Emu71 documentation

1. Presentation:

Emu71 is a software emulator of the HP-71B machine.
Main features:
- runs under DOS or in a command box in Windows (95,98,2000,NT, ...).
- text mode application consistent with the HP-71B system look and feel.
- can emulate any numbers of ROM or RAM modules.
- very fast emulation engine written in optimized assembly language,
- runs correctly even on slow 186 (e.g. HP200LX), 286 systems although at reduced speed.
- FREEWARE: free for non commercial usage,
  (a short mail with your opinion to the author would be appreciated),
- emulation of the HP-IL loop and 8 HP-IL devices (one video display, three mass storage units, one printer port, one DOS interface, two serial COM),
- direct access to LIF image file archives,
- supports 43/50 lines video modes (very convenient!),
- compatible with cut&paste operation in Windows environment (very convenient!),
- support of the PC serial COM ports, even through USB bridge,
- optional control of external HP-IL loop from Emu71, and/or communication with an external HP-71B (using a ISA HPIL board or a PIL-Box).

NOTE: Emu71 will not run on 8088 and 8086 systems and needs a 80186 or higher uP. The reason is that Emu71 heavily uses the 4 bits shift opcode (eg. shr al,4) to extract the nibbles, this opcode exists only on 80186 and following x86 processors.

2. Preparation for use:

Emu71 uses images of HP-71B ROM and modules (Math, Forth, ...).

THESES FILES ARE NOT PROVIDED WITH EMU71
FOR OBVIOUS COPYRIGHT REASONS

You need to transfer these ROMs on PC. The easiest way to do it is to use the HP9114 drive. An other way is to use a HP-IL-RS232 interface, or the HP-IL card on PC:

- **You have a HP9114 drive**. You can make a dump on a LIF disk, then read it on the PC with a LIF/DOS transfer utility (one is provided: vlif.exe).
The following program can do the job on the HP-71B:

```
10 DIM A$[64]
20 CREATE TEXT "ROM71:TAPE",140000
30 ASSIGN #1 TO "ROM71:TAPE"
40 A=HTD("0")   ! start address of ROM to transfer
50 FOR I=0 TO 64*32-1  ! length of ROM (here for 64Kb)
60 A$=PEEK$(DTH$(A+I*64),64)
70 PRINT #1;A$
80 NEXT I
90 ASSIGN #1 TO *
```

Now, read this LIF disk on the PC with the following command:

```
vlif ROM71 >rom71.dmp
```

Note that ROM71 is the name of the file created by the HP-71B in UPPERCASE.

- **You don't have a HP9114** or compatible drive, but you can make a link between your HP-71B and a PC with the HP82164A HP-IL/RS232 converter or the HP82973A HP-IL interface card in PC (you have to know how to make these interfaces work).
You can dump the ROMs by a program such as this one:

```
10 DIM A$[64]
20 A=HTD("0")   ! start address of ROM to transfer
30 FOR I=0 TO 64*32-1  ! length of ROM (here for 64Kb)
40 A$=PEEK$(DTH$(A+I*64),64)
50 OUTPUT :1 ;A$
60 NEXT I
```

save the data in *.dmp files, eg: ROM71.DMP, MATHROM.DMP, ...

After the end of transfer (disc or link), you must have a ASCII file with 64 nibbles per line named ROM71.DMP (135168 bytes long).
Use the DMP2BIN command to build the binary files *.bin that can be used by Emu71.
eg: `DMP2BIN ROM71.DMP ROM71.BIN`
ROM71.BIN must be 65536 bytes long.
Repeat the same operations for all the ROMs you need.

To find the starting address of a ROM module:
- find the name of the first file inside the module with the CAT function:
e.g. for MATH module, the first (and only) file is "MATHROM"
- find the starting address of this file:
e.g. : `ADDR$("MATHROM")` may return B0008
- to build the module address, keep only the 2 highest nibbles and set the other to 0: B0xxx --> B0000

Note that some module ROMs are protected and can't be read with the standard `PEEK` function, so you must have an extra PEEK LEX such as the one in the JPC ROM. You can use the same way to transfer your own programs saved in independant RAM (IRAM).

