SDIO Stack for FreeBSD: Status update

Ilya Bakulin
`ilya@bakulin.de`

FreeBSD Developer Summit
Hilton Conference Centre
St. Julian's, Malta
September 26 – 28, 2013

What is it?

SDIO may refer to:

- Secure Digital Input Output, a type of Secure Digital card interface. It can be used as an interface for input or output devices.
- Strategic Defense Initiative Organization, an organization set up to oversee the Strategic Defense Initiative; now known as the Missile Defense Agency.

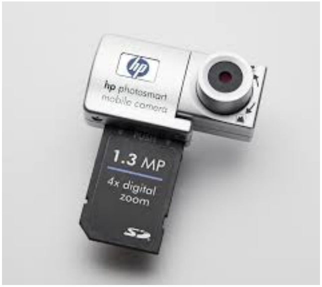
FreeBSD

Tell me moar!

- An SDIO card is an extension of the SD specification to cover I/O functions.
- GPS receivers, barcode readers, RFID readers, digital cameras, Wi-Fi, Bluetooth, ...
- It is now more common for I/O devices to connect using the USB interface.
- Pandaboard (some TI card), GlobalScale DreamPlug (Marvell SD8787 SDIO WLAN), Atheros AR6xxx WIFia.

An SDIO card may be directly connected to the SD interface, as on DreamPlug.

How does it look like?



FreeBSD

How does it look like?

SDIO vs SD

- Electrically compatible with SD
- Initialization process is a bit different
- SDIO card in non-SDIO aware host doesn't cause it to fail – it just doesn't respond

OS support

- Linux: supported, a lot of drivers for WiFi chipsets, sometimes even vendor-supplied
- OpenBSD: support limited, exactly one WiFi card is supported
- FreeBSD: not supported

SDIO4FreeBSD

- Being developed on GlobalScale DreamPlug
- Target device is the integrated SD8787 WiFi/BT adapter
- Using mv_sdio driver from Semihalf (not committed to the src/ tree)
- Using Linux mwifiex driver as a reference implementation of WIFI driver

When I have spare time, just for fun

What works: SDIO stack

- Initialization process for SDIO-only cards. No combo cards.
- Setting card speed, bus width; getting card functions
- Enable/disable SDIO functions, bus methods: write/read registers/FIFOs
- Attaching child drivers, including fixes for MMCSD driver

What works: SD8787 driver

- Firmware loading (and the firmware even starts after load)
- Some initial setup
- Sending commands to firmware (limited, need rework)

FreeBSD

What doesn't work

- Support for interrupts from the SDIO card!
- Child drivers cannot setup/teardown their own ISRs
- Controller drivers are not able to signal the interrupt from the card

FreeBSD

What's being worked on

- Support for interrupts from the SDIO card!!!
- Currently only for mv_sdio: lack of hardware (no sdhci_pci compatible h/w)
- Further work on SD8787 driver – code cleanup and refactoring
- I need to improve my Linux debugging knowledge to understand the Linux driver better (Marvell-supplied mwifiex)

Demo time

FreeBSD

Porting from Linux to FreeBSD

Both systems have similar concepts and primitives. APIs are sometimes different.

Linux	FreeBSD
create_workqueue()	taskqueue_create()
INIT_WORK()	TASK_INIT()
queue_work()	taskqueue_enqueue()
"waitqueue"	"channel"
wait_event_interruptible()	cv_wait_sig()
sk_buffs	mbufs

The plan

- Solve interrupts problem
- Integrate mv_sdiowl in the net80211 layer (tutorial promised by Adrian about writing network drivers will be useful here)
- Push SDIO-related changes upstream. Anyone willing to become a mentor? :-)
- Futher development of mv_sdiowl, ask Marvell to give the documentation

FreeBSD

Thank you for your attention!

ask questions

FreeBSD