GO9800 – Series 9800 Emulator

Release 1.60

User Manual

Contents

0. Preface

GO9800 is intended as a complete emulation of the hardware of the famous first family of Hewlett-Packard desktop calculators, i.e. HP9810A, HP9820A, HP9821A, and HP9830A/B.

The emulator is able to run the complete ROM-based firmware of the above models and the application programs written in the model specific programming language. Moreover the user interfaces (display, keyboard, printer), mass memory devices (magnetic card reader, tape drive, disc drive), and other external devices can be emulated to resemble the complete functionality of the original machine.

To enhance the realism of the emulation part of the outside appearance is graphically displayed and some of the mechanical sounds (beeper, cooling fan, drive motors, etc.) can be output via a sound card.

The emulator is completely written in Java 2 and platform independent.

1. General Features

The GO9800 kernel is an emulation of the CPU which was common to all calculators of the 9800-family. All CPU instructions are implemented, including I/O and floating point. The instruction set is briefly described in the U.S. patent 3,859,635. During development some of the descriptions turned out to be incomplete or incorrect. In such cases the behaviour of these instructions was modified to achieve an error free operation of the machine.

The CPU communicates with the 'I/O-register and gate interface' and the memory unit, consisting of read only (ROM) and read/write memory (RWM, nowadays called 'RAM'). Via the I/O-bus the IO-register is connected to all internal and external devices (display, keyboard, printer, etc.). The complete device I/O protocols are implemented, including service request (interrupt). The peripheral devices work in separate program threads, so that parallel and real time operation of the calculator mainframe and peripheral devices is possible. The machine firmware, originally residing in ROMs, was extracted from the original hardware and is executed without any modification by the CPU emulation. The ROM areas of the memory are write protected in the emulator and behave as the original. The complete memory equipment is configurable to resemble several memory expansion options. The presence of plug-in ROM blocks can also be configured and ROM blocks can be exchanged at runtime.

The emulator also contains a complete disassembler, which, when activated, disassembles each CPU instruction prior to execution, dumps the contents of the CPU registers A, B, and E, the IO register, various flags, and the pseudo registers AR1 and AR2, used for floating point operations. The disassembler can be activated either interactively or by setting breakpoints or watchpoints in the calculator memory. The disassembler output is displayed in a separate, scrollable window.

In a special trace mode, activated either manually in the disassembler window or by a breakpoint, the cpu instructions can be executed step by step or in a 'slow motion' with variable time intervals.

1.1 Machine Configuration Files

The emulated machines can be customised by means of a configuration file. The configuration file defines the layout and size of the calculator memory areas, type and position of ROM modules, and external devices which are connected to the machine. For one specific calculator model there can be several configuration files for different combinations of RWM, ROM, and peripheral devices.

The configuration file is a plain text file and contains one configuration item per line. Each configuration item consists of a type, name, address or select code fields, and an internal slot identification.

The configuration file may contain comment lines, which are not evaluated. Comment lines have to start with a semicolon.

Example:

```
; This is a comment line
```

Since release 1.50 the emulator loads the Java classes for each model, peripheral device, and device interface dynamically using the Reflection API. This has the advantage that additional model and device classes may be added without changing the source code of the emulator. The names of calculator models and devices must correspond to an existing Java class. If in a configuration file a model or device name is used for which no matching Java class is found an appropriate error message is created and the emulator is terminated.

1.1.1 Peripheral Device Configuration Files

Since release 1.50 each peripheral device has its own configuration file, in which the appropriate calculator interface, the possible select codes, and the suitable calculator models are defined.

The configuration file comprises the following configuration items:

```
Model <List of suitable calculator models>
Name <Java class name for the device (usually the HP-number)>
Title <Name which is displayed in the startup log>
Interface <Java class name for the appropriate interface (usally the HP-number)>
Selectcode <List of allowed select codes>
RWM <Optional starting address and length of extended memory>
Parameters <Optional device specific parameters>
```

The select code defined in a calculator configuration file are checked against the allowed select codes in the device configuration file.

Here is an example configuration file for the HP11305A disk controller with HP11273A interface containing 256 words of extended memory, connected to one HP9880A disk drive with two disks (Parameters 1 2), which may only be used in conjunction with the HP9830A:

```
Model HP9830A
Name HP11305A
Title MASS_MEMORY_CONTROLLER
Interface HP11273A
Selectcode 11
RWM 077000 000400
Parameters 1 2
```

1.1.2 Search Strategy for Configuration Files

Every configuration file is searched first in the current working directory (normally the directory where the emulator is started from).

If there is no customized configuration file found in this place, the emulator looks for the standard configuration file in the installation directory, where the file GO9800.jar is located.

If this is also not present, the configuration is loaded from the directory /config/ in the GO9800.jar itself.

If the configuration file not found in any of these locations the emulator is terminated.

1.1.3 Calculator Model Configuration

The first line of the configuration file must define the calculator model and an optional version. In this release the version information is used only for the HP9810A which exists in version 1 and 2. If there is only one version the version number has to be omitted.

Examples:

```
Model HP9830A
Model HP9810A 2
```

In the current release the following Java classes (calculator models) exist:

```
HP9810A
HP9810A2
HP9820A
HP9821A
HP9830A
```

The following lines define memory blocks of different type and sizes. There are two types of memory: RWM (Read-Write-Memory) and ROM (Read-Only-Memory). The areas where RWM and ROM are located are specific and fixed for each calculator model.

1.1.4 ROM configuration

The ROM areas may be divided in several sections, e.g. for build-in system ROM and additional ROM modules. Each ROM block has a 16bit start address and block size. Both values have to be given as octal values. Each ROM block has a name which references a file which contains the ROM data. At startup of the emulator the referenced ROM files are read into the calculator memory.

Examples:

```
ROM 016000 001000 HP9830A_System8 Block16
ROM 040000 000400 HP9830A_System9 Block40
ROM 020000 002000 HP11274B Slot1
ROM 022000 002000 HP11271B Slot2
```

A system ROM block has a specific start address and can only be used at this position. The block name has the form <Calculator Model>_System#. Each memory block (ROM or RWM) needs a unique block-id. The block-id for system ROM block is arbitrary but has to be unique amongst all memory blocks, conventionally the block-id contains the octal starting address of the block. For plug-in ROMs the block-id must be in the form Slot#, where # is the position of the calculators ROM slot. In the real calculator each slot corresponds to a fixed address range (e.g. in the HP9810A Slot1 uses addresses from 002000 to 003777 octal). In the emulator one can assign a different address range although this will probably lead to malfunction or hangup of the emulation.

Add-on ROM blocks may be positioned at different start addresses although not every position is useful. E.g. the HP9810A Mathematics ROM block may be plugged in every of the three ROM slots of the calculator but will only be functional in the first slot (address 002000). The block name is identical to the HP product number of the original ROM, e.g. HP11220A.

Since release 1.40 each ROM has an additional description file which contains the valid slots and/or address positions for the particular ROM and calculator. During startup each ROM in the calculator configuration is validated against the corresponding ROM description. If a conflict is detected, the cause is logged in the command console and the emulator startup is aborted. If a ROM is later plugged in manually the same check is performed and the ROM is activated only if no conflict is detected (see chapter 1.2).

The assigned address range for ROM blocks is marked read-only for the calculator CPU, so the contents may not be changed by any program.

1.1.5 Empty ROM Slots

If any ROM slot should be left empty at startup, then the corresponding configuration items have to be set to a dummy ROM block HP11XXXX, otherwise the related memory range will not be initialized correctly. E.g. if the ROM slots 2 and 3 of the HP9810A should be empty at startup, the configuration file must contain:

```
ROM 006000 002000 HP11XXXX Slot2
ROM 010000 002000 HP11XXXX Slot3
```

The HP11XXXX dummy ROM simply contains 0 (zero) values of each memory location.
See also chapter 1.2.

1.1.6 RWM configuration

Areas of Read-Write-Memory are defined by RWM blocks. A RWM block is defined similar to a ROM block with start address and block size (in octal values), a block name, and a slot-id. The block name is arbitrary and for information purpose only. RWM block names are typically System or HP product numbers of memory expansions. The slot-id is also arbitrary but has to be unique between all memory blocks (RWM and ROM). Memory blocks with the same block-id 'overwrite' each other in the internal configuration.

Examples:

```
RWM 001400 000400 HP9830A_System Block1
RWM 040400 007400 HP9830A_User Block40.4
RWM 050000 010000 HP11275F Block50
```

1.1.7 Peripheral Device Configuration

Peripheral devices which shall be attached to the calculator are defined by configuration items of type DEV. The name field contains the HP product number of the device (e.g. HP9862A for a plotter). The last entry of the configuration line is a optional select code. Some devices have fixed select code which can not be changed (e.g. the HP9862A has fixed select code 14). In such cases the select code entry is ignored. Other devices need a specific select code which has to be unique between all internal and external devices. Refer to the original documentation of the specific peripheral device and calculator for possible select codes.

Examples:

```
DEV HP9862A
DEV HP9865A 5
```

1.1.8 Breakpoint Configuration

The emulator contains a run-time disassembler for inspection and analysis of machine programs (appendix C). Normally the disassembler is inactive and can be enabled manually in the disassembler dialog window.

In the configuration file breakpoint addresses can be defined at machine instruction level. As soon as the CPU program counter reaches a breakpoint address the execution of the machine instructions is halted and the disassembler dialog window is automatically opened. The instruction at the breakpoint address is shown in the disassembler list but not yet executed. The execution can be continued by either using the single step mode or resuming to the normal run mode.

Breakpoint addresses have to be given as octal values from 0 to 77777. There can be a arbitrary number of breakpoints defined without significant performance drawback. Breakpoints are only evaluated when the associated memory address is accessed.

Example:

```
Breakpoint 05733
```

1.1.9 Watchpoint Configuration

In the configuration file memory watchpoint addresses can be defined. A watchpoint is a memory location which is checked every time it is accessed by a CPU instruction either in read or write mode. A watchpoint can be conditional or unconditional. A conditional watchpoint is further characterised by a watch condition and a test value. The content of the watched memory location is compared with the test value by a comparison operator. If the result is true, the watchpoint condition is met. In an unconditional watchpoint the memory content is ignored.

As soon as a CPU instruction accesses the memory location identified by the watchpoint address, the contents of this memory location is tested against the watchpoint condition. If the condition is met, the current instruction is finished, then the machine program execution is halted and the disassembler dialog window is automatically opened. The next CPU instruction is shown in the disassembler list but not yet executed. The execution can be continued by either using the single step mode or resuming to the normal run mode.

Watchpoint addresses have to be given as octal values from 0 to 77777. For conditional watchpoints a test value and a watch condition may be defined. The test value can be any unsigned 16 bit octal value between 0 and 177777. The watch condition is defined by one of the comparison operators (=, <, >).

There can be a arbitrary number of watchpoints defined without significant performance drawback. Watchpoints are only evaluated when the associated memory address is accessed.

Examples:

```
Watchpoint 01000 07777 =
Watchpoint 05600
```

1.1.10 Example Configurations

This is an example of a complete configuration file for the HP9830A calculator:

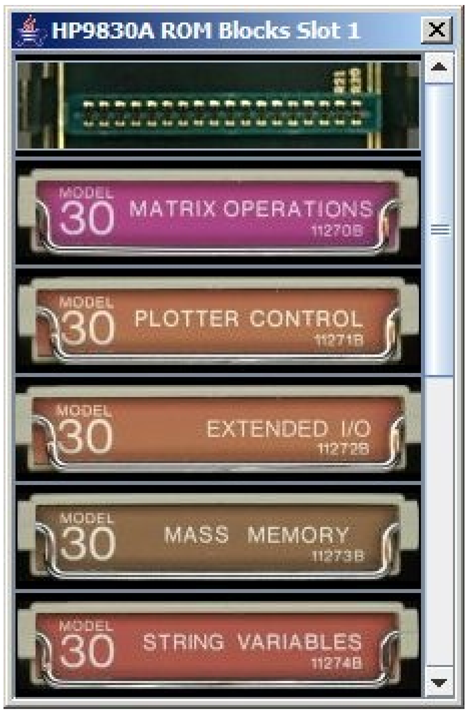
```
Model HP9830A
ROM 000000 001400 HP9830A_System1 Block0
ROM 002000 002000 HP9830A_System2 Block2
ROM 004000 002000 HP9830A_System3 Block4
ROM 006000 002000 HP9830A_System4 Block6
ROM 010000 002000 HP9830A_System5 Block10
ROM 012000 002000 HP9830A_System6 Block12
ROM 014000 002000 HP9830A_System7 Block14
ROM 016000 001000 HP9830A_System8 Block16
ROM 040000 000400 HP9830A_System9 Block40.0
RWM 001400 000400 HP9830A_System Block1
RWM 040400 007400 HP9830A_User Block40.4
RWM 050000 010000 HP11275F Block50
RWM 060000 016000 HP11276F Block60
ROM 017000 001000 INFOTEK_FB2 Int0
ROM 024000 002000 INFOTEK_FB1 Int1
ROM 030000 002000 HP11270B Int2
ROM 032000 002000 HP11272B Int3
ROM 036000 002000 HP11273B Slot1
ROM 034000 002000 HP11274B Slot2
ROM 020000 002000 HP11289B Slot3
ROM 022000 002000 HP11279B Slot4
ROM 026000 002000 HP11271B Slot5
DEV HP9860A
DEV HP9861A 8
DEV HP9862A
DEV HP9865A 5
DEV HP9866B 15
DEV HP9880B
DEV HP11202A 1
```

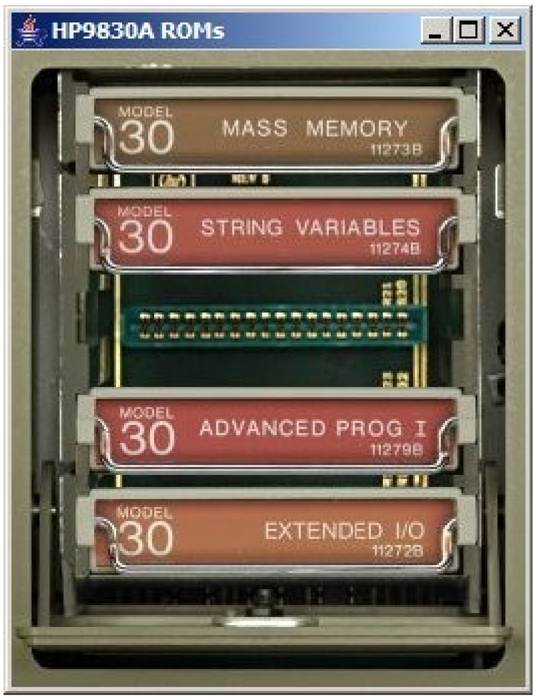
1.2 Exchanging of ROM blocks

Plug-in ROM blocks may be exchanged in all emulated calculators at run-time. To achieve this in the HP9810A or HP9820A click with the left mouse-button on the module to be exchanged (on the calculator left side above the LED display). Then a dialog window is opened in which all available plug-in ROMs are visible. Click on one of the symbolic ROMs to select it for use in the same slot as the previous. If no ROM has to be used (the slot should be empty) click on the first symbol showing the slot cover. In this case a dummy ROM block HP11XXXX with 0 (zero) byte values will be used in the address range of the slot.

Since release 1.4 the chosen ROM slot is validated against the ROM description file (see chapter 1.1.4). If the ROM is not compatible with this slot it is unloaded and the previous ROM in this slot is reloaded. This may occur with several HP9810A ROMs. E.g. The HP11211A PRINTER ALPHA ROM can only be plugged into slot 3.

ROM exchange in the HP9830A is somewhat tricky since the plug-in ROMs are covered in the left side of the calculator housing and normally not visible. One has to click somewhere in an area in the LED display, above the function keys. Then a dialog window is opened which shows the ROM slots. Now click on the ROM position to be exchanged. A Second dialog window is opened which shows the available plug-in ROMs. Click on one of the symbolic ROMs to select it for use in the same slot as the previous. If no ROM has to be used click on the first symbol showing an empty slot. The newly select ROM is now displayed in position. Choose another ROM to exchange or close the slot window by clicking on the upper right corner.


HP9830A ROM Blocks Slot 1
MODEL 30 MATRIX OPERATIONS 11270B
MODEL 30 PLOTTER CONTROL 11271B
MODEL 30 EXTENDED I/O 11272B
MODEL 30 MASS MEMORY 11273B
MODEL 30 STRING VARIABLES 11274B
HP9830A ROM selector dialog

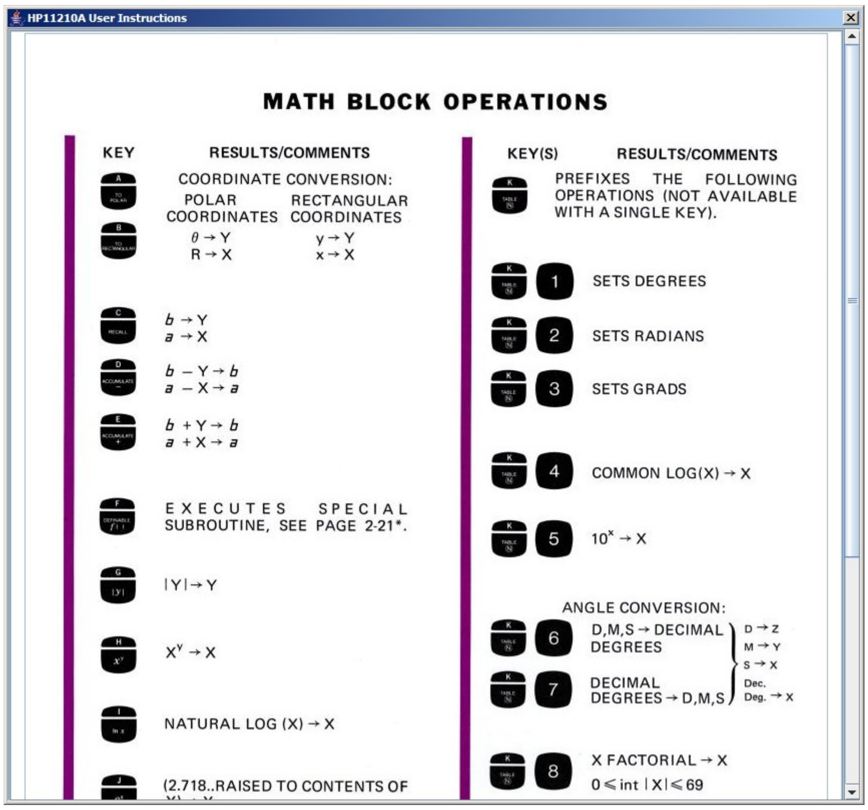

HP9830A ROMs
MODEL 30 MASS MEMORY 11273B
MODEL 30 STRING VARIABLES 11274B
MODEL 30 ADVANCED PROG I 11279B
MODEL 30 EXTENDED I/O 11272B
HP9830A ROM housing with one empty slot

After exchange of a ROM block the calculator should be restarted by the key combination Ctrl+Alt+R. This has the same effect as switching the real calculator off and on. For certain ROMs this procedure is necessary to prevent the calculator from erratic behaviour or hang-up.

1.3 Display of User Instructions

There is a facility to display one ore more pages of user instructions for each plug-in ROM as well as for the calculator itself. The user instructions where extracted from the original operating manuals as far as these had been available and contained instruction summary pages or error codes. If there is a keyboard overlay for a ROM (only available for the HP9810A and HP9820/21A half-keys), this overlay will also be shown enlarged.

The ROM specific user instructions may be displayed by clicking with the *right* mouse-button on the plugged ROM module. On the HP9810A, HP9820A, and HP9821A a right-click anywhere on the keyboard overlay area also shows the user instructions. On the HP9830A the ROM slot window has to be opened before the instructions can be displayed (see above ch. 1.2).


HP11210A User Instructions

MATH BLOCK OPERATIONS

KEY	RESULTS/COMMENTS
A TO POLAR	COORDINATE CONVERSION: POLAR COORDINATES $\theta \rightarrow Y$, $R \rightarrow X$
B TO RECTANGULAR	RECTANGULAR COORDINATES $y \rightarrow Y$, $x \rightarrow X$
C RECALL	$b \rightarrow Y$ $a \rightarrow X$
D ACCUMULATE −	$b - Y \rightarrow b$ $a - X \rightarrow a$
E ACCUMULATE +	$b + Y \rightarrow b$ $a + X \rightarrow a$
F DEFINABLE f( )	EXECUTES SPECIAL SUBROUTINE, SEE PAGE 2-21*.
G |y|	$\|Y\| \rightarrow Y$
H x^y	$X^Y \rightarrow X$
I ln x	NATURAL LOG (X) → X
J	(2.718..RAISED TO CONTENTS OF

KEY(S)	RESULTS/COMMENTS
K TABLE (N)	PREFIXES THE FOLLOWING OPERATIONS (NOT AVAILABLE WITH A SINGLE KEY).
K 1	SETS DEGREES
K 2	SETS RADIANS
K 3	SETS GRADS
K 4	COMMON LOG(X) → X
K 5	$10^x \rightarrow X$
	ANGLE CONVERSION:
K 6	D,M,S → DECIMAL DEGREES (D → Z, M → Y, S → X)
K 7	DECIMAL DEGREES → D,M,S (Dec. Deg. → X)
K 8	X FACTORIAL → X $0 \leqslant \text{int} \|X\| \leqslant 69$

User Instructions for the HP9810A Mathematics ROM

Then a popup window opens and shows the first instructions page. If there are more pages clicking with the left mouse-button on the instruction page or pressing the space-bar loads the next one. If the last page was displayed the cycle begins again with page one.

Calculator specific user instructions for the HP9810A and HP9820A may be displayed by clicking with the right mouse-button on the handle of the top cover (above the printer and magnetic card reader). For the HP9830A right-clicking on the 'hot spot' for the ROM window does the job (see above ch. 1.2).

1.4 Keyboard Configuration Files

The assignment of keys on the host-PC keyboard to key codes of the emulated machines is controlled by a calculator specific configuration file. By that means user- or language-specific keyboard customizations can be achieved.

The name of the configuration file is

```
    <Machine>-keyb.cfg
```

where <Machine> is the name of the machine configuration file. E.g. if the machine configuration file is MyHP9830A.cfg then the corresponding keyboard configuration file is MyHP9830A-keyb.cfg.

The keyboard configuration file is a plain text file and contains one key code assignment per line. Each assignment consists of the octal calculator key code, the decimal PC key code, and an optional modifier key, each separated by space or tabulator.

The octal key code may be found in the operation manual of the individual calculator or can be determined using the GO9800 Console key-log function (see appendix C.3).

The host-PC key code may be found elsewhere or also using the key-log function as described above.

The modifier key defines one to three PC-keys which have to be pressed simultaneously with the function key. The possible modifier keys are

- S = Shift
- A = Alt
- C = Control

Example: If the calculator key code is 77 (octal) and on the PC the keys Alt+Sift+X have to be pressed to activate that key, the configuration line is:

```
    77   88   AS
```

The configuration file may contain comment lines, which are not evaluated. Comment lines have to start with a semicolon. Comments can also be used as last part of a assignment line.

Example:

```
; This is a comment line
```

To ease the creation of keyboard configurations for the HP9830A, most alphanumeric keys are inherited from the host PC. PC keys which are not listed in the configuration file are treated as follows:

If the corresponding octal scan code is between 40 and 172 (inclusive) it is used as the calculator key code. For compatibility with the HP9830A lower case characters are converted to upper case and upper case characters are or-ed with the shift-bit (octal 200).

If the scan code is outside this range, the key is ignored. So normally only function keys and language specific keys have to be configured.

1.4.1 Display of the Keyboard Mapping

The current mapping of PC keys to calculator keys can be temporary displayed as a keyboard overlay by pressing Ctrl+K. For each calculator key the configured PC key combination is displayed beneath that key. If no special PC key is assigned, the calculator key code converted to the corresponding ASCII character is displayed. Modifiers Alt, Ctrl, and Shift are displayed as A+, C+, and S+ respectively, followed by the normal key.

In order to have a readable representation of special PC keys like function or cursor keys there is an editable file

```
keynames.cfg
```

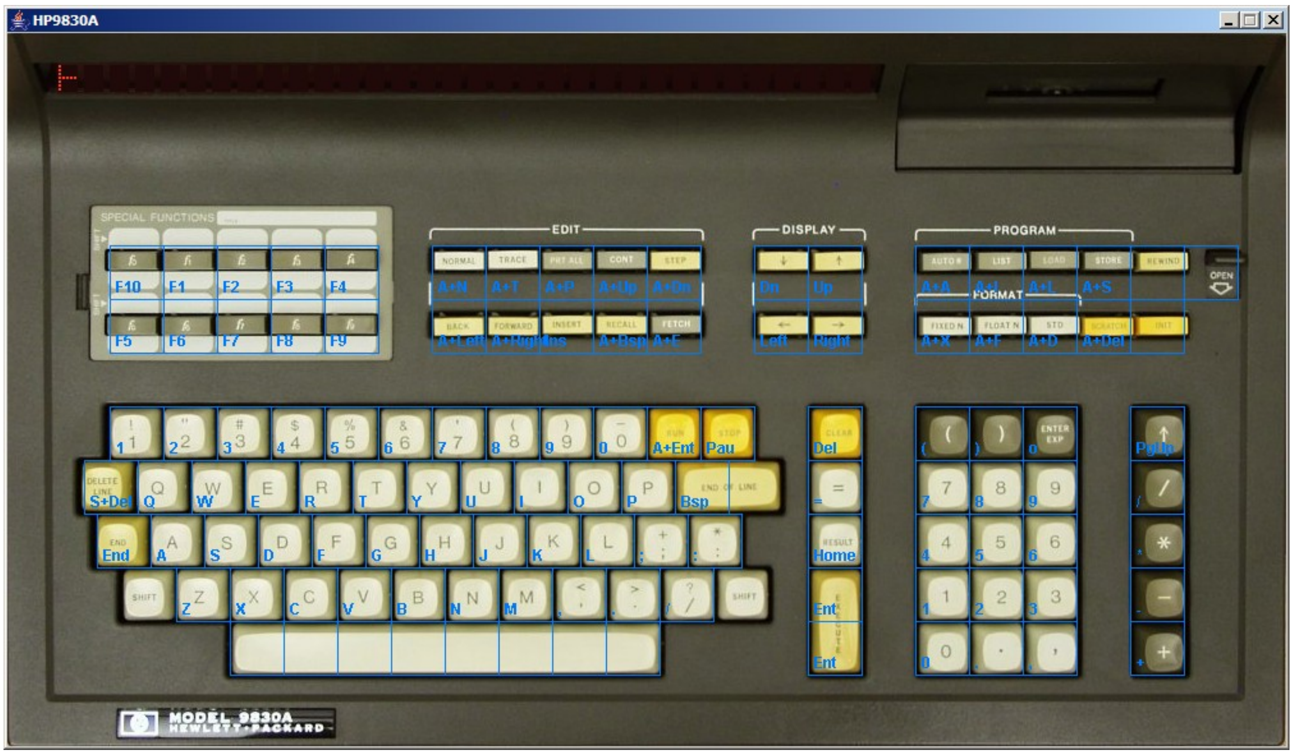
which contains the translation of JAVA key codes to display character strings.