3. Emu71 basic functions

3.1 Running the emulator:

```
Emu71 [option]
```

options:
  /d : call debugger at startup
  /t : system opcodes trace (configuration)
  /s : slow host

/s option is used to ajust the emulation speed to slow host processor like a 8-MHz 286 system, or the HP100LX/200LX

Performances:
A 8MHz 286 emulates the HP-71B at about half speed,
a Pentium 166 MHz emulates the HP-71B at about 30x speed,
a Pentium Centrino 2.4 GHz emulates the HP-71B at about 250x speed,

3.2 Exiting the emulator:

Normal exit by the HP-71B OFF command or shift-F1, with automatic saving of CPU and RAMs (main and independant).
Emergency exit (emulated CPU hanged, or user choice not to save RAMs) by ctrl-Z, then 'q' (see debugger description below).
In case of difficulty, you can make a master reset (memory lost) by deleting the sys71.dat file.

3.3 Files used by Emu71 (minimum configuration):

```
emu71.exe   : emulator
vlif.exe    : utility to read LIF disc
dmp2bin.exe : utility to build *.bin files
emu71.ini   : emu71 initialisation file, see description bellow
sys71.dat   : system file (CPU registers and I/O area backup)
rom71.bin   : image of HP-71B 64Kb ROM (to be built by user)
hpilrom.bin : image of HP-IL ROM (to be built by user)
ram00.bin   : 32Kb main RAM module in port 0
iram50.bin  : 32Kb independant module
hdrive1.dat : mass storage file (HDRIVE1 HP-IL device)
```

3.4 Debugger:

A debugger is present in Emu71.
You call it with ctrl-Z, or with the DEBUG command if the Forth/Assembler ROM is present (Emu71 emulates the hardware debugger at F0000 called by the DBGLEX file of this ROM).
To return to Emu71, use the 'r' (run) command.
The 'q' command is used to exit from Emu71. In this case, RAM is NOT saved.
'?' can be used to get the list of all available commands.
This debugger has been written to help the debugging of Emu71, it lacks a lot of commands, for example you can't change the values of registers or memory locations (because I didn't need it).

3.5 emu71.ini file

This file describes the modules plugged into the emulated machine, as well as the virtual HP-IL devices.
This file is made of two sections, each declared by a header:
[modules]: start of the memory modules declaration section,
[devices]: start of the HP-IL devices declaration section.

- Memory modules
The main RAM, ROMs et independant RAMs are only limited by the addressing space of the Saturn processor (512Kb total) and by the available DOS memory.
Each module, of any size, uses at least 32Kb of the Saturn addressing space (so, the modules are aligned at X0000 addresses). 64Kb modules are supported!
Format of the lines in emu71.ini :
port(0-5) type(ROM,RAM,HRD) size(Kb) file
Emu71 support up to 10 softwired modules (ROM or RAM), plus one
hardwired module at E0000 (HRD).
Exemple of emu71.ini file with 32Kb main RAM (port 0), Math (port 1), Forth (port 2), and 32Kb independant RAM (port 5) modules :

```
 0 RAM 32 ram00.bin
 1 ROM 32 mathrom.bin
 2 ROM 16 forthrom.bin
 2 HRD 32 hrdforth.bin
 5 RAM 32 iram50.bin
```

(note that the Forth module contains one 16Kb softwired ROM and one 32Kb hardwired ROM).

Caution: only port 0 RAM modules can be used as main memory. Don't use CLAIM PORT on other ports than 0.xx. See advanced functions for more details.

Emu71 basic memory space layout:

```
 00000-1FFFF : 64Kb internal ROM  (hardwired)
 20000-2FFFF : I/O and display RAM
 30000-3FFFF : 32Kb main RAM
 40000-DFFFF : softwired modules (up to ten 32Kb modules)
 E0000-EFFFF : 32Kb hardwired ROM (FORTH or HP41/71 Translator)
 F0000-FFFFF : hardware debugger
```

- HP-IL devices
The [devices] section is used to declare what are the active internal HP-IL devices, and what is the device order, as well as the external HP-IL interface (XIL). The internal device order is chosen by the user, but it is not allowed to declare twice the same device.
It is recommended to put the XIL declaration after all the others, so the external devices are after the internal devices.
See more details on HP-IL at chapters 4.4 and 4.5.
Example:

```
  [devices]
  DISPLAY
  HDRIVE1
  LPRTER1
  DOSLINK
```

3.6 HP-71B keyboard emulation:

HP-71B special keys are mapped to the following PC keys:

HP-71B	PC	
ATTN	Escape	
I/R	Insert	
-CHAR	Delete	
<-	<-	
->	->	
v	v	cursor down
^	^	cursor up
|<-	Home	cursor at beginning of line
->|	End	cursor at end of line
vv	Page v	cursor at bottom
^^	Page ^	cursor at top
BACK	<--	backspace
CMDS	Ctrl-Enter	
-LINE	Ctrl-Delete or Ctrl-Backspace	
EDIT	Ctrl-E	
FETCH	Ctrl-F	
RUN	Ctrl-R	
ON-/	Ctrl-/ or Ctrl-Page^	

For convenience, the F1 to F4 key functions are mapped to HP-71B specific keys, and F5 to F12 to various functions.

