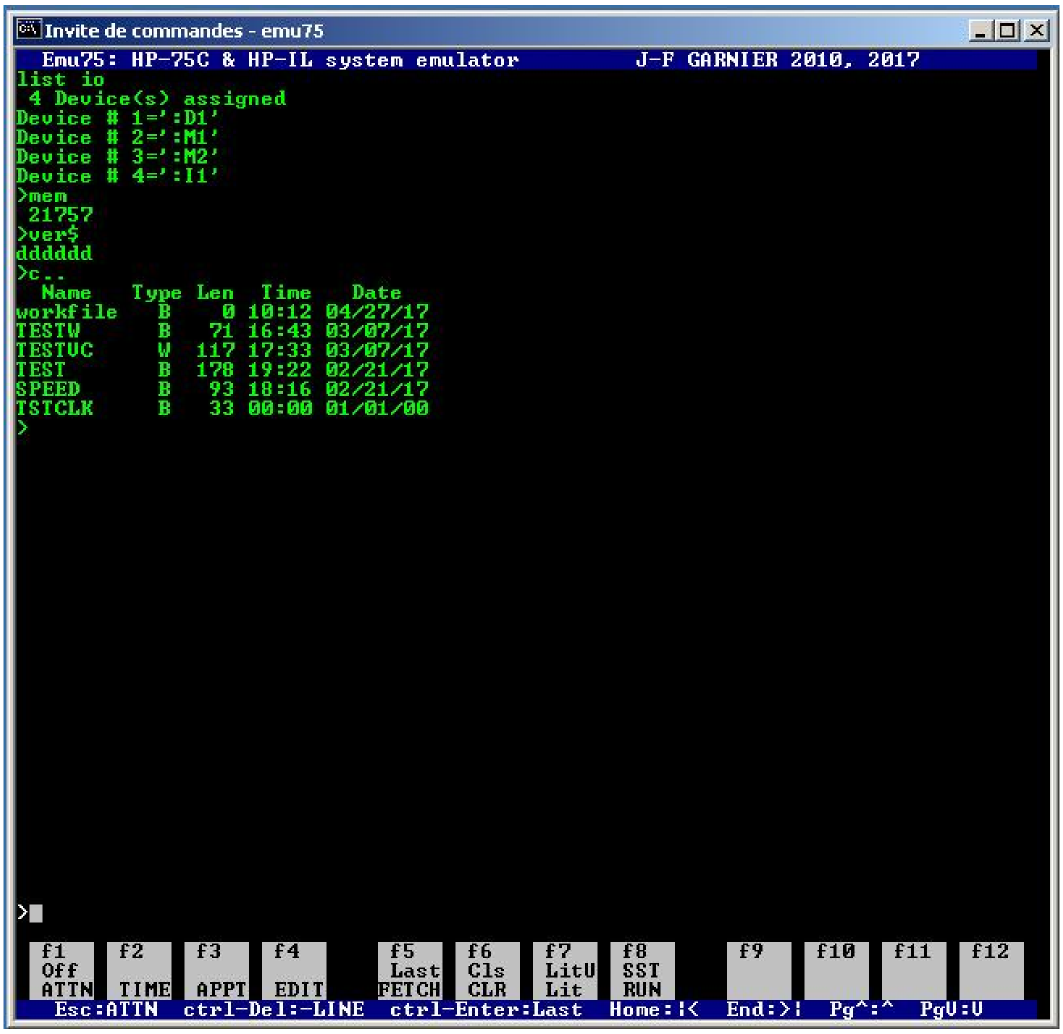
Emu75 documentation
[Version 1.3 - October 2018]

1. Presentation:

Emu75 is a software emulator of the HP-75C machine.



```
  Emu75: HP-75C & HP-IL system emulator          J-F GARNIER 2010, 2017
list io
 4 Device(s) assigned
Device # 1=':D1'
Device # 2=':M1'
Device # 3=':M2'
Device # 4=':I1'
>mem
 21757
>ver$
dddddd
>c..
  Name     Type Len  Time    Date
workfile    B     0 10:12 04/27/17
TESTW       B    71 16:43 03/07/17
TESTVC      W   117 17:33 03/07/17
TEST        B   178 19:22 02/21/17
SPEED       B    93 18:16 02/21/17
TSTCLK      B    33 00:00 01/01/00
>

>
 f1    f2    f3    f4        f5    f6    f7    f8        f9    f10   f11   f12
 Off                         Last  Cls   LitU  SST
 ATTN  TIME  APPT  EDIT      FETCH CLR   Lit   RUN
   Esc:ATTN  ctrl-Del:-LINE  ctrl-Enter:Last  Home:|<  End:>|  Pg^:^  PgV:V
```