The configuration file of the distribution looks as follows:

```
8    Bsp
10   Ent
19   Pau
27   Esc
32   Spc
33   PgUp
34   PgDn
35   End
36   Home
37   Left
38   Up
39   Right
40   Dn
96   K0
97   K1
98   K2
99   K3
100  K4
101  K5
102  K6
103  K7
104  K8
105  K9
106  K*
107  K+
109  K-
110  K.
111  K/
112  F1
113  F2
114  F3
115  F4
116  F5
117  F6
118  F7
119  F8
120  F9
121  F10
```

122	F11
123	F12
127	Del
153	<
155	Ins
520	#
521	~

The key map also shows the 'hot areas' for each calculator key in which it can be actuated by a mouse-click.



HP9830A with key map showing 'hot areas' and PC key combinations

1.4.2 Predefined Keys

Some functions of the emulator itself may be controlled by the PC keyboard. The key codes used for these functions are hard coded and can not be changed or used in keyboard configuration files. These functions and corresponding keys are as follows.

PC keys	Function
Ctrl + D	Opens the emulator Console.
Ctrl + F	Toggles the sound output of the calculator cooling fan.
Ctrl + K	Toggles display of the PC to HP calculator key map.
Ctrl + P	Generates a hardcopy of the output of the build-in printer.
Shift + Ctrl + P	Opens the page format dialog for the hardcopy function.
Alt + Ctrl + R	Resets the calculator to power-on condition.
Ctrl + S	Toggles the complete sound output of the calculator on and off.
Ctrl + T	Starts and stops the CPU instruction execution timer.
Ctrl + Page↑	Moves the paper of the built-in printer one page up.
Ctrl + Page↓	Moves the paper of the built-in printer one page up.
Ctrl + Del	Clears the output of the built-in printer.
Ctrl + Home	Advances the paper of the built-in printer.

1.5 General Limitations

When executed on a modern PC hardware (some GHz) the emulated calculator is much faster than the original machine (typically 30 times). Programs which use loops to generate time delays may not run correctly. For internal timing of the emulator the JAVA class Thread is used. On most platforms the method Thread.sleep(milliseconds) is inaccurate, esp. for small values of 1-3 ms. This may lead to slower display and printer output and other timing deviations.

Some emulated peripheral devices are timing-critical because they deliver a continuous data stream to the calculator. Amongst these devices are the built-in magnetic card reader, the HP9860A marked card reader, and the HP9865A cassette memory. The emulator accounts for the behaviour of the real devices where asynchronous mechanical transports determine the rate of incoming data. If for any reason the calculator is not able to receive data at this given rate, information maybe lost and the data unusable. With the HP9865A this may result in check-sum errors. So care should be taken that the host PC is not too busy with other programs when using the peripheral devices mentioned.

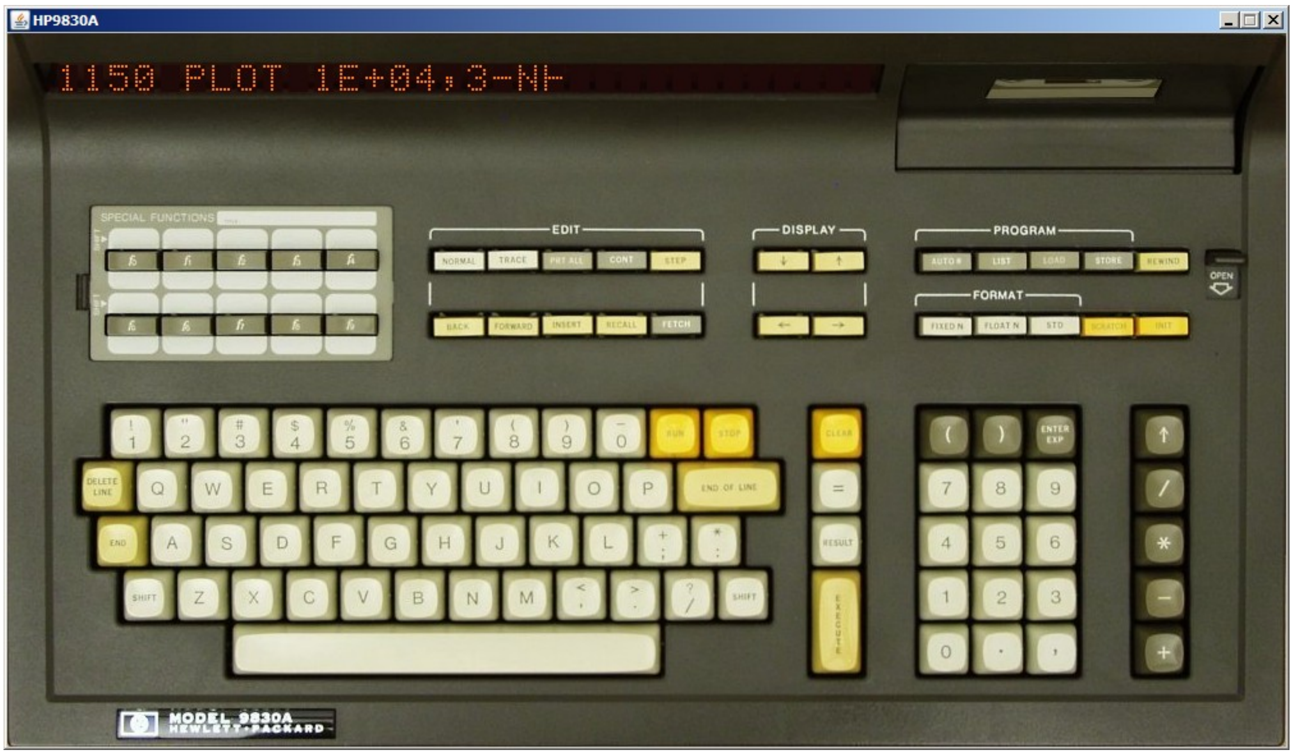
From release 1.41 there is an automatic timing calibration running at startup of the emulator. During this calibration the critical timing constants are “measured” and corrected if necessary. As a result the timing of peripheral devices should be more accurate and platform independent.

On some platforms, esp. Linux-Systems, the sound output maybe distorted or may exhibit some crackling. The is due to characteristics of the Java sound engine and some sound cards. If this is perturbing the sound may be completely switched off using the key combin-ation Ctrl+S.

2. Implemented Machines

The HP9800 emulator is able to run with different personalities which resemble the individual hardware features of the real models. In this release the personalities of the HP9810A, HP9820A, HP9821A, and HP9830A are implemented. The HP9810A is available in two versions: version one with single-digit LED display modules, and version two with five-digit LED modules. Which calculator personality is used by emulator is defined in the Model item of the configuration file. The configuration file is given the the emulator as start parameter.

2.1 HP9830A Personality



2.1.1 Implemented Features

2.1.1.1 ROM

For the HP9830A the following system ROM blocks must be defined in the configuration file:

HP9830A_System1 to HP9830A_System9

The following add-on ROMs are optional:

HP11270B Matrix Operations
HP11271B Plotter Control
HP11272B Extended I/O
HP11273B Mass Memory
HP11274B String Variables

HP11277B Terminal I
HP11278B Batch Basic
HP11279B Advanced Programming I
HP11283B Printer Control
HP11289B Advanced Programming II
HP11296B Data Comm. I
Infotek Fast Basic I
Infotek Fast Basic II
Infotek Fast Basic III
Infotek Fast Basic IV

2.1.1.2 RWM

The minimal RWM blocks which have to be configured are:

HP9830A_System
HP9830A_User (3840 words)

The following memory expansions are optional:

HP11275F (additional 4096 words)
HP11276F (additional 8192 words)

2.1.1.3 I/O Devices

The following internal devices are implemented:

Display
Sound (for output of BEEP command and error messages)
Keyboard (select code 13)
Tape drive (select code 10)

The following external devices can be configured:

HP9860A Marked Card Reader (no select code)
HP9861A Typewriter (select code 1 to 14)
HP9862A Plotter (fixed select code 14)
HP9865A Cassette Tape Drive (select code 1 to 9)
HP9866A/B Thermoelectric Line Printer (fixed select code 15)
HP9880B Mass Memory (fixed select code 11)
HP11202A Interface for file I/O on host PC (select code 1 to 9)

2.1.2 Limitations

When executed on a modern PC hardware (some GHz) the emulated HP9830A is much faster than the original machine (typically 30 times). Programs which use loops to generate time delays may not run correctly. Instead the WAIT instruction generates a correct delay in milliseconds. See also General Limitations (chapter 1.5).

Special attention should be payed when a HP9880A/B Mass Memory is connected to the HP9830A. Refer to chapter 3.6.4 for futher informations.

2.1.3 User Interfaces

2.1.3.1 Display

The original 5*7 dot matrix LED display is simulated. In the original machine the display is multiplexed completely by the CPU. The calculator controls the display by the signal DEN (Display Enable). When the calculator is busy, DEN is false and the display is dark.

In the emulator the display multiplexing would lead to flickering and high load of the host PCs CPU. Therefore the display content is buffered and refreshed only if it has changed. When DEN is false for a period of more than 200ms (which means that the calculator is busy with other things) the emulated display will get dark.

The output timing is nearly like the original, visible delays in display output are intended.

2.1.3.2 Keyboard

The representation of the HP9830 keyboard on the host PC keyboard can be configured. The configuration is stored in the text file HP9830A-keyb.cfg and can be changed with any text editor. More on keyboard configuration can be found in chapter 1.4.

Besides the predefined emulator keyboard functions (see chapter 1.4.2) the distributed HP9830A keyboard configuration is as follows.

Main keyboard, unshifted:

PC Key	**HP9830 Key**
A - Z	A - Z
0 - 9	0 - 9
space	space
symbols (+-*, ...)	symbols (+-*, ...)
backspace	END OF LINE
return	EXECUTE
insert	INSERT
delete	CLEAR
home	RESULT
end	END
page ↑	↑ (power-of operator)
pause	STOP
← ↑ → ↓	DISPLAY ← ↑ → ↓
F1-F9	f1-f9
F10	f0

Main keyboard, shifted:

PC Key	HP9830 Key
A - Z	a - z (displayed as A - Z)
0	=
1	!
2	"
3	
4	$
5	%
6	&
7	/
8	(
9	)
symbols (+-*, ...)	symbols (+-*, ...)
del	DELETE LINE
F1-F9	f11-f19
F10	f10

Numpad keys:

PC Key	HP9830 Key
0 - 9	0 - 9
+ - * /	+ - * /
enter	EXECUTE

Alt + Key:

PC Key	HP9830 Key
←	BACK
→	FORWARD
↓	STEP
↑	CONT
return	RUN
backspace	RECALL
del	SCRATCH
A	AUTO#
D	STD
F	FLOAT

PC Key	HP9830 Key
I	LIST
L	LOAD
N	NORMAL
P	PRT ALL
S	STORE
T	TRACE
X	FIXED

2.1.4 Internal Devices

2.1.4.1 Tape Drive

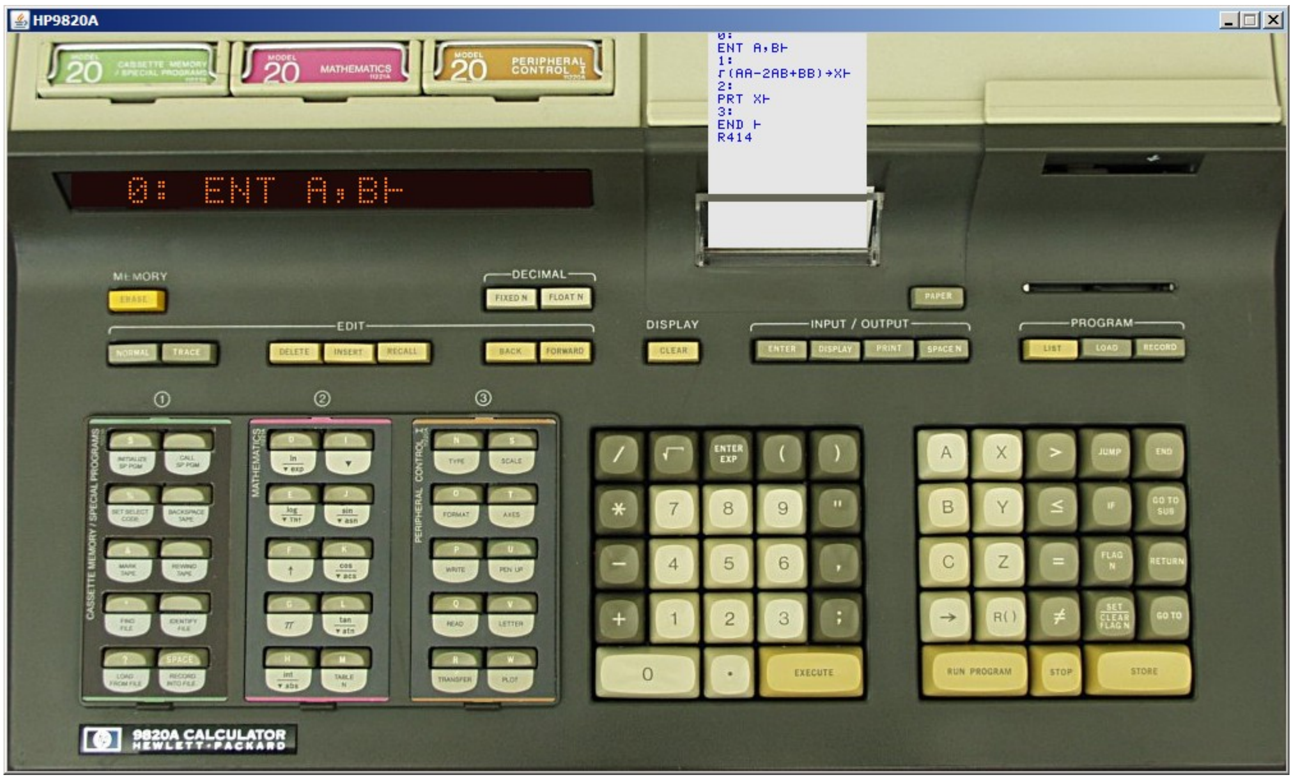
The internal tape drive is functional identical to the HP9865A external tape drive, so the emulator uses the same Java classes for internal and external drives. See the description of the HP9865A (chapter 3.4) for more detailed information.

The original tape cassettes are emulated by files in the host file system. To prepare a new host tape-file one needs a 'blank' tape. A original blank tape is not really empty, but contains at least a Begin-Of-File marker (hex 3C + control bit). Without that BOF no new files can be MARKed on that tape. In the GO9800 distribution there is file named 'blank.tape' which should be kept as origin for new tapes. To create new tape make a copy of the file 'blank.tape' and give it a name of your choice. For convention it should have the extension .tape.

To load a tape in the HP9830A emulation, mouse-click on the 'OPEN' door lever. The cassette door will be opened and a file selector dialog is displayed in which you can select your new empty tape. If you dismiss this dialog with the Cancel button, the cassette door stays open. After successful selection of a tape the door will be closed again and a loaded cassette is visible in the door window.

Now the tape is ready for the usual HP9830 tape commands, like MARK, STORE, and LOAD. See the original HP9830A manual for reference.

2.2 *HP9820A Personality*



```
0:
ENT A,B⊢
1:
√(AA-2AB+BB)→X⊢
2:
PRT X⊢
3:
END ⊢
R414
```

2.2.1 Implemented Features

2.2.1.1 ROM

For the HP9820A the following system ROM blocks must be defined in the configuration file:

HP9820A_System1 to HP9820A_System6

The following add-on ROMs are optional:

HP11221A Mathematics
HP11222A User Defined Functions
HP11223A Cassette Memory / Special Programs
HP11224A Peripheral Control I

2.2.1.2 RWM

The minimal RWM blocks which have to be configured are:

HP9820A_System
HP9820A_User (768 words = 192 registers)

The following memory expansions are optional:

HP11228A (additional 1024 words = 256 registers)

2.2.1.3 I/O Devices

The following internal devices are implemented:

Display
Magnetic Card Reader
Printer
Keyboard (select code 13)

The following external devices can be configured:

HP9860A Marked Card Reader (no select code)
HP9861A Typewriter (fixed select code 15)
HP9862A Plotter (fixed select code 14)
HP9865A Cassette Tape Drive (select code 1 to 9)
HP9866A/B Thermoelectric Line Printer (fixed select code 15)
HP11202A Interface for file I/O on host PC (select code 1 to 9)

2.2.2 Limitations

When executed on a modern PC hardware (some GHz) the emulated HP9820 is much faster than the original machine (typically 30 times). Programs which use loops to generate time delays may not run correctly. See also general limitations (chapter 1.5).

2.2.3 User Interfaces

2.2.3.1 Display

The original 5*7 dot matrix LED display is simulated. In the original machine the display is multiplexed completely by the CPU. The calculator controls the display by the signal DEN (Display Enable). When the calculator is busy, DEN is false and the display is dark.

In the emulator the display multiplexing would lead to flickering and high load of the host PCs CPU. Therefore the display content is buffered and refreshed only if it has changed. When DEN is false for a period of more than 200ms (which means that the calculator is busy with other things) the emulated display will get dark.

2.2.3.2 Keyboard

The representation of the HP9820 keyboard on the host PC keyboard can be configured. The configuration is stored in the text file HP9820A-keyb.cfg and can be changed with any text editor. More on keyboard configuration can be found in chapter 1.4.

Besides the predefined emulator keyboard functions (see chapter 1.4.2) the distributed HP9820A keyboard configuration is as follows.

Main keyboard:

PC Key	**HP9820 Key**
A - C	A - C
D - W	D - W (half keys)
X - Z	X - Z
0 - 9	0 - 9
$ % & ' ?	$ % & ' ?
,	,
;	;
~	√x
#	≠
=	=
<	≤
>	>
^	GOTO SUB
backspace	STORE
space	SPACE

Numpad keys:

PC Key	**HP9820 Key**
0 - 9	0 - 9
+ - * /	+ - * /
.	.
enter	EXECUTE

Special and Function Keys:

PC Key	**HP9820 Key**
insert	INSERT
delete	DELETE
end	END
backspace	STORE
pause	STOP
←	BACK
→	FORWARD

PC Key	**HP9820 Key**
F1	NORMAL
F2	TRACE
F3	FIXED
F4	FLOAT
F5	ENTER
F6	DISPLAY
F7	PRINT
F8	SPACE
F9	LIST
F10	LOAD
F11	RECORD
F12	

Alt + Key:

PC Key	**HP9820 Key**
C	SET / CLEAR FLAG N
E	END
F	FLAG N
G	GO TO
R	R()
T	RETURN
U	GO TO SUB
X	ENTER EXP
<	≠
→	→
return	RUN PROGRAM
backspace	RECALL
del	CLEAR

Alt + Shift + Key:

PC Key	**HP9820 Key**
del	ERASE

2.2.4 Internal Devices

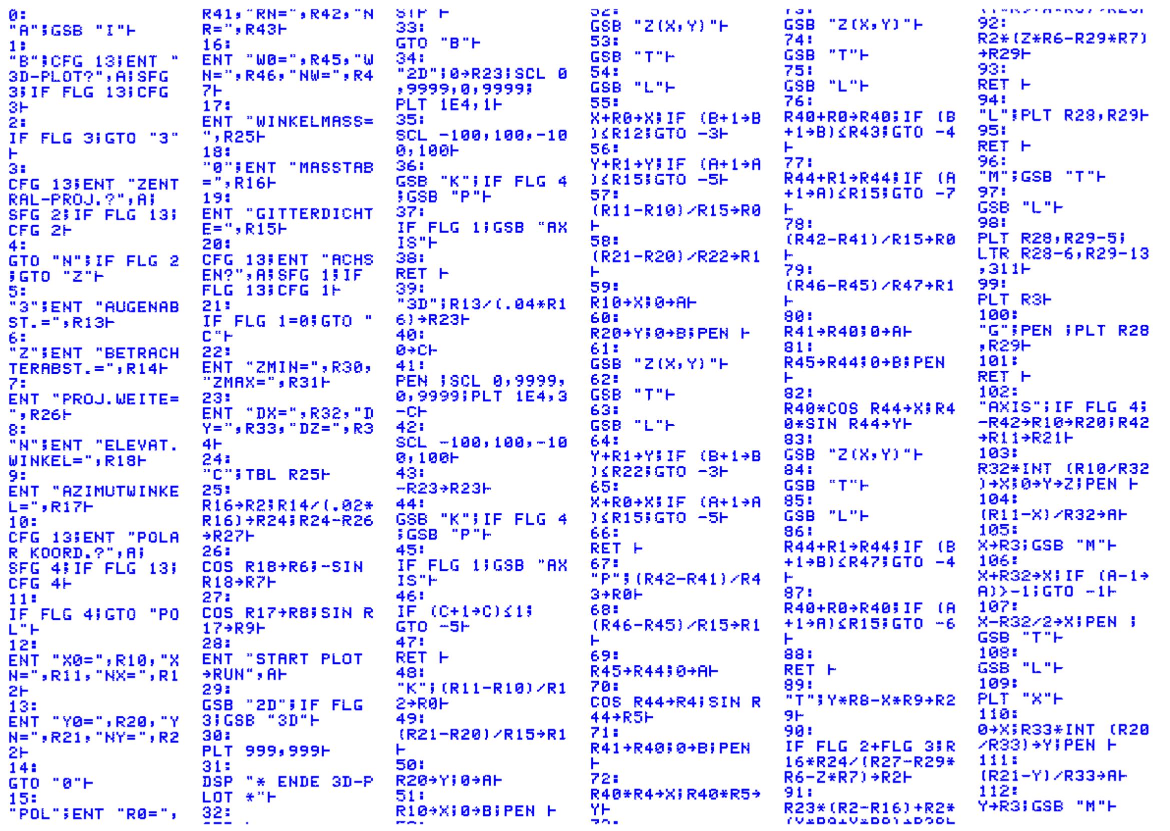
2.2.4.1 Printer

The HP9820A was equipped with a 16 character wide thermo electric printer. Use of the printer is identical to the original. The printer output is graphically emulated and the graphics are internally buffered. The 'paper' advances to the top of the displayed window area. Since the HP9820 generates the output as printer dots, not as characters, it is impossible to store it in a text file.

The output can be scrolled back and forth with the PC keys Ctrl+Page↑ and Ctrl+Page↓. The complete output can be erased by Ctrl+Del.

The PAPER advance key can be operated by mouse click or Ctrl+Home keys. Click on the PAPER key and hold the mouse button down to achieve a continous paper feed.

