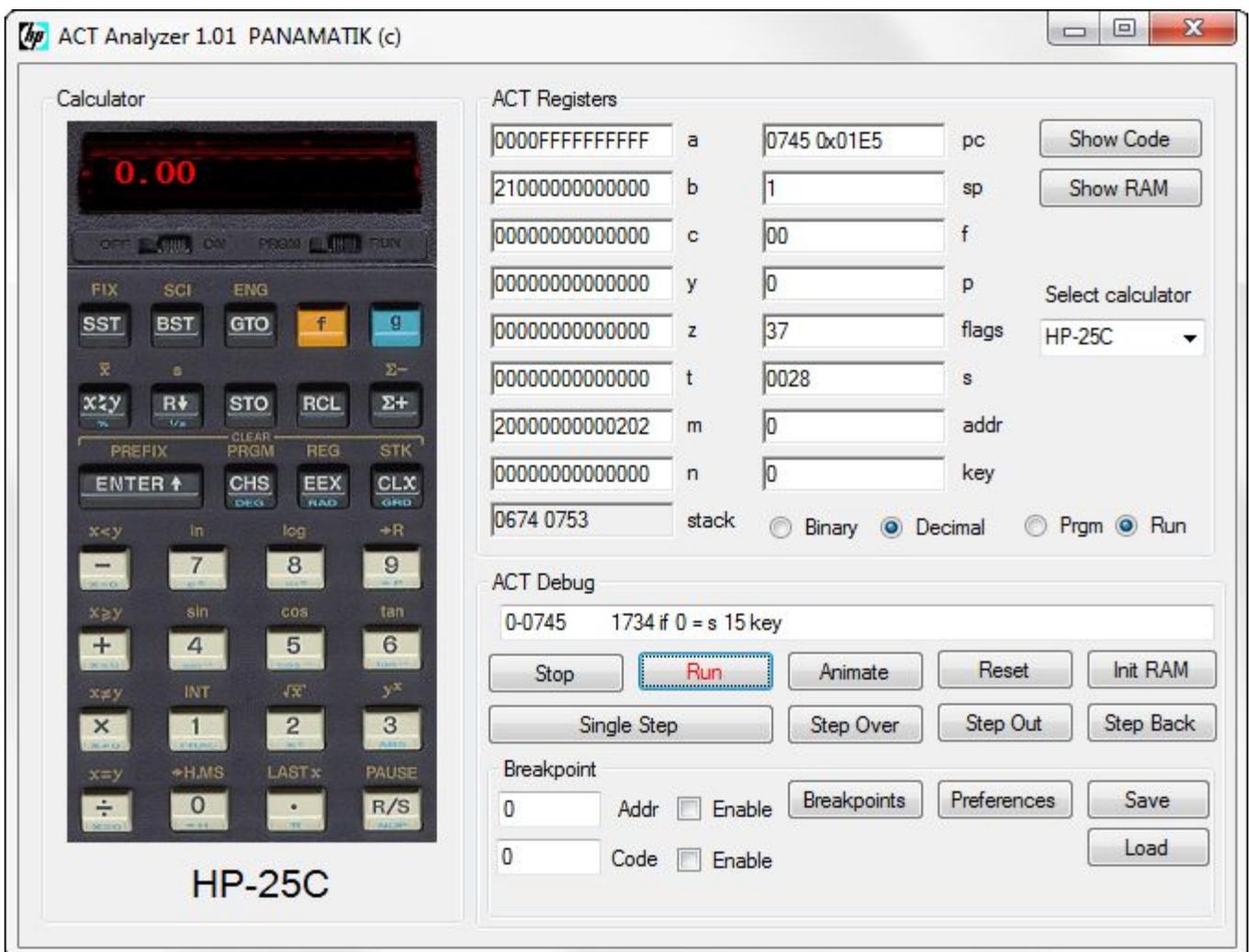


# ACT Analyzer

## User Manual



## ACT Analyzer

### Contents

1. Introduction.....	3
2. The Analyzer.....	4
3. Disassembly window.....	7
3.1. Comment Entry.....	7
3.2. Show Program Flow.....	8
4. RAM registers.....	9
5. Select Calculator.....	10
6. Breakpoints.....	11
7. Cycles.....	11
8. The ACT Assembler.....	12
8.1. Labels.....	13
9. Context Menus.....	13
10. Trace Window.....	14
11. Preferences.....	15
11.1 Save.....	16
11.2. Load.....	16
HP-65.....	17
HP-01.....	18
HP-19C.....	19
Appendix A.....	20
Files.....	20
Appendix B.....	21
ACT and NUT Assembler Syntax.....	21
Classic Assembler Syntax.....	25
HP-01 Assembler Syntax.....	28

Rev 1.00 May 2020 Initial release

Rev 1.01 June 2020 chapter ACT Assembler, Assembler Syntax added

Rev 1.02 June 2020 NUT Syntax added, Trace Window added

## ACT Analyzer

### 1. Introduction

The „ACT Analyzer“ is a mighty emulator and the ultimate Analyzing and Debugging Tool for these old HP LED calculators like HP-25, HP-34C, HP-67, which use the ACT processor (Arithmetic Control Timing). It emulates the original ROM codes of all Woodstock Spice and Classic models. This manual describes how to use the Analyzer to understand the secrets and the mathematics behind these old calculators and to be able to retrace how they work.

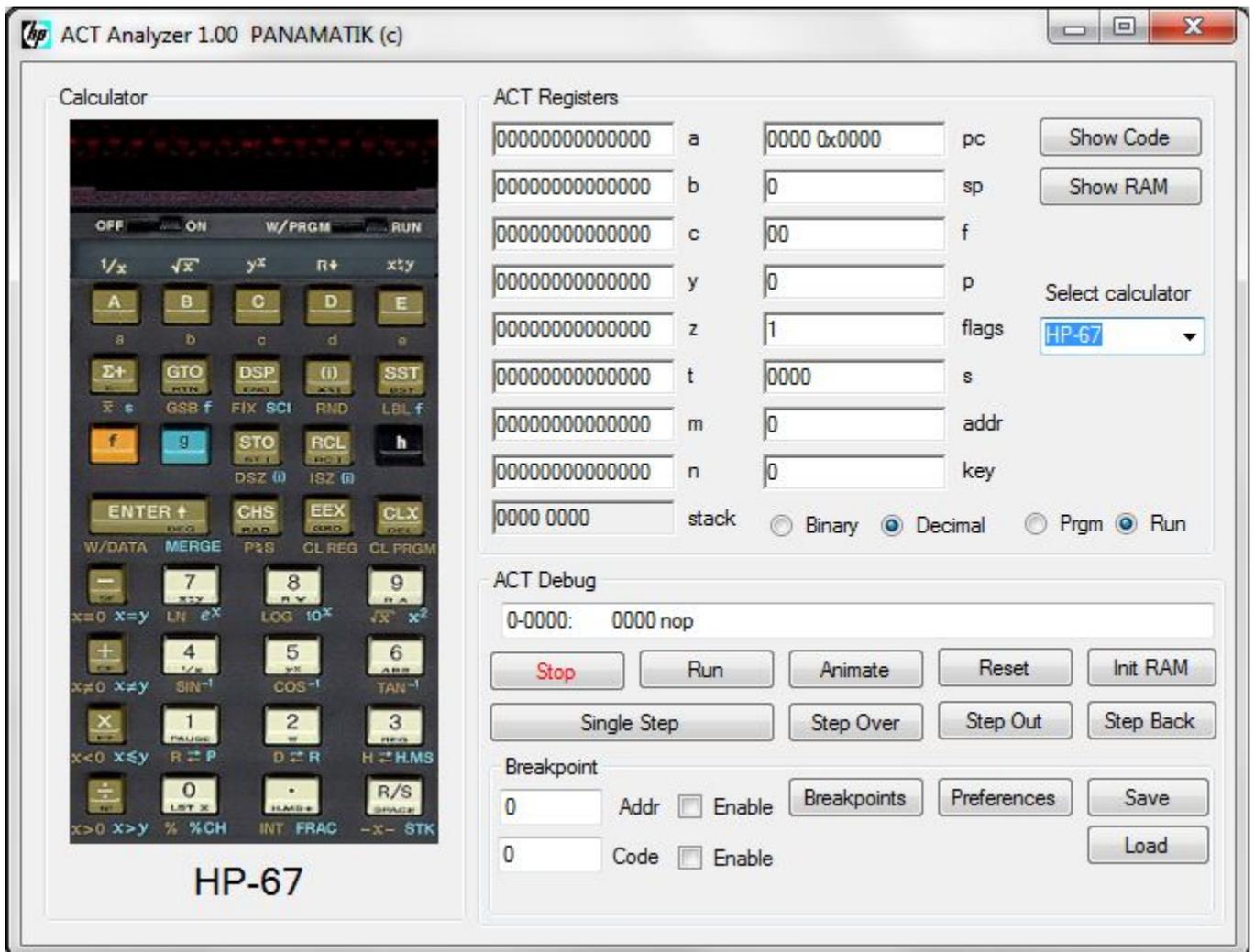
The application is very easy to use and pops up as a dialog when started. It is running under Windows XP/7/Vista/8/10 using the MicroSoft .NET framework version 3.5. This framework is preinstalled in Windows 7 and later, you have to download and install the framework when using the Windows XP operating system.