Emu75 running in 50-line mode in a command box on a Windows 2000 system

Main features:
- runs under DOS or in a command box in 32-bit Windows (95,98,2000, XP, ...), or in virtual environment such as DOSBox in 64-bit OS,
- text mode application consistent with the HP-75C/D system look and feel,
- can emulate any numbers of ROM modules (up to the limit of the HP75 OS),
- support of the Plug-in Module Simulator (PMS) and the Expansion Pod memory,
- emulation of the HP-IL loop with integrated video display, mass storage units and DOS interface,
- direct access to LIF image file archives,
- supports 43/50 lines video modes,
- compatible with cut&paste operation in 32-bit Windows environments.

2. Preparation for use:

Emu75 uses images of HP-75C ROM and modules (I/O ROM, ...).
The basic ROM images of the HP75 machine and I/O ROM are available on my site.
Other ROM images may be found on the Web.

To dump the ROM content from a HP75 requires a LEX file with a `PEEK` function.
Below are some indications on how to dump the ROM from a HP75 machine, using a PIL-Box and the ILPer software:

- load the MEMXLEX file from the CHHU01 or SWAP09 archive into the HP75,
- write a HP75 program using the `PEEK` or `ROMPEEK` function to read the ROM and send it as ASCII characters to the ILPer printer area,
- copy and paste the results from the ILPer printer area to a text file,
- on the PC, write a small utility in your preferred programming language (or use my 16-bit DOS program dmp2bin) to convert the various 8KB dumped blocks into binary files,
- assemble the 8KB blocks to create the system ROM and modules.

Here is a HP75 program I used almost 20 years ago, slightly adapted to the PIL-Box and ILPer:

```
10 ! --- rdump2 ---
20 ! romdump: dump HP-75 ROM
30 ! jfg 8 mar.98 - revised april-may 2017(!) for use with PIL-Box & ILPer
40 DIM B$[80]
50 ! GOTO 120 ! uncomment this line to skip step 1
60 ! step 1: lower 24K ROM & upper 8K ROM
70 R=0
80 A=0 @ F$="Blk0" @ GOSUB 250
90 A=8*1024 @ F$="Blk1" @ GOSUB 250
100 A=16*1024 @ F$="Blk2" @ GOSUB 250
110 A=56*1024 @ F$="Blk7" @ GOSUB 250
120 ! step 2: paged 8K ROM
130 R=32
140 A=24*1024
150 ON ERROR GOTO 230
160 X=ROMPEEK(A,R)
170 OFF ERROR
190 F$="rom"&CHR$(R+16) ! start at "rom0"
200 GOSUB 250
210 R=R+2 @ GOTO 150
230 OFF ERROR @ DISP "Finished !" @ BEEP
240 END
250 ! dump 8K block
300 PRINT @ PRINT F$
320 FOR I=A TO A+8191 STEP 32
325 DISP F$;I @ B$=" "
340 IF R>0 THEN 400
350 FOR J=I TO I+31
360 X=PEEK(J)
370 B$=B$&CHR$(48+X\16)&CHR$(48+MOD(X,16))
380 NEXT J
390 GOTO 440
400 FOR J=I TO I+31
410 X=ROMPEEK(J,R)
420 B$=B$&CHR$(48+X\16)&CHR$(48+MOD(X,16))
430 NEXT J
440 PRINT B$
450 NEXT I
490 RETURN
```

The step 1 can be skipped to dump only a new ROM module. Remove all but the module that must be dumped otherwise the blocks from the different modules will be mixed.
The last three dumped 8K blocks are the internal HP75 ROM.

3. Emu75 basic functions

3.1 Running the emulator:

```
Emu75 [option]
```

options:
  /d : call debugger at startup
  /m : HP-IL frame monitor
  /f : fast host

By default, Emu75 should work well in slow environments, such as DosBox that is needed to run Emu75 on modern 64-bit OS.
The /f option can be used to optimize the performance on native 16/32 bit OS.