Since release 1.50 it is possible to send a the printer output to an arbitrary printer of the host PC. Pressing the PC keys Shift+Ctrl+P opens the standard page-setup dialog of the host system. Pressing Ctrl+P opens the standard print dialog of the host system, where the printer may be selected and the hardcopy started. The output is automatically arranged in multiple columns per page, the number depends of the page size and orientation.



```
0:
"A";GSB "I"⊢
1:
"B";CFG 13;ENT "
3D-PLOT?",A;SFG
3;IF FLG 13;CFG
3⊢
2:
IF FLG 3;GTO "3"
⊢
3:
CFG 13;ENT "ZENT
RAL-PROJ.?",A;
SFG 2;IF FLG 13;
CFG 2⊢
4:
GTO "N";IF FLG 2
;GTO "Z"⊢
5:
"3";ENT "AUGENAB
ST.=",R13⊢
6:
"Z";ENT "BETRACH
TERABST.=",R14⊢
7:
ENT "PROJ.WEITE=
",R26⊢
8:
"N";ENT "ELEVAT.
WINKEL=",R18⊢
9:
ENT "AZIMUTWINKE
L=",R17⊢
10:
CFG 13;ENT "POLA
R KOORD.?",A;
SFG 4;IF FLG 13;
CFG 4⊢
11:
IF FLG 4;GTO "PO
L"⊢
12:
ENT "X0=",R10,"X
N=",R11,"NX=",R1
2⊢
13:
ENT "Y0=",R20,"Y
N=",R21,"NY=",R2
2⊢
14:
GTO "0"⊢
15:
"POL";ENT "R0=",
```

```
R41,"RN=",R42,"N
R=",R43⊢
16:
ENT "W0=",R45,"W
N=",R46,"NW=",R4
7⊢
17:
ENT "WINKELMASS=
",R25⊢
18:
"0";ENT "MASSTAB
=",R16⊢
19:
ENT "GITTERDICHT
E=",R15⊢
20:
CFG 13;ENT "ACHS
EN?",A;SFG 1;IF
FLG 13;CFG 1⊢
21:
IF FLG 1=0;GTO "
C"⊢
22:
ENT "ZMIN=",R30,
"ZMAX=",R31⊢
23:
ENT "DX=",R32,"D
Y=",R33,"DZ=",R3
4⊢
24:
"C";TBL R25⊢
25:
R16→R2;R14/(.02*
R16)→R24;R24-R26
→R27⊢
26:
COS R18→R6;-SIN
R18→R7⊢
27:
COS R17→R8;SIN R
17→R9⊢
28:
ENT "START PLOT
→RUN",A⊢
29:
GSB "2D";IF FLG
3;GSB "3D"⊢
30:
PLT 999,999⊢
31:
DSP "* ENDE 3D-P
LOT *"⊢
32:
```

```
33:
GTO "B"⊢
34:
"2D";0→R23;SCL 0
,9999,0,9999;
PLT 1E4,1⊢
35:
SCL -100,100,-10
0,100⊢
36:
GSB "K";IF FLG 4
;GSB "P"⊢
37:
IF FLG 1;GSB "AX
IS"⊢
38:
RET ⊢
39:
"3D";R13/(.04*R1
6)→R23⊢
40:
0→C⊢
41:
PEN ;SCL 0,9999,
0,9999;PLT 1E4,3
-C⊢
42:
SCL -100,100,-10
0,100⊢
43:
-R23→R23⊢
44:
GSB "K";IF FLG 4
;GSB "P"⊢
45:
IF FLG 1;GSB "AX
IS"⊢
46:
IF (C+1→C)≤1;
GTO -5⊢
47:
RET ⊢
48:
"K";(R11-R10)/R1
2→R0⊢
49:
(R21-R20)/R15→R1
⊢
50:
R20→Y;0→A⊢
51:
R10→X;0→B;PEN ⊢
```

```
GSB "Z(X,Y)"⊢
53:
GSB "T"⊢
54:
GSB "L"⊢
55:
X+R0→X;IF (B+1→B
)≤R12;GTO -3⊢
56:
Y+R1→Y;IF (A+1→A
)≤R15;GTO -5⊢
57:
(R11-R10)/R15→R0
⊢
58:
(R21-R20)/R22→R1
⊢
59:
R10→X;0→A⊢
60:
R20→Y;0→B;PEN ⊢
61:
GSB "Z(X,Y)"⊢
62:
GSB "T"⊢
63:
GSB "L"⊢
64:
Y+R1→Y;IF (B+1→B
)≤R22;GTO -3⊢
65:
X+R0→X;IF (A+1→A
)≤R15;GTO -5⊢
66:
RET ⊢
67:
"P";(R42-R41)/R4
3→R0⊢
68:
(R46-R45)/R15→R1
⊢
69:
R45→R44;0→A⊢
70:
COS R44→R4;SIN R
44→R5⊢
71:
R41→R40;0→B;PEN
⊢
72:
R40*R4→X;R40*R5→
Y⊢
```

```
GSB "Z(X,Y)"⊢
74:
GSB "T"⊢
75:
GSB "L"⊢
76:
R40+R0→R40;IF (B
+1→B)≤R43;GTO -4
⊢
77:
R44+R1→R44;IF (A
+1→A)≤R15;GTO -7
⊢
78:
(R42-R41)/R15→R0
⊢
79:
(R46-R45)/R47→R1
⊢
80:
R41→R40;0→A⊢
81:
R45→R44;0→B;PEN
⊢
82:
R40*COS R44→X;R4
0*SIN R44→Y⊢
83:
GSB "Z(X,Y)"⊢
84:
GSB "T"⊢
85:
GSB "L"⊢
86:
R44+R1→R44;IF (B
+1→B)≤R47;GTO -4
⊢
87:
R40+R0→R40;IF (A
+1→A)≤R15;GTO -6
⊢
88:
RET ⊢
89:
"T";Y*R8-X*R9→R2
9⊢
90:
IF FLG 2+FLG 3;R
16*R24/(R27-R29*
R6-Z*R7)→R2⊢
91:
R23*(R2-R16)+R2*
```

```
92:
R2*(Z*R6-R29*R7)
→R29⊢
93:
RET ⊢
94:
"L";PLT R28,R29⊢
95:
RET ⊢
96:
"M";GSB "T"⊢
97:
GSB "L"⊢
98:
PLT R28,R29-5;
LTR R28-6,R29-13
,311⊢
99:
PLT R3⊢
100:
"G";PEN ;PLT R28
,R29⊢
101:
RET ⊢
102:
"AXIS";IF FLG 4;
-R42→R10→R20;R42
→R11→R21⊢
103:
R32*INT (R10/R32
)→X;0→Y→Z;PEN ⊢
104:
(R11-X)/R32→A⊢
105:
X→R3;GSB "M"⊢
106:
X+R32→X;IF (A-1→
A)>-1;GTO -1⊢
107:
X-R32/2→X;PEN ;
GSB "T"⊢
108:
GSB "L"⊢
109:
PLT "X"⊢
110:
0→X;R33*INT (R20
/R33)→Y;PEN ⊢
111:
(R21-Y)/R33→A⊢
112:
Y→R3;GSB "M"⊢
```

Example of a printer hardcopy

2.2.4.2 Magnetic Card Reader

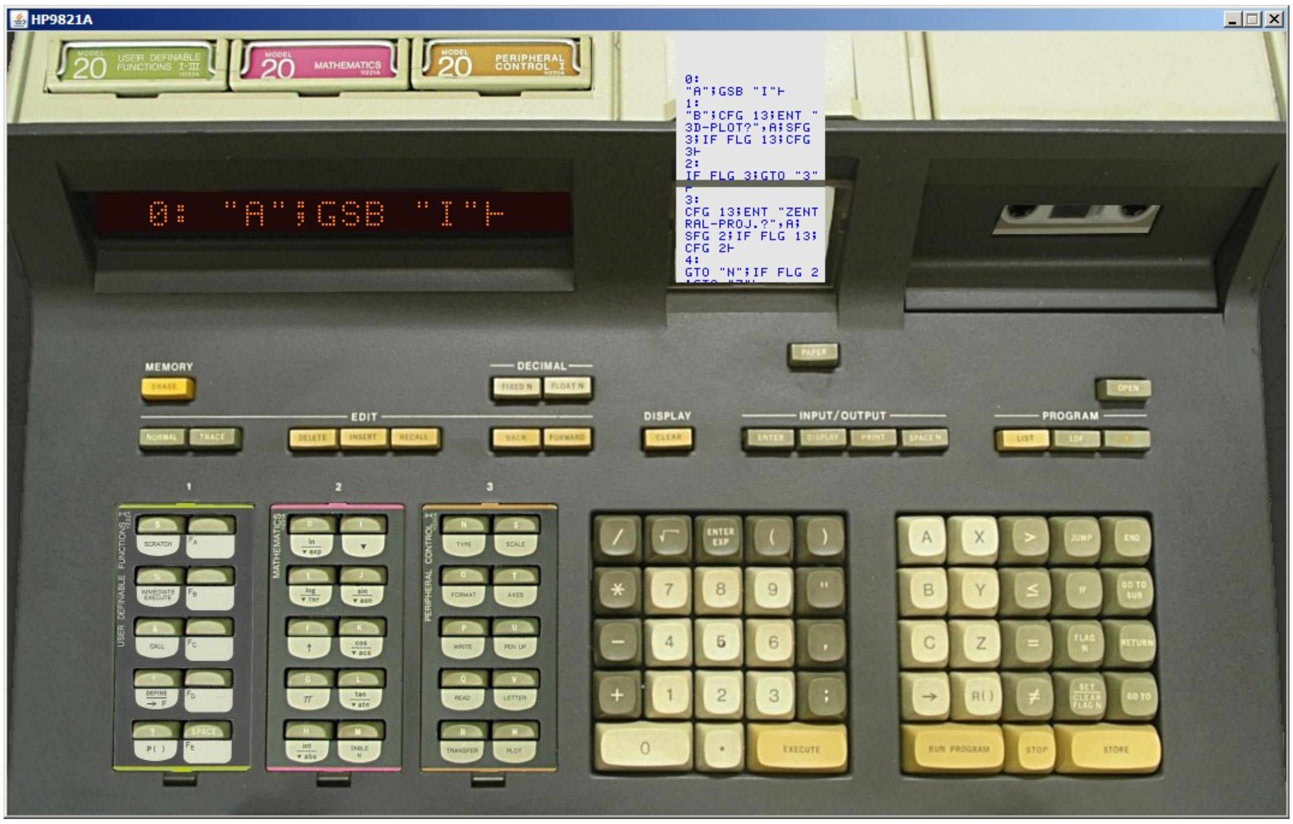
Like the HP9865A the internal card reader emulates the original magnetic media by files in the host file system. To prepare a new host card-file one needs a 'blank' card. This can be any simple empty (0 byte length) file, e.g. created with a text editor. Give the file a name of your choice. For convention it should have the extension .mcard.

To write a program in HP9820 main memory to a magnetic card simply click on the RECORD key. Then the display goes out and the sound of the transport motor is audible. A file selector dialog is displayed in which you can select your card file. If you dismiss this dialog with the Cancel button, the motor keeps running and has to be stopped with the STOP key. After successful selection of a card the program will be stored in the card file. As in the original every previously stored contents will be overwritten. Finally the motor sound stops.

Unlike the original a program of any size can be stored on one single card.

Reading of a program stored in a magnetic card is straight forward. Also writing and reading of data registers. See the original HP9820A manual for reference.

2.3 *HP9821A Personality*



```
0: "A";GSB "I"⊢
```

```
0:
"A";GSB "I"⊢
1:
"B";CFG 13;ENT "
3D-PLOT?",A;SFG
3;IF FLG 13;CFG
3⊢
2:
IF FLG 3;GTO "3"
⊢
3:
CFG 13;ENT "ZENT
RAL-PROJ.?",A;
SFG 2;IF FLG 13;
CFG 2⊢
4:
GTO "N";IF FLG 2
```

The HP9821A is an advancement of the HP9820A and is identical to it in the most functions. The main differences are:

- A cassette tape drive instead of a magnetic card reader
- Extended memory (up to 1400 registers)
- Different design of the chassis

Most user programs for the HP9820A will also run on the HP9821A.

2.3.1 Implemented Features

2.3.1.1 ROM

For the HP9821A the following system ROM blocks must be defined in the configuration file:

HP9821A_System1 to HP9821A_System6

The following add-on ROMs are optional:

HP11221A Mathematics
HP11222A User Defined Functions
HP11224A Peripheral Control I

The HP11223A Cassette Memory / Special Programs ROM can not be used with the

HP9821A since the functionality is already contained in the system ROM.

2.3.1.2 RWM

The minimal RWM blocks which have to be configured are:

HP9821A_System
HP9821A_User (1792 words = 448 registers)

Up to two of the following memory expansions are optional:

HP11255A (additional 2048 words = 512 registers)

2.3.1.3 I/O Devices

The following internal devices are implemented:

Display
Printer
Sound (for output of ♦BEL command and error messages)
Keyboard (select code 13)
Cassette Tape Drive (select code 10)

The following external devices can be configured:

HP9860A Marked Card Reader (no select code)
HP9861A Typewriter (fixed select code 15)
HP9862A Plotter (fixed select code 14)
HP9865A Cassette Tape Drive (select code 1 to 9)
HP9866A/B Thermoelectric Line Printer (fixed select code 15)
HP11202A Interface for file I/O on host PC (select code 1 to 9)

2.3.2 Limitations

When executed on a modern PC hardware (some GHz) the emulated HP9821 is much faster than the original machine (typically 30 times). Programs which use loops to generate time delays may not run correctly. See also general limitations (chapter 1.5).

2.3.3 User Interfaces

2.3.3.1 Display

The original 5*7 dot matrix LED display is simulated. In the original machine the display is multiplexed completely by the CPU. The calculator controls the display by the signal DEN (Display Enable). When the calculator is busy, DEN is false and the display is dark.

In the emulator the display multiplexing would lead to flickering and high load of the host PCs CPU. Therefore the display content is buffered and refreshed only if it has changed. When DEN is false for a period of more than 200ms (which means that the calculator is busy with other things) the emulated display will get dark.

2.3.3.2 Keyboard

The representation of the HP9821 keyboard on the host PC keyboard can be configured. The configuration is stored in the text file HP9821A-keyb.cfg and can be changed with any text editor. More on keyboard configuration can be found in chapter 1.4.

The distributed PC keyboard configuration is the same as for the HP9820A.

2.3.4 Internal Devices

2.3.4.1 Printer

The HP9821A was equipped with a 16 character wide thermo electric printer. Use of the printer is identical to the original. The printer output is graphically emulated and the graphics are internally buffered. The 'paper' advances to the top of the displayed window area. Since the HP9821A generates the output as printer dots, not as characters, it is impossible to store it in a text file.

The output can be scrolled back and forth with the PC keys Ctrl+Page↑ and Ctrl+Page↓. The complete output can be erased by Ctrl+Del.

The PAPER advance key can be operated by mouse click or Ctrl+Home keys. Click on the PAPER key and hold the mouse button down to achieve a continuous paper feed.

Since release 1.50 it is possible to send a the printer output to an arbitrary printer of the host PC. Pressing the PC keys Shift+Ctrl+P opens the standard page-setup dialog of the host system. Pressing Ctrl+P opens the standard print dialog of the host system, where the printer may be selected and the hardcopy started. The output is automatically arranged in multiple columns per page, the number depends of the page size and orientation.

2.3.4.2 Tape Drive

The internal tape drive is functional identical to the HP9865A external tape drive, so the emulator uses the same Java classes for internal and external drives. See the description of the HP9865A (chapter 3.4) for more detailed information.

The original tape cassettes are emulated by files in the host file system. To prepare a new host tape-file one needs a 'blank' tape. A original blank tape is not really empty, but contains at least a Begin-Of-File marker (hex 3C + control bit). Without that BOF no new files can be MARKed on that tape. In the GO9800 distribution there is file named 'blank.tape' which should be kept as origin for new tapes. To create new tape make a copy of the file 'blank.tape' and give it a name of your choice. For convention it should have the extension .tape.

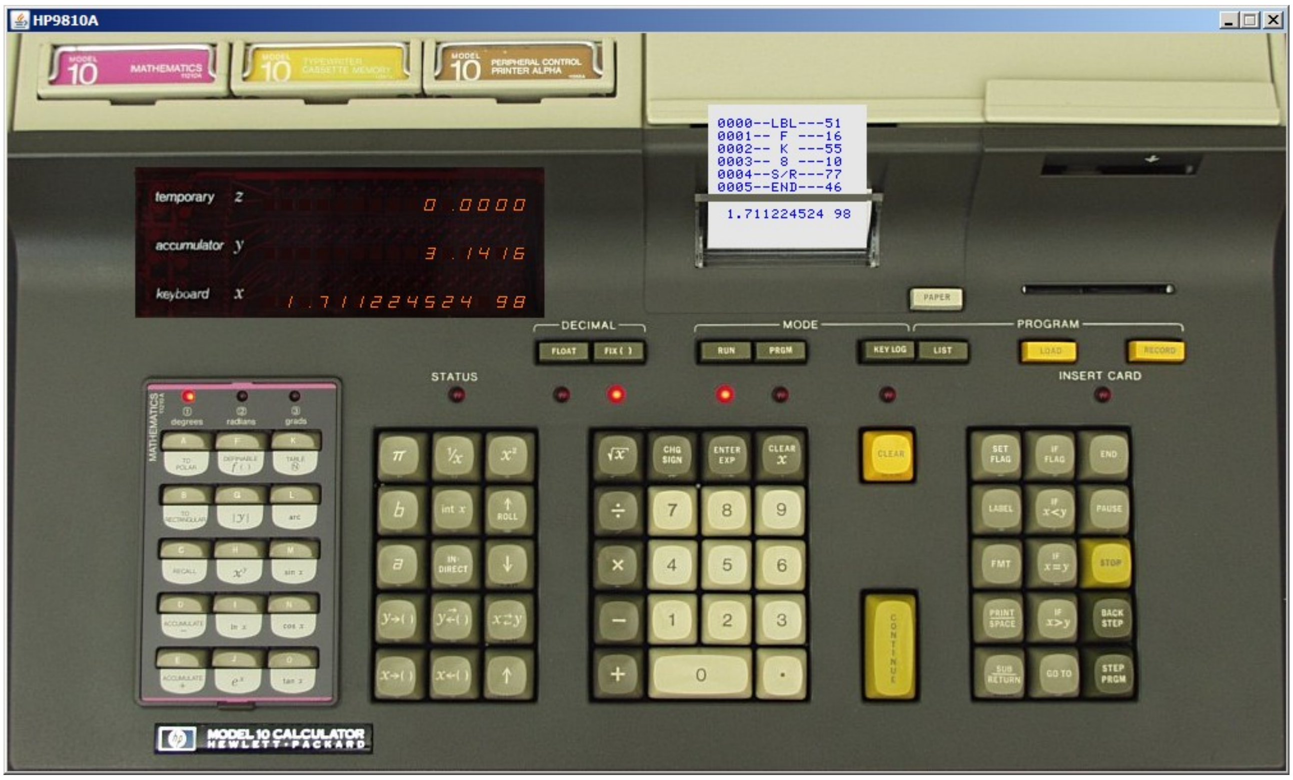
To load a tape in the HP9821A emulation, mouse-click on the 'OPEN' key. A file selector dialog is displayed in which you can select your new empty tape. After successful selection of a tape a loaded cassette is visible in the door window.

Now the tape is ready for the usual HP9821A tape commands, like MRK, STF, and LDF. See the original HP9821A manual for reference.

Most tape functions and some special functions are obtained by pressing the prefix key ♦ followed by another key. On the original HP9821A these secondary functions are printed on the front side of the key caps but are not visible in the emulator. For the ease of use here is a list of the secondary functions and their corresponding key sequences.

Function	**Mnemonic**	**Key Sequence**
Load File	LDF	LDF
Record File	RCF	♦ LDF
Find File	FDF	♦ >
Rewind	REW	♦ JUMP
Back Space	BKS	♦ ≤
Mark	MRK	♦ IF
Set Select Code	SSC	♦ =
Identify File	IDF	♦ FLAG N
Call Special Program	CSP	♦ ≠
Initialize Special Program	ISP	♦ SET│CLEAR FLAG N
Bell	BEL	♦ SPACE N

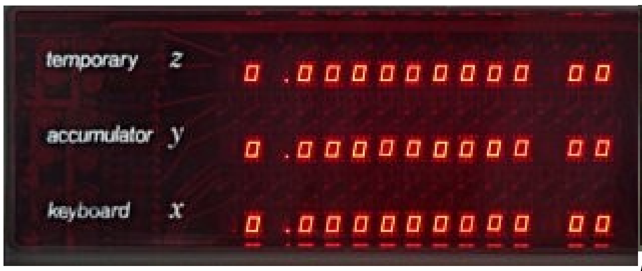
2.4 *HP9810A Personality*


```
0000--LBL---51
0001-- F ---16
0002-- K ---55
0003-- 8 ---10
0004--S/R---77
0005--END---46

 1.711224524 98
```

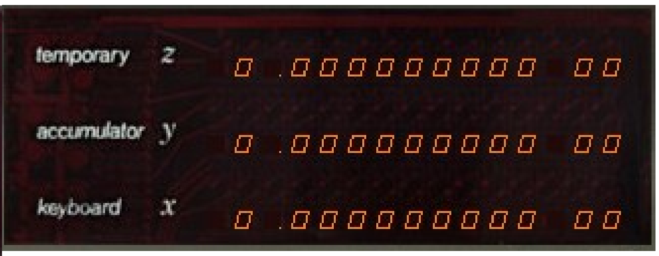
2.4.1 Implemented Features

The HP9810 comes in two slightly different versions. The first version was produced between 1971 and (perhaps) 1973 and used a single LED display module for each digit. The second version used the same 5-digit modules (HP1990-7405 or HP5082-7405) as the classic pocket calculators, like the HP-35. The digits of this display are significantly smaller then those of the first version. Also the second version has a bug fix in the system ROM affecting the square root function[1].

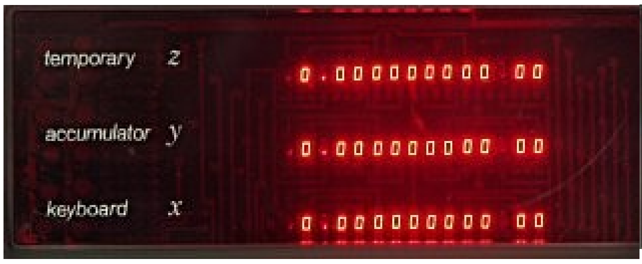
1 In the first version the square root of arguments near to squares of integers, e.g. √36.0000000005, causes an arbitrary change of the exponent of the Z-register.



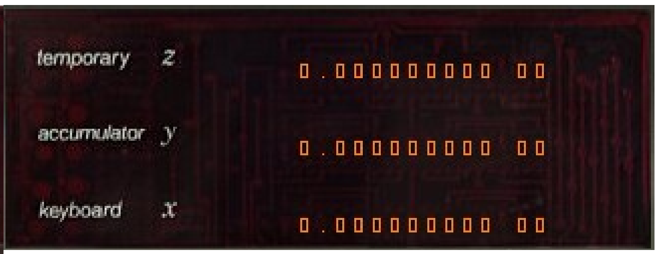
HP9810A version 1 real display



HP9810A version 1 emulator display



HP9810A version 2 real display



HP9810A version 2 emulator display

Which version is emulated is controlled by the configuration file (chapter 1.1). To emulate the earlier version the first configuration item has to be

```
Model HP9810A 1
```

To emulate the second version the first configuration item has to be

```
Model HP9810A 2
```

2.4.1.1 ROM

For the HP9810A the following system ROM blocks must be defined in the configuration file:

HP9810A_System1 to HP9810A_System3

Alternatively the never system ROMs of the second version can be used:

HP9810A2_System1 to HP9810A2_System3

The following add-on ROMs are optional:

HP11210A Mathematics
HP11211A Printer Alpha
HP11213A User Definable Functions
HP11214A Statistics
HP11215A Plotter
HP11252A Peripheral Control II
HP11261A Plotter / Printer Alpha Combo
HP11262A Peripheral Control / Cassette Memory Combo
HP11266A Peripheral Control / Printer Alpha Combo
HP11267A Typewriter / Cassette Memory Combo

Note 1: The Printer Alpha ROM (HP11211A) and all combo ROMs with this can only be used in ROM slot 3. Use in another slot will lead to erratic output of the printer.

Note 2: The ROMs which require the left block of the so called half-keys and a corresponding keyboard overlay will work in ROM slot 1 only.

2.4.1.2 RWM

The minimal RWM blocks which have to be configured are:

HP9810A_Data (109 registers)
HP9810A_Program (500 program steps)

The following memory expansions are optional:

HP11217A (512 additional program steps)
HP11218A (1024 additional program steps)

2.4.1.3 I/O Devices

The following internal devices are implemented:

Display
Magnetic Card Reader
Printer
Keyboard

The following external devices can be configured:

HP9860A Marked Card Reader (no select code)
HP9861A Typewriter (select code 15)
HP9862A Plotter (fixed select code 14)
HP9865A Cassette Tape Drive (select code 1 to 9)
HP9866A/B Thermal Line Printer (select code 15)
HP11202A Interface for file I/O on host PC (select code 1 to 9)

2.4.2 Limitations

When executed on a modern PC hardware (some GHz) the emulated HP9810 is much faster than the original machine (typically 30 times). Programs which use loops to generate time delays may not run correctly. See also general limitations (chapter 1.5).

2.4.3 User Interfaces

2.4.3.1 Display

The original 7 segment LED display is simulated. There are two different versions implemented: the older version 1 uses single digit display modules, the newer version 2 5-digit modules. In the original machine the display is multiplexed completely by the CPU. The calculator controls the display by the signal DEN (Display Enable). When the calculator is busy, DEN is false and the display is dark.

In the emulator the display multiplexing would lead to flickering and high load of the host PCs CPU. Therefore the display content is buffered and refreshed only if it has changed. When DEN is false for a period of more than 200ms (which means that the calculator is busy with other things) the emulated display will get dark.

2.4.3.2 Keyboard

The representation of the HP9810 keyboard on the host PC keyboard can be configured. The configuration is stored in the text file HP9810A-keyb.cfg and can be changed with any text editor. More on keyboard configuration can be found in chapter 1.4.

Besides the predefined emulator keyboard functions (see chapter 1.4.2) the distributed HP9810A keyboard configuration is as follows.

Main keyboard:

PC Key	**HP9810 Key**
A - O	A - O (half keys)
P	π
Q	b
R	a
S	y→()
T	x→()
U	1/x
V	int x
W	INDIRECT
X	$y\overset{\rightarrow}{\leftarrow}()$
Y	x←()
Z	x^2
0 - 9	0 - 9
-	CHG SIGN
<	IF x<y
>	IF x>y
=	IF x=y
#	ENTER EXP
~	$\sqrt{x}$
^	SUB/RETURN
backspace	CLEAR x
return	CONTINUE

Numpad keys:

PC Key	**HP9810 Key**
0 - 9	0 - 9
+ - * /	+ - * /
enter	CONTINUE

Special and Function Keys:

PC Key	**HP9810 Key**
del	CLEAR
page ↑	ROLL↑
page ↓	x⇄y
pause	STOP
↑	↑
↓	↓
←	BACK STEP
→	STEP PRGM
F5	FLOAT
F6	FIX
F7	RUN
F8	PRGM
F9	KEY LOG
F10	LIST
F11	LOAD
F12	RECORD

Alt + Key:

PC Key	**HP9810 Key**
A	a
B	b
E	END
F	FORMAT
G	GOTO
I	IF FLAG
L	LABEL
P	PRINT
R	SUB/RETURN
S	SET FLAG
U	PAUSE
X	x→()
Y	y→()

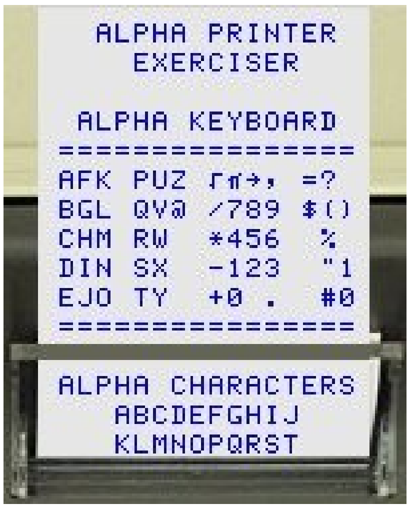
2.4.4 Internal Devices

2.4.4.1 Printer

For the HP9810A the thermo electric printer was optional (Option 004). The emulated HP9810A is equipped with this printer. Use of the printer is identical to the original. The printer output is graphically emulated and the graphics are internally buffered. The 'paper' advances to the top of the displayed window area. Since the 9810 generates the output as printer dots, not as characters, it is impossible to store it in a text file.

The output can be scrolled back and forth with the PC keys Ctrl+Page↑ and Ctrl+↓. The complete output can be erased by Ctrl+Del.