This manual revision describes the „ACT Analyzer“ Program version 1.02

# ACT Analyzer

## 2. The Analyzer

When starting the ACT Analyzer application you will discover on the left side of the screen an image of the actually selected HP calculator and many buttons and entry fields on the right. These are the internals of the calculator, the ACT registers. With the corresponding buttons you can single step or run through the ROM code of the calculator.



The ACT register description is far too extensive to be described here. The register usage differs from calculator to calculator and it is an adventure to understand how the calculators really work. But you can dive into this adventure when you enter key strokes and see how the calculator performs arithmetic or trigonometric functions step by step until it returns to the loop where it is waiting for another key stroke.

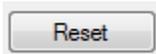
There are three possible states: Stopped, Running and Animate. All registers can be manipulated by entering new values, when the calculator is stopped. When the calculator is running it is executing all functions in real time. In Stop mode it waits for the next button, which can be a „Single Step“ or it performs a complete Subroutine with „Step Over“ or the remaining part of a subroutine until a

## ACT Analyzer

return instruction will be found with „Step Out“. „Animate“ differs from „Run“ in that way, that after each instruction the registers will be displayed. This is very slow compared to „Run“ mode, but shows very nicely how the register contents changes with each instruction.

The text field above the buttons shows the next instruction which will be executed, it contains the address, the instruction code and, if present, also your comments. After executing a „Single Step“ or a subroutine, the register contents are displayed and if some of them has changed they are displayed in **red text**. Registers that didn't have changed are displayed in black. Thus you can easily see what's going on.

„Reset“



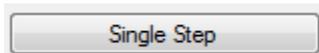
The „Reset“ button stops the calculator and sets the program counter pc to 0000. Here you can start your investigation and see how the initialization routine of the calculator is performed.

„Init RAM“



This will reset the Continuous Memory of the calculator to all FFs (all bits set), or the value you have chosen in the preferences. This simulates the calculator state as if a new battery is inserted and some of the models will show „Pr Error“, indicating that the „Continuous Memory“ contents is lost.

„Single Step“



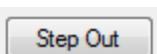
Pressing this button will execute a single HP machine instruction. You will see the new register values if any of them have changed. The program counter pc increments by one or jumps to a new location.

„Step Over“



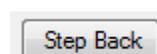
Executes the complete subroutine if the program counter is located at a JSB instruction, otherwise a normal single step is executed like pressing the „Single Step“ button. The „Local Cycles“ counter is reset before the JSB instruction is executed. If the program counter does not point to a JSB instruction a normal Single Step is performed.

„Step Out“



Runs the HP machine code within a subroutine until a RETURN instruction is executed. This will stop execution after at the end of the subroutine. The „Local Cycles“ counter is reset before execution.

„Step Back“



This special button will execute the instructions backwards. Up to 100 previous instructions can be executed backwards. If you go back some steps and then run the program from here it will execute still the correct code. This might be useful, if you want to see which register contents had changed and you didn't remember what it was showing before your „Single Step“. If you click „Run“ or use „Step Over“ and „Step Out“, the program counter will be set to the instruction before you started

## ACT Analyzer

the program.

„Breakpoint“

You can define a breakpoint for executing the HP machine code until a specific address is encountered. Enter a 12-bit breakpoint address as octal number. If enabled with the checkbox, the execution will stop when the program reaches the address when program counter and breakpoint address matches. There are more breakpoints in the „Breakpoints“ dialog.

„Code Breakpoint“

You also can define a code breakpoint. Enter an 10-bit instruction code as octal number. Then press „Run“. If the breakpoint is enabled the execution will stop as soon the executed instruction and the code breakpoint matches. There are more code breakpoints in the „Breakpoints“ dialog.

„Binary Decimal“  Binary  Decimal

This bit is also part of the ACT chip, it changes when the „binary“ or „decimal“ command is executed. It can be altered like the register contents can be changed.

„Prgm Run“  Prgm  Run

This changes from Program to Run mode and is the same as using the PRGM/RUN slider switch.

Some calculators have additional buttons like HP-55 ,  Timer , which has a timer mode. Or the HP-19C which has three print modes.

Man  Norm  
 Trace

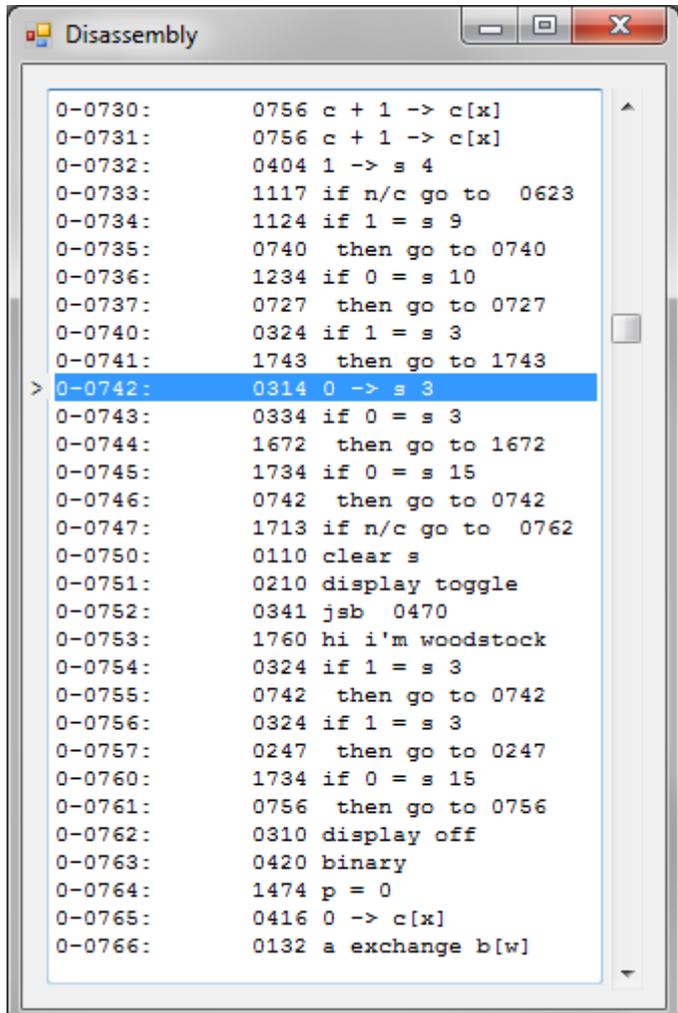
### 3. Disassembly window

To understand the program flow you need to see the disassembled instructions. This is done by clicking the „Show Code“ button. A window with the instructions around the actual program counter will appear. You can enlarge or reduce the window size to your needs. Whenever you stop the calculator or during a „Single Step“ or single step „Animation“ the actual program counter address, instruction code and mnemonics of the instructions will be shown.

The next instruction line, that will be executed is shown in blue color and marked with an arrow character.