Performances:
Emu75 runs at about 40x the native speed on a Pentium-4 at 3GHz with a 32-bit Windows 2000.
Emu75 runs at about the native HP75 speed in DosBox on a Core-i3 with a 64-bit Windows OS.

3.2 Exiting the emulator:

Normal exit by the HP-75C `off` command or shift-F1, with automatic saving of the system status and RAM.
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 do a master reset (memory lost) by deleting the sys75.dat file.

3.3 Files used by Emu75 (minimum configuration):

```
 emu75.exe   : emulator
 emu75.ini   : emu75 initialisation file, see description bellow
 sys75.dat   : system file (I/O area backup)
 ram75.dat   : 24KB main RAM
 sysrom.bin  : image of HP-75C system (to be provided by user)
 basrom.bin  : image of HP-75C basrom (to be provided by user)
 altrom.bin  : image of HP-75C altrom (to be provided by user)
 melrom.bin  : image of HP-75C melrom (to be provided by user)
 hdrive1.dat : mass storage file (HDRIVE1 HP-IL device)
 dmp2bin.exe : utility to build the HP75 *.bin files (see §2)
```

Note: this dmp2bin utility is different from the Emu71 version. Don't mix the versions.

3.4 Debugger:

A debugger is present in Emu75.
The debugger is called with ctrl-Z. To return to Emu75, use the 'r' (run) command.
The 'q' command is used to exit from Emu75. In this case, RAM is NOT saved.
'?' can be used to get the list of all available commands.
This debugger has been used to help the debugging of Emu75, 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 Emu75.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 ROM module section,
[devices]: start of the HP-IL device section.

- ROM modules

This section describes the internal and external ROMs.
The internal ROMs are made of the 24KB system ROM, the 8KB “Basic ROM” and the two bank-switched “ALT ROM” and “MEL ROM”.
The port 0 is reserved for these internal ROMs.
Format of the lines in emu75.ini :

```
port(0) type(HRD/ROM) size(KB) file
```

These ROM must be declared as:

```
 0 HRD 24 sysrom_d.bin   system ROM 0000-5FFF
 0 HRD  8 basrom_d.bin   basic  ROM E000-FEFF
 0 ROM  8 altrom1d.bin
 0 ROM  8 melrom_d.bin
```

The names of the bank-switched altrom and melrom must start with "alt" and "mel" respectively for Emu75 to properly allocate them.

The other plug-in ROM modules are only limited by the bank addressing space of the HP75 (16 external 8 KB banks = 128 KB total) and by the available DOS memory.
Format of the lines in emu75.ini :

```
port(1-3) type(ROM/RAM) size(KB) file
```

The port number (1-3) is irrelevant – but must not be zero.
The RAM type is used for the PMS, see section 4.3.
Example of the I/O ROM:

```
 1 ROM 24 iorom.bin
```

- HP-IL devices

The [devices] section is used to declare what are the active internal HP-IL devices, and what is the device order. The internal device order is chosen by the user, but it is not allowed to declare the same device twice.
See more details on HP-IL at chapters 4.4 and 4.5.
Example:

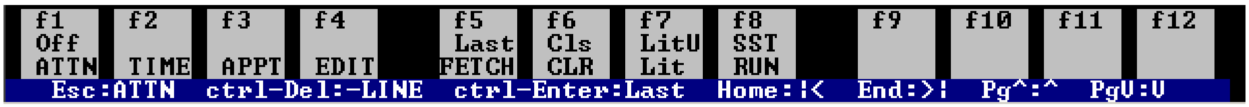
```
  [devices]
  DISPLAY
  HDRIVE1
  HDRIVE2
  DOSLINK
```

3.6 HP-75C keyboard emulation:

HP-75C special keys are mapped to the following PC keys:

HP-75C	**PC**	
ATTN	Escape	
I/R	Insert	
-CHAR	Delete	
<-	<-	
->	->	
v	v	cursor down
^	^	cursor up
Shift <-	Home	cursor at beginning of line
Shift ->	End	cursor at end of line
Shift v	Page v	cursor at bottom
Shift ^	Page ^	cursor at top
TAB	Tab	
BACK	<--	backspace
Ctrl-FETCH	Ctrl-Enter	recall last input (also mapped to shift-F5)
Shift-DEL	Ctrl-Delete or Ctrl-Backspace	
EDIT	Ctrl-E	
FETCH	Ctrl-F	
RUN	Ctrl-R	
Shift-RUN	Ctrl-S	single step (SST)