3.7 Display emulation:

The PC screen is shared by the HP-71B LCD emulation and the HP-IL video display (if used).

Emu71 now supports 43 and 50 lines display mode. To set this mode, use the MODE CO80,43 or CO80,50 command (DOS, MS-DOS command in Win95/98), or set the number of lines in the Command Box Properties (Win2000, WinXP, etc, ...) before launching Emu71.

The LCD is emulated in text mode. An extra feature is that Emu71 displays up to 78 characters of the output buffer (96 char.), instead of the regular HP-71B 22 characters.
See HP-IL chapter 4.3 for details on HP-IL video emulation.

4. Advanced functions

4.1 Emulation speed

Emu71 runs at the maximum speed allowed by the host processor.
On modern Pentium-based machines, the actual speed is much higher than the original HP-71B one and I didn't try to slow down the speed to the original one, because it doesn't make sense to me. Thanks to this high speed and to the large memory space available, complex calculations which were out of reach to the HP-71B can now be done on Emu71.

On very slow systems (186 and 286 based), the actual speed is slower than the original one.
Main drawback is the slow key response time.
To improve the speed on these systems, you can use the /s option which control the cpu time between display/keyboard handling and HP-71B emulation.
On 186 and 8MHz 286 systems, I recommend the /s4 option.

A SPEED utility is provided to check the speed improvement (see §4.6).

4.2 Clock emulation

Emu71 fully supports the HP-71B real time clock.
However, the clock is not automaticaly updated at each Emu71 startup. Emu71 places the actual date and time values at startup at certain I/O addresses.
You must use the 'MAJDATE' subroutine found in the iram50 file (i.e. PORT(5)).
The easiest way is to use the HP-71B STARTUP feature to automaticly launch it at each Emu71 startup:

```
STARTUP "CALL MAJDATE"
```

(Note: in French, "MAJ" means "Mise A Jour" i.e. update).

4.3 Support for various modules

The various modules must be declared in the [modules] section of the initialization file (see chapter 3.5).

- Standard memory modules

Emu71 supports standard 16k, 32k, 64k, 128k RAM or ROM memory modules.
Firms like CMT or HHP made large memory pacs (96k, 128k, 160k) which were actually combinations of 32k and 64k modules. To emulate such kind a pac, you have to declare each individual 'module'.
Example of a 96k pac (32k+64k modules) in port 5:

```
5 RAM 32 iram50.bin
5 RAM 64 iram51.bin
```

This will create a 32Kb port(5) and a 64Kb port(5.01)
When you declare new independant RAM modules in emu71.ini, the memory will be merged to the main RAM, and you have to use the `FREE PORT` command to configure them as independant RAM. (caution: it's not safe to use ports other than 0 as main RAM, see below).
To add a ROM module in Emu71, you may use the `ROMCOPY` function (included in iram50.bin module) to copy a ROM module from your HP-71B to a mass storage unit, then read it back into an independant RAM module in Emu71.

- Main memory

Emu71 supports main memory size larger than the initial 32Kb.
However, there are a few limitations:
- Users of previous Emu71 versions (1.x) should declare the first main RAM module (was internally set to ram71.bin in version 1.x) before adding other RAM modules in port 0:

```
  0 RAM 32 ram71.bin  ! was an implicit declaration in Emu71 1.x
```

If there is no RAM declaration in port 0, Emu71 still uses the implicit ram71.bin file.
- main memory must be made only from port 0 modules.
It means that you should not CLAIM PORT other than 0 to use it as main memory.
You can temporarily use CLAIM PORT on a RAM module from other ports, but only in the goal to reset its content: immediatly do a FREE PORT to put it back as independant port.
- to increase the main RAM size, just add a 32k module declarations to port 0 in emu71.ini.
example; to increase main memory to 96Kb, add two lines after the initial one:

```
  0 RAM 32 ram00.bin   ! initial declaration
  0 RAM 32 ram01.bin
  0 RAM 32 ram02.bin
```

This declaration will create the ports 0.01 and 0.02 at the next startup, and merge them to the main RAM.
- Important: use only 32k modules for main RAM. 64k modules will certainly create problems when used as main RAM.
- if you need to reduce the main memory size, you should take some precautions (as on the real HP-71B) to avoid memory lost. Always use FREE PORT before removing the declaration in emu71.ini, and always remove the ports in high to low order: e.g. 0.02 before 0.01.
As main memory size reduction is risky (memory lost in case of mistake), save all important files on mass storage first.