The left column shows the address of the instruction in octal notation<sup>1</sup>. The preceding number 0 is the actual ROM Bank. Each ROM bank ranges from 0000-7777 octal, which is 4k bytes or 12-bit. There is a theoretical maximum of 16 banks possible, which would allow a maximum of 32k ROM code. The HP-34C, the biggest model in our selection, however uses „only“ 8k.

The next column shows the 10-bit instruction code in octal notation. Notice, that there are no digits above 7 used. The right side of the window shows the mnemonics of the instructions. What you see in the window is a text file named „rom\_25.dis“ for the HP-25 calculator. Every calculator has an associated text file, which you can edit with a text editor if you like. You can add your remarks in each line and they will be shown the next time in the application. But beware to remove or add lines when using an external editor, the executed instructions will not match any longer the text lines. Better use the comment entry, which is described next, which prohibits to insert or delete lines.



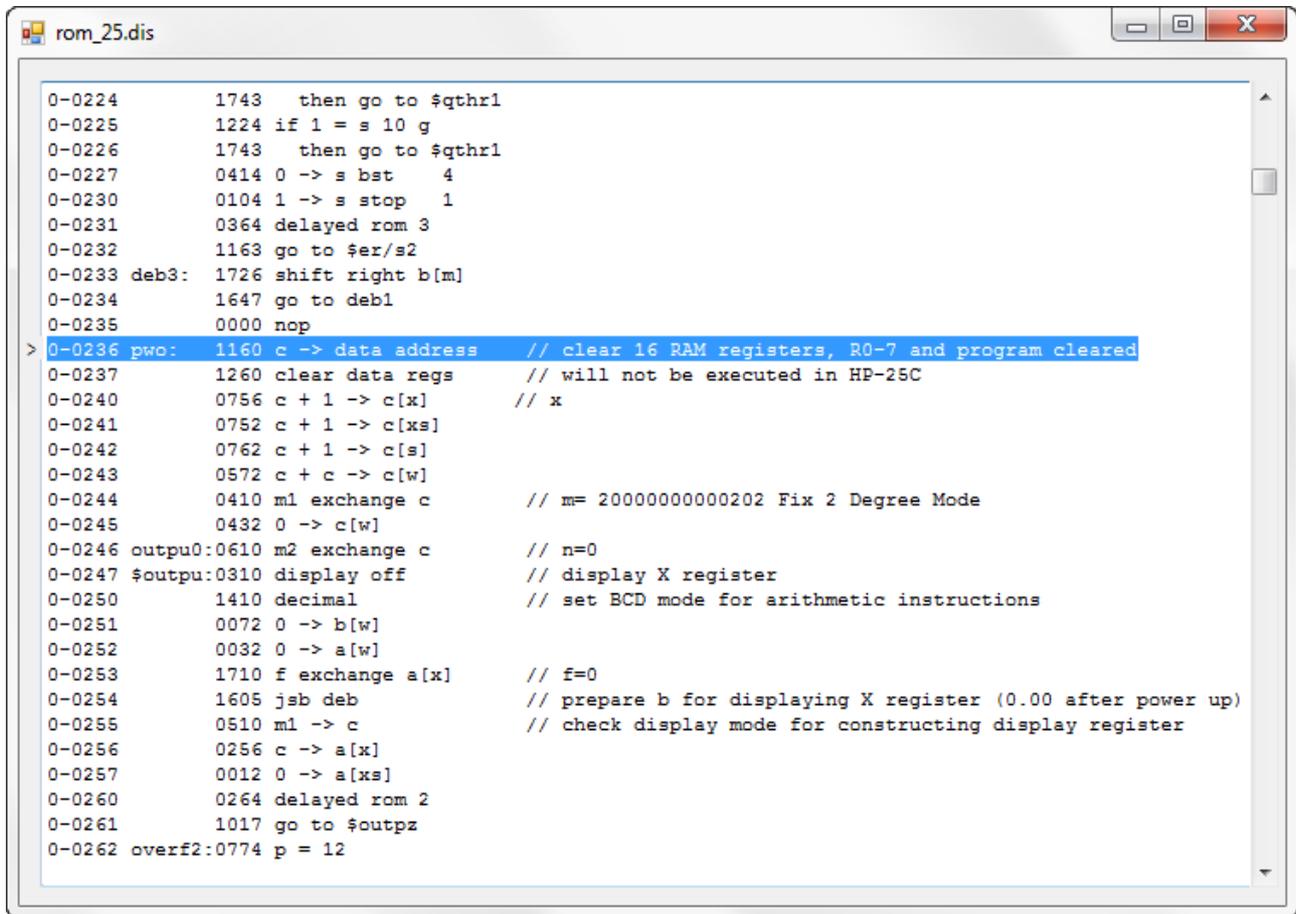
<sup>1</sup> Alternatively hexadecimal notation of the address and operation codes is possible if selected in the preferences.

#### 3.1. Comment Entry

When you are analyzing the calculator firmware you want to enter your own line comments. This is easily done just by entering your text behind each line. It is not possible to overwrite or remove the address and opcode, but any text at the right can be entered. You can even remove the Mnemonic code and replace it by your own text, but this is usually not advisable. Between the address and the

## ACT Analyzer

opcode there is space for program labels. Some of the calculator ROM files contain already labels, but others, which are not yet analyzed, don't. You can also enter your own labels here, as long as the checkbox „Show Program Flow“ is not activated.



```
rom_25.dis
0-0224      1743  then go to $qthrl
0-0225      1224  if 1 = s 10 g
0-0226      1743  then go to $qthrl
0-0227      0414  0 -> s bst    4
0-0230      0104  1 -> s stop   1
0-0231      0364  delayed rom 3
0-0232      1163  go to $er/s2
0-0233 deb3: 1726  shift right b[m]
0-0234      1647  go to deb1
0-0235      0000  nop
> 0-0236 pwo: 1160  c -> data address // clear 16 RAM registers, R0-7 and program cleared
0-0237      1260  clear data regs // will not be executed in HP-25C
0-0240      0756  c + 1 -> c[x] // x
0-0241      0752  c + 1 -> c[xs]
0-0242      0762  c + 1 -> c[s]
0-0243      0572  c + c -> c[w]
0-0244      0410  m1 exchange c // m= 20000000000202 Fix 2 Degree Mode
0-0245      0432  0 -> c[w]
0-0246 outpu0:0610 m2 exchange c // n=0
0-0247 $outpu:0310 display off // display X register
0-0250      1410  decimal // set BCD mode for arithmetic instructions
0-0251      0072  0 -> b[w]
0-0252      0032  0 -> a[w]
0-0253      1710  f exchange a[x] // f=0
0-0254      1605  jsb deb // prepare b for displaying X register (0.00 after power up)
0-0255      0510  m1 -> c // check display mode for constructing display register
0-0256      0256  c -> a[x]
0-0257      0012  0 -> a[xs]
0-0260      0264  delayed rom 2
0-0261      1017  go to $outpz
0-0262 overf2:0774 p = 12
```

Example of HP-25 code with comments entered.

If you close the window you will be asked, whether the comments should be saved to file. This will overwrite your original .dis file for this calculator. If you want to keep the original you should make a copy before or download it again from PANAMATIKs website.