Function keys F1 to F12:



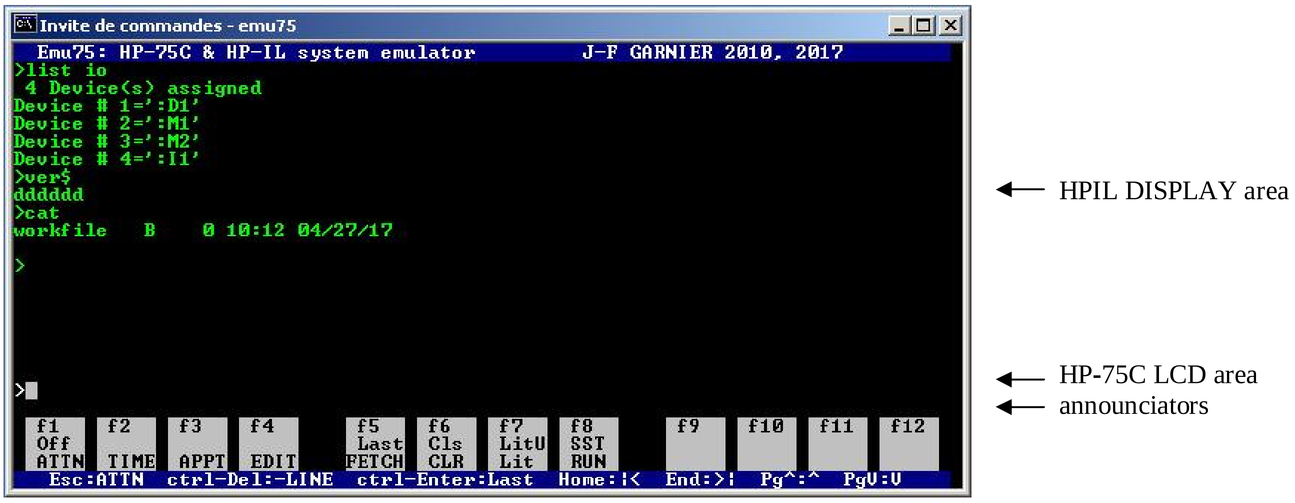
```
 f1    f2    f3    f4        f5    f6    f7    f8        f9    f10   f11   f12
 Off                         Last  Cls   LitU  SST
 ATTN  TIME  APPT  EDIT     FETCH  CLR   Lit   RUN
  Esc:ATTN  ctrl-Del:-LINE  ctrl-Enter:Last  Home:|<  End:>|  Pg^:^  PgV:V
```

F1 to F4 are mapped to the four top left HP75 keys: `ATTN`, `TIME`, `APPT` and `EDIT`.
Shift-F1 is the `Off` command used to quit Emu75, shift-F2 to shift-F4 functions are depending on the context.
F5, F6 and F8 are mapped to `FETCH`, `CLR` and `RUN`.
F7 and shift-F7 are mapped to the HP75 "literalize" keys `Shift-I/R` and `Ctrl-I/R` (`Ctrl-I/R` is the "literalize with underline" key, usable when the I/O ROM is present).
Cls (shift-F6) is the "clear screen" key (`Ctrl-CLR`) usable when the I/O ROM is present.
The keys F9 to F12, and shift-F9 to shift-F12 are mapped to `Ctrl-A` to `Ctrl-D` and `Ctrl-N` to `Ctrl-P`. They can be used as programmable keys by assigning key definition with `DEF KEY`.

3.7 Display emulation:

The PC screen is shared by the HP-75C LCD emulation and the HP-IL video display (if used). Emu75 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 expand the size of the command box (Win2000, WinXP, etc, ...) before starting Emu75.

An extra feature is that Emu75 displays up to 78 characters of the output buffer (96 char.), instead of the regular HP-75C 32 characters.
See HP-IL chapter 4.3 for details on HP-IL video emulation.



```
  Emu75: HP-75C & HP-IL system emulator         J-F GARNIER 2010, 2017
>list io
 4 Device(s) assigned
Device # 1=':D1'
Device # 2=':M1'
Device # 3=':M2'
Device # 4=':I1'
>ver$
dddddd
>cat
workfile   B    0 10:12 04/27/17

>




>
 f1    f2    f3    f4       f5    f6    f7    f8       f9    f10   f11   f12
 Off                       Last  Cls   LitU  SST
 ATTN  TIME  APPT  EDIT   FETCH  CLR   Lit   RUN
   Esc:ATTN  ctrl-Del:-LINE  ctrl-Enter:Last  Home:|<  End:>|  Pg^:^  PgV:V
```

4. Advanced functions

4.1 Emulation speed
Emu75 always runs at the maximum speed permitted by the host system.
A SPEED utility is provided to check the speed (see §4.6).