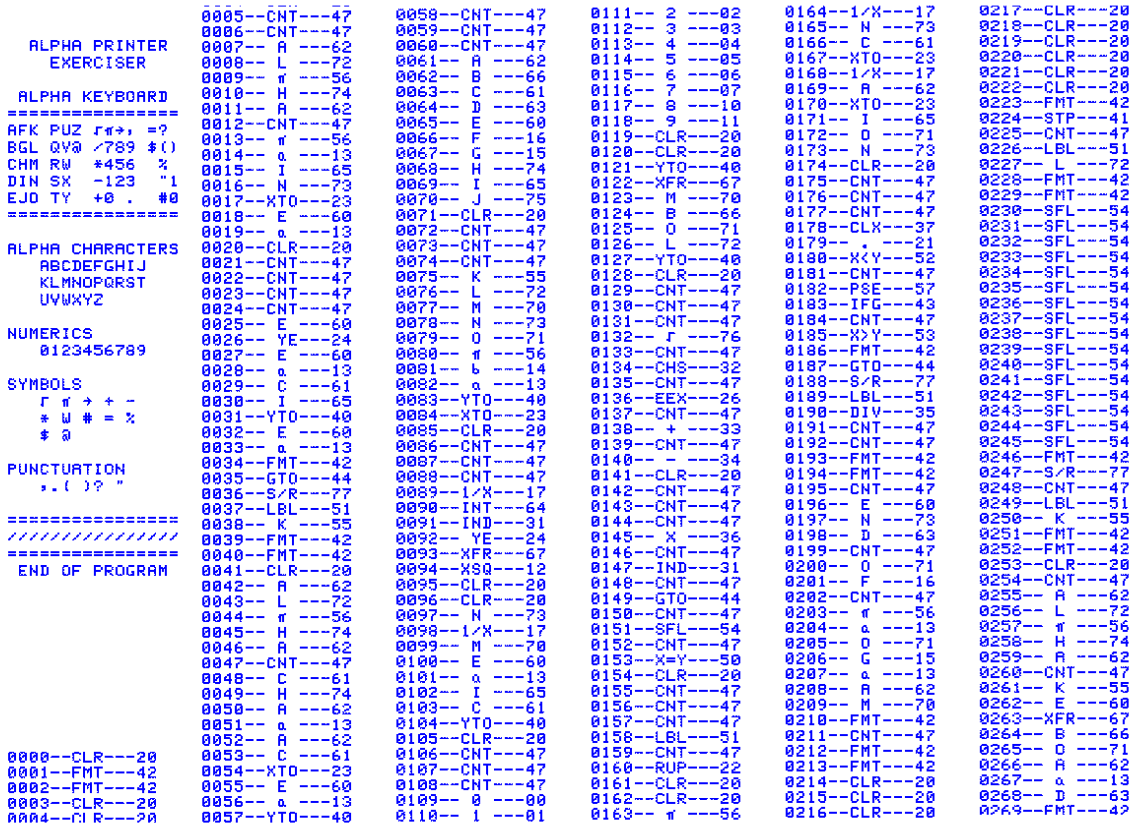
The PAPER advance key can be operated by mouse click or Ctrl+Home keys. Click on the PAPER key and hold the mouse button down to achieve a continuous paper feed.



```
  ALPHA PRINTER
    EXERCISER

 ALPHA KEYBOARD
================
AFK PUZ Γπ→, =?
BGL QV@ /789 $()
CHM RW  *456  %
DIN SX  -123  "1
EJO TY  +0 .  #0
================

ALPHA CHARACTERS
   ABCDEFGHIJ
   KLMNOPQRST
```

HP9810A sample printer output

Since release 1.50 it is possible to send a the printer output to an arbitrary printer of the host PC. Pressing the PC keys Shift+Ctrl+P opens the standard page-setup dialog of the host system. Pressing Ctrl+P opens the standard print dialog of the host system, where the printer may be selected and the hardcopy started. The output is automatically arranged in multiple columns per page, the number depends of the page size and orientation.



```
  ALPHA PRINTER
    EXERCISER

 ALPHA KEYBOARD
================
AFK PUZ Γπ→, =?
BGL QV@ /789 $()
CHM RW  *456  %
DIN SX  -123  "1
EJO TY  +0 .  #0
================

ALPHA CHARACTERS
   ABCDEFGHIJ
   KLMNOPQRST
   UVWXYZ

NUMERICS
   0123456789

SYMBOLS
   Γ π → + -
   * W # = %
   $ @

PUNCTUATION
   ,.( )? "

================
////////////////
================
 END OF PROGRAM

0000--CLR---20
0001--FMT---42
0002--FMT---42
0003--CLR---20
0004--CLR---20
```

```
0005--CNT---47
0006--CNT---47
0007-- A ---62
0008-- L ---72
0009-- π ---56
0010-- H ---74
0011-- A ---62
0012--CNT---47
0013-- π ---56
0014-- a ---13
0015-- I ---65
0016-- N ---73
0017--XTO---23
0018-- E ---60
0019-- a ---13
0020--CLR---20
0021--CNT---47
0022--CNT---47
0023--CNT---47
0024--CNT---47
0025-- E ---60
0026-- YE---24
0027-- E ---60
0028-- a ---13
0029-- C ---61
0030-- I ---65
0031--YTO---40
0032-- E ---60
0033-- a ---13
0034--FMT---42
0035--GTO---44
0036--S/R---77
0037--LBL---51
0038-- K ---55
0039--FMT---42
0040--FMT---42
0041--CLR---20
0042-- A ---62
0043-- L ---72
0044-- π ---56
0045-- H ---74
0046-- A ---62
0047--CNT---47
0048-- C ---61
0049-- H ---74
0050-- A ---62
0051-- a ---13
0052-- A ---62
0053-- C ---61
0054--XTO---23
0055-- E ---60
0056-- a ---13
0057--YTO---40
```

```
0058--CNT---47
0059--CNT---47
0060--CNT---47
0061-- A ---62
0062-- B ---66
0063-- C ---61
0064-- D ---63
0065-- E ---60
0066-- F ---16
0067-- G ---15
0068-- H ---74
0069-- I ---65
0070-- J ---75
0071--CLR---20
0072--CNT---47
0073--CNT---47
0074--CNT---47
0075-- K ---55
0076-- L ---72
0077-- M ---70
0078-- N ---73
0079-- O ---71
0080-- π ---56
0081-- b ---14
0082-- a ---13
0083--YTO---40
0084--XTO---23
0085--CLR---20
0086--CNT---47
0087--CNT---47
0088--CNT---47
0089--1/X---17
0090--INT---64
0091--IND---31
0092-- YE---24
0093--XFR---67
0094--XSQ---12
0095--CLR---20
0096--CLR---20
0097-- N ---73
0098--1/X---17
0099-- M ---70
0100-- E ---60
0101-- a ---13
0102-- I ---65
0103-- C ---61
0104--YTO---40
0105--CLR---20
0106--CNT---47
0107--CNT---47
0108--CNT---47
0109-- 0 ---00
0110-- 1 ---01
```

```
0111-- 2 ---02
0112-- 3 ---03
0113-- 4 ---04
0114-- 5 ---05
0115-- 6 ---06
0116-- 7 ---07
0117-- 8 ---10
0118-- 9 ---11
0119--CLR---20
0120--CLR---20
0121--YTO---40
0122--XFR---67
0123-- M ---70
0124-- B ---66
0125-- O ---71
0126-- L ---72
0127--YTO---40
0128--CLR---20
0129--CNT---47
0130--CNT---47
0131--CNT---47
0132-- Γ ---76
0133--CNT---47
0134--CHS---32
0135--CNT---47
0136--EEX---26
0137--CNT---47
0138-- + ---33
0139--CNT---47
0140-- - ---34
0141--CLR---20
0142--CNT---47
0143--CNT---47
0144--CNT---47
0145-- X ---36
0146--CNT---47
0147--IND---31
0148--CNT---47
0149--GTO---44
0150--CNT---47
0151--SFL---54
0152--CNT---47
0153--X=Y---50
0154--CLR---20
0155--CNT---47
0156--CNT---47
0157--CNT---47
0158--LBL---51
0159--CNT---47
0160--RUP---22
0161--CLR---20
0162--CLR---20
0163-- π ---56
```

```
0164--1/X---17
0165-- N ---73
0166-- C ---61
0167--XTO---23
0168--1/X---17
0169-- A ---62
0170--XTO---23
0171-- I ---65
0172-- O ---71
0173-- N ---73
0174--CLR---20
0175--CNT---47
0176--CNT---47
0177--CNT---47
0178--CLX---37
0179-- . ---21
0180--X<Y---52
0181--CNT---47
0182--PSE---57
0183--IFG---43
0184--CNT---47
0185--X>Y---53
0186--FMT---42
0187--GTO---44
0188--S/R---77
0189--LBL---51
0190--DIV---35
0191--CNT---47
0192--CNT---47
0193--FMT---42
0194--FMT---42
0195--CNT---47
0196-- E ---60
0197-- N ---73
0198-- D ---63
0199--CNT---47
0200-- O ---71
0201-- F ---16
0202--CNT---47
0203-- π ---56
0204-- a ---13
0205-- O ---71
0206-- G ---15
0207-- a ---13
0208-- A ---62
0209-- M ---70
0210--FMT---42
0211--CNT---47
0212--FMT---42
0213--FMT---42
0214--CLR---20
0215--CLR---20
0216--CLR---20
```

```
0217--CLR---20
0218--CLR---20
0219--CLR---20
0220--CLR---20
0221--CLR---20
0222--CLR---20
0223--FMT---42
0224--STP---41
0225--CNT---47
0226--LBL---51
0227-- L ---72
0228--FMT---42
0229--FMT---42
0230--SFL---54
0231--SFL---54
0232--SFL---54
0233--SFL---54
0234--SFL---54
0235--SFL---54
0236--SFL---54
0237--SFL---54
0238--SFL---54
0239--SFL---54
0240--SFL---54
0241--SFL---54
0242--SFL---54
0243--SFL---54
0244--SFL---54
0245--SFL---54
0246--FMT---42
0247--S/R---77
0248--CNT---47
0249--LBL---51
0250-- K ---55
0251--FMT---42
0252--FMT---42
0253--CLR---20
0254--CNT---47
0255-- A ---62
0256-- L ---72
0257-- π ---56
0258-- H ---74
0259-- A ---62
0260--CNT---47
0261-- K ---55
0262-- E ---60
0263--XFR---67
0264-- B ---66
0265-- O ---71
0266-- A ---62
0267-- a ---13
0268-- D ---63
0269--FMT---42
```

Example of a printer hardcopy

2.4.4.2 Magnetic Card Reader

Like the HP9865A the internal card reader emulates the original magnetic media by files in the host file system. To prepare a new host card-file one needs a 'blank' card. This can be any simple empty (0 byte length) file, e.g. created with a text editor. Give the file a name of your choice. For convention it should have the extension .mcard.

To write a program in the 9810 memory to a magnetic card simply click on the RECORD key. Then the LED 'INSERT CARD' lights and the sound of the transport motor is audible. A file selector dialog is displayed in which you can select your card file. If you dismiss this dialog with the Cancel button, the motor keeps running and has to be stopped with the STOP key. After successful selection of a card the program will be stored in the card file. As in the original every previously stored contents will be overwritten. Finally the 'INSERT CARD' LED goes out and the motor sound stops.

Unlike the original a program of any size can be stored on one single card.

Reading of a program stored in a magnetic card is straight forward. Also writing and reading of data registers. See the original HP9810A manual for reference.

2.4.5 Notes on use of the HP9865A tape drive

In the meantime the original manual for the Cassette Memory ROM became available to the author. In this manual the functions and FMT-key sequences for the HP9810A are described in detail. A copy of the manual is available from the author on request.

The functions and FMT key sequences are also described in the ROM instructions page implemented in the emulator (see chapter 1.3).

3. External Devices

3.1 HP9860A Marked Card Reader

3.1.1 General Information

The HP9860A Marked Card Reader is an input device for programs and data. In fact it is a keyboard-like device which reads octal key codes from optical markings on a paper card and sends them to the calculator (using interrupts). For the original device there where cards with 30 and 50 lines (key codes) available, so one needed many cards to store a long program. The codes had to be marked by hand with a pencil.

In GO9800 the cards are emulated by a single file in the host PC file system. The card-file has to be written by hand using a simple text editor. The HP9860A can be used in conjunction with all calculators models. With the HP9810A and HP9820A the key codes have to be entered as octal characters, one key code per text line. The first line of the card file has to be 'OCT', e.g.

```
OCT
44
01
00
00
46
```

The octal key codes can be found in the original calculator manuals. For convention the octal card files should have the extension .oct.

In addition to the normal calculator key codes there is a pseudo code 200 (octal) for the Skip marker. On the original cards there was a special skip-column which caused the whole character to be ignored (skipped). This feature could be used to insert a pause in the reading process to allow for completion of somewhat lengthy operations (like STORE on HP9820). This feature is also available on the emulated HP9860A with one difference: the pseudo code 200 creates a wait time of 300ms instead of 30ms like in the original. When skip is combined with the key code 177 (octal) the pseudo key code 377 may be used as end-of-file marker. As on the original all codes behind skip+177 are ignored.

For the HP9830 key codes may also be octal coded but the emulated 9860 has a special 'convenience mode' which accepts BASIC programs stored as normal ASCII text. Such a program file has to begin with the first line 'BAS', e.g.

```
BAS
10 FOR I=1 TO 10
20 DISP I
30 NEXT I
40 END
```

For convention the card files containing BASIC programs should have the extension .bas.

3.1.1.1 Calculator Configuration

The HP9860A has no specific select code and the configuration item is:

```
DEV HP9860A
```

3.1.1.2 *Device Configuration File*

The device configuration file for the HP9860A is:

```
Model HP9810A HP9820A HP9821A HP9830A
Name HP9860A
Title MARKED_CARD_READER
Interface HP11200A
Selectcode 12
```

3.1.2 Usage of the HP9860A

In GO9800 the marked card reader has its own window and runs in a parallel program thread.

To read a card-file into the calculator bring the HP9860A to the foreground and mouse-click on the card input slot (the silver metallic area below the type label). Alternatively the Return or Enter key on the host PC may be pressed. A file selector dialog is displayed in which you can select the prepared card file. After successful selection of a file a marked card is displayed in the reader output area and the transport motor sound is audible. After the complete reading of the card-file the motor sound stops and the marked card is removed from the display window. The reading process may be stopped at any time by mouse-click on the marked card in the output area or by pressing Pause on the host PC keyboard.

Note: Pressing STOP on the calculator doesn't affect the reader operation but may lead to loss of data.

3.1.3 Reading of HP9810A programs

When reading programs into the HP9810A one has to change to PRGM mode before reading the card file. Alternatively one can store the mode switch in the beginning of the card file, e.g.

```
OCT
107
46
106
```

This sequence executes RUN, END, PRGM before reading the program.

3.1.4 Reading of HP9820/21A programs

HP9820A programs have to be octal coded. The octal key codes may be found in the Appendix II of the original HP9820A manual. To avoid loss of characters each STORE command (code 002) has to followed by one or more Skips (code 200) to give the calculator time to execute the STORE.

3.1.5 Reading of HP9830A programs

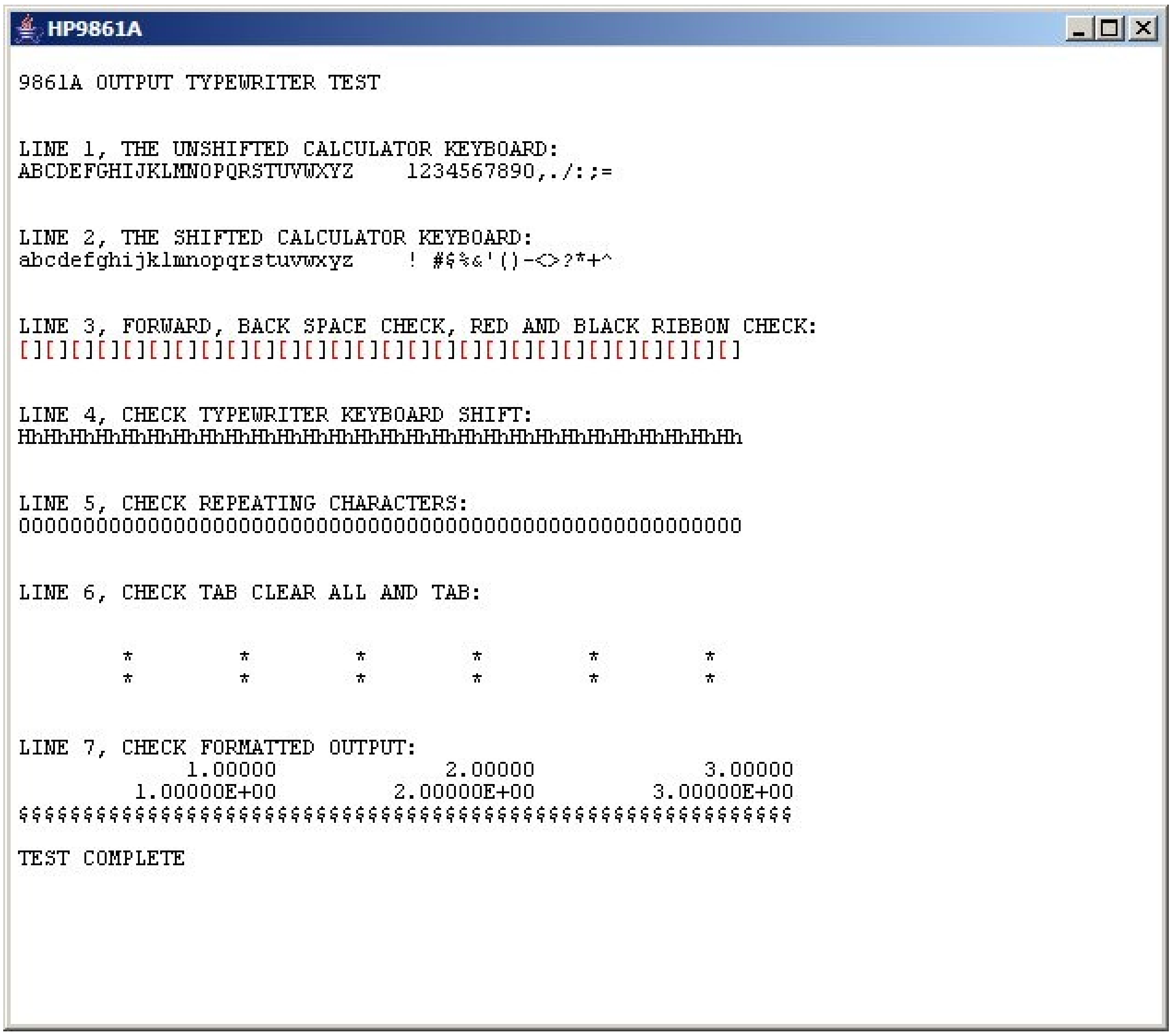
HP9830A programs can be BASIC files coded in ASCII. The emulated HP9860A automatically inserts a 300ms pause after each program line to allow for execution of the END OF

LINE key.

3.2 HP9861A Output Typewriter

3.2.1 General Information

The original HP9861A was a ASCII text impact printer based on a Facit 3841 electric type-writer. The maximum line width was 162 characters. For use with the HP9810A the Type-writer or Peripheral Control ROM is necessary. The Peripheral Control ROM is also re-quired for use with the HP9820/21A.



```
9861A OUTPUT TYPEWRITER TEST


LINE 1, THE UNSHIFTED CALCULATOR KEYBOARD:
ABCDEFGHIJKLMNOPQRSTUVWXYZ    1234567890,./:;=


LINE 2, THE SHIFTED CALCULATOR KEYBOARD:
abcdefghijklmnopqrstuvwxyz    ! #$%&'()-<>?*+^


LINE 3, FORWARD, BACK SPACE CHECK, RED AND BLACK RIBBON CHECK:
[][][][][][][][][][][][][][][][][][][][][][][][][][][][]


LINE 4, CHECK TYPEWRITER KEYBOARD SHIFT:
HhHhHhHhHhHhHhHhHhHhHhHhHhHhHhHhHhHhHhHhHhHhHhHhHhHhHhHh


LINE 5, CHECK REPEATING CHARACTERS:
00000000000000000000000000000000000000000000000000000000


LINE 6, CHECK TAB CLEAR ALL AND TAB:


        *        *        *        *        *        *
        *        *        *        *        *        *


LINE 7, CHECK FORMATTED OUTPUT:
             1.00000             2.00000             3.00000
         1.00000E+00         2.00000E+00         3.00000E+00
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

TEST COMPLETE
```

HP9861A output of HP9830A printer test program. Note the red ribbon color below LINE 3.

3.2.1.1 Calculator Configuration

In the configuration file the select code has to be set to an unused value between 1 and 15. The standard value which is assumed in the HP9810A and HP9820A ROMs is 15:

```
DEV HP9861A 15
```

For use with the HP9810A or HP9820A an additional Typewriter or Peripheral Control ROM is necessary and has to be listed in the configuration file.

3.2.1.2 Device Configuration File

The device configuration file for the HP9861A is:

```
Model HP9810A HP9820A HP9821A HP9830A
Name HP9861A
Title TYPEWRITER
Interface HP11201A
Selectcode 1 2 3 4 5 6 7 8 9 15
```

3.2.2 Usage of the HP9861A

The typewriter is controlled by the HP9810A using the FORMAT key and by the HP9820/21A using the TYPE key. Refer to the original manuals of the Typewriter and Peripheral Control ROMs for further informations.

On the HP9830A the typewriter is controlled by the WRITE command. Also the PRINT # and LIST # commands with the proper select code may be used.

The HP9861A uses its own window for output. For text output the JAVA Monospaced font is used, so some characters may look different from the original. Red and black ribbon colors are supported as well as backspace / overtype and tabulator functions. The font size may be increased and decreased in integer steps between 8 and 40. The output is automatically scrolled line by line in the window and is internally buffered. The window may be resized manually.

The buffered output maybe controlled by the following PC keys:

PC key	**Function**
page ↑	Scroll output one window page up
page ↓	Scroll output one window page down
home	Position to first (top) window page
end	Position to last window page
delete	Delete complete output
shift + delete	Delete complete output, reset to default font and window size
insert	Generates a hardcopy of the output
shift + insert	Opens the page format dialog for the hardcopy function.
+	Increase font size (max. 40)
-	Decrease font size (min. 8)
S	Toggle high-speed mode

It is possible to send a the output to an arbitrary printer of the host PC. Pressing the PC keys Shift+Ctrl+P opens the standard page-setup dialog of the host system. Pressing Ctrl+P opens the standard print dialog of the host system, where the printer may be selected and the hardcopy started. The hardcopy is automatically scaled to fill the available page size.

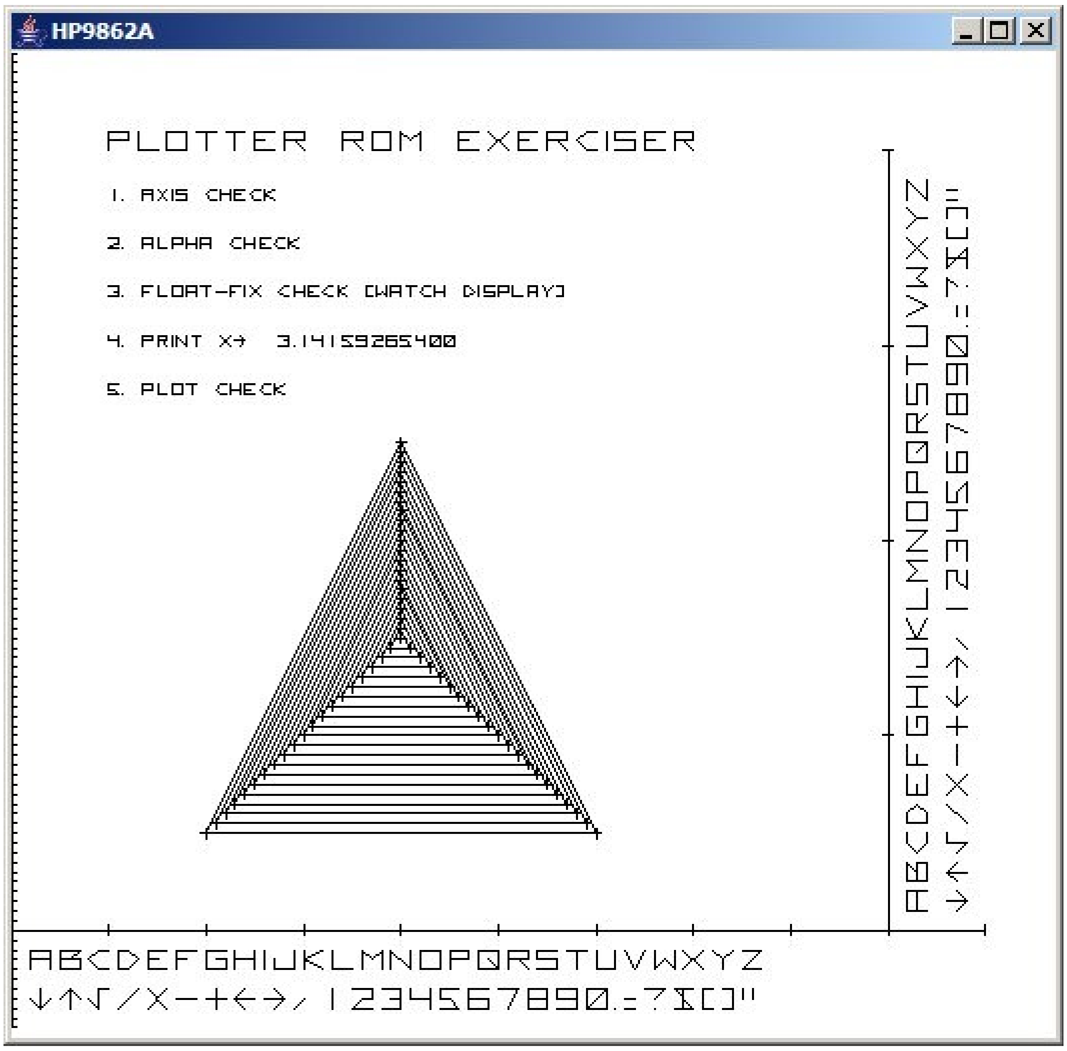
The emulated HP9861A has a special high-speed output mode, where all artificial delays are eliminated and the print output is generated at the maximum possible speed. In normal output mode the delay resembles the timing of the real device. In high-speed mode the printing sound is also disabled.