### 3.2. Show Program Flow

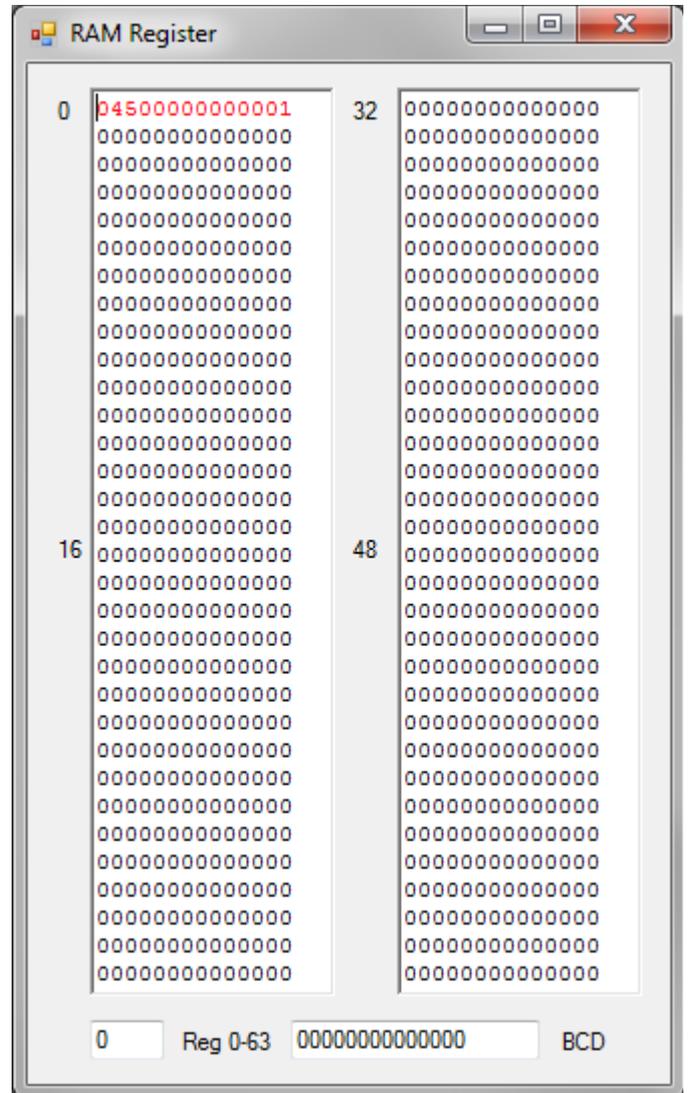
If you have activated the „Show Program Flow“ checkbox in the „Preferences“ dialog and you run the calculator code, either by „Single Step“ or „Run“, a number is shown between the address and opcode in the label area, where normally the program labels are shown. This number shows you how often this line was executed. It gives you information about which lines are used by the program or how often a loop is executed. If a line is executed more than 10000 times „>9999“ will be shown. This is normally the case in the „wait for keyboard entry“ loop.

## 4. RAM registers

Some models have additional RAM, which in fact is true for all models except HP-21 and HP-01. You can show the RAM contents by pressing the „Show RAM“ button. A window with 64 registers, the maximum amount which only the HP-34 and HP-67 can provide, are shown. This window shows the actual register contents. For some models this is the „Continuous Memory“, for others it is just the volatile registers. Please note, that the HP-25 has only 16 registers, only the left upper part of the window will change if you store a 49 step program or store variables in the registers.

Whenever a register value has changed since the last step of the program it will be marked in red color. This makes it more easy to follow any changes.

You can also manipulate the register contents by entering a new value. Enlarge the window at the bottom and you will see the entry fields. First chose the register number then enter its value.



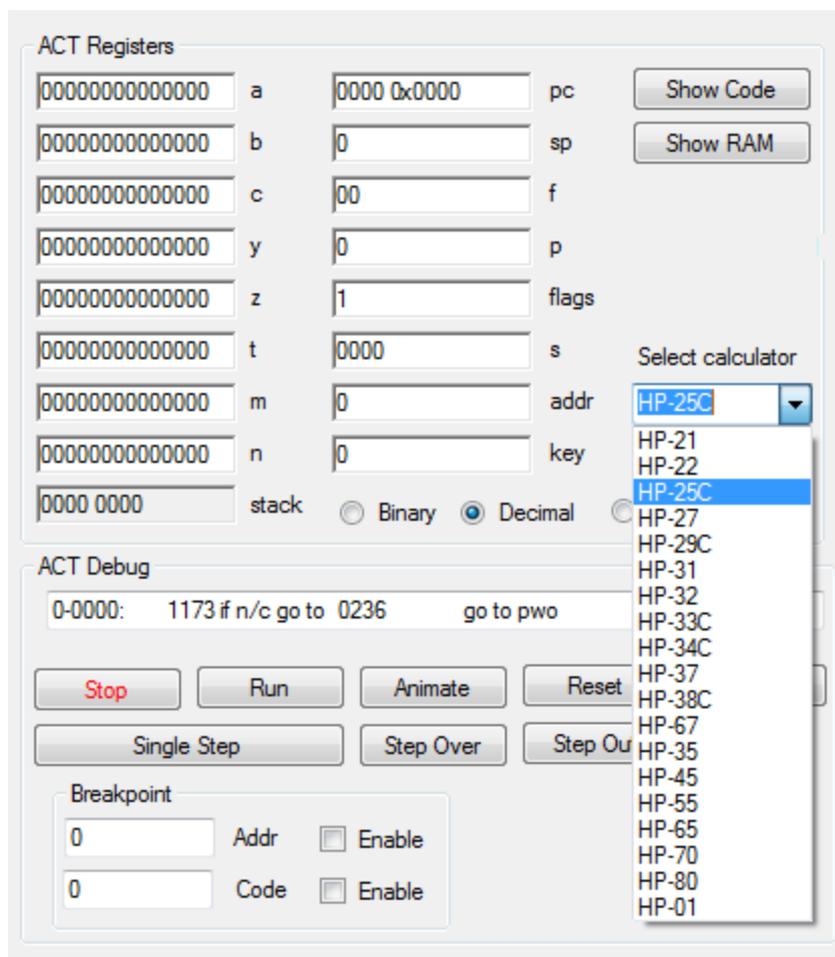
## ACT Analyzer

### 5. Select Calculator

You can select nearly any HP LED calculator from a combobox and this calculator will appear on your screen, ready to start.

By using the „Run“ button or „Single Step“ the original HP machine code will be emulated by the Analyzer. Besides the well known Woodstock and Spice models also the Classics like HP-35 HP-55, HP-65 and the HP-19C printing calculator, and even the famous HP-01 watch calculator, can be selected.

After starting the application, your preferred calculator, as defined in the preferences file, is shown. If you click on the „Select calculator“ comboBox, you can select any other calculator. After



selection, the calculator appears on the screen and is reset and normally stopped. Press the „Run“ button to see how the calculator starts and how it shows something in the display. This startup is equivalent as if you inserted new batteries. All previous data is lost. There is no Continuous Memory.

The main idea behind the emulator is to give you the possibility to analyze the HP code of any Woodstock, Spice and Classic model and of course do calculations.

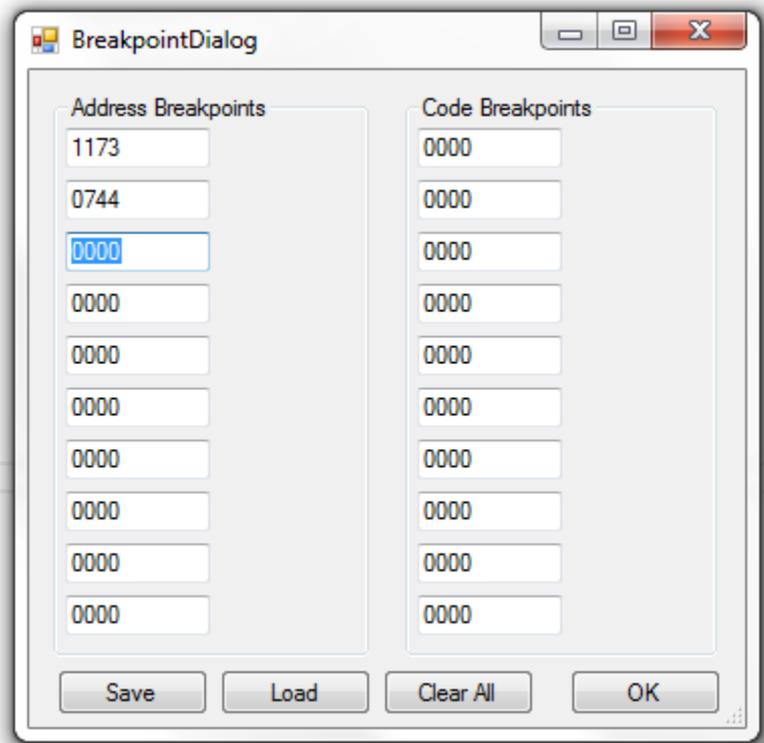
## 6. Breakpoints

The Analyzer has more capabilities. You can set more than one breakpoint by clicking the „Breakpoints“ button.