4.2 Clock emulation
Emu75 fully supports the HP-75C real time clock.
The emu75 clock is automatically updated from the host system time at each Emu75 startup.
It is still needed to set the date after the first Emu75 power-up (cold-start).

4.3 Support for various modules

The various modules must be declared in the [modules] section of the emu75 initialization file.

- Main memory
Emu75 provides the full 24KB main memory.
No declaration is needed in emu75.ini.

- Standard ROM memory modules
Emu75 supports standard 8k, 16k, 24k, and 32k ROM modules.
The 16k, 24k and 32k ROM modules can be loaded using either a single file with the full content of the ROM, or several individual 8k files, for instance for the 24KB I/O ROM:
```
 1 ROM 24 iorom.bin
```
or
```
 1 ROM  8 iorom1.bin
 1 ROM  8 iorom2.bin
 1 ROM  8 iorom3.bin
```

- Diagnostic ROM (Dingo)
Emu75 (from version 1.1) partially supports the "Dingo" diagnostic ROM.
The ROM operates properly, but several tests fail and/or lock Emu75.
The diagnostic ROM name must start with "dia" and must be declared in port 0 as:
```
  0 ROM  8 diag75.bin
```
Note that the diagnostic ROM is usable in DOSBox, because the execution speed is close to the real machine. In a native environment (such as a 32-bit Windows), the execution is way too fast and the user messages can't be read.

- Plug-in Module Simulator (PMS) HP 82173A
Emu75 supports the PMS.
To declare a PMS in emu75.ini, use the RAM module type. The PMSCMDS LEX is needed to manage the PMS ports (warning: a PODPMS1C LEX exists that is a patched version of the original LEX with a behaviour different than documented in the HP82173A manual).
The PMS RAM file can be 8, 16 or 24 KB and must be declared in first position in the [MODULES] section.
Example of a 16 KB PMS :
```
 1 RAM 16 pmsram.bin
```
The content of the PMS RAM is automatically saved at Emu75 exit.

- Expansion Pod HP 82718A

Emu75 (from version 1.3) supports the Pod memory.
To declare the expansion Pod ROM:

```
 3 ROM 16 exppod.bin
```

or using individual ROM image files:

```
 3 ROM 8 mmaster1.bin
 3 ROM 8 modem.bin
```

The Pod is using a special "MEMIC" device that uses a select code and existed in versions 32 or 64 KB. Emu75 uses a special configuration in the `[OPTION]` section of emu75.ini to define the MEMIC select code and the memory size:

```
  [OPTION]
  MEMIC 2 32 podmem.dat
```

For emulation of the genuine 32 or 64 KB Pod, use the select code 2 only. The next parameter specifies the Pod memory in KB, the last parameter specifies the name of the image file.
If the MEMIC declaration is changed, then it is needed to initialize the Pod memory again with:

```
  INITIALIZE ":XMEM"
```

Advanced Pod use and configurations:

- The genuine MEMIC can manage up to 128 KB of memory space. It is possible to emulate this configuration with:

```
  MEMIC 2 128 podmem.dat
```

Note, however, that only 120 KB are usable by `XMEM` because the last 8 KB space is occupied by the `MODEM` device.

- Using the `IMEM` device:
A hidden feature of the Pod ROM code is the support of the `IMEM` device. The genuine Pod doesn't provide the `IMEM` memory but Emu75 fully supports it.
To use the `IMEM` device, the Pod memory declaration must be larger than 128 KB, the first 128 KB will be allocated to `IMEM` and the rest (up to 128 KB) to `XMEM`.
For example, the configuration

```
  MEMIC 2 224 podmem.dat
```

declares a 128 KB `IMEM` and a 96 KB `XMEM`.
The `IMEM` device operates in the same way than `XMEM` but must be manually initialized with

```
  INITIALIZE ":IMEM".
```

As for `XMEM`, only 120 KB are actually usable by `IMEM.`

- Using the MEMIC select code 1 (experimental):
By using the MEMIC select code 1, it is possible to extend the memory capability of the original MEMIC and manage up to 16 MB (16000 KB). The `IMEM` is still limited to 128 KB.
Example:

```
  MEMIC 1 1024 podmem.dat
```

declares a 128 KB `IMEM` and a 896 KB `XMEM`.
With select code 1, if no memory image file exists, a "Ram is invalid" error may happen and emu75 must be ended and re-started to ensure the memory image file to be correctly initialized.
The support of select code 1 is provided for Pod experimentation only, the `XMEM` performance is reduced due to the limitations of the DOS memory management.