When the HP9861A window has the focus pressing the key S toggles between high-speed and normal speed output. The high-speed mode is visualized in the window title bar.

3.3 *HP9862A Plotter*

3.3.1 General Information

The original HP9862A is a single pen flatbed x/y plotter with a resolution of 10000 by 10000 units. The emulation uses its own window for drawing. The graphic output is internally buffered so the window can be resized as desired and the graphics will be redrawn.



```
PLOTTER ROM EXERCISER
1. AXIS CHECK
2. ALPHA CHECK
3. FLOAT-FIX CHECK [WATCH DISPLAY]
4. PRINT X→ 3.14159265400
5. PLOT CHECK
ABCDEFGHIJKLMNOPQRSTUVWXYZ
↓↑√/X-+←→, 1234567890.:?Σ[]"
```

HP9862A output of HP9810A plotter ROM exerciser program

Different from the original the emulated HP9862A is able to change the pen color. This is achieved by driving the pen position beyond the right border and 'grab' one of the 15 virtual color pens. To change the pen color to N (1 <= N <= 15) output a plot command with x,y coordinates of (10000,N), e.g. on the HP9830A:

```
SCALE 0,9999,0,9999
PLOT 10000,N.
```

The x-value of the right border of course depends on the scaling factor.

The colors are defined as follows:

1	black (default color)
2	green
3	red
4	blue
5	cyan
6	magenta
7	yellow
8	orange
9-15	black

3.3.1.1 Calculator Configuration

The select code of the HP9862A is fixed to 14 by the interface. In the configuration file the select code has to be omitted:

```
DEV HP9862A
```

3.3.1.2 Device Configuration File

The device configuration file for the HP9862A is:

```
Model HP9810A HP9820A HP9821A HP9830A
Name HP9862A
Title PLOTTER
Interface HP9862Interface
Selectcode 14
```

3.3.2 Usage of the HP9862A

In GO9800 the plotter has its own window and runs in a parallel program thread. Plot commands can only be issued by the calculator. Every calculator needs a special add-on ROM to control the plotter, these are delivered with the GO9800 emulator. See the appropriate original manual for further informations.

The buffered output maybe controlled by the following PC keys:

PC key	Function
delete	Delete complete output
shift + delete	Delete complete output, reset to default window size
insert	Generates a hardcopy of the output
shift + insert	Opens the page format dialog for the hardcopy function.
S	Toggle high-speed mode

The plot output may be cleared by the Del key on the host PC. When the Shift key is pressed simultaneously, the output window will be resized to the default size of 500x500.

It is possible to send a the output to an arbitrary printer of the host PC. Pressing the PC keys Shift+Ctrl+P opens the standard page-setup dialog of the host system. Pressing Ctrl+P opens the standard print dialog of the host system, where the printer may be selec-ted and the hardcopy started.

The emulated HP9862A has a special high-speed output mode, where all artificial delays are eliminated and the plot output is generated at the maximum possible speed. In normal output mode the delay resembles the timing of the real device. In high-speed mode the plotting sound is also disabled.

When the HP9862A window has the focus pressing the key S toggles between high-speed and normal speed output. The high-speed mode is visualized in the window title bar.

3.4 HP9865A Cassette Memory

3.4.1 General Information

The original HP9865A used modified audio compact cassettes for storage of programs and data. The original tape cassettes are emulated by files in the host PC file system. One host tape-file can contain a unlimited number of HP calculator files. Like on a real tape these files have to be created by the calculator using a MARK command. The movement of the tape is emulated by a read/write position in the host file. When a tape is loaded in the HP9865A the r/w position is 0 (zero). The position moves forward when reading or writing HP files on the tape. It moves backward when a REWIND or FIND is executed. When the r/w position reaches the begin or end of the host file a 'clear leader' is signalled to the calculator, which automatically stops the HP9865A. So the emulated device behaves like the real one.

To prepare a new host tape-file one needs a 'blank' tape. A original blank tape is not really empty, but contains at least eight 0-value bytes and a Begin-Of-File marker (hex 3C + control bit). Without that initial BOF no new files can be MARKed on that tape. The zero value bytes before BOF are necessary to prevent the drive running into a 'clear leader' position during FIND or LOAD, which would result into an error message on the HP9830A. In the GO9800 distribution there is file named 'blank.tape' which should be kept as source for new tapes. To create new tape make a copy of the file 'blank.tape' and give it a name of your choice. For convention it should have the extension .tape.

3.4.1.1 Calculator Configuration

In the configuration file the select code has to be set to an unused value between 1 and 9. The standard value which is assumed in the HP9810A and HP9820A cassette memory ROMs is 5:

```
DEV HP9865A 5
```

3.4.1.2 Device Configuration File

The device configuration file for the HP9865A is:

```
Model HP9810A HP9820A HP9821A HP9830A
Name HP9865A
Title CASSETTE_MEMORY
Interface HP9865Interface
Selectcode 5 1 2 3 4 6 7 8 9 10
```

3.4.2 Usage of the HP9865A

In GO9800 the cassette memory has its own window and runs in a parallel program thread.

To load a tape in the HP9865A bring the HP9865A to the foreground and mouse-click on the 'OPEN' door lever. The cassette door will be opened and a file selector dialog is displayed in which you can select your new empty tape. After successful selection of a tape the door will be closed again and a loaded cassette is visible in the door window.

The tape can only be read or written by commands issued on the calculator. On the HP9830A these commands are build-in, on the HP9810A and HP9820A one needs a

'Cassette Memory' ROM block. See the original manuals for reference.

The only tape operation which can be directly executed by the HP9865A is rewind. Mouse-click on the REWIND button places the read/write position to the beginning of the tape.

In the emulated drive the tape activity is not obvious since there are no moving reels in the emulator and the only feedback is acoustical by the simulated motor sounds. Therefore a tape activity indicator was added which shows if the tape drive is busy by using colored chevrons.

The tape activity indicators have the following meaning:

>	move tape forward slow
<	move tape backward slow
>>	move tape forward fast
<<	move tape backward fast

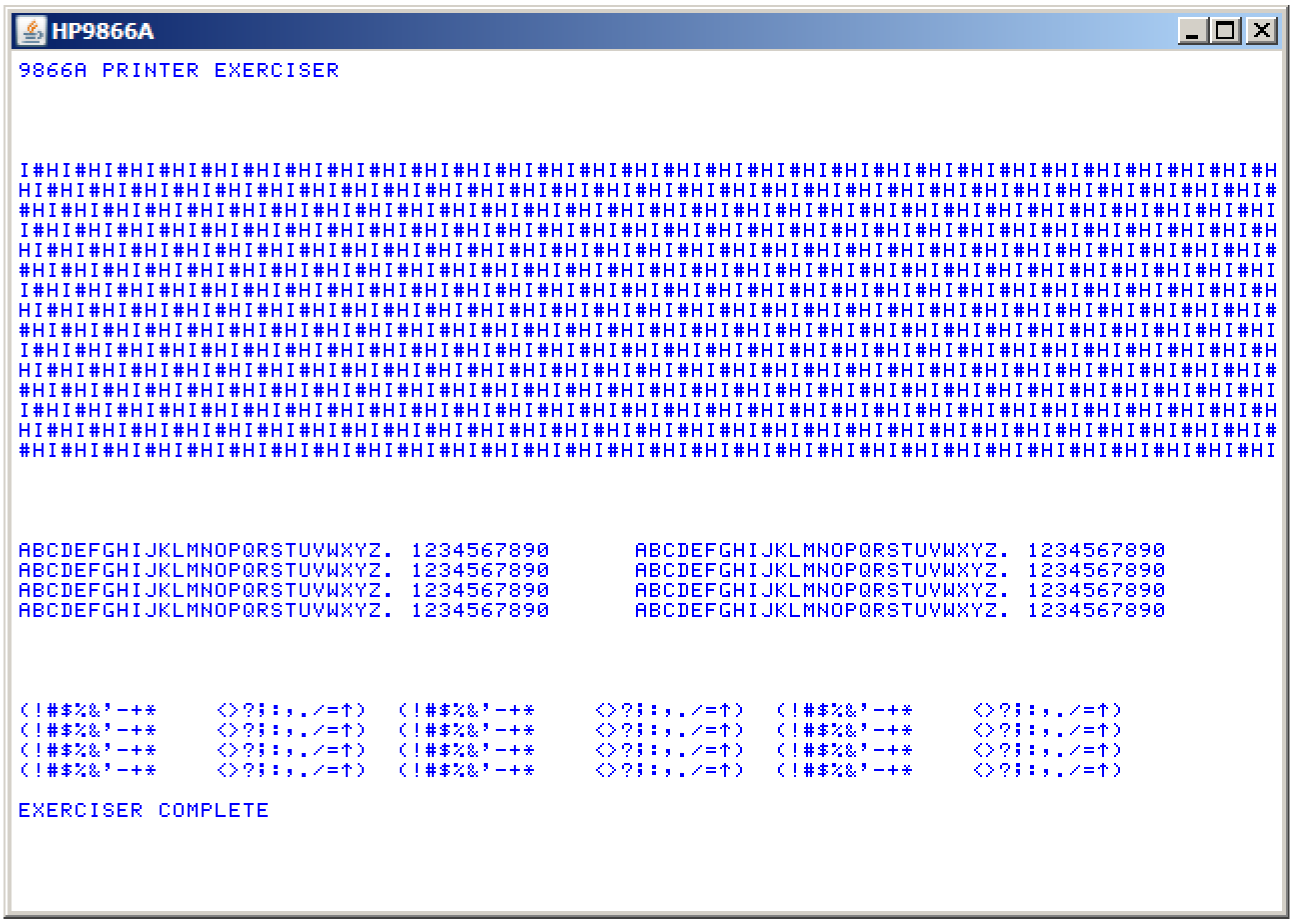
The color coding is:

yellow	search for control word (begin-of-file)
green	read from tape
red	write to tape

3.5 HP9866A/B Thermal Line Printer

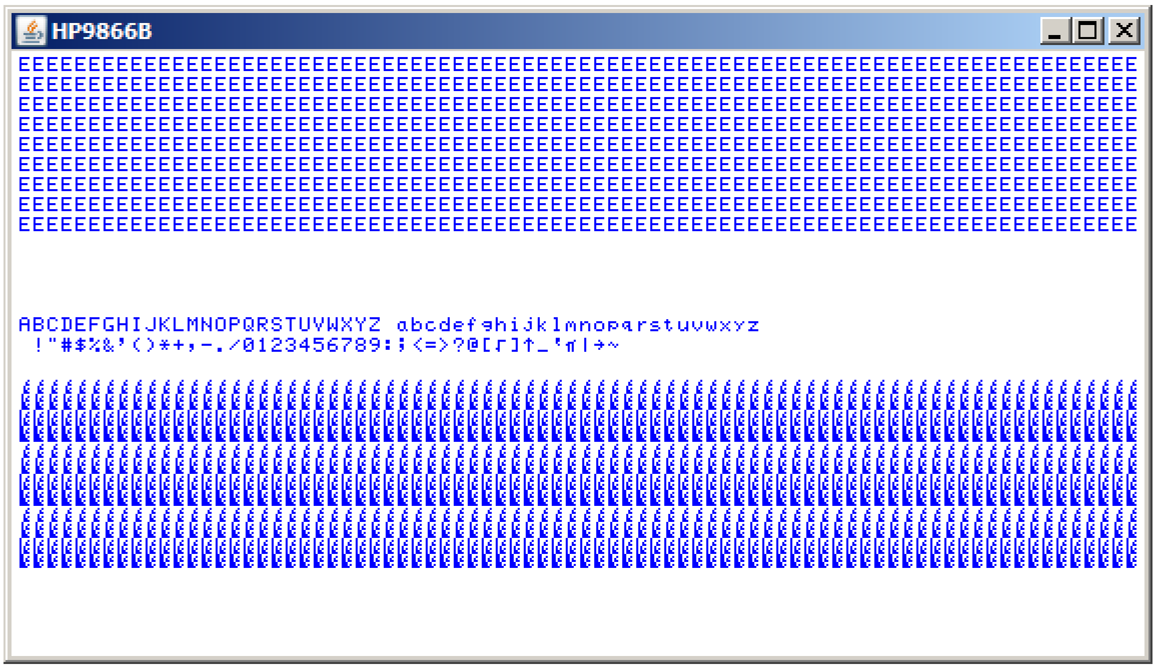
3.5.1 General Information

The original HP9866A was a ASCII text printer which prints one complete line at a time on thermo sensitive roll paper. It was the primary printer for the HP9830A which had no internal printer like the HP9810 and HP9820. The HP9830A had a build-in interface for direct connection of the printer. For use with other calculators a special interface card was necessary. The HP9866B was the successor of the HP9866A with extended character set and graphics capability.



```
9866A PRINTER EXERCISER



I#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#H
HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#
#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI
I#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#H
HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#
#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI
I#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#H
HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#
#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI
I#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#H
HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#
#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI
I#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#H
HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#
#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI#HI



ABCDEFGHIJKLMNOPQRSTUVWXYZ. 1234567890      ABCDEFGHIJKLMNOPQRSTUVWXYZ. 1234567890
ABCDEFGHIJKLMNOPQRSTUVWXYZ. 1234567890      ABCDEFGHIJKLMNOPQRSTUVWXYZ. 1234567890
ABCDEFGHIJKLMNOPQRSTUVWXYZ. 1234567890      ABCDEFGHIJKLMNOPQRSTUVWXYZ. 1234567890
ABCDEFGHIJKLMNOPQRSTUVWXYZ. 1234567890      ABCDEFGHIJKLMNOPQRSTUVWXYZ. 1234567890



(!#$%&'-+*    <>?;:,./=↑)  (!#$%&'-+*    <>?;:,./=↑)  (!#$%&'-+*    <>?;:,./=↑)
(!#$%&'-+*    <>?;:,./=↑)  (!#$%&'-+*    <>?;:,./=↑)  (!#$%&'-+*    <>?;:,./=↑)
(!#$%&'-+*    <>?;:,./=↑)  (!#$%&'-+*    <>?;:,./=↑)  (!#$%&'-+*    <>?;:,./=↑)
(!#$%&'-+*    <>?;:,./=↑)  (!#$%&'-+*    <>?;:,./=↑)  (!#$%&'-+*    <>?;:,./=↑)

EXERCISER COMPLETE
```

HP9866A output of HP9830A printer test program



```
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE


ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz
 !"#$%&'()*+,-./0123456789:;<=>?@[┌]↑_'π|→~
```

HP9866B output of HP9830A printer test program

3.5.1.1 *Calculator Configuration*

In the GO9800 the HP9866A/B can be used with the HP9830A as well as with the other calculators. For use with the HP9830A/B the select code must be set to 15.

Typical configuration item:

```
DEV HP9866A 15
```

For use with the HP9810A or HP9820A an additional Peripheral Control ROM is necessary and has to be listed in the configuration file. The select code has to be set to an unused value between 1 and 9.

Example:

```
DEV HP9866B 8
```

The HP9866A/B may also be controlled by the HP9810A Typewriter ROM. The select code must then be set to 15.

3.5.1.2 *Device Configuration File*

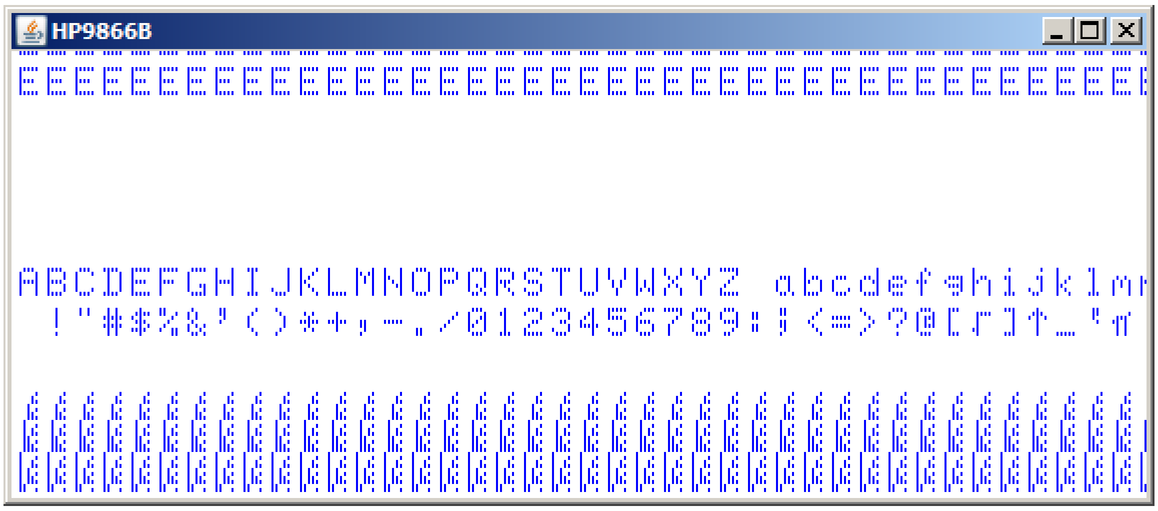
The device configuration file for the HP9866A is:

```
Model HP9810A HP9820A HP9821A HP9830A
Name HP9866A
Title LINE_PRINTER
Interface HP9866Interface
Selectcode 1 2 3 4 5 6 7 8 9 15
```

3.5.2 Usage of the HP9866A/B

The HP9830A commands PRINT, LIST, and TLIST can be used like on the original machine.

The HP9866A uses its own window for output. For text output the internal dot matrix font is emulated. The font size may be changed in fixed steps, each step increases the hight of a print dot by one display pixel. The original printer dots where slightly elliptic, this is only visible with font sizes larger than 7. The HP9866B has an additional graphics printing mode which is enabled for one text line by output of a DC1 character (decimal code 17). The lower five bits of each character in the line are printed as dot pattern. Detailed description of the HP9866B graphics may be found in the “HP9866A/B Options 10, 20, 21 and 30 Operating Manual” on pages 3-3 ff. A copy of this manual is available from the author on request. The output is automatically scrolled line by line in the window and is internally buffered. The window may be resized manually.



```
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE



ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklm
 !"#$%&'()*+,-./0123456789:;<=>?@[\]↑_'
```

HP9866B output with font size 2

The buffered output maybe controlled by the following PC keys:

PC key	**Function**
page ↑	Scroll output one window page up
page ↓	Scroll output one window page down
home	Position to first (top) window page
end	Position to last window page
delete	Delete complete output
shift + delete	Delete complete output, reset to default font and window size
insert	Generates a hardcopy of the output
shift + insert	Opens the page format dialog for the hardcopy function.
+	Increase font size (max. 10)
-	Decrease font size (min. 1)
S	Toggle high-speed mode

It is possible to send the output to an arbitrary printer of the host PC. Pressing the PC keys Shift+Ctrl+P opens the standard page-setup dialog of the host system. Pressing Ctrl+P opens the standard print dialog of the host system, where the printer may be selected and the hardcopy started. The hardcopy is automatically scaled to fill the available page size. The scaling accounts for the selected font size.

The emulated HP9866A/B has a special high-speed output mode, where all artificial delays are eliminated and the print output is generated at the maximum possible speed. In normal output mode the delay resembles the timing of the real device. In high-speed mode the printing sound is also disabled.

When the HP9866A window has the focus pressing the key S toggles between high-speed and normal speed output. The high-speed mode is visualized in the window title bar.

3.6 HP9880A/B Mass Memory System

3.6.1 General Information

The original HP9880A/B was a storage system with fixed or removable magnetic discs with a capacity of 2.5 Mbytes each. It comprised of a HP11273B interface, a HP11305A disc controller unit and one or more HP9867A/B disc drives. Each HP11305A controller was able to connect up to four disc units, either up to four HP9867A single platter or two HP9867B dual platter drives. In the emulator both the HP9880A (with HP9867A drives) and the HP9880B (with HP9867B drives) is available.

The HP9867A has a single removable disc, while the HP9867B has one fixed (lower) and one removable (upper) disc. Each disc can be accessed by a unit number between 0 and 3. In the HP9867B the upper disc has the unit number 0 or 2, and the lower, fixed disc the unit number 1 or 3.

The physical disc is emulated by a binary file in the host PC file system. Before it can be used a new disc has to be initialized. The procedure is described below. Alternatively an initialized empty disc may be used. In the GO9800 distribution there is file named 'empty.disc' which should be kept as source for new discs. To create new disc make a copy of the file 'empty.disc' and give it a name of your choice. For convention it should have the extension .disc.

3.6.1.1 Calculator Configuration

In the GO9800 the HP9880A/B can be used with the HP9830A only. The select code is internally fixed to 11. There are 5 device configuration files for different combinations of drive and disk numbers.

```
HP9880A   (one HP9880A drive with one disk)
HP9880A_2 (two HP9880A drives with one disk each)
HP9880A_4 (four HP9880A drives with one disk each)
HP9880B   (one HP9880B drives with two disks)
HP9880B_2 (two HP9880B drives with two disks each)
```

Each of these device configurations may be referred in the calculator configuration.

Typical configuration items:

```
DEV HP9880A_4
DEV HP9880B
```

For use with the HP9830A an additional Mass Memory ROM is necessary and has to be listed in the configuration file. The ROM has to be placed in the uppermost slot in order to be functional. The corresponding configuration item is:

```
ROM 036000 002000 HP11273B Slot1
```

The HP11273B interface connects the HP11305A controller to the calculator and contains a 256-word cache memory which is mapped in the main memory at address 77000-77377. This cache is automatically configured in the HP9880A configuration file.

3.6.1.2 Device Configuration File

The device configuration file for the HP9880A is:

```
Model HP9830A
Name HP11305A
Title MASS_MEMORY_CONTROLLER
Interface HP11273A
Selectcode 11
RWM 077000 000400
Parameters 1 1
```

3.6.2 Usage of the HP9880A/B

In the emulator the HP9880 has a user interface showing the front panel of each HP9867 disc drive in its own window.

At startup each configured disc unit is loaded with a standard disc named

'HP9880-UNIT*n*.disc'

where *n* is the unit number.

A disc can be exchanged by mouse-clicking on the 'LOAD' switch on the right side of the HP9867. The 'DRIVE READY' lamp will go out and the 'DOOR UNLOCKED' lamp will be lighted. Then a file selector dialog is displayed in which a new disc-file can be selected. After successful selection of a disc the 'DRIVE READY' lamp will be lighted again and the other lamp will go out. If the dialog is dismissed without selecting a disc-file the drive will stay in the not-ready state and can not be operated until another disc is loaded.

The HP9867A/B drives have a write-protect mechanism which can be activated separately for each disc. Mouse-click on the 'PROTECT U.D.' or 'PROTECT L.D.' area of the HP9867 front panel and the corresponding disc will be write protected. In the HP9867A only the upper disc is present and can be protected.

In the emulated drive the disc activity is not obvious, therefore a disc activity indicator was added which shows which which disc record is currently accessed. The physical records have numbers between 0 and 9743 (corresponding to 2 surfaces with 203 cylinders and 24 sectors each).

The disc activity indicators have the following meaning:

Unnnn	HP9867B upper disc record nnnn
Lnnnn	HP9867B lower disc record nnnn
nnnn	HP9867A disc record nnnn

The color coding is:

yellow	Record being initialized
green	Record being read from disc
red	Record being written to disc

The emulated HP9867A/B has a special high-speed output mode, where all artificial delays are eliminated and the disc access is performed at the maximum possible speed. In normal output mode the delay approximates the timing of the real device.

When the HP9867A/B window has the focus pressing the key S toggles between high-speed and normal speed output. The high-speed mode is visualized in the window title bar.

3.6.3 Initializing discs

Before a new disc can be used for data storage it has to be initialized (formatted). This can be accomplished by a utility program provided on the HP9880 Systems Tape. This tape is included in the GO9800 distribution as file

```
applications/HP9830/HP11273-60004 SYSTEMS TAPE.tape
```

The initialization procedure is described in the HP9880 operation manual and shortly outlined here. Load the systems tape into the HP9830 tape drive and execute

LOAD BIN 60

When the file is loaded execute

INITIALIZE

and follow the instructions printed on the HP9866A printer. When you are requested to set the mode switch to INITIALIZE on the controller, simply ignore this and continue.

3.6.4 Note on Usage of the Infotek Fast Basic II ROM

The Infotek Fast Basic II ROM contained in the GO9800 distribution has a feature which automatically executes the command

GET “↑”,10,10

when the HP9830 calculator is switched on. This command searches for a BASIC program file named “↑” on disc unit 0. If present, this file is loaded and started from line 10.

If at startup of the HP9830 the disc unit 0 is not ready or not present, the calculator beeps periodically and waits for unit 0 to become available. This can be abandoned pressing the STOP key.

If the file “↑” is not present on the disc ERROR 94 is displayed. If the file is not executable ERROR 98 occurs.

If the Infotek Fast Basic II ROM is not necessary and the above behavior is bothersome the ROM should be removed from the HP9830A configuration file. The following configured line should be deleted:

```
ROM 017000 001000 INFOTEK_FB2 Int0
```

3.7 HP11202A I/O Interface

3.7.1 General Information

The original HP11202A interface was a bidirectional 8 bit TTL interface. Like other peripheral interfaces it was plugged into one of the four slots on the rear side of the calculator. It could be connected by a cable to output devices like printers (e.g. HP9866A, HP9871A) or input devices like a card reader (HP9869A).

For control of the HP11202A and devices connected to it by the calculator functions from special add-on ROM blocks (Peripheral Control Block I or II for the HP9810A, HP9820A, and Extended I/O Block for the HP9830A) had to be used. Some simple functions where also implemented in the basic machines. The basic HP9830A was able to read programs from a HP9869A (PTAPE#) or output to a connected printer (LIST#).

3.7.1.1 Calculator Configuration

In the configuration file the select code has to be set to an unused value between 1 and 9. A standard value which is assumed for HP9869A paper tape reader and other devices is 1:

```
DEV HP11202A 1
```

3.7.1.2 Device Configuration File

The device configuration file for the HPA is:

```
Model HP9810A HP9820A HP9821A HP9830A
Name HP11202HostFileIO
Title Host_File_IO
Interface HP11202A
Selectcode 1 2 3 4 5 6 7 8 9
Parameters 16
```

The HP11202A can handle I/O data in different formats: as binary bytes or number strings in an arbitrary radix, e.g. as decimal, octal or hexadecimal numbers. The radix is configurable as an additional parameter.