There are 10 additional „Address breakpoints“ and 10 „Code breakpoints“ available. Breakpoints with value 0000 are inactive. You can enter any octal 12-bit address value or 10-bit operation code to activate the breakpoints.

Your breakpoints can be saved to file and loaded again. This is useful if you want to continue your analyzing session the next time.

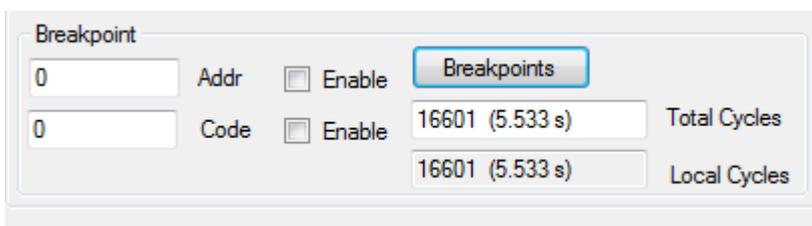
The additional breakpoints in this dialog are only enabled when the according breakpoint checkbox is active.



## 7. Cycles

When a program is running the number of machine cycles are counted and shown in the textboxes „Total Cycles“ and „Local Cycles“. Normally the HP calculators execute about 3000 instructions per second. The elapsed time will be shown also in the cycles textboxes.

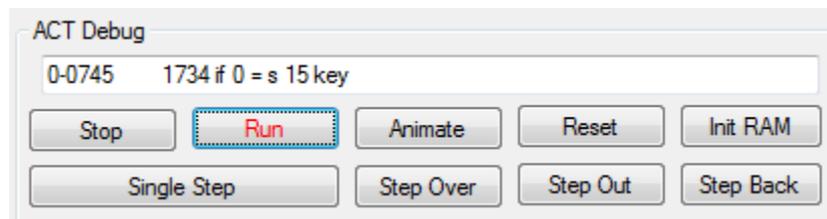
„Local Cycles“ counts the machine cycles within a subroutine when you use „Step Over“ or „Step Out“. The „Total cycles“ can be set to zero or to any number. It will also be set to zero if you reset the program by the „Reset“ button or select a new calculator.



## ACT Analyzer

### 8. The ACT Assembler

The disassembly window shows the original HP code of the actual calculator. And as described before you can add your own line comments to make the code more understandable for you and others. But lets go a step further. The ACT Analyzer offers you to modify the HP code and write your own ACT Assembler code. Of course you need to learn the instruction set of the old ACT processor, which is a BCD arithmetic oriented processor with special instructions to modify its 56-bit registers, before you can start this enterprise. I don't think that there is a real need for writing a complete new code, because the Woodstock and Spice calculators contain already highly optimized code, hand crafted by HP assembler specialists of their time. But still there were some minor bugs found in some of the calculators and if you like, you can fix them. And if you are more creative you nevertheless could add some new functionality.



To modify a code line you have two possibilities: First you can enter the assembler mnemonic in the entry field above the ACT Debug buttons. You can enter any instruction, according to the allowed syntax of the ACT or NUT processor. The mnemonic must be entered on the right side after the 4-digit octal or 3-digit hexadecimal operation code. You cannot alter the operation code directly. Also the address and the optional program label cannot be overwritten. If you finish the entry with „Return“ key or the „Tab“ key the entry will be interpreted and a new operation code is displayed.

If the mnemonic you entered is unknown, because you typed a wrong syntax or misspelled something, the opcode will not be changed, but the text „Syntax Error“ will be displayed. Press the „Stop“ Button to get the original line displayed again and correct the syntax until it will be accepted.

The second possibility is to enter the mnemonics anywhere in the disassembly window. Here you even can change the labels. This is more practical, because you can change any line you want independent of the line which is shown in the ACT Debug Entry field. If the syntax is not correct you will be shown '???' in the opcode field. Try again until a valid opcode is displayed.

When you close the disassembly window you will be asked, if you want to save your changes. Keep in mind, that you store a copy of the file somewhere to get back the original code.

There are two independent mnemonics you can use to enter your code. One is the more laborious ACT code, which was used at the time when the „Woodstock“ models were developed. Another is more compact and is similar to the NUT processor syntax, which was used when the HP-41C arrived.

An example for the ACT Syntax is:  $a + b \rightarrow c[w]$  where the NUT syntax is simply  $C=A+B$  W. Note that we use lower case letters for the ACT syntax and upper case for the NUT syntax by convention. In the Appendix the possible mnemonics are shown for both processors.

# ACT Analyzer

## 8.1. Labels

Each instruction can be preceded by a label. A label can be any word up to 6 characters and should be followed by a colon „:“. This allows you to enter „go to“ or „jsb“ instructions with labels instead of addresses. For example if you enter „go to pwo“ the label „pwo“ will be searched through the entire code and if found, its address will be calculated into the operation code of the jump instruction.

Be careful, when you alter the instructions, they will be saved whenever you confirm to save the code together with your comments. Keep a copy of the original code as a separate file or download it again from the [www.panamatik.de](http://www.panamatik.de) website.

Now have fun to write you own calculator code.

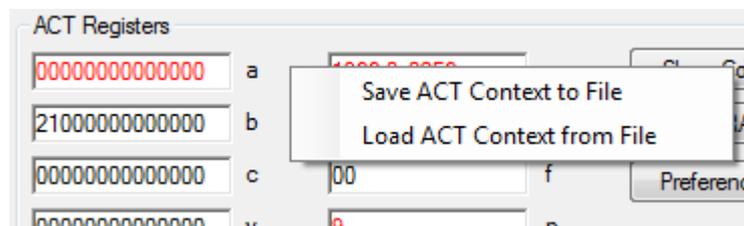
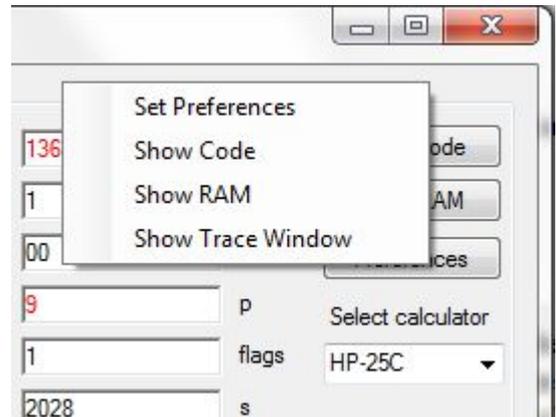
The instruction set of the ACT is partially documented in the „Museum of HP calculators“ website: <https://www.hpmuseum.org/techcpu.htm>.

The instruction set of the HP-01 calculator watch is well documented in the patent US4158285, which can be found in the internet.

## 9. Context Menus

Whenever you click the right mouse button, a context menu will appear. This menu allows you to perform some additional function, which may not have found a place for a buttons on the screen, like „Show Trace Window“.