- Hardwired ROM modules

Emu71 supports hardwired ROM modules. As far as I know, this technique was used by HP on two modules only: the Forth/Assembler module and the HP41/71 translator module (which was actually based on the Forth core).
Hardwired module means that the address is fixed by the hardware. The HP-71B operating system supports a 32k hardwired space at E0000.
To declare a harwired module in Emu71, use the HRD module type.
Example of Forth module in port 2:

```
 2 ROM 16 forthrom.bin
 2 HRD 32 hrdforth.bin
```

Note that the Forth module contains one 16Kb softwired ROM and one 32Kb hardwired ROM.

- Hardware debugger ROM

Emu71 provides a minimum support for the hardware debugger in the sense that it supports the DEBUG keyword found in the Forth/Assembler module.
Support includes:
- debugger version returned in VER$ as "DBG:X"
- when DEBUG is executed, control is returned to the Emu71 debugger. Usually, you will use a sentance like `'DEBUG @ MYFUNCTION'` and you will continue execution up to a suitable breakpoint (e.g. your function entry point) by using the 'r addr' or 'g addr' command.
I would be very happy to provide additional support if I had more information on the hardware debugger. Please contact me if you had ever used the real hardware debugger.

- HP-IL module
The HP-IL module is supported. The HP-IL module ROM is 16k long, and can be declared in any port, e.g.:

```
1 ROM 16 hpilrom.bin
```

(note: for proper operation, the rom filename must start with 'hpil')
I recommend to use a port number other than 0 to keep port 0 for main RAM modules only.
The main goal of the HP-IL emulation is to provide video and mass storage functions.
Emu71 can only be loop controller, it can not PASS CONTROL or be a HPIL device, but it can give up CONTROL OFF, in this case Emu71 is no more in the HPIL loop.
Some very advanced or very low level features of the ROM module (like manual mode or auto IDY) are not supported.
The emulated HP-IL devices are described in the next section.

4.4 HP-IL support - internal virtual devices

Emu71 emulates the HP-IL interface, and provides 8 internal (or virtual) devices. These devices must be declared in the [devices] section of the emu71.ini initialization file (see chapter 3.5).
Below are described the internal devices:

- DISPLAY
The DISPLAY device emulates the HP82163 video interface.

```
AID=48
ID="DISPLAY"
```

Declaration in Emu71.ini:

```
DISPLAY
```

To use the display, do:

```
DISPLAY IS DISPLAY
```

To disable the HP-IL display and use only the single-line LCD emulation, do:

```
DISPLAY IS *
```

- HDRIVE1
The HDRIVE1 device emulates a mass storage unit compatible with the HP9114 by using a DOS image file.

```
AID=16
ID="HDRIVE1"
```

Declaration in Emu71.ini:

```
HDRIVE1 {name}
```

'name' is optional, and set the name of the image file. Default is: hdrive1.dat.
DDT and DDL command set compatible with the HP82161 and HP9114 units.
The data are recorded in a image file in the Emu71 working directory. Maximum size of this file is 640k.
This file is created with `INITIALIZE :HDRIVE1`. When created, the file is 32k long, its size is then increased when needed, up to 640k.
The format of this file is compatible with LIF disk images that can be found in some Web archives. So it is possible to directly read these files from these archives to Emu71 (then to copy them to an external HP-71B with the XIL option).

- HDRIVE2

The HDRIVE2 is similar to HDRIVE1.
Main usage of HDRIVE2 is to access LIF image archive, whereas HDRIVE1 is used as working drive.
Typical HDRIVE1/HDRIVE2 usage example:

```
HDRIVE1 mydisc.dat              my personal disc
HDRIVE2 swap\lexfl1.lif         path to a LIF archive image
```

Then it is possible to read files from HDRIVE2, and save them on HDRIVE1 if needed.
Tips: It may be convenient to set the archive as read-only file, so Emu71 will not overwrite any archive file by mistake (but instead will report 'Write Protected Medium' error).
It is easy to get confused with HP-IL addresses and do `COPY TO :3` instead of `COPY TO :2`. To do so, use the File Properties under Windows, or the ATTRIB utility under DOS.