`Parameters Bin`
configures binary format. Data is input or output as binary bytes.

`Parameters <n>`
configures input and output as number string in radix *n*. Each value is separated by a newline character. E.g.

`Parameters 8`
configures input and output as octal number string (e.g. 77). This is also the default format if the parameter is omitted.

3.7.2 Usage of the HP11202A

From the beginning there is no special user dialog for the HP11202A. Instead when one of the input or output commands is executed for the first time with the interface select code, a file dialog box is displayed. Then the location and name of an input or output file has to be entered. This file will be used for the current and following input resp. output commands. There are two different files possible: one for input and another for output operations. So if both input and output operations occur over the same interface, two file selector dialogues are displayed.

Once an output or input file is opened an audit trail window is drawn in which every output byte will be displayed. The audit trail may be scrolled, formatted and printed in the same way as in the HP9861A (see chap. 3.2.2). Manual closing of the audit trail window causes any output or input file also to be closed.

When the end-of-file condition of an input file is met during an input operation, another file selector box will be displayed. If this dialog is dismissed by Cancel or no file is selected, a pause of 5 seconds is carried out. During this pause the input process can be cancelled by pressing the calculator STOP key. After the 5 second pause the file dialog is displayed again and the user gets a second chance.

3.7.3 Saving and Loading of HP9810A programs

Using the integrated Peripheral Control ROM HP9810A programs can be saved to a file in the host PC and later reloaded without using slow magnetic cards. To save a program press the following keys:

```
FORMAT  4  0  1  RECORD
```

To reload a saved program press:

```
FORMAT  3  0  1  GOTO
```

After the complete program is loaded the reading doesn't stop automatically but the file selector is displayed again. Dismiss this dialog by pressing Cancel and press STOP on the HP9810A emulation.

3.7.4 Saving and Loading of HP9830A programs

HP9830A programs can be saved to a host PC file using the LIST command:

```
LIST #1
```

The resulting text file may be opened by a text editor and printed.

To load a program from a text file enter:

```
PTAPE #1
```

After the complete program is loaded the reading doesn't stop automatically but the file selector is displayed again. Dismiss this dialog by pressing Cancel and press STOP on the HP9830 emulation.

4. Installation and Running

The emulator requires a JAVA runtime environment 1.6.0 or higher. The JAVA runtime may be downloaded from http://java.sun.com/javase/downloads/index.jsp. The emulator consists of a single JAR-archive which can be placed anywhere in the file system.

To run specific calculator a configuration file is needed (see chapter 1.1), which contains informations about the calculator model, the memory, and attached peripheral devices. The name of the configuration file is arbitrary.

Run the emulator by entering the command line

```
    java -jar GO9800.jar <configuration-file>
```

During startup of the emulator a copyright note and various messages about configured components and devices are output to the command console. At the beginning a timing calibration is performend to evaluate critical timing constants which may be influenced by the host platform.

A typical console output looks like this:

```
HP Series 9800 Emulator Release 1.60, Copyright (C) 2006-2012 Achim Buerger

GO9800 comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to
redistribute it under certain conditions.

GO9800 is in no way associated with the Hewlett-Packard Company.
Hewlett-Packard, HP, and the HP logos are all trademarks of the Hewlett-Packard
Company.
This software is intended solely for research and education purposes.


HP9800 I/O bus loaded.
Timing calibration in progress  1/1 3/3 7/7 10/10 12/12 16/16 30/30 32/32
100/100 done.
HP9800 I/O register loaded.
HP9800 CPU loaded.
Custom configuration file HP9810A.cfg loaded.
HP9810A_System1 ROM at 0-777, Block0 (SYSTEM1) loaded.
HP9810A_System2 ROM at 4000-5777, Block4 (SYSTEM2) loaded.
HP9810A_System3 ROM at 16000-16777, Block16 (SYSTEM3) loaded.
HP9810A_Data RWM at 1000-1777, Block1 initialized.
HP9810A_Program RWM at 12000-12777, Block12 initialized.
HP11217A RWM at 13000-13777, Block13 initialized.
HP11218A RWM at 14000-15777, Block14 initialized.
HP11210A ROM at 2000-3777, Slot1 (MATHEMATICS) loaded.
HP11267A ROM at 6000-7777, Slot2 (TYPEWRITER/CASSETTE_MEMORY) loaded.
HP11262A ROM at 10000-11777, Slot3 (PERIPHERAL_CONTROL/CASSETTE_MEMORY) loaded.
HP9860A MARKED_CARD_READER HP11200A, select code 12 loaded.
HP9861A TYPEWRITER HP11201A, select code 15 loaded.
HP9862A PLOTTER HP9862Interface, select code 14 loaded.
HP9865A CASSETTE_MEMORY HP9865Interface, select code 5 loaded.
HP9866A LINE_PRINTER HP9866Interface, select code 8 loaded.
HP11202HostFileIO Host_File_IO HP11202A, select code 1  mode=16 loaded.
HP11202HostFileIO Host_File_IO HP11202A, select code 2  mode=Bin loaded.
```

```
Default configuration file C:\Java\9800 Emulator/config/keynames.cfg loaded.
Default configuration file C:\Java\9800 Emulator/config/HP9810A-keyb.cfg loaded.
HP9800 Printer loaded.
HP9800 Magnetic Card Reader loaded.
HP9810 Display loaded.
HP9810 KeyLEDs loaded.
HP9810 Keyboard loaded.
HP9810 Mainframe loaded.
HP9800 Emulator started.
```

5. Known Issues

5.1 HP9830A

Writing and reading of files in the internal or external HP9865A tape device is timing critical and may hang up on some machines. This due to timing and CPU load problems on the host machine.

5.2 Peripheral Devices

The devices with critical timing and operation in background, like HP9865A and the internal card reader, may not work properly on slow host machines. Symptoms are error messages, loss of data, and hangup of the emulated machine.

5.3 Java Issues

5.3.1 Direct3D Problem

In the Java 6 runtime there is an issue with Microsoft DirectX output of graphics which may lead to rather slow drawing or flickering of bitmaps. This is mainly visible with the output of the LED display on the HP9820A. There is a workaround in deactivating the usage of Direct3D acceleration by the Java runtime. This is accomplished by adding the option -Dsun.java2d.d3d=false when starting the Java runtime.

This is already done in the startup CMD files for each calculator contained in the distribution. A typical Java call looks like this:

```
java -Dsun.java2d.d3d=false -jar GO9800.jar HP9810A
```

You may try if this option has an effect on drawing speed on your particular host system.

5.3.2 Window Resize Problem

In Java implementations for certain operating systems, e.g. MS Windows, resizing of an application window may lead to irregular or incomplete repaint of the window contents. This occurs especially with the resizable windows of the HP9862A and HP9866A/B devices. When the window size is enlarged by dragging the window corner with the mouse, the paint() method of the Frame is called very frequently. But obviously only the new exposed areas of the window are part of a system generated clipping area, so only these are painted according to the new window size. Since the scaling and position of the graphics (plot or print) depends on the actual window size, the new painted area doesn't fit in most cases to the existing graphics.

This problem is partially solved by painting the complete window once again after the window resize is finished by releasing the mouse button.

6. The Project

The objectives of this project where to understand and preserve the firmware of the Hewlett-Packard series 9800 calculators and to create a fully functional real-time simulation of the complete hardware.

Of course there where some legal concerns which had to be examined. As of April 2007 all patents are long since expired and there are no registered trademarks affected except the HP logo. The firmware ROMs, software tapes and magnetic cards don't carry any copyright notice. Also the contents of the most ROMs are disclosed as source code in the appropriate patents. Nevertheless all HP firmware and software is still regarded as copyrighted and used as is, without any changes.

The project began with acquisition of the ROM contents of the HP9810A and HP9820A by means of a logic analyser. The next step was programming a disassembler in Java to get a deeper understanding of the 9800 CPU instructions. Furthermore by analysing the disassembled firmware it was possible to understand the I/O procedure and interrupt structure.

One more result was the ability to create my own assembler programs for the 9800 CPU and run them on the real hardware. I discovered undocumented features in the 9810 Peripheral Control ROM which allowed to store and run binary machine programs. The first one was a kind of PEEK function for the HP9810A. By means of this the complete memory contents could be dumped. It was a good verification of the correctness of the logic analyzer read out. The next assembler program was a PEEK function for the HP9830A, whose firmware contents was not available so far. This also succeeded in a complete dump of the system ROM and additional plug-in ROMs.

The last phase of the project was the Java programming of a 9800 CPU emulator. Because of the properties of the display and keyboard I/O devices the HP9830A seemed to be the easiest candidate for emulation of the whole hardware. There where several risks in this phase as it was not sure whether the CPU instructions where completely documented in the various U.S. patents of the series 9800 machines. Also it was difficult to estimate the execution speed of the emulator. The machines should at least run in real time speed. Moreover the I/O principles where very poor documented and difficult to understand in detail.

In fact during the first tests of the emulator it showed that some CPU instructions had to behave different from the available 'documentation'. Esp. the floating point macro instructions took some weeks of debugging and step-by-step analyzing until they produced correct results. Also the stack handling of JSM /RET and the interrupt service routine gave some unexpected surprises. Late in the project, when testing the HP9820A together with the HP9865A, there showed one more anomaly in the IO interface. After many hours of testing, debugging, and re-programming it was only one possible solution left: the handling of the CEO handshake in the instruction STF 1 was wrong. There was only one constellation in the firmware of all three calculators where this special situation occurred.

With the core functionalities of the HP9830 and getting the machine up running with a very simple output and input interface, the proof of concept was done. After that I added a simulated keyboard and LED matrix display. Photographs and sound recordings of the original machine where used to resemble a most naturalistic simulation.

The display simulation is somewhat tricky: in the original machine the display output is controlled and multiplexed by the CPU. That means the CPU outputs every displayed character many times a second to give to the human eye a quasi steady display. When implemented 1:1 in software, this resulted in heavy flickering and high load of the host CPU. So I decided to update the display output only if a character changed, i.e. when a new output was generated by DISP command, user entry and the like. This has only on disadvantage: the display keeps visible even if the calculator is busy. This issue finally is solved by clearing the simulated display if the DEN (display enable) signal is false for more than a certain amount of executed CPU instructions (5000).

In the next steps several peripheral devices where added: the HP9860A marked card reader, the HP9866A thermal printer, and the HP9862A plotter. The device which caused the most work was the HP9865A cassette tape drive, which is available build-in to the HP9830A and as external device. The I/O protocol is very complicated and the timing crucial, since the 9865 reads and writes asynchronously to the main machine. Some tape operations run completely in background and send interrupts to the calculator. JAVA multi threading techniques had to be used intensely. It took several weeks until the 9865 worked satisfactory.

When the HP9830 and the peripherals where completed the next machine to be implemented was the HP9810A. Reusing or extending most of the classes of the HP9830, the 9810 was basically brought up in a few days. Again the display simulation required a special timing logic. Most work had to be spend in the magnetic card reader. It has a similar I/O protocol as the HP9865, but the timing even more critical. Again it took more than two weeks to get a stable solution.

The last one was the HP9820A. With the experience of the first two machines it would have take only a few days to complete the 9820. If there didn't occur this annoying problem with the cassette memory ROM and the 9865 (see above).

In several releases of the emulator more add-on ROM blocks and application programs where added to the distribution. Some of them where borrowed by contributors (see chapter 7) and read out using the mentioned assembler programs. Some ROM blocks are obviously very rare and could not be acquired until now. Amongst these are the HP9810A User Defined Functions ROM and the Typewriter ROM. Since their contents are completely listed in the HP9810A patent I decided to re-create them from scratch and type the octal values from the patent listings. By disassembling and cross-checking the instructions all typing errors could be eliminated. The graphics of the ROM blocks and keyboard overlays where also created artificially with GIMP on basis of other existing ROMs.

In 2007 the HP9880A/B mass memory system was added to the project, in 2008 the HP9861A impact printer, and in 2011 the HP9866B graphics capable thermal printer. At this time the complete printer and plotter output was reworked for scalability and hardcopy to a real printing device. The last major addition and intended finish of the project was the HP9821A in 2011. This was made possible by a acquisition of the original HP9821A system ROMs.

As of May 2012 the project took more than 1000 hours of time for research, design, programming, and testing.

6.1 Literature and Links

The architecture of the HP9800 series is most completely described in the following patents available from the U.S. patent office (www.uspto.gov):

HP9810A:	3,859,635	Programmable Calculator
HP9820A:	3,839,630	Programmable Calculator Employing Algebraic Language

HP9830A:	4,012,725	Programmable Calculator

There are also german patents available from Deutsches Patent- und Markenamt (www.dpma.de):

HP9810A:	2264920, 2264923, 2264871
HP9820A:	2262725, 2264896, 2264897, 2264898
HP9830A:	2333908, 2365567, 2365568, 2365570

Further information about the history and technology of HP calculators can be found at

www.hpmuseum.org

www.hp9825.com

www.hp9831.com

www.hp9845.net

http://www.classiccmp.org/hp/9800.htm

Most calculator and peripheral manuals are available for download from

www.hpmuseum.net

7. Contributions to the Project

The author doesn't own all option ROMS, interfaces and peripheral devices. So all users are invited to contribute by borrowing or donating items which are still missing. These can be analyzed or read out in the authors lab.

Also wanted is original software for all models. There are methods to transfer program listings or file dumps to a PC, these programs could be made available for use with the GO9800.

You may also join as a developer. The IDE used for this project is Eclipse 3.2.

7.1 Contributors

Jon Johnston (hpmuseum.net)

provided several ROM blocks (HP11252A, HP11262A, HP11222A, HP11283B). The ROM contents where read out and made available in the emulator.

He also provided photographs of an original HP9821A, and the components of the HP9880B mass memory system.

Moreover he borrowed several original HP tape cassettes and magnetic cards containing applications, diagnostics, and system programs for the HP9810, 9820, and 9830. The programs where extracted and included in the applications folder of the emulator.

Tim Eliseo

has created several add-on software packages for the HP9830A and has provided several bug fixes and performance improvements for the emulator.

David Bryan

was the initiator of the keyboard configuration file and provided several hints for functional improvement and internationalization. He also tested pre-releases of the emulator and localized a lot of bugs. Dave also created internationalized keyboad configurations for the HP9820A and HP9830A which especially matches US keyboards.

Philippe Sonnet

borrowed several magnetic cards with programs for the HP9810A. The contents where extracted and are included in the applications folder of the emulator.

Appendix

A Creation and Execution of Assembler Programs

Originally the calculators of the 9800 series where not designed to execute user programs in assembler or machine language. The machine instructions of the 9800 CPU where not well documented (only as part of some U.S. patents describing the machines) and there was no assembler to generate machine instructions from an assembler source code.

Nevertheless there where so called special programs available from HP, which served for certain applications esp. in controlling of peripheral devices. These special programs where created outside of the calculators and sold by HP.

During realization of the HP9800 emulator most of the system and plug-in ROMs where disassembled and analyzed. Several ways where found to insert user created machine programs into the calculator memory and execute them. Of course detailed knowledge of the assembler language, which is common to all calculators, and of the machine instructions is necessary. There is no assembler available to generate machine instructions from an assembler source. The coding has to be done by hand. It is beyond the scope of this manual to describe the 9800 CPU instructions in detail.

Starting from an assembler source the first step is to obtain the instruction opcodes, a 16-bit code of each CPU instruction.

A.1 HP9810A

To execute assembler programs on a HP9810A the presence of a Peripheral Control ROM 1 or 2 is necessary. These ROMs contain undocumented features to store a machine language program in the user RWM and execute it.

Machine programs are stored in the user RWM, precisely in data registers, beginning from the highest available register (108) and continuing with register 107, 106, etc. The data registers are located at the octal memory addresses 001000 (register 108), 001004 (register 107), and so on. So the machine instructions are stored from address 001000 upward. In fact the first word at 001000 is used as an internal program length counter, so the user program starts at address 001001.

A.1.1 Coding the Program

For preparation all opcodes of the machine program have to be represented as 4 digit (16 bit) hexadecimal values. This can conveniently done using OpenOffice Calc. Next every hexadecimal value has to be represented by HP9810A keys with matching key codes by the following scheme:

Hex digit	**HP9810A key**
0 – 9	0 - 9
A	x^2
B	a
C	b
D	G
E	F
F	1/x

Example:

Instruction	Oct. opcode	Hex. opcode	HP9810A keys			
LDB B,I	074537	795F	7	9	5	1/x
STB 01717	035717	3BCF	3	a	b	1/x
LDA 016	020016	200E	2	0	0	F
CLR	170000	F000	1/x	0	0	0

A.1.2 Entering the Program

Before entering the coded program the length counter at address 001000 has to be cleared. This is done by storing 0 to register 108.

Press: 0 x→() 1 0 8

Then each opcode has to be entered using the undocumented format sequence

FORMAT 3 b key1 key2 key3 key4

Each time this sequence is entered the original 16 bit value is reconstructed by the calculator and stored in a memory position given by the said length counter plus 01001. After that the length counter is incremented by one.

For the example from above the keys to be pressed would be:

```
FORMAT  3  b  7  9  5  1/x
FORMAT  3  b  3  a  b  1/x
FORMAT  3  b  2  0  0  F
FORMAT  3  b  1/x  0  0  0
```

This procedure is of course lengthy and error prone. Mistakes can not be corrected. So the best way is to enter the whole sequence in the program memory and save it on a magnetic card. The code then can be corrected re-entered as necessary.

A.1.3 Executing the Program

The stored machine program can be executed by the format sequence

FORMAT 3 CONTINUE

The machine program is executed beginning at address 001001. The last executed instruction should be RET (opcode 170402).

A.1.4 Example Program

Here is an example assembler program which implements a 'PEEK' function. The program takes the value of the X register, converts it to a 16 bit address value, reads the memory location pointed to by this address, and returns the value in the X register.

The resulting format sequences are stored on a magnetic card, which is contained in the GO9800 distribution (applications/HP9810/HP9810A PEEK.mcard).

```
01001 020004            LDA D4     ; A = addr of X
01002 024015            LDB D15    ; B = addr of AR1
01003 170004            XFR        ; transfer X->AR1
01004 074742            SBR 16     ; B = 0
01005 035717            STB D1717  ; (BTMP) = B
01006 021744            LDA D1744  ; A = Exponent word of AR1
01007 070113            SAP *+2    ; A>=0?
01010 067024            JMP J1024  ; If not: address=0
01011 070342      J1011 SAR 8      ; A = Exponent byte
01012 024063            LDB D63    ; B = 1
01013 070037            ADB A      ; B = A+1
01014 000023            ADA D23    ; A = A-5
01015 070112            SAM *+2    ; Exponent >4 ?
01016 025065            LDB D1065  ; B = 5          ; max. 5 digits
01017 035716      J1017 STB D1716  ; (addrC) = B    ; Address counter
01020 175400      J1020 DLS        ; get highest digit of AR1 and shift AR1
left
01021 063056            JSM J1056  ; find binary equivalent
01022 055716            DSZ D1716  ; all digits processed?
01023 067020            JMP J1020  ; no, repeat
;
01024 074537      J1024 LDB B,I    ; B = (B)   ; load contents of memory
location in B
01025 035717            STB D1717  ; (BTEMP) = B
01026 020016            LDA D16    ; A = addr of AR2
01027 170000            CLR        ; Clear AR2
01030 021066            LDA D1066  ; A = 16    ; process 16 bits
01031 031716            STA D1716  ; initialize counter
01032 025717      J1032 LDB D1717  ; B = (BTEMP)
01033 074133            SBP *+2,C  ; Is bit15 of B zero?
01034 072075            SEC *+1,S  ; Set E to 1111 binary
01035 074744            SBL 1      ; B = B<<1
01036 035717            STB D1717  ; (BTEMP) = B
01037 020016            LDA D16    ; A = addr of AR2
01040 024015            LDB D15    ; B = addr of AR1
01041 170004            XFR        ; transfer AR2->AR1
01042 170560            FXA        ; AR2 = AR2+AR1+E = 2*AR2 + bit15(BTEMP)
01043 055716            DSZ D1716  ; all bits processed?
01044 067032            JMP J1032  ; no, continue
01045 171450            NRM        ; normalize AR2
01046 074056            CMB        ; B = !B = -B-1
01047 004106            ADB D106   ; B = 12-B-1 = 11-B
01050 074404            SBL 8      ; B = B<<8  ; exponent byte
01051 035754            STB D1754  ; Exponent word of AR2 = B
01052 020016            LDA D16    ; A=addr of AR2
```

```
01053 024004            LDB D4      ; B=addr of X
01054 170004            XFR         ; transfer AR2->X
01055 170402            RET
;
01056 025717      J1056 LDB D1717   ; B = (BTEMP)
01057 074704            SBL 2       ; B = 4*B
01060 005717            ADB D1717   ; B = B+BTEMP = 5*BTEMP
01061 074037            ADB B       ; B = 10*BTEMP
01062 070037            ADB A       ; B = B+A = 10*BTEMP+A
01063 035717            STB D1717   ; (BTEMP) = B
01064 170402            RET
;
01065 000005      D1065 OCT 00005   ; 5
01066 000020      D1066 OCT 00020   ; 16
```

A.2 HP9830A

To create assembler programs on a HP9830A and store them permanently on tape the presence of a Infotek Fast Basic ROM II is necessary. This ROM can be used to directly read and write memory locations by means of two undocumented functions. Additionally with the FDATA command it is possible to create binary files and store machine programs on tape.

Machine programs are stored in the user RWM, at the upper end of the RWM area. The structure of a machine program is the same as of an add-on ROM module. In fact machine programs are managed by the system exactly like ROM blocks. A machine program can contain one or more commands or functions with their own keywords. These keywords can be used in the same way in BASIC programs or command lines like with build-in functions.

The ROM block structure is described in the U.S. patent 4,012,725 (Programmable Calculator). Below follows an excerpt from this patent.

Communication with the routines in the various plug-in read-only memory modules is accomplished through a series of mnemonic tables and jump tables. The standard firmware, the cassette operating firmware, and each of the plug-in read-only memory modules all contain the following tables:

1. A statement mnemonic table
2. A statement syntax jump table
3. A statement execution jump table
4. A system command mnemonic table
5. A system command execution jump table
6. A function mnemonic table
7. A function execution jump table
8. A non-formula operator mnemonic table

All of these tables, with the exception of the statement execution jump table, may appear anywhere within a memory module. The last five words in each module are used by the table scan routines [...] to find the actual location of the tables. The last word of each module contains a unique operation code word for that particular module. The second from the last word contains a relative address of the statement mnemonic table. The third from the last word contains a relative address of the system command mnemonic table. The fourth from the last word contains a relative address to the function mnemonic table. The fifth from the last word contains a relative address to the non-formula operator table. The jump tables for statement syntax, system command execution, and function execution are located directly above their respective mnemonic tables. The statement execution jump table is located directly above the fifth from the last word of each module. The complete set of tables for the standard firmware is shown in the firmware listings [...].

Each of the mnemonic tables consists of a string of seven-bit ASCII character and six-bit operation code characters packed two characters per sixteen-bit word. The eighth bit of each character is used to indicate whether that character is ASCII or an operator code. A zero in the eighth bit indicates ASCII and a one indicates an operation code. The seventh bit of each operation code character is used to indicate whether that operation code is the last character in that table. The jump table address for each mnemonic is found by subtracting the operation code for that mnemonic from the starting address of the associated mnemonic table. The internal stored format for program statements consists of a series of operation codes, operand codes, and other special codes. The first word of each statement contains the line number of that statement in binary format. The second word contains both the operation code for that particular statement mnemonic and also the length of the statement. The length information is used by various firmware routines to scan from one statement to the next. The third word contains the operation code for the table or optional read-only memory module, and it also contains the first operand code. The remainder of the statement is stored with one operator code and one operand code in each word.

A.2.1 Coding the Program

The opcodes of the machine program have to be supplemented by a pointer structure and memory for the keyword(s) as explained above. This can be demonstrated best by means of a sample program.

The following program represents a PEEK function which returns the contents of a memory location whose address is given as function parameter. In this example the program is located at the end of first memory page 000000-001777. Generally the machine program should be written page relocatable, that means that only page relative addressing should be used. Later the program will be loaded at the end of the last RWM page.

```
001750      4          4           ; Execution Jump Table: +4 for Opcode=1
001751      50105      "PE"
001752      42513      "EK"
001753      140400     OPCODE=1    ; Opcode-Bit(0x80) + Last-Opcode(0x40) + 1
001754      060445     JSM XFAR2+1 ; Transfer Argument -> AR2
001755      160225     JSM FLTRA,I ; Convert Float AR2 -> Int B
001756      074742     SBR 16      ; If Overflow: B=0
001757      074537     LDB B,I     ; load contents of memory location in B
001760      164227     JMP FXFLA,I ; Convert Int to Float ->AR2 and return
001761      000000                 ; unused memory
001762      000000                 ;
001763      000000                 ;
001764      000000                 ;
001765      000000                 ;
```

```
001766      000000                  ;
001767      000000                  ;
001770      000000                  ;
001771      000000                  ; unused memory
001772      177777      -1          ; Statement Jump Table
001773      177777      -1          ; Ptr. to Non-formula Operator Mnemonic
Table
001774      177755      -19         ; Ptr. to Function Mnemonic Table
001775      177775      -3          ; Ptr. to Command Mnemonic Table
001776      177774      -4          ; Ptr. to Statement Mnemonic Table
001777      060000                  ; Module operation code 60000
```

A.2.2 Entering the Program

For preparation all opcodes of the machine program have to be represented as decimal integer values between -32767 and +32767. This can conveniently done using OpenOffice Calc. Next a BASIC program has to be written which creates a binary tape file from the assembler opcodes. The decimal opcodes have to be stored in DATA statements.

For creation of the binary file a special plug-in ROM is necessary, which is supplied with the GO9800 emulator. The Infotek Fast Basic ROM II contains the FDATA instruction which allows creation of arbitrary file types. For further informations refer to the original Infotek manual.