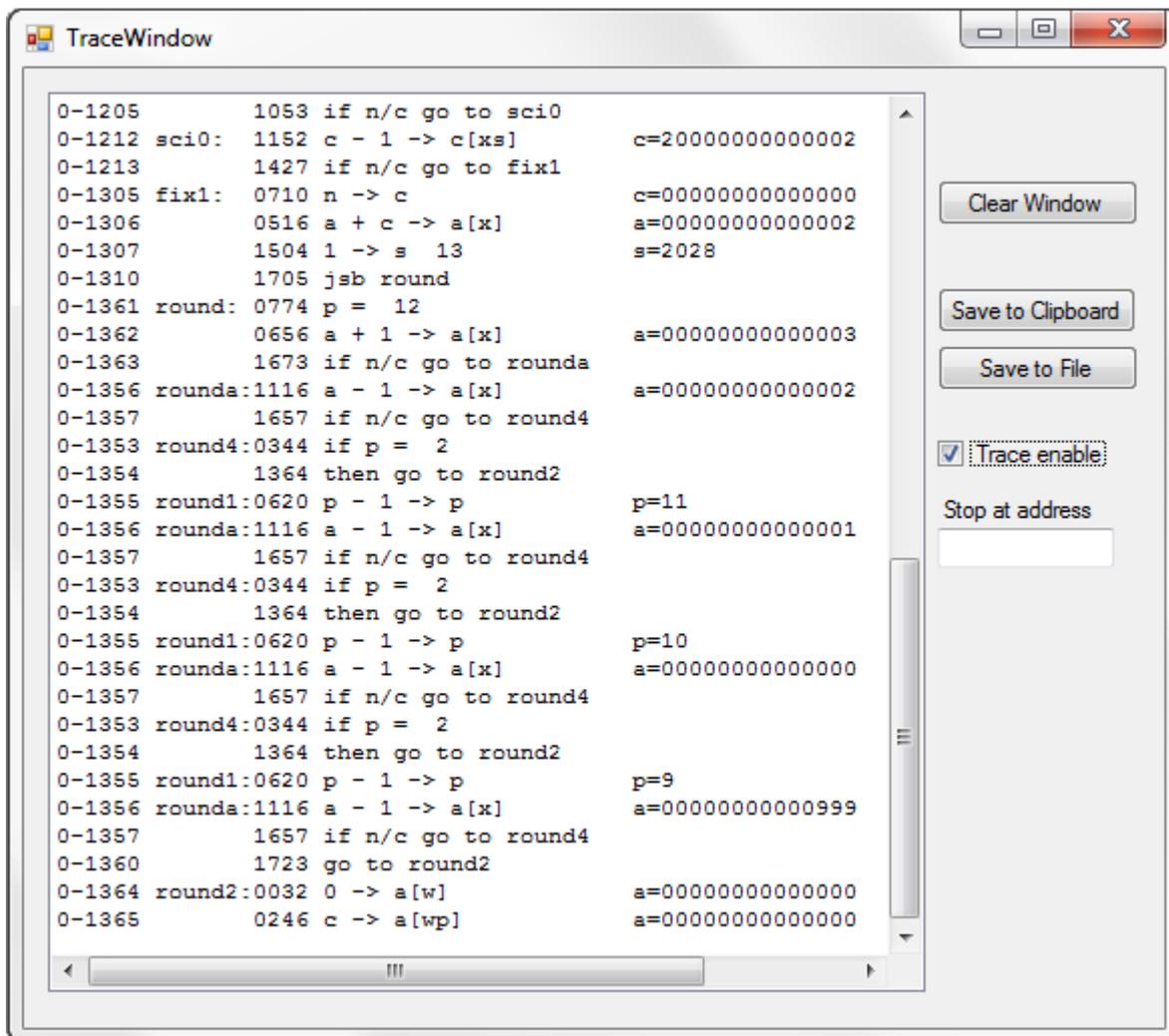
If you click within the „ACT Registers“ group box another context menu will pop up, where you can save the or load the ACT processor state.



## 10. Trace Window

For better understanding a program flow of a complete mathematical function you can trace all executed instructions. The „Trace Window“ can be opened by right mouse click from the context menu.

It shows alle executed instructions since you started the tracing with the checkbox.



As you can see, not only the instructions are shown but also the registers that may have been changed by the instruction. Most instructions change only one register, but there are some, which can affect two or more registers. For displaying all of them you can use the horizontal scroll bar at the bottom.

Normally after execution of a mathematical function the program will return to the entry loop waiting for another key stroke. You can enter the address of the entry loop to automatically stop the tracing after the function is finished to avoid endless recording of the same instructions in the waiting loop. Alternatively you can also stop the program with a normal breakpoint. This will also stop the recording, because the program is stopped.

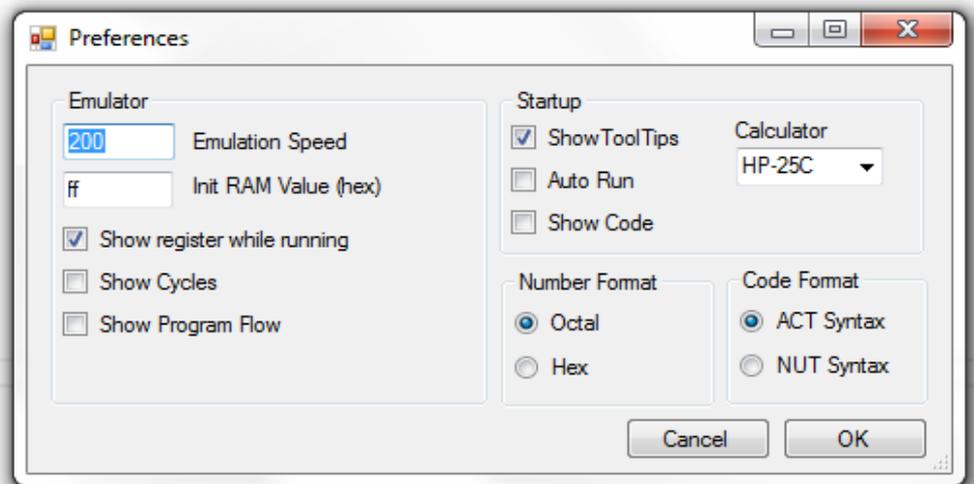
## ACT Analyzer

For Analyzing the recorded lines you can „Save“ them either to file or to the clipboard, where you can move it to any editor you want.

„Clear Window“ will clear the entire screen and start a new recording. The window contents is also cleared if you close and reopen the „Trace Window“.

## 11. Preferences

By clicking the „Preferences“ Button a dialog appears to enter some convenient personal likings.



The Emulation speed is a number between 1 and 1000. It determines how many machine cycles are executed before any display refresh is done and makes the calculations faster or slower. Normal values are between 50 and 500.

The Init Value will be used when you fill der RAM registers with a value by the „Init RAM“ button. Any value between 00 and ff can be entered as hexadecimal number.

„Show registers while running“ refreshes the registers while the calculator is running. Deactivating avoids continuously changing values on the screen.

„Show cycles“ adds two text fields, which show the elapsed calculator time in machine cycles and seconds. If you don't need it just deactivate.

„Show Program Flow“ shows in the disassembly window the number of times an instruction was executed. The numbers are show in each line after the address instead of the program labels. When running or single stepping through the program you can see how often the same program line was executed. After „Reset“ the count is set to 0. When unchecking this checkbox the labels will become visible again but the counters are still counting.

*More preferences affect the behaviour at startup.*

Each button has a small text to explain its functionality. You can activate it by „Show Tool Tips“

„Auto Run“ starts the selected calculator at startup and „0.00“ will automatically appear

„Show Code“ opens the disassembly window automatically.

## ACT Analyzer

With the „Calculator“ box you can select your preferred calculator, which appears at startup.

### *Number Format and Code Format*

Here you can choose the format of numbers, addresses and operation codes, which are displayed in the disassembly window, and also the format of Mnemonics.

The „Octal“ notation uses the digits from 0-7 to describe 3 binary bits, whereas the „Hex“ notation uses 10 digits 0-9 and 6 characters A-F to describe 4 binary bits. This notation is more compact.

A typical line in octal mode looks like this

```
0-0013 LABEL: 1566 if c[m] # 0
```

The same line in hex notation would look like

```
000B: LABEL: 376 if c[m] # 0
```

The normal code syntax is the ACT Syntax, but you can prefer the NUT CPU syntax, which is more compact and better readable. The NUT Syntax can only be used for the „Woodstock“, „Spice“ models and the HP-67, which contain the ACT processor. The Classic calculators and HP-01 always use their specific Syntax.

When you press the „OK“ button the preferences are saved to file „ACTAnalyzer.prefs, which is automatically loaded at startup. If you close the dialog without saving, your entries are discarded. If no preference file is present at startup, default values are used.

### **11.1 Save**

If you want to save your session use this button and the complete processor context, all ACT registers and all RAM registers, are saved to a file of your choice. You can load this file later and the calculator code will proceed exactly from this point.

### **11.2. Load**

When you have saved your code analyzing session you can load one of the files to proceed where you have stopped perhaps the day before.