- FDRIVE1

The FDRIVE1 device emulates a mass memory unit compatible with HP9114 by using the PC 3 .5” floppy drive. The goal is to allow reading the old LIF discs created with the HP9114 unit.

```
AID=16
ID="FDRIVE1"
```

Declaration in Emu71.ini:

```
FDRIVE1 {A|B}
```

The optional A (default) or B parameter set the PC drive.
DDT and DDL command set compatible with the HP82161 and HP9114 units.
This device doesn't allow to format discs. Nevertheless, it can read and write on discs previously formatted with the HP9114 unit.
Note: It is recommended to use this device only to read old LIF discs, it is also recommended to write protect the discs.
Note: This emulation works on true DOS and Windows 95/98 systems, but doesn't work on modern OS like Windows NT, Me, 2000, XP. Reason is that this emulation relies on low level BIOS calls to manage LIF formatted discs (256 bytes/sector). These calls are historic (there are not used by DOS/Windows) and seem to have been removed (or at least are no more supported) in modern PC OS.

- DOSLINK

The DOSLINK device provides an interface with the DOS file system.
It is possible to import data from a DOS file, and export data to a DOS file.

```
AID=78
ID="DOSLINK"
```

Declaration in Emu71.ini:

```
DOSLINK
```

Data sent to DOSLINK are written into the emu_out.dat file.
Data received from DOSLINK are read from the emu_in.dat file.
These files are in the Emu71 working directory.
Tips: sending a `CLEAR :LOOP` or `CLEAR :DOSLINK` close the files. Next DOSLINK access will open the files again from the start: emu_out.dat will be overwritten and emu_in.dat will read again from the start. This can be useful to save emu_out.dat content to an other file name, or to load the emu_in.dat with new data.
Note: These files are managed as binary files by DOSLINK, so you can read/write binary data. To write DOS text file, use PRINT instructions and CR/LF line terminator (see `ENDLINE` keyword).

- LPRTER1
The LPRTER1 device is a pure interface that uses the parallel port.
Although its name is LPRTER1, it is recognized as an interface similar to the HP82165 GPIO interface or the HP82166 converter that are commonly used as parallel printer interface.

```
AID=64
ID="LPRTER1"
```

Declaration in Emu71.ini:

```
LPRTER1
```

To use it as printer, do:

```
PRINTER IS :LPRTER1
```

- SERIAL1
The SERIAL1 device is a pure interface that uses a serial port.

```
AID=66
ID="SERIAL1"
```

Declaration in Emu71.ini:

```
SERIAL1 [ComPort[,InitByte]]
```

ComPort and InitByte are two optional parameters. Initbyte is a hex number.
ComPort specifies the COM port. Default value is 1, for COM1.
InitByte specified the communication parameters. It is a hex value with the meanings:

```
bit  7 6 5          4 3       2         1 0
     speed          parity    stop      length
     010=300        x0=none   0=1bit    10=7bits
     011=600        01=odd    1=2bits   11=8bits
     100=1200       11=even
     101=2400
     110=4800
     111=9600
```

Default value is E3(hex) = 9600 bauds, no parity, 1 stop, 8 bits.
Actually, this is the value send to the INT14, function 0

- SERIAL2
Similar to SERIAL2, but ID="SERIAL2" and defaults to COM2.

- Important notes about SERIALx:
Emu71 uses the BIOS functions (INT 14) to communicate with the COM port.
The BIOS functions are very limited and some points must be understood:
- Usually, PC BIOS expects the CTS and DSR lines to be active to be able to send a character. If these lines are inactive, no character will be transmited.
See the correct cable wiring to a HP82164 interface in an associate document on my site.
- In Windows environment, Emu71 must be active i.e. not in idle state, but in a running program or running a I/O function (`ENTER`, `COPY` for instance) for reliable reception of characters from the Windows OS.
- USB/serial bridges seem to work. If a USB/RS232 interface installs itself as COM4 for instance, declare it as:

```
SERIAL1 4   (or SERIAL2 4)
```

- The serial support is provided for experimental or hobby purpose only.
It is believed to be reasonnably reliable for downloads from PC to HP71 through the HP82164A.
Upload from the HP71 to the PC is not guaranteed.
I can't provide any support or help for problem solving.