The appropriate BASIC program for the above PEEK example is presented here:

```
10 DIM HI[7],BI[24]
20 FOR I=1 TO 7
30 READ H[I]
40 NEXT I
50 FOR I=1 TO 24
60 READ B[I]
70 NEXT I
80 FDATA S,H[1],B
90 STOP
100 DATA 0,24,1,50,0,0,0
110 DATA 4,20549,17739,-16128,24869,-8043,31202,31071,-5993,0,0,0
120 DATA 0,0,0,0,0,0,-1,-1,-19,-3,-4,24576
130 END
```

The DATA statement in line 100 defines the header of the binary file. The first value is the file number (in this example 0), the second the aktual file size (24 words), the third value is the file type (HP9830 binary=1), and the fourth value is the absolute file size.

Before executing the program MARK an empty file 0 (or whatever was defined as file number) of the same absolute size as in the above DATA statement. Then position the tape to the beginning of the MARKed file using FIND 0 (or whatever). Then RUN the program.

A.2.3 Executing the Program

An assembler program can be loaded from a binary tape file into the calculator memory by the command

LOAD BIN <file number>

After that the assembler program can be executed by entering the corresponding keyword, which is defined in the program.

The example program above can be executed by entering PEEK <address>, where address is an integer value. PEEK is a function and returns an integer value which can be used in expressions like any other value.

A.3 HP9820A / HP9821A

At the moment there is no method known to directly create assembler programs on a HP9820A. Nevertheless it is possible to create binary files for this machine using a HP9830A and to load and execute machine programs on the HP9820/21A.

To execute assembler programs on a HP9820A the presence of the HP11223A Cassette Memory / Special Programs ROM is necessary. The HP9821A contains the required functions in system ROM.

Machine programs are stored in the user RWM, at the lower end of the RWM area, beginning with address 016426 (octal). The structure of a machine or so called 'special' program is much similar to that used for the HP9830A, which is described above. A machine program can contain one or more commands or functions identified by keywords. A program is found by its keyword using the instruction CSP "*keyword*" (Call Special Program). The memory used by the program has to be a integer multiple of 8 words plus 3 and the last word has to be 0177777 octal (-1 decimal).

To create HP9820/21A assembler programs on a HP9830A and store them permanently on tape the presence of a Infotek Fast Basic ROM II is necessary. The FDATA command is used to create binary files and store machine programs on tape.

A.3.1 Coding the Program

The opcodes of the machine program have to be supplemented by a pointer structure and memory for the keyword(s). This can be demonstrated best by means of a sample program.

The following program represents a PEEK function which returns the contents of a memory location whose address is given in register Z. The result is returned in register Z. The program code is nearly identical to that of the HP9810A example program above.

```
16426 016430          OCT D16430  ; Pointer to first name
16427 177777          -1          ; End of list
16430 050105   D16430 "PE"
16431 042513          "EK"
16432 177773          -4-1        ; -(len+1) = End of name
16433 016434          J16434      ; Pointer to program start
16434 020442   J16434 LDA D442    ; A = addr of A-reg.
16435 000451          ADA D451    ; A = A+024 = addr of Z-reg.
16436 024447          LDB D447    ; B = addr of AR1
16437 170004          XFR         ; transfer Z->AR1
16440 074742          SBR 16      ; B = 0
16441 035717          STB D1717   ; (BTMP) = B
16442 021744          LDA D1744   ; A = Exponent word of AR1
16443 070113          SAP *+2     ; A>=0?
16444 067460          JMP J16460  ; If not: address=0
16445 070342   J16445 SAR 8       ; A = Exponent byte
16446 024403          LDB D403    ; B = 1
16447 070037          ADB A       ; B = A+1
```

```
16450 000411            ADA D411    ; A = A-5
16451 070112            SAM *+2     ; Exponent >4 ?
16452 024377            LDB D377    ; B = 5           ; max. 5 digits
16453 035716    J16453  STB D1716   ; (addrC) = B     ; Address counter
16454 175400    J16454  DLS         ; get highest digit of AR1 and shift AR1
left
16455 062513            JSM J16513  ; find binary equivalent
16456 055716            DSZ D1716   ; all digits processed?
16457 066454            JMP J16454  ; no, repeat
;
16460 074537    J16460  LDB B,I     ; B = (B)   ; load contents of memory
location in B
16461 035717            STB D1717   ; (BTEMP) = B
16462 020450            LDA D450    ; A = addr of AR2
16463 170000            CLR         ; Clear AR2
16464 020364            LDA D364    ; A = 16    ; process 16 bits
16465 031716            STA D1716   ; initialize counter
16466 025717    J16466  LDB D1717   ; B = (BTEMP)
16467 074133            SBP *+2,C   ; Is bit15 of B zero?
16470 072075            SEC *+1,S   ; Set E to 1111 binary
16471 074744            SBL 1       ; B = B<<1
16472 035717            STB D1717   ; (BTEMP) = B
16473 020450            LDA D450    ; A = addr of AR2
16474 024447            LDB D447    ; B = addr of AR1
16475 170004            XFR         ; transfer AR2->AR1
16476 170560            FXA         ; AR2 = AR2+AR1+E = 2*AR2 + bit15(BTEMP)
16477 055716            DSZ D1716   ; all bits processed?
16500 066466            JMP J16466  ; no, continue
16501 171450            NRM         ; normalize AR2
16502 074056            CMB         ; B = !B = -B-1
16503 004370            ADB D370    ; B = 12-B-1 = 11-B
16504 074404            SBL 8       ; B = B<<8  ; exponent byte
16505 035754            STB D1754   ; Exponent word of AR2 = B
16506 020450            LDA D450    ; A=addr of AR2
16507 024442            LDB D442    ; B=addr of A-reg.
16510 004451            ADB D451    ; B = B+024 = addr of Z-reg.
16511 170004            XFR         ; transfer AR2->Z
16512 170402            RET
;
16513 025717    J16512  LDB D1717   ; B = (BTEMP)
16514 074704            SBL 2       ; B = 4*B
16515 005717            ADB D1717   ; B = B+BTEMP = 5*BTEMP
16516 074037            ADB B       ; B = 10*BTEMP
16517 070037            ADB A       ; B = B+A = 10*BTEMP+A
16520 035717            STB D1717   ; (BTEMP) = B
16521 170402            RET
16522 000000            0           ; Filler word to int. mult. of 8 words
16523 000000            0           ; Filler word to int. mult. of 8 words
16524 000000            0           ; Filler word to int. mult. of 8 words
16525 000000            0           ; Filler word to int. mult. of 8 words
16526 000000            0           ; Filler word
16527 000000            0           ; Filler word
16528 177777            -1          ; End of memory for binary program
```

A.3.2 Entering the Program

For preparation all opcodes of the machine program have to be represented as decimal integer values between -32767 and +32767. This can conveniently done using OpenOffice Calc. Next a BASIC program has to be written which creates a binary tape file from the assembler opcodes. The decimal opcodes have to be stored in DATA statements.

For creation of the binary file a special plug-in ROM is necessary, which is supplied with the GO9800 emulator. The INFOTEK Fast Basic ROM II contains the FDATA instruction which allows creation of arbitrary file types. For further informations refer to the original Infotek manual.

The appropriate BASIC program for the above PEEK example is presented here:

```
10 DIM HI[7],BI[60]
20 FOR I=1 TO 7
30 READ H[I]
40 NEXT I
50 FOR I=1 TO 60
60 READ B[I]
70 NEXT I
80 FDATA S,H[1],B
90 STOP
100 DATA 0,60,28,100,0,0,0
110 DATA 7448,-1,20549,17739,-5,7452,8482,297
120 DATA 10535,-4092,31202,15311,9188,28747,28464,28898
130 DATA 10499,28703,265,28746,10495,15310,-1280,25931
140 DATA 23502,27948,31071,15311,8488,-4096,8436,13262
150 DATA 11215,30811,29757,31204,15311,8488,10535,-4092
160 DATA -3728,23502,27958,-3288,30766,2296,30980,15340
170 DATA 8488,10530,2345,-4092,-3838,11215,31172,3023
180 DATA 30751,28703,15311,-3838
190 DATA 0,0,0,0,0,0,-1
200 END
```

The DATA statement in line 100 defines the header of the binary file. The first value is the file number (in this example 0), the second the actual file size (60 words), the third value is the file type (HP9820 binary=28), and the fourth value is the absolute file size.

Before executing the program MARK an empty file 0 (or whatever was defined as file number) of the same absolute size as in the above DATA statement (100). Then position the tape to the beginning of the MARKed file using FIND 0 (or whatever). Then RUN the program.

A.3.3 Executing the Program

A HP9820A machine program can be loaded from a binary tape file into the calculator memory by the command

LDF <file number>

After that the program can be executed this the CSP instruction with the corresponding keyword parameter, which is defined in the program.

The example program above can be executed by entering

<address> → Z

CSP “PEEK”

The register Z now contains the memory value at the given address.

A.3.4 Saving and Deleting the Program

There are two more functions in the Cassette Memory / Special Programs ROM for managing special programs by means of the key ISP (Initialize Special Program).

A special program already loaded in the calculator memory may be saved to a new tape file be means of the command

ISP 1,<file number>

Later it can be loaded again from this file.

A HP9820A machine program can normally not deleted by pressing the ERASE key. Instead the command

ISP 2

has to be executed. This carries out a soft reset of the machine (like power-on).

WARNING: the complete RWM is erased by ISP 2, including user programs and data!

B Machine Instruction Set

The design of the series 9800 calculators, a firmware listing, and the description of the HP9800 CPU instruction set is contained in the following U.S. patents:

HP9810A:	3,859,635	Programmable Calculator
HP9820A:	3,839,630	Programmable Calculator Employing Algebraic Language
HP9830A:	4,012,725	Programmable Calculator

Below is an excerpt of the HP9830A patent which describes the CPU instructions and their corresponding opcodes.

BASIC INSTRUCTION SET

Every routine and subroutine of the calculator comprises a sequence of one or more of 71 basic sixteen-bit instructions listed below. These 71 instructions are all implemented serially by the micro-processor in a time period which varies according to the specific instruction, to whether or not it is indirect, and to whether or not the skip condition has been met.

Upon completion of the execution of each instruction, the program counter (P register) has been incremented by one except for instructions JMP, JSM, and the skip instructions in which the skip condition has been met. The M-register is left with contents identical to the P-register. The contents of the addressed memory location and the A and B registers are left unchanged unless specified otherwise.

Memory Reference Group

The 14 memory reference instructions refer to the specific address in memory determined by the address field <*m*>, by the ZERO/CURRENT page bit, and by the DIRECT/INDIRECT bit. Page addressing and indirect addressing are both described in detail in the reference manuals for the Hewlett-Packard Model 2116 computer (hereinafter referred to as the HP 2116).

The address field <*m*> is a 10 bit field consisting of bits 0 through 9. The ZERO/CURRENT page bit is bit 10 and the DIRECT/INDIRECT bit is bit 15, except for reference to the A or B register in which case bit 8 becomes the DIRECT/INDIRECT bit. An indirect reference is denoted by a <, I> following the address <*m*>.

REGISTER REFERENCE OF A OR B REGISTER: If the location <A> or <B> is used in place of <*m*> for any memory reference instruction, the instruction will treat the contents of A or B exactly as it would be contents of location <*m*>. See the note below on the special restriction for direct register reference of A or B.

ADA *m*,I Add to A. The contents of the addressed memory location m are added (binary add) to contents of the A register, and the sum remains in the A register. If carry occurs from bit 15, the E register is loaded with 0001, otherwise E is left unchanged.

ADB *m*,I Add to B. Otherwise identical to ADA.

CPA *m*,I Compare to A and skip if unequal. The contents of the addressed memory location are compared with the contents of the A register. If the two 16-bit words are different, the next instruction is skipped; that is, the P and M registers are advanced by two instead of one. Otherwise, the next instruction will be executed in normal sequence.

CPB *m*,I Compare to B and skip is unequal. Otherwise identical to CPA.

LDA *m*,I Load into A. The A register is loaded with the contents of the addressed memory location.

LDB *m*,I Load into B. The B register is loaded with the contents of the addressed memory location.

STA *m*,I Store A. The contents of the A register are stored into the addressed memory location. The previous contents of the addressed memory location are lost.

STB *m*,I Store B. Otherwise identical to STA.

IOR *m*,I Inclusive OR to A. The contents of the addressed location are combined with the contents of the A register as an INCLUSIVE OR logic operation.

ISZ *m*,I Increment and Skip if Zero. The ISZ instruction adds ONE to the contents of the addressed memory location. If the result of this operaion is ZERO, the next instruction is skipped, that is, the P and M registers are advanced by TWO instead of ONE. The incremental value is writted back into the addressed memory location. Use of ISZ with the A or B register is limited to indirect reference; see footnote on restrictions.

AND *m*,I Logical AND to A. The contents of the addressed location are combined with the contents of the A register as an AND logic operation.

DSZ *m*,I Decrement and Skip if Zero. The DSZ instruction subtracts ONE from the contents of the addressed memory location. If the result of this operation is zero, the next instruction is skipped. The decremented value is writted back into the addressed memory location. Use of DSZ with the A or B register is limited to indirect reference; see footnote on restrictions.

JSM *m*,I Jump to Subroutine. The JSM instruction permits jumping to a subroutine in either ROM or R/W memory. The contents of the P register is stored at the address contained in location 1777 (stack pointer). The contents of the stack pointer is incremented by one, and both M and P are loaded with the referenced memory location.

JMP *m*,I Jump. This instruction transfers control to the contents of the addressed location. That is, the referenced memory location is loaded into both M and P registers, effecting a jump to that location.

Shift-Rotate Group

The eight shift-rotate instructions all contain a 4 bit variable shift field <*n*> which permits a shift of one through 16 bits; that is, $1 \leq n \leq 16$. If <*n*> is omitted, the shift will be treated as a one bit shift. The shift code appearing in bits 8,7,6,5 is the binary code for *n*-1, except for SAL and SBL, in which cases the complementary code for *n*-1 is used.

AAR *n* Arithmethic right shift of A. The A register is shifted right *n* places with the sign bit (bit 15) filling all vacated bit positions. That is, the *n*+1 most significant bits become equal to the sign bit.

ABR *n* Arithmetic right shift of B. Otherwise identical to AAR.

SAR *n* Shift A right. The A register is shifted right *n* places with all vacated bit positions cleared. That is, the *n* most significant bits become equal to zero.

SBR *n* Shift B right. Otherwise identical to SAR.

SAL *n* Shift A left. The A register is shifted left *n* places with the *n* least significant bits equal to zero.

SBL *n* Shift B left. Otherwise identical to SAL.

RAR *n* Rotate A right. The A register is rotated right *n* places, with bit 0 rotated around to bit 15.

RBR *n* Rotate B right. Otherwise identical to RAR.

Alter-Skip Group

The sixteen alter-skip instructions all contain a 5-bit variable skip field <*n*> which, upon meeting the skip condition, permits a relative branch to any one of 32 locations. Bits 9,8,7,6,5 are coded for positive or negative relative branching in which the number <*n*> is the number to be added to the current address, (skip in forward direction), and the number <-*n*> is the number to be subtracted from the current address, (skip in negative direction). If <*n*> is omitted, it will be interpreted as a ONE.

<*n*>=0	CODE=00000 REPEAT SAME INSTRUCTION
<*n*>=1	CODE=00001 DO NEXT INSTRUCTION
<*n*>=2	CODE=00010 SKIP ONE INSTRUCTION
<*n*>=15	CODE=01111 ADD 15 TO ADDRESS
<*n*>=-1	CODE=11111 DO PREVIOUS INSTRUCTION
<*n*>=-16	CODE=10000 SUBTRACT 16 FROM ADDRESS
<*n*>=nothing	CODE=00001 DO NEXT INSTRUCTION

The alter bits consist of bits 10 and bits 4. The letter <S> following the instruction places a ONE in bit 10 which causes the tested bit to be set after the test. Similarly the letter <C> will place a ONE In bit 4 to clear the test bit. If both a set and clear bit are given, the set will take precedence. Alter bits do not apply to SZA, SZB, SIA, and SIB.

SZA *n* Skip if A zero. If all 16 bits of the A register are zero, skip to location defined by *n*.

SZB *n* Skip if B zero. Otherwise identical to SZA.

RZA *n* Skip if A not zero. This is a Reverse Sense skip of SZA.

RZB *n* Skip if B not zero. Otherwise identical to RZA.

SIA *n* *S*kip if A zero; then increment A. The A register is tested for zero, then incremented by one. If all 16 bits of A were zero before incrementing, skip to location defined by *n*.

SIB *n* Skip if B zero; then increment B. Otherwise identical to SIA.

RIA *n* Skip if A not zero; then increment A. This is a Reverse Sense skip of SIA.

RIB *n* Skip if B not zero; then increment B. Otherwise identical to RIA.

SLA *n*,S/C Skip if Least Significant bit of A is zero. If the least significant bit (bit 0) of the A register is zero, skip to location defined by *n*. If either S or C is present, the test bit is altered accordingly after test.

SLB *n*,S/C Skip if Least Significant bit of B is zero. Otherwise identical to SLA.

SAM *n,*S/C Skip if A is Minus. If the sign bit (bit 15) of the A register is a ONE, skip to location defined by *n*. If either S or C is present, bit 15 is altered after the test.

SBM *n*,S/C Skip if B is Minus. Otherwise identical to SAM.

SAP *n*,S/C Skip if A is Positive. If the sign bit (bit 15) of the A register is a ZERO, skip to location defined by *n*. If either S or C is present, bit 15 is altered after the test.

SBP *n*,S/C Skip if B is Positive. Otherwise identical to SAP.

SES *n*,S/C Skip if Least Significant bit of E is Set. If bit 0 of the E register is a ONE, skip to location defined by *n*. If either S or C is present, the entire E register is set or cleared respectively.

SEC *n*,S/C Skip if Least Significant bit of E is Clear. If bit 0 of the E register is a ZERO, skip to location defined by *n*. If either S or C is present, the entire E register is set or cleared respectively.

Complement-Execute-DMA Group.

These seven instructions include complement operations and several special-purpose instructions chosen to speed up printing and extended memory operations.

CMA Complement A. The A register is replaced by its One's complement.

CMB Complement B. The B register is replaced by its One's complement.

TCA Two's Complement A. The A register is replaced by its One's Complement and incremented by one.

TBC Two's complement B. The B register is replaced by its One's Complement and incremented by one.

EXA Execute A. The contents of the A register are treated as the current instruction, and executed in the normal manner. The A register is left unchanged unless the instruction code causes A to be altered.

EXB Execute B. Otherwise identical to EXA.

DMA Direct Memory Access. The DMA control in Extended Memory is enabled by setting the indirect bit in M and giving a WTM instruction. The next ROM clock transfers A→M and the following two cycles transfer B→M. ROM clock then remains inhibited until released by DMA control.

***Note:* Special Restriction for Direct Register Reference of A or B**

For the five register reference instructions which involve a write operation during execution, a register reference to A or B must be restricted to an INDIRECT reference. These instructions are STA, STB, ISZ, DSZ, and JSM. A DIRECT register reference to A or B with these instructions may result in program modification. (This is different from the hp 2116 in which a memory reference to the A or B register is treated as a reference to locations 0 or 1 respectively.) A reference to location 0 or 1 will actually refer to locations 0 or 1 in Read Only Memory.

Input/Output Group (IOG)

The eleven IOG instructions, when given with a select code, are used for the purpose of checking flags, setting or clearing flag and control flip-flops, and transferring data between the A/B registers and the I/O register.

STF <SC> Set the flag. Set the flag flip-flop of the channel indicated by select code <SC>.

CLF <SC> Clear the flag flip-flop of the channel indicated by select code <SC>.

SFC <SC> Skip if flag clear. If the flag flip-flop is clear in the channel indicated by <SC>, skip the next instruction.

SFS <SC> H/C Skip if flag set. If the flag flip-flop is set in the channel indicated by <SC>, skip the next instruction. H/C indicates if the flag flip-flop should be held or cleared after executing SFS.

CLC <SC> H/C Clear control. Clear the control flip-flop in the channel indicated by <SC>. H/C indicates if the flag flip-flop should be held or cleared after executing CLC.

STC <SC> H/C Set Control. Set the control flip-flop in the channel indicated by <SC>. H/C indicates if the flag flip-flop should be held or cleared after executing STC.

OT* <SC> H/C   Output A or B. Sixteen bits from the A/B register are output to the I/O register. H/C allows holding or clearing the flag flop after execution of OT*. The different select codes allow different functions to take place after loading the I/O register.

SC=00   Data from the A or B register is output eight bits at a time for each OT* instruction given. The A or B register is rotated right eight bits.

SC=01   The I/O register is loaded with 16 bits from the A/B registers.

SC=02   Data from the A/B register is output one bit at a time for each OT* instruction for the purpose of giving data to the Magnetic Card Reader. The I/O register is unchanged.

SC=04   The I/O register is loaded with 16 bits from the A/B register and the control flip flop for the printer is then set.

SC=08   The I/O register is loaded with 16 bits from the A/B register and the control flip flop for the display is then set.

SC=16   The I/O register is loaded with 16 bits from the A/B register and then data in the I/O register is transferred to the switch latches.

LI* <01> H/C   Load into A or B. Load 16 bits of data into the A/B register from the I/O register. H/C allows holding or clearing the flag flop after LI* has been executed.

LI* <00>   The least significant 8 bits of the I/O register are loaded into the most significant locations in the A or B register.

MI* <01> H/C   Merge into A or B. Merge 16 bits of data into the A/B register from the I/O register by inclusive or. H/C allows holding or clearing the flag flop after MT* has been executed.

MI* <00>   The least significant 8 bits of the I/O register are combined by inclusive OR with the least significant 8 bits of the A or B register, and rotated to the most significant bit locations of the A or B register.

Mac instruction Group

A total of 16 MAC instructions are available for operation
a. with the whole floating-point data (like transfer, shifts, etc), or
b. with two floating-point data words to speed up digit and word loops in arithmethic routines.

NOTE: <A_{0-3}> means: contents of A-register bit 0 to 3

AR1 is a mnemonix for arithmetic pseudo-register located in R/W memory on addresses 1744 to 1747 (octal)

AR2 is a mnemonix for arithmetic pseudo-register located in R/W memory on addresses 1754 to 1757 (octal)

D_i means: mantissas i-th decimal digit;
most significant digit is D1
least significant digit is D12
decimal point is located between D1 and D2

Every operation with mantissa means BCD-coded decimal operation.

RET   Return
16-bit-number stored at highest occupied address in stack is transferred to P- and M-registers. Stack pointer (=next free address in stack) is decremented by one. <A>, <B>, <E> unchanged.

MOV   Move overflow
The contents of E-register is transferred to A_{0-3}. Rest of A-register and E-register are filled by

zeros. <B> unchanged.

CLR Clear a floating-point data register in R/W memory on location <A>
Zero→<A>, <A>+1, <A>+2, <A>+3
<A>, <B>, <E> unchanged

EXF Floating-point data transfer in R/W memory from location <A> to location <B>.
Routine starts with exponent word transfer.
Data on location <A> is unchanged.
<E> unchanged.

MRX AR1 Mantissa is shifted to right *n*-times. Exponent word remains unchanged.
$<B_{0-3}> = n$ (binary coded)
1st shift: $<A_{0-3}> \rightarrow D_1$; $D_i \rightarrow D_{i+1}$; D_{12} is lost
*j*th shift: $0 \rightarrow D_1$; $D_i \rightarrow D_{i+1}$; D_{12} is lost
*n*th shift: $0 \rightarrow D_1$; $D_i \rightarrow D_{i+1}$; $D_{12} \rightarrow A_{0-3}$
$0 \rightarrow E, A_{4-15}$
each shift: $<B_{0-3}> - 1 \rightarrow B_{0-3}$
$<B_{4-15}>$ unchanged

MRY AR2 Mantissa is shifted to right *n*-times. Otherwise identical to MRX

MLS AR2 Mantissa is shifted to left once. Exponent word remains unchanged.
$0 \rightarrow D_{12}$; $D_i \rightarrow D_{i-1}$; $D_1 \rightarrow A_{0-3}$
<B> unchanged

DRS AR1 Mantissa is shifted to right once Exponent word remains unchanged
$0 \rightarrow D_1$; $D_i \rightarrow D_{i+1}$; $D_{12} \rightarrow A_{0-3}$
$0 \rightarrow$ E and A_{4-15}
<B> unchanged

DLS AR1 Mantissa is shifted to left once. Exponent word remains unchanged.
$<A_{0-3}> \rightarrow D_{12}$; $D_i \rightarrow D_{i-1}$; $D_1 \rightarrow A_{0-3}$
$0 \rightarrow E, A_{4-15}$
<B> unchanged

FXA Fixed-point addition Mantissas in pseudo-registers AR2 and AR1 are added together and result in placed into AR2. Both exponent words remain unchanged. When overflow occurs 0001 is set into E-reg., in opposite case <E> will be zero.
$<AR2> + <AR1> + DC \rightarrow AR2$
DC = 0 if <E> was 0000 before routine execution
DC = 1 if <E> was 1111 before routine execution
<B>, <AR1> unchanged

FMP Fast multiply
Mantissas in pseudo-registers AR2 and AR1 are added together $<B_{0-3}>$-times and result is placed into AR2. Total decimal overflow is placed to A_{0-3}. Both exponent words remain unchanged.
$<AR2> + <AR1> * <B_{0-3}> + DC \rightarrow AR2$
DC = 0 if <E> was 0000 before routine execution
DC = 1 if <E> was 1111 before routine execution
$0 \rightarrow E, A_{4-15}$
<AR1> unchanged

FDV Fast divide
Mantissas in pseudo-registers AR2 and AR1 are added together so many times until first decimal overflow occurs. Result is placed into AR2. Both exponent words remain unchanged. Each addi-

tion without overflow causes +1 increment of <B>.
1st addition: $<AR2> + <AR1> + DC \rightarrow AR2$
DC = 0 if <E> was 0000 before routine execution
DC = 1 if <E> was 1111 before routine execution
next additions: $<AR2> + <AR1> \rightarrow AR2$
$0 \rightarrow E$
<AR1> unchanged

CMX 10's complement of AR1 mantissa is placed back to AR1, and ZERO is set into E-register. Exponent word remains unchanged
<B> unchanged

CMY 10's complement of AR2 mantissa.
Otherwise identical to CMY

MDI Mantissa decimal increment.
Mantissa on location <A> is incremented by decimal ONE on D_{12} level, result is placed back into the same location, and zero is set into E-reg.
Exponent word is unchanged.
When overflow occurs, result mantissa will be 1,000 0000 0000 (dec)
and 0001 (bin) will be set into E-reg.
<B> unchanged.