# ACT Analyzer

## HP-65

Calculator

ACT Registers

00000000000999	a	0766 0x01F6	pc	Show Code
02009999999999	b	0	sp	Show RAM
00000000000221	c	04	f	
00000000000000	y	1	p	
00000000000000	z	37	flags	
00000000000000	t	0000	s	Select calculator
00000000000000	m	9	addr	HP-65
00000000000000	n	0	key	
4177 0000	stack			

Binary  Decimal  Prgm  Run

ACT Debug

0-0766: 0542 c - 1 -> c[p]

Stop Run Animate Reset Init RAM

Single Step Step Over Step Out

Breakpoint

0	Addr	<input type="checkbox"/> Enable
0	Code	<input type="checkbox"/> Enable

The HP-65 was the first programmable HP calculator. It uses an instruction set, which is related to the Classic ACT processor, but uses special pointer instructions for handling the program memory. Have fun to see how immensely complicated the firmware does its job. Of course there is no Card Reader functionality possible with this virtual calculator.

## ACT Analyzer

### HP-01

This is the first HP-01 emulator ever with debugging capability. See how the Code is executed in single step. Also the Time and Date and Stopwatch will be simulated but without providing accuracy.

The screenshot displays the ACT Analyzer software interface. On the left is a virtual HP-01 calculator with a digital display showing '11:54 04'. Below the calculator are fields for 'HP-01 registers' including 'dsp', 'date/time', 'alarm', 'stopwatch', and 'HP-01 flags'. The 'date/time' field shows '00000000115404'. In the center, 'ACT Registers' are shown with values for 'a' (0141 0x0061) and 'b' (1). Below that is an 'ACT Debug' section with a 'Stop' button. On the right, a 'Disassembly' window is open, showing a list of instructions with addresses and mnemonics. The instruction at address 0-0141 is highlighted: '1534 DSSCWP'. Other instructions include GOYES, GOTO, ENSCWP, GONC, SLEEP, GOKEYS, and SWSTOP.

There are many new instructions and registers, especially invented for the HP-01 hardware. Examples are SWSTOP, which stops the stopwatch counter registers, or SW- which makes the stopwatch counting backwards, when started, or ALTOG which toggles the Alarm active bit, or BLINK which makes the display blinking. These special instructions, which can be found only in the HP-01 firmware, interact with the surrounding hardware registers and gates. The processor is still in many ways related to the Woodstock ACT, but has only 48-bit instead of 56-bit registers and there are many other changes in detail. But it is still a BCD architecture with exponent arithmetics.

The HP-01 code for emulation is the same as the published prototype code in the patent US4158285. It has some minor differences to the original HP-01 code and has no percent button. The patent can give you valuable information about the firmware and hardware of the HP-01 wristwatch calculator (the first smartwatch of the 70s).

# ACT Analyzer

## HP-19C

Finally also the HP-19C printing calculator can be emulated, including its printer function. This was not easy to implement. The HP engineers used a lot of tricks in the hardware and software for using the standard ACT chip controlling the keyboard and printer switch and printer home contact by multiplexing these inputs. Emulating the original HP-19C code needed a lot of research in the firmware and in the schematics for getting the right behaviour of the printer and interfacing with the peripheral components. Also the PIK (Printer Interface Keyboard) chip had to be analyzed to achieve the perfect emulation. Explore the HP-19C machine code if you like. Many comments are already written in the code lines and they can be completed by you.

The printer output is identical to the original calculator, except that the paper is only for 5 printer lines.

# Appendix A

## **Files**

The ACT Analyzer application uses some files, which must be located in the same directory than the .exe application.

### *Application file*

The executable is named ACTAnalyzer.exe. There is no installation necessary. Just start the application from its directory.

### *Bitmap Files*

Each calculator has its own .bmp image file, which is loaded when the calculator is selected: hp21.bmp hp22.bmp hp25.bmp etc. If the file is not present the application will show the HP-25 bitmap, although the buttons will react differently.

### *Disassembly files*

These files with ending .dis contain the machine code for each calculator. They are essential to emulate the calculator and show and execute its machine code. You can edit these files to enter your comments, but it is not allowed to insert or delete lines. If you are very experienced and change the machine operation codes you can change the calculators behaviour. For adding your comments use the „Show Code“ window. It is save because you cannot unintentionally delete lines or change the code.

### *Preferences*

This small file ACTAnalyzer.prefs contains your last saved preferences. If it is not present default values are used.

### *Breakpoints*

When you download the ACTAnalyzer, there are no predefined breakpoint files. However you can save your own files with your file names. The file extension is normally .bkp. Don't confuse the files with backup files, which sometimes use the same extension.

## ACT Analyzer

### Appendix B

#### **ACT and NUT Assembler Syntax**

for HP-21/22/25/25C/27/29C 31E/32E/33E/33C/34C/37E/38E/38C HP-67 HP-19C

#### Instructions without argument

Code	ACT Syntax	NUT Syntax
0000	"nop",	"NOP",
0010	"clear regs",	"REG=0",
0110	"clear s",	"S=0",
0210	"display toggle",	"DISTOG",
0310	"display off",	"DISOFF",
0410	"m exch c",	"C<>M",
0510	"m -> c",	"C=M",
0610	"n exch c",	"C<>N",
0710	"n -> c",	"C=N",
1010	"stack -> a",	"A=STK",
1110	"down rotate",	"ROTDN",
1210	"y -> a",	"A=Y",
1310	"c -> stack",	"STK=C",
1410	"decimal",	"SETDEC",
1610	"f -> a", "f -> a[x]",	"A=F",
1710	"f exch a", "f exch a[x]",	"A<>F",
0020	"keys -> rom address",	"GTOKEY",
0120	"keys -> a",	"A=KEY",
0220	"a -> rom address",	"ROM=A",
0320	"reset twf",	"RESTWF",
0420	"binary",	"SETHEX",
0520	"rotate a left",	"RAL",
0620	"p - 1 -> p",	"-P",
0720	"p + 1 -> p",	"+P",
1020	"return",	"RTN",
1060	"bank switch",	"BANKSW",
0070	"c -> addr",	"ADDR=C",
1260	"clear data registers",	"CLREGS",
1360	"c -> data",	"DATA=C",
1460	"rom checksum",	"CHKSUM",
1760	"hi i'm woodstock"	"WOODSTOCK"

## ACT Analyzer

### Instructions with argument nnnn = 0-15

Code	ACT Syntax	NUT Syntax
nnnn000100	"1 -> s n",	"SF n",
nnnn001100	"0 -> s n",	"CF n",
nnnn010100	"if 1 = s n",	"?FS n",
nnnn011000	"load constant n",	"LC n",
nnnn011100	"if 0 = s n",	"?FC n",
nnnn100000	"select rom n",	"SELROM n",
nnnn100100	"if p = n",	"?PT= n",
nnnn101000	"c -> data register n",	"REG=C n",
nnnn101100	"if p # n",	"?PT# n",
nnnn110100	"delayed rom n",	"DELROM n",
nnnn111000	"data register -> c n",	"C=REG n",
nnnn111100	"p = n"	"P= n"

### Jump instructions

Code	ACT Syntax	NUT Syntax
xxxxxxxx01	"jsb"	"XSUB"
xxxxxxxx11	"if n/c go to" „go to“ „goto“	"NCGO"
nnnnnnnnnn	"then go to"	"CGO"

## ACT Analyzer

Arithmetic instructions with 3-bit field specifier "p", "wp", "xs", "x", "s", "m", "w", "ms"

