Description of the Philog USB stack

The ***Philog***'s USB stack is fully portable.

To simplify the adaptation to other OS and hardware (for instance, the stack exists as well on Nucleus and Windows NT respectively on ARM9 and Intel platforms), all OS and hardware dependant functions are isolated from the main code.

The stack ships in binary form for the core stack and source code for the BSP part and optional device drivers. The ***Philog***'s license is an unlimited copy license.

Philog can do portage, support, maintenance, evolutions and device driver development.

Design features:

- **USB 2.0**
  Supports the USB 2.0 satndard, compatibility with the USB 1.1 peripherals guaranted.

- **UHCI, OHCI, and EHCI Host Controllers**
  All the USB 1.1 and 2.0 EHCI, UHCI, OHCI host controllers are supported as well as the ISP1161, CY7C67300, CY7C67200 controllers.

- **HUB driver**
  Each module, EHCI, OHCI or other, has a driver for the HUB-class peripherals. The API that ***Philog*** provides, enables to know the exact topology of the bus and to run the hubs' ports one by one.

- **Transfer in Control, Bulk, Interrupt and t Isochroneous modes**
  Any type of transfer can be used to ensure a maximum rate with the peripherals.

- **Low-, Full-, and High-Speed Data Transfer**
  The 2 Mb/s on USB 1.1 and 480 Mb/s on USB 2.0 speeds are supported.

- **Embedded oriented**
  Fast and compact : approximately 30ko of RAM and 60ko of code per controller. The USB stack generates its own 1ms tick.
  Low real time resources consumptions: 1 interrupt vector, 3 semaphores and 1 task are required, the stack uses its own timers and queue mechanisms.

- **Plug-and-Play and Hot Plugging**
  Dynamic device drivers registration to the stack, plug-and-play and hot plugging management, easy supervision of the device tree.

- **Class Drivers**
  The ***Philog*** stack has a Host controller driver and a driver for the HUB-class peripherals. Class drivers are also available, either as finished products (serial CDC emulation, Still Image) or as incomplete implementations which do take in charge the USB management only, the upper interface being made up customer's material (Mass storage BOT, HID).

- **High level API**
  An interface library provides the developers of device driver with a simple way to interface the stack. Device drivers register dynamically and are notified whenever a matching device is connected or removed.

- **Portability**

  The USB stack requires a single task or thread to run in. All the tasks, scheduling, signaling etc. are based on internal mechanisms. It uses the hardware interrupt issued by the Host Controller to generate an 1ms tick. The SUB stack is fully written in ANSI C and supports the little- et big-endian CPU.