NRM Normalization
Mantissa in pseudo-register AR2 is rotated to the left to get $D_1 \neq 0$. Number of these 4-bit left shifts is stored in B_{0-3} in binary form ($<B_{4-15}>=0$)
when $< B_{0-3} > = 0,1,2,. . . . , 11$ (dec) $\rightarrow <E> = 0000$
When $< B_{0-3} > = 12$ (dec) $\rightarrow$ mantissa is zero, and <E>= 0001
Exponent word remains unchanged
<A> unchanged.

The binary codes of all of the above instructions are listed in the following coding table, where * implies the A or B register, D/I means direct/indirect, A/B means A register/B register, Z/C means zero page (base page) / current page, H/S means hold test bit/set test bit, and H/C means hold test bit/clear test bit. D/I, A/B, Z/C, H/S and H/C are all coded as 0/1.

CODING TABLE

GROUP	OCTAL	INSTR	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MEMORY	-0----	AD*	D/I	0	0	0	A/B	Z/C	←			MEMORY ADDRESS					→	
REFERENCE	-1----	CP*	D/I	0	0	1	A/B	Z/C										
GROUP	-2----	LO*	D/I	0	1	0	A/B	Z/C										
	-3----	ST*	D/I	0	1	1	A/B	Z/C										
	-4----	IOR	D/I	1	0	0	0	Z/C										
	-4----	ISZ	D/I	1	0	0	1	Z/C										
	-5----	AND	D/I	1	0	1	0	Z/C										
	-5----	DSZ	D/I	1	0	1	1	Z/C										
	-6----	JSM	D/I	1	1	0	0	Z/C										
	-6----	JMP	D/I	1	1	0	1	Z/C										
SHIFT-	07---0	A*R	0	1	1	1	A/B	-		-	←SHIFT CODE→				0	0	0	0
ROTATE	07---2	S*R	0	1	1	1	A/B	-		-					0	0	1	0
GROUP	07---4	S*L	0	1	1	1	A/B	-		-					0	1	0	0
	07---6	R*R	0	1	1	1	A/B	-		-					0	1	1	0
ALTER-	07---0	SZ*	0	1	1	1	A/B	0	←	SKIP CODE			→	0	1	0	0	0
SKIP	07---0	RZ*	0	1	1	1	A/B	1						0	1	0	0	0
GROUP	07---0	SI*	0	1	1	1	A/B	0						1	1	0	0	0
	07---0	RI*	0	1	1	1	A/B	1						1	1	0	0	0
	07---1	SL*	0	1	1	1	A/B	H/S						H/C	1	0	0	1
	07---2	S*M	0	1	1	1	A/B	H/S						H/C	1	0	1	0
	07---3	S*P	0	1	1	1	A/B	H/S						H/C	1	0	1	1
	07---4	SES	0	1	1	1	A/B	H/S						H/C	1	1	0	0
	07---5	SEC	0	1	1	1	A/B	H/S						H/C	1	1	0	1
D/I	07--17	ADA	0	1	1	1	A/B	-	-	D/I	0	0	0	0	1	1	1	1
REFERENCE	07--37	ADB	0	1	1	1	A/B	-	-	D/I	0	0	0	1	1	1	1	1
GROUP	07--57	CPA	0	1	1	1	A/B	-	-	D/I	0	0	1	0	1	1	1	1
	07--77	CPB	0	1	1	1	A/B	-	-	D/I	0	0	1	1	1	1	1	1
	07--17	LDA	0	1	1	1	A/B	-	-	D/I	0	1	0	0	1	1	1	1
	07--37	LDB	0	1	1	1	A/B	-	-	D/I	0	1	0	1	1	1	1	1
	07-557	STA	0	1	1	1	A/B	-	-	1	0	1	1	0	1	1	1	1
	07-577	STB	0	1	1	1	A/B	-	-	1	0	1	1	1	1	1	1	1
	07--17	IOR	0	1	1	1	A/B	-	-	D/I	1	0	0	0	1	1	1	1
	07-637	ISZ	0	1	1	1	A/B	-	-	1	1	0	0	1	1	1	1	1
	07--57	AND	0	1	1	1	A/B	-	-	D/I	1	0	1	0	1	1	1	1
	07-677	DSZ	0	1	1	1	A/B	-	-	1	1	0	1	1	1	1	1	1
	07-717	JSM	0	1	1	1	A/B	-	-	1	1	1	0	0	1	1	1	1
	07--37	JMP	0	1	1	1	A/B	-	-	D/I	1	1	0	1	1	1	1	1
COMP	07-016	EX*	0	1	1	1	A/B	-	-	-	-	-	0	0	1	1	1	0
EXECUTE	070036	DMA	0	1	1	1	0	-	-	-	-	-	0	1	1	1	1	0
DMA	07-056	CM*	0	1	1	1	A/B	-	-	-	-	-	1	0	1	1	1	0
	07-076	TC*	0	1	1	1	A/B	-	-	-	-	-	1	1	1	1	1	0
INPUT	1727--	STF	1	1	1	1	-	1	0	1	1	1	1	←	SELECT CODE			→
OUTPUT	1737--	CLF	1	1	1	1	-	1	1	1	1	1	1					
GROUP	17-7--	SFC	1	1	1	1	-	1	H/C	1	1	1	0					
	17-5--	SFS	1	1	1	1	-	1	H/C	1	0	1	0					
	17-5--	CLC	1	1	1	1	-	1	H/C	1	0	1	1					
	17-6--	STC	1	1	1	1	-	1	H/C	1	1	0	0					
	17-1--	OT*	1	1	1	1	A/B	1	H/C	0	0	1	1					
	17-2--	LI*	1	1	1	1	A/B	1	H/C	0	1	0	1					
	17-0--	MI*	1	1	1	1	A/B	1	H/C	0	0	0	1					

GROUP	OCTAL	INSTR	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAC	170402	RET	1	1	1	1	0	0	0	1	0	0	0	0	0	0	1	0
GROUP	170002	MOV	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0
	170000	CLR	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
	170004	XFR	1	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0
	174430	MRX	1	1	1	1	1	0	0	1	0	0	0	1	1	0	0	0
	174470	MRY	1	1	1	1	1	0	0	1	0	0	1	1	1	0	0	0
	171400	MLS	1	1	1	1	0	0	1	1	0	0	0	0	0	0	0	0
	170410	DRS	1	1	1	1	0	0	0	1	0	0	0	0	1	0	0	0
	175400	DLS	1	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0
	170560	FXA	1	1	1	1	0	0	0	1	0	1	1	1	0	0	0	0
	171460	FMP	1	1	1	1	0	0	1	1	0	0	1	1	0	0	0	0
	170420	FDV	1	1	1	1	0	0	0	1	0	0	0	1	0	0	0	0
	174400	CMX	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0
	170400	CMY	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0
	170540	MDI	1	1	1	1	0	0	0	1	0	1	1	0	0	0	0	0
	171450	NRM	1	1	1	1	0	0	1	1	0	0	1	0	1	0	0	0

The following describes the function of each I/O instruction with the 5 allowable select codes.

STF <SC> Set the flag. STF is a 240 nanosecond positive true pulse which accomplishes the following with the various select codes.
STF 00 Not used by the calculator.
STF 01 a. Sets the "Service Inhibit" flip-flop to the true state ($\overline{SIH}$ = Low; interrupt not allowed).
b. Causes parallel input data and status to be loaded into the I/O register.
STF 02 Generates a 240 nanosecond positive true MCR pulse.
STF 04,08,16 Not used by the calculator.
CLF <SC> Clear the flag. CLF is a 240 ns positive true pulse which accomplishes the following with the various select codes.
CLF 00 Not used by the calculator
CLF 01 a. Clears the "Service Inhibit" flip-flop to the false state. ($\overline{SIH}$ = High; interrupt allowed.)
b. Loads address locations in I/O register gister with 0's. (0 = Low)
c. Clears "Device Ready" flip-flop ($\overline{CEO}$ = High).
CLF 02 Clears MCR flag flip-flop.
CLF 04 Clears PEN flip-flop ($\overline{PEN}$ = Low).
CLF 08 Clears DEN flip-flop ($\overline{DEN}$ = High).
CLF 16 Generates a 240 nanosecond positive true KLS pulse
SFC <SC> H/C Skip if flag clear. SFC is a 240 ns positive true pulse which accomplishes the following with the various select codes. If C is given a 240 nanosecond CLF pulse is given after SFC.
SFC 00 Causes the next instruction to be skipped if the STOP key has not been depressed.
SFC 01 Causes the next instruction to be skipped

if Device Ready is true ($\overline{\text{CEO}}$ = Low).
SFC 02   Causes the next instruction to be skipped
if the MCR flag flip-flop is clear.
SFC 04   Causes the next instruction to be skipped
if the PEN flip-flop is clear. ($\overline{\text{PEN}}$ = Low).
SFS <SC> H/C Skip is flag set. SFS is a 240 nanosecond
positive true pulse which accomplishes
the following with the various codes.
If C is given then a 240 nanosecond CLF Pulse
is issued after SFS.
SFS 00   Causes the next instruction to be skipped
if the STOP key is depressed.
SFS 01   Causes the next instruction to be skipped
if "Device Ready" is false ($\overline{\text{CEO}}$ = High).
SFS 02   Causes the next instruction to be skipped
if the MCR flag flip-flop is set.
SFS 04   Causes the next instruction to be skipped
if the PEN flip-flop is set ($\overline{\text{PEN}}$ = High).
CLC <SC> H/C Clear Control. $\overline{\text{CLC}}$ is a 240 nanosecond negative
true pulse and is not used by the calculator.
If C is given then a 240 nanosecond positive true
CLF pulse is given after CLC.
STC <SC> H/C Set the Control. STC is a 240 nanosecond posi-
tive true pulse which accomplishes the
following with the various select codes.
If C is given a 240 nanosecond CLF pulse is
issued after STC.
STC 00   Not used by the calculator.
STC 01   Sets the "Device Ready" flip-flop ($\overline{\text{CEO}}$ = Low).
STC 02   Generates a 240 nanosecond positive true MLS
pulse for the magnetic card reader.
STC 04,08,16 Not used by the calculator.
OTX <SC> H/C Output A or B causes data bits from
A or B to be shifted to the I/O register
and accomplishes the following with the
various select codes. If C is given, a
240 nanosecond CLF pulse is given after OTX
is executed.
OTX 00   The 8 least significant bits in the A
or B register are shifted non-inverted
to the 8 least significant locations in
the I/O register, and 120 nanosecond after the
8th shift the "Device Ready" flip-flop is
set ($\overline{\text{CEO}}$ = Low). The 8 most significant
bits are shifted right 8 places and the
least 8 significant bits are recirculated
to the 8 most significant locations in
the A or B registers. The 8 most signi-
ficant bits in the I/O register are un-
touched.
OTX 01   Sixteen bits from the A or B re-
gister are shifted non-inverted
to the I/O register. The data in
A or B recirculates.
OTX 02   Not used by the calculator
OTX 04   Same as OTX 01 and in addition, 120 ns
after the 16th bit has been shifted nanoseconds
printer enable flip-flop is set
OTX 03   Same as OTX 01 and in addition, 120 nanoseconds
after the 16th bit has been shifted the
display enable flip-flop is set.

OTX 16 Same as OTX 01 and in addition, 120 nanoseconds after the 16th bit has been shifted the 240 nanosecond KLS signal is generated

LIX <SC> H/C Load into A or B. Loads data bits from the I/O register into the A or B register and accomplishes the following with the various select codes. If C is given, a 240 nanosecond CLF pulse is given after LIX is executed.

LIX 00 The eight least significant bits in the I/O register are shifted inverted to the eight most significant locations of A or B, and 120 nanoseconds after the 8th shift the "Device Ready" flip-flop is set ($\overline{\text{CEO}}$ = Low). A or B is shifted right eight places as the I/O register data comes in. The 8 most significant bits in the I/O register are untouched.

LIX 01 The 16 bits of the I/O register are transferred inverted to the A or B register. Data in the I/O register is lost.

LIX 02,04,08,16 Not used by the calculator.

MIX <SC> H/C Merge into A or B. Merges data from the I/O register into A or B registers and accomplishes the following with various select codes. If C is given, a 240 nanosecond CLF pulse is given after MIX is executed.

MIX 00 The eight least significant bits in the I/O register are merged with the eight least significant bits of the A or B register and shifted to the 8 most significant locations of A or B; 120 nanosecond after the merge takes place the Device Ready flip-flop is set ($\overline{\text{CEO}}$ = Low). A or B shifts right 8 places as the data is merged and shifted to the most significant locations. The 8 most significant bits of the I/O register are untouched.

MIX 01 The 16 bits of the I/O register are merged with the 16 bits of the A or B register and contained in the A or B register.

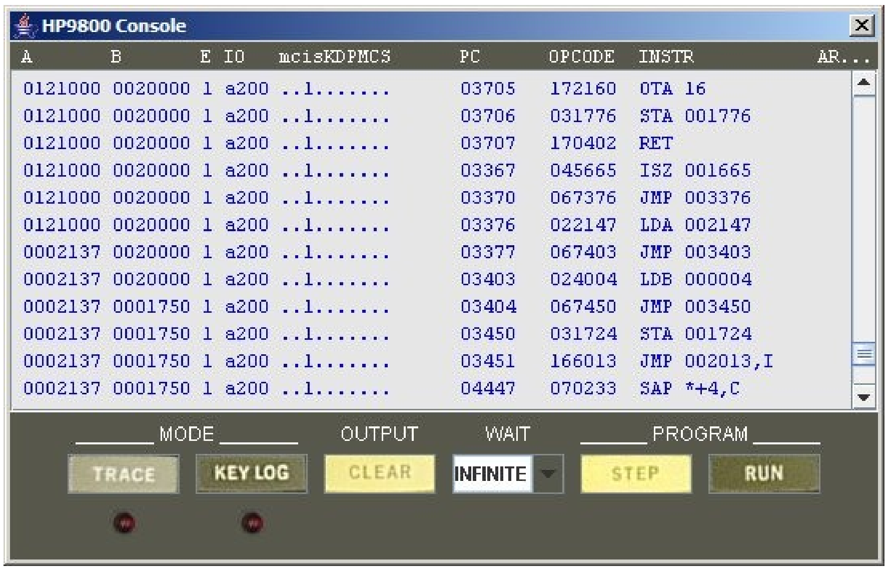
MIX 02,04,08,16 Not used by the calculator.

C Using the GO9800 Console

The emulator includes a console which allows inspection of the CPU registers and analysis of machine programs (either in ROM or RWM) at run-time. Additionally there is a special key-log function which allows to conveniently create key assignment configuration files.

C.1 Disassembler Functions

The disassembler may be controlled in a separate console dialog window. The console can be opened from any of the emulated calculator models by pressing the keys Ctrl+D. The dialog is also automatically opened, when a breakpoint or watchpoint condition is met (chapter 1.1.8). It can be closed by the window close icon or by again pressing the keys Ctrl+D when the calculator has input focus.



```
A         B         E IO    mcisKDPMCS        PC       OPCODE  INSTR              AR...
0121000 0020000 1 a200 ..1.......        03705    172160  OTA 16
0121000 0020000 1 a200 ..1.......        03706    031776  STA 001776
0121000 0020000 1 a200 ..1.......        03707    170402  RET
0121000 0020000 1 a200 ..1.......        03367    045665  ISZ 001665
0121000 0020000 1 a200 ..1.......        03370    067376  JMP 003376
0121000 0020000 1 a200 ..1.......        03376    022147  LDA 002147
0002137 0020000 1 a200 ..1.......        03377    067403  JMP 003403
0002137 0020000 1 a200 ..1.......        03403    024004  LDB 000004
0002137 0001750 1 a200 ..1.......        03404    067450  JMP 003450
0002137 0001750 1 a200 ..1.......        03450    031724  STA 001724
0002137 0001750 1 a200 ..1.......        03451    166013  JMP 002013,I
0002137 0001750 1 a200 ..1.......        04447    070233  SAP *+4,C
```

Figure C-1: The console dialog.

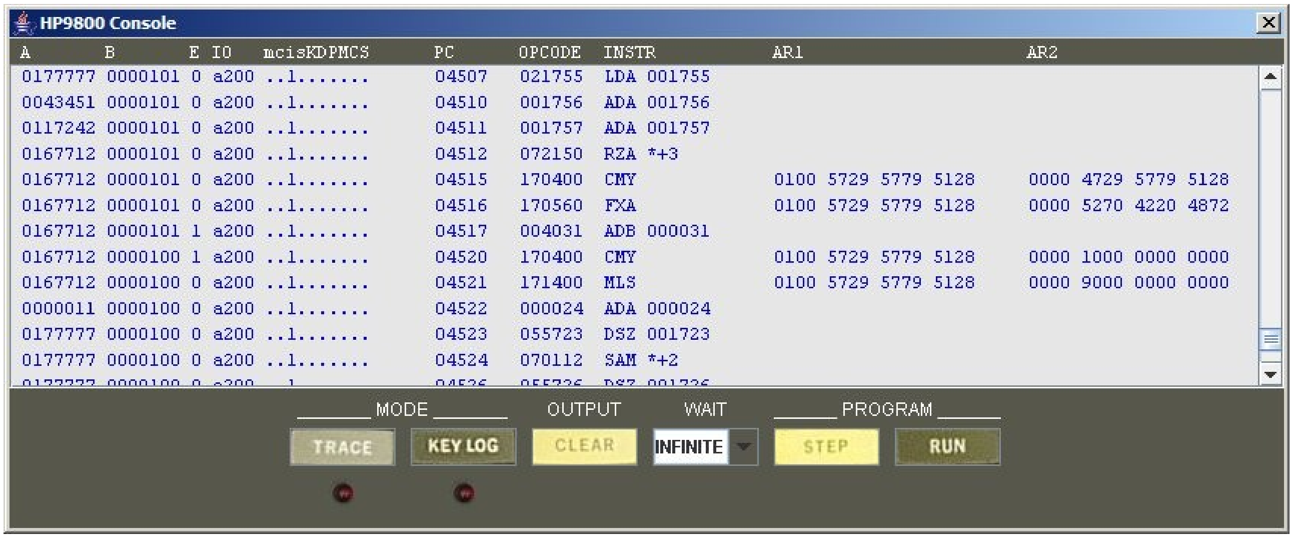
In the dialog window each diassembled CPU instruction is output *before* its execution together with the current contents of the following registers and flags in one text line:

- CPU registers A and B (16 bit octal values)
- Extend register E (4 bit hexadecimal value)
- I/O register (16 bit hexadecimal value)
- I/O flags (1 bit value, either 1 or 0, symbolized by a dot):
  - m (MLS): magnetic card reader control
  - c (MCR): magnetic card reader output
  - i (SIH): service inhibit
  - s (SSF): single service flip-flop
  - K (KLS): keyboard LEDs enable
  - D (DEN): display enable

  - P (PEN): printer enable
  - M (MFL): magnetic card input flag
  - C (CEO): device control enable output
  - S (STP): stop flag
- Program counter PC (16 bit octal value)
- Instruction OPCODE (16 bit octal value)
- Disassembled instruction

See appendix B for a detailed description of the CPU instructions. All memory addresses are given as octal values, relative branch values are given as offset to the program counter (*+n or *-n).

Additionally for BCD arithmetic instructions the contents of the two pseudo registers AR1 (memory address 01744-01747) and AR2 (address 01754-01757) are shown at right of the instruction. They can be made visible by enlarging the disassembler window. The values of AR1 and AR2 are given as four hexadecimal 16-bit words.



```
A       B       E IO   mcisKDPMCS    PC     OPCODE  INSTR             AR1                     AR2
0177777 0000101 0 a200 ..l.......    04507  021755  LDA 001755
0043451 0000101 0 a200 ..l.......    04510  001756  ADA 001756
0117242 0000101 0 a200 ..l.......    04511  001757  ADA 001757
0167712 0000101 0 a200 ..l.......    04512  072150  RZA *+3
0167712 0000101 0 a200 ..l.......    04515  170400  CMY               0100 5729 5779 5128     0000 4729 5779 5128
0167712 0000101 0 a200 ..l.......    04516  170560  FXA               0100 5729 5779 5128     0000 5270 4220 4872
0167712 0000101 1 a200 ..l.......    04517  004031  ADB 000031
0167712 0000100 1 a200 ..l.......    04520  170400  CMY               0100 5729 5779 5128     0000 1000 0000 0000
0167712 0000100 0 a200 ..l.......    04521  171400  MLS               0100 5729 5779 5128     0000 9000 0000 0000
0000011 0000100 0 a200 ..l.......    04522  000024  ADA 000024
0177777 0000100 0 a200 ..l.......    04523  055723  DSZ 001723
0177777 0000100 0 a200 ..l.......    04524  070112  SAM *+2
```

Figure C-2: Disassembler output showing the arithmetic pseudo registers.

C.2 Usage of the Disassembler

C.2.1 Online Disassembler Mode

The disassembler may be started by clicking on the TRACE button on the left side. Since the output of each executed instruction takes a certain amount of time, the speed of the calculator is slowed down significantly. Clicking again on on the TRACE stops the online disassembler mode.

The disassembled instructions can be inspected using the scroll bar on the right side. There are a maximum of 4096 lines kept in the window output buffer. Output of one additional line at the end results in deletion of the first line. The content of the buffer may be marked by mouse-dragging and copied and pasted in a text editor for later analysis.

The output buffer may be completely emptied using the CLEAR button.

C.2.2 Single Step Mode

It is also possible to step through a machine program, either manually or automatically at a selectable speed. Clicking on the STEP button halts the normal execution and switches the emulator to the single step mode. The next machine instruction is disassembled but the execution is delayed until either STEP is pressed again or the number of milliseconds selected in the drop-down list has elapsed. The default behaviour is infinite waiting for pressing of the STEP key. This may changed by selecting one of alternative values in the drop-down list or by keying in an arbitrary integer value.

The single step mode may be cancelled and normal execution resumed by clicking the RUN button.

The single step mode is automatically activated when a breakpoint or watchpoint condition is met.

C.2.3 Breakpoints

Breakpoints are locations in a machine program at which the normal execution is automatically halted and the disassembler single-step mode is entered. They can be defined in the calculator configuration file (see chapter 1.1.8).

Breakpoints can be set for any valid memory address. They become effective only when the program counter reaches the breakpoint address, not when the corresponding memory location is read or written by a CPU instruction.

C.2.4 Watchpoints

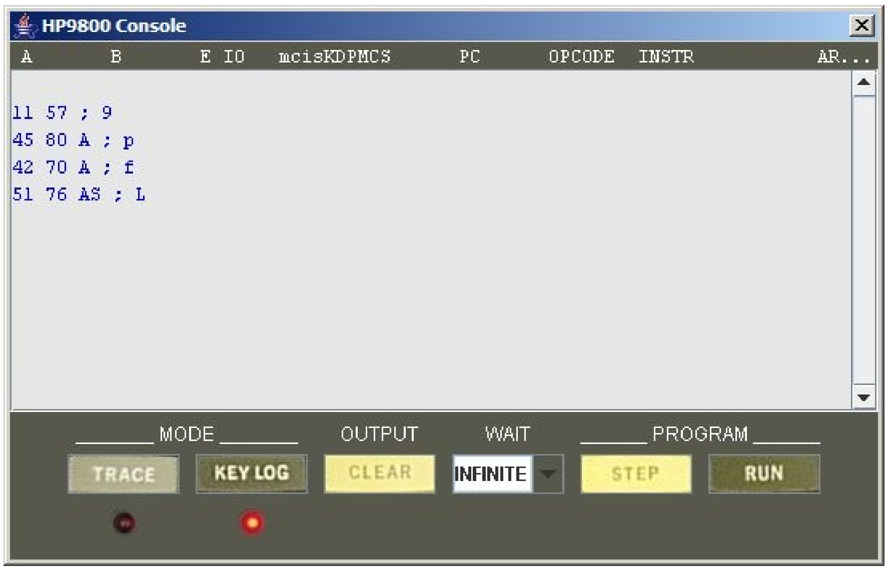
Watchpoints are locations in the calculator memory which are continuously monitored for any read or write access by a CPU instruction. There are two types of breakpoints: conditional and unconditional ones. If a memory location is accessed and there is a unconditional watchpoint defined, the disassembler single-step mode is activated. The watchpoint address and the current content of the memory location are output to the disassembler window.

If a memory location is accessed and there is a conditional watchpoint defined, the current content of that memory location is compared with a test value by a comparison operator (=, <, or >). If the result is logically true, the single-step mode is activated.

Note: A watchpoint is also effective if the CPU program counter reaches the watchpoint address.

C.3 Key-Log Mode

To assist in creation of keyboard configuration files the console implements a key-log mode with which every calculator and PC keystroke is recorded in the console log. To open the Console window press Ctrl+D.



```
11 57 ; 9
45 80 A ; p
42 70 A ; f
51 76 AS ; L
```

Figure C-3: Console with key-log mode activated.

To activate this mode press the KEY LOG button. The LED below this button is lit when key-log is active. In key-log mode all keys pressed on the PC keyboard and all mouse clicks on calculator keys are not sent to the calculator but their corresponding key codes are output in the console window.

C.3.1 Creating Keyboard Configuration Files

Before creating a new keyboard configuration the console window should be cleared using the CLEAR button. Click on the calculator window to give it the input focus.

To record one complete calculator / PC key assignment proceed as follows.

1. Decide which calculator key has to be assigned.
2. Mouse-click on the the calculator key. The corresponding octal key code is displayed in a new line.
3. Press the desired PC key together with the necessary modifier key(s) (Shift, Alt, Ctrl). The corresponding decimal key code and modifier shortcuts (S, A, C) are displayed on the same line. The equivalent ASCII character is added as a comment.
4. Mistakes can be corrected directly in the console output by mouse-clicking in the line to be corrected and using the common editor and text keys. Detailed comments maybe typed behind the semicolon.
5. Repeat from step 1 until all desired calculator keys are assigned.

Finally open an existing or new text file (e.g. the existing keyboard configuration of that calculator) in a text editor of your choice. Then, in the console output, mark all key assignment lines using the mouse or cursor keys, copy them to the clipboard (Ctrl+C), and paste them to the open text file. The text file should now be saved with the name <Model>-keyb.cfg in the installation folder of the emulator.