Code	ACT Syntax	NUT Syntax
00000 fff 10	"0 -> a",	"A=0",
00001 fff 10	"0 -> b",	"B=0",
00010 fff 10	"a exchange b",	"A<>B",
00010 fff 10	"a -> b",	"B=A",
00011 fff 10	"a exchange c",	"A<>C",
00100 fff 10	"c -> a",	"A=C",
00101 fff 10	"b -> c",	"C=B",
00110 fff 10	"b exchange c",	"B<>C",
00111 fff 10	"0 -> c",	"C=0",
01000 fff 10	"a + b -> a",	"A=A+B",
01001 fff 10	"a + c -> a",	"A=A+C",
01010 fff 10	"c + c -> c",	"C=C+C",
01011 fff 10	"a + c -> c",	"C=A+C",
01100 fff 10	"a + 1 -> a",	"A=A+1",
01101 fff 10	"shift left a",	"ASL",
01110 fff 10	"c + 1 -> c",	"C=C+1",
01111 fff 10	"a - b -> a",	"A=A-B",
10001 fff 10	"a - c -> c",	"C=A-C",
10010 fff 10	"a - 1 -> a",	"A=A-1",
10011 fff 10	"c - 1 -> c",	"C=C-1",
10100 fff 10	"0 - c -> c",	"C=-C",
10101 fff 10	"0 - c - 1 -> c",	"C=-C-1",
10110 fff 10	"if b = 0",	"?B=0",
10111 fff 10	"if c = 0",	"?C=0",
11000 fff 10	"if a >= c",	"?A>=C",
11001 fff 10	"if a >= b",	"?A>=B",
11010 fff 10	"if a # 0",	"?A#0",
11011 fff 10	"if c # 0",	"?C#0",
11100 fff 10	"a - c -> a",	"A=A-C",
11101 fff 10	"shift right a",	"SRA",
11110 fff 10	"shift right b",	"SRB",
11111 fff 10	"shift right c",	"SRC",

## ACT Analyzer

### HP-19C Printer Interface Keyboard instructions

Code	ACT Syntax	NUT Syntax
1120	"pik home?",	"PIKHOME?",
1220	"pik cr?",	"PIKCR?",
1320	"pik keys?",	"PIKKEYS?",
1720	"pik print3",	"PIKPRINT3",
1660	"pik print6",	"PIKPRINT6",

### HP-67 Card Reader Controller instructions

Code	ACT Syntax	NUT Syntax
0100	"crc motor on?",	"MOTON?",
1700	"crc wr prot?",	"WRPROT?",
0560	"crc card in?",	"CARDIN?",
0300	"crc test f1",	"CRCF1?",
0500	"crc test f2",	"CRCF2?",
0700	"crc test f3",	"CRCF3?",
1100	"crc test f4",	"CRCF4?",
1200	"crc set f0",	"CRCSF0",
1400	"crc set f1",	"CRCSF1",
0600	"crc set f3",	"CRCSF3",
1300	"crc clear f0",	"CRCCF0",
1500	"crc clear f1",	"CRCCF1",
0260	"crc motor on",	"MOTON",
0360	"crc motor off",	"MOTOFF",
0660	"crc test prot",	"TESTPROT"
0760	"crc clear write mode",	"CLWRMODE",
1000	"rom address -> buffer",	"BUF=ROM",

## ACT Analyzer

### Classic Assembler Syntax

for HP-35/45/55/65/70/80

Instructions without argument

Code	Classic Syntax	
	"nop",	
	"display toggle",	
	"m exch c",	
	"c -> stack",	
	"stack -> a",	
	"display off",	
	"m -> c",	
	"down rotate",	
	"clear regs",	
	"p - 1 -> p",	
	"p + 1 -> p",	
	"return",	
	"clear s",	
	"keys -> rom address",	
	"c -> addr",	
	"c -> data",	
	"data register -> c",	

Instructions with argument n = 0-15

Code	Classic Syntax	
	"1 -> s n",	
	"p = n",	
	"select rom n",	
	"if 0 = s n",	
	"load constant n",	
	"0 -> f n",	
	"1 -> f n",	
	"0 -> s n",	
	"if p # n",	
	"delayed select rom n",	
	"delayed select group n",	

## ACT Analyzer

### Arithmetik instructions

Code	Classic Syntax	
	"if b = 0",	
	"0 -> b",	
	"if a >= c",	
	"if c # 0",	
	"b -> c",	
	"0 - c -> c",	
	"0 -> c",	
	"0 - c - 1 -> c",	
	"shift left a",	
	"a -> b",	
	"a - c -> c",	
	"c - 1 -> c",	
	"c -> a",	
	"if c = 0",	
	"a + c -> c",	
	"c + 1 -> c",	
	"if a >= b",	
	"b exchange c",	
	"shift right c",	
	"if a # 0",	
	"shift right b",	
	"c + c -> c",	
	"shift right a",	
	"0 -> a",	
	"a - b -> a",	
	"a exchange b",	
	"a - c -> a",	
	"a - 1 -> a",	
	"a + b -> a",	
	"a exchange c",	
	"a + c -> a",	
	"a + 1 -> a",	

## ACT Analyzer

### Jump instructions

Code	Classic Syntax	
xxxxxxxx01	"jsb"	
xxxxxxxx11	"if n/c go to" „go to“	
nnnnnnnnnn	"then go to"	

### HP-65 instructions

Code	HP-65 Syntax	
	"memory full",	
	"buffer -> romaddr",	
	"memory insert",	
	"mark and search",	
	"memory delete",	
	"rom address -> buffer",	
	"search for label",	
	"pointer advance",	
	"memory initialize",	

## ACT Analyzer

### HP-01 Assembler Syntax

#### Instructions without argument

Code	HP-01 Syntax	Operation
0000	"NOP",	no operation
	"S1-7=0",	clear status bits 0-7
	"S8-15=0",	clear status bits 8-15
	"P=P+1",	increment P register
	"P=P-1",	decrement P register
	"SLEEP",	enter sleep mode
	"CLRREG",	clear registers
	"CD EX",	exchange C and D
	"C=M",	
	"C=D",	
	"M=C",	
	"DSPON",	Display on
	"DSPOFF",	Display off
	"BLINK",	Display is blinking
	"A(P)=F",	
	"F=A(P)",	
	"ENSCWP",	
	"DSSCWP",	
	"A=DSP",	
	"A=CL",	
	"CLRS=A",	
	"CL=A",	
	"DSP=CL",	Show Clock
	"AL=A",	
	"A=SW",	
	"SW=A",	
	"DSP=SW",	Show Stopwatch
	"SW-",	Stopwatch backward
	"DSP=AL",	show Alarm
	"SW+",	Stopwatch forward
	"DSP=A",	Show A register
	"A=AL",	
	"SWSTRT",	Stopwatch Start
	"SWSTOP",	Stopwatch Stop
	"ALTOG",	Alarm Toggle

## ACT Analyzer

### Instructions with argument n= 0-1 or n=0-15

Code	HP-01 Syntax	
	"P= n",	
	"? Sn = 0",	
	"A(P)= n",	
	"GOROM n",	
	"? P# n",	
	"GOROMD n",	
	"S0= 0", "S0= 1",	
	"S1= 0", "S1= 1",	
	"S2= 0", "S2= 1",	
	"S3= 0", "S3= 1",	
	"S4= 0", "S4= 1",	
	"S5= 0", "S5= 1",	
	"S6= 0", "S6= 1",	
	"S7= 0", "S7= 1",	
	"S8= 0", "S8= 1",	
	"S9= 0", "S9= 1",	
	"S10= 0", "S10= 1",	
	"S11= 0", "S11= 1",	
	"S12= 0", "S12= 1",	
	"S13= 0", "S13= 1",	
	"S14= 0", "S14= 1",	
	"S15= 0", "S15= 1",	

### Jump instructions

Code	Classic Syntax	
	"GOSUB"	jump to subroutine
	"GONC" „GOTO“ „GOYES“	jump if not carry
	"GOTOX"	
	"GOKEYS"	jump by key
	"RETURN",	return from Subroutine

## ACT Analyzer

### Arithmetik instructions

Code	HP-01 Syntax	
	"? C=0",	
	"C=0",	
	"C=C+1",	
	"C=C-1",	
	"? C#0",	
	"C=C+C",	
	"C=-C",	
	"C=-C-1",	
	"AC EX",	
	"A=C",	
	"A=A+C",	
	"A=A-C",	
	"? A>=C",	
	"? A#0",	
	"C=A+C",	
	"C=A-C",	
	"? A>=B",	
	"A=0",	
	"A=A+1",	
	"A=A-1",	
	"B=A",	
	"AB EX",	
	"A=A+B",	
	"A=A-B",	
	"B=0",	
	"? B=0",	
	"C=B",	
	"BC EX",	
	"A SR",	
	"A SL",	
	"B SR",	
	"C SR",	