Note: the Pod memory image file holds both the `IMEM` and `XMEM` memories. First 128 kB is the `IMEM` even if not used, then the `XMEM` up to the declared memory size.

4.4 HP-IL support - internal virtual devices

Emu75 emulates the HP-IL interface, and provides 8 internal (or virtual) devices. These devices must be declared in the [devices] section of the emu75.ini initialization file (see chapter 3.5). Below are described the internal devices:

- DISPLAY
The DISPLAY device emulates a video interface compatible with the HP82163.
```
  AID=48
  ID="DISPLAY"
```
Declaration in Emu75.ini:
```
  DISPLAY
```
To use the display, do:
```
  display is ':d1'
```
(assuming the display has been assigned to the ':D1' identifier).
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 Emu75.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 Emu75 working directory.
This file is created with `initialize ':m1'`.
(assuming the unit has been assigned to the ':M1' identifier).
When created, the file is 32k long, its size is then increased when needed.
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 Emu75.

- 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\chhu1.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 Emu75 will not overwrite any archive file by mistake (but instead will report `'mass mem error'`).
To do so, use the File Properties under Windows, or the ATTRIB utility under DOS.

- 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 Emu75.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 Emu75 working directory.
Tips: sending a `clear loop` or `clear ':i1'` closes the files.
(assuming the DOS interface has been assigned to the ':I1' identifier).
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

```
  printer is ':i1'
```

(assuming the DOS interface has been assigned to the ':I1' identifier),
and use `print` instructions with CR/LF line terminator (see the `endline` keyword).

- FDRIVE1 ***[Obsolete – kept for reference only]***
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 Emu75.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.

- LPRTER1 ***[Obsolete – kept for reference only]***
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 Emu75.ini:

```
  LPRTER1
```

To use it as printer, do:

```
  printer is ':i2'
```

(assuming the printer interface has been assigned to the ':I2' identifier),

- SERIAL1 ***[Obsolete – kept for reference only]***
The SERIAL1 device is a pure interface that uses a serial port.

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

Declaration in Emu75.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 ***[Obsolete – kept for reference only]***
Similar to SERIAL2, but ID="SERIAL2" and defaults to COM2.

- Important notes about SERIALx:
Emu75 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 transmitted.
See the correct cable wiring to a HP82164 interface in an associate document on my site.
- In Windows environment, Emu75 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 reasonably reliable for downloads from PC to HP75 through the HP82164A.
Upload from the HP75 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 *[Obsolete – kept for reference only]*

Emu75 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.
The declaration of the external loop is made with the XIL declaration in the [devices] section of the emu75.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 communication port (VCP) provided for instance by a USB/RS232 converter, with the following parameters: 9600 bps, 8 bits, no parity.
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.

The only usage of the external HP-IL interface within Emu75 is:

- control of external devices from Emu75
Emu75 manages the devices connected to the PC. The external HP-IL loop should be closed, and all devices should be turned on.

Note: the parts marked as ***Obsolete – kept for reference only*** are still present and functional in Emu75, but are limited to obsolete systems (true DOS, parallel and serial interfaces, ISA board). The native support of the PIL-Box is not recommended on modern OS, the ILPer software is now the natural gateway between Emu75 and the HP-75C machine.

4.6 Utilities:

The “SPEED” program is on the hdrive1.dat image file.

5. Limitations:

- The `BEEP` command does nothing.
- The modem of the expansion Pod is not supported.
- The PMS is limited to 24 KB and must be declared in first position in the [modules] section of emu75.ini.

Author:
Emu75 has been written by Jean-Francois Garnier.
For more information or any update, please visit my home page at:
`http://www.jeffcalc.hp41.eu/emu75`

J-F Garnier, Oct 2018.