4.5 Support of the external HP-IL interface

Emu71 allows also to manage real external devices using the HP82973A HP-IL board for PC (or the equivalent board rebuilt by Christoph Klug) or using a PIL-Box. It also allows to communicate with an external HP-71B.
The declaration of the external loop is made with the XIL declaration in the [devices] section of the emu71.ini file:

```
XIL {addr}
```

or

```
XIL COMx
```

If the parameter is a hexadecimal number, then it specifies the address of a HP-IL/PC ISA board, default value is 1700(hex).
If the parameter is the string "COMx" and 'x' is a number from 1 to 4, then it specifies a HPIL/ serial converter (PIL-Box) connected to a COM port, either a native COM1 or COM2, or a virtual comunication port (VCP) provided for instance by a USB/RS232 converter.
Note: a HP-IL/serial converter (PIL-Box) is NOT a HP82164 HP-IL/RS232 interface. Visit my Web site for information on the PIL-Box.
Although not mandatory, it is recommended to put the XIL declaration after all the internal device declarations.

There are three usages of the external HP-IL interface:

- control of external devices from Emu71
Emu71 manages the devices connected to the PC. The external HP-IL loop should be closed, and all devices should be turned on.
Emu71 includes a very nice feature: the external loop integrity is checked each time a `RESTORE IO` is issued. If the messages don't return in a short time, then the external loop is assumed to be broken, and is bypassed until a next `RESTORE IO` shows to Emu71 that the external loop is valid.
In this way, we can use Emu71 with an open external loop, or with shutdown devices, and still have access to the internal devices. To have access to the external devices, just close the loop, turn on all devices and do `RESTORE IO`.

- communication with an external HP-71B
In this case, Emu71 is always the loop controller.
To exchange program and data files with an external HP-71B, just make it a device by typing 'CONTROL OFF' on the HP-71B. Then do 'RESTORE IO' on Emu71.
You can now copy files between Emu71 and HP-71B.
Emu71 to HP-71B:
do `'COPY :LOOP'` on the HP-71B
do `'COPY file TO :HP71'` on Emu71
HP-71B to Emu71:
do `'COPY file TO :LOOP'` on the HP-71B
do `'COPY :HP71'` on Emu71
Always send the command on HP-71B side first, then on Emu71.
If you are an experimented HP-71B and HP-IL user, you even can use the HP-71B remote capability and type all commands on Emu71 side.

For example, to transfer a file from HP-71B to Emu71, type on Emu71 side:

```
REMOTE
OUTPUT :HP71;"COPY file TO :LOOP"
LOCAL
COPY :HP71
```

Of course, this can be made under program control, providing automatic backup functionality.

- access to this internal device from an external HP-71B

Although Emu71 is unable to `PASS CONTROL` or be a HPIL device, it can give up `CONTROL OFF`. In this case, Emu71 is no more in the HPIL loop, and an external HP-71B can access the internal devices.

To make the external HP-71B the loop controller:

do `CONTROL OFF` on Emu71,

do `CONTROL ON, RESTORE IO` on the HP-71B.

To give back HPIL control to Emu71:

do `CONTROL OFF` on the HP-71B,

do `CONTROL ON` on Emu71.

It is possible to use this feature to exchange program and data files, both Emu71 and the HP-71B can access the HDRIVE1 (for instance) drive in turn to store/retrieve files.

4.6 Utilities:

The JFG file in the iram50.bin module provides some utilities:

`SUB MAJDATE`: used to setup the time system at each startup.

To be called as `STARTUP "CALL MAJDATE".`

`SUB CLOCK`: Display a running clock. Useful to verify the correct emulation of the clock system. If the clock is running late, slow down the emulation speed with the /s option of Emu71.

`SUB SPEED`: useful to compare the emulator speed with the real HP-71B.

`SUB LOOPCAT`: list all HP-IL devices.

5. Limitations:

- The BEEP and GDISP commands do nothing (but the GDISP$ function works!).
- The KEYDOWN function always returns 0.
- The ROM modules are not write protected (ROM image files are not affected).
- Emu71 can not PASS CONTROL or be a HPIL device (but it can give up CONTROL OFF).
- Multiple HP-IL is not supported ("NO LOOP" error for loop number 2 or 3).
- By default, Emu71 will shutdown itself after 10 minutes of inactivity, to avoid this, set flag -3 (SFLAG -3).

Author:
EMU71 was written by Jean-Francois Garnier
For more information or any update, please visit my home page at:
http://membres.lycos.fr/jeffcalc

J-F Garnier, March 2010.