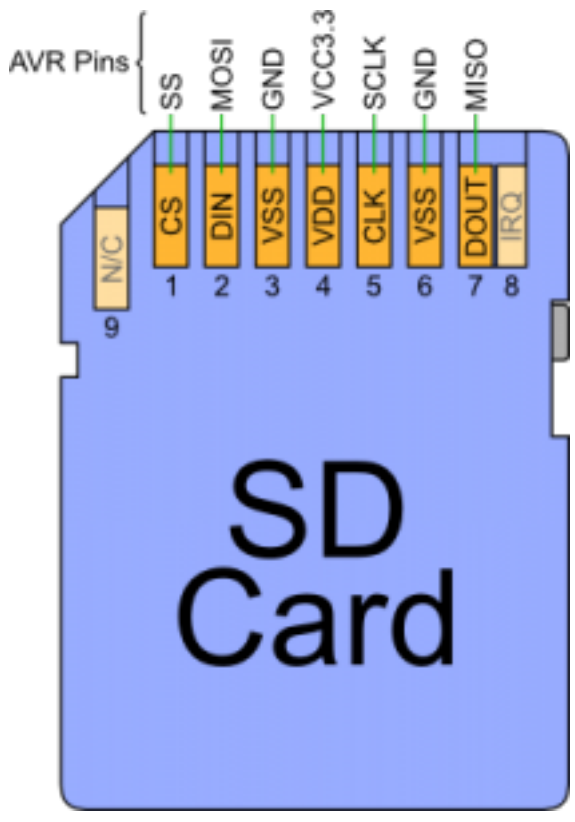
# Lab 23, hard disk or SD card

*By* ADMIN | *Published:* JANUARY 2, 2014

Before this Lab, the only way to bypass information to our Raspberry Pi machine, was the TFTP which downloads kernel from development machine (used Mac). Because kernel also includes root filesystem, we can embed some files there to have minimal read-only filesystem which allows to run dis files and do some testing, like experimenting with USB support, etc.

Sure, there were no way to do something and keep results on storage as there is no one. In this lab we try to create a partition to have possibility to save files.




Let’s start:
First I compared files in **port/** versus same in Plan9. Inferno’s versions were more old, so I used **sd.h** and **devsd.c** from Plan9. Inferno does not have **port/sdmmc.c**, so this one was also added. Examined with compilation in Inferno environment and applied few trivial changes (like *up->errstr* to *up->env->errstr*). That was minor.

Now codes specific to Raspberry: we borrow from 9pi. We need **dma.c** as it is time that we use the DMA for fast data transfer from SD. **emmc.c** written by Richard Miller for 9pi for bcm2835 mass media controller. Modify **rpi** spec file to include *sd* in *dev* section, *dma* and *sdmmc emmc* into misc section. Add **/dis/disk/*.dis**, **zeros**, **dd** to embedded filesystem.

Some changes are needed for our header files: in **mem.h** we need specify *#define BUSIO* to have controller communications. That should have addresses as seen from the videocore gpu, which was my fault as I thought same address as *PHYSIO*, but fixed later.

Add *Devport* and *DevConf* to **dat.h**:

```
01 /*
02  *  hardware info about a device
03  */
04 typedef struct {
05         ulong   port;
06         int     size;
```


*Categories*

- Blog
- Boost
- C++
- Cryptography
- Embedding
- Hybrids
- Inferno OS
- MacAppStore
- Misc
- Models
- Projects
- PyQt
- PySide
- Qt
- QtSpeech
- Raspberry Pi
- Research
- Ru
- TogMeg
- Trac
- TTS
- Tutorial
- Undo
- Usb
- Web

```
} Devport;


struct DevConf
{
        ulong   intnum;                 /* interrupt number */
        char    *type;                  /* card type, malloced */
        int     nports;                 /* Number of ports */
        Devport *ports;                 /* The ports themselves */
};
```

Add SD card dev file server bind to **rpiinit.b** to see it on device:

```
sys->bind("#S", "/dev", sys->MAFTER);   # sdcard subsystem
```

Try to compile, all looks okay. Attempt to boot and see message:

```
eMMC external clock 50 MHz
```

Have a look at **/dev**, there is **/dev/sdMo/**

```
/dev/sdM0/ctl
/dev/sdM0/data
/dev/sdM0/raw
```

To see partition table we use **disk/fdisk** from Inferno. Amazing, but it work ok:

```
$ disk/fdisk /dev/sdM0/data
>>> p
cylinder = 1936384 bytes
   p1                    0 66          (66 cylinders, 121.88 MB) FAT32LBA
   p2                   66 595         (529 cylinders, 976.89 MB) LINUX
   p3                  595 859         (264 cylinders, 487.52 MB) LINUXSWAP
   empty               859 2047        (1188 cylinders, 2.14 GB)
 >>>
```

Looks like I have SD card with some linux onboard. If we would like to see partitions as files in **/dev/sdMo/**, we just do:

```
$ disk/fdisk -p /dev/sdM0/data > /dev/sdM0/ctl
$ ls /dev/sdM0
/dev/sdM0/ctl
/dev/sdM0/data
/dev/sdM0/dos
/dev/sdM0/linux
/dev/sdM0/raw
```

Now about filesystem that we can use as root of our system. This is **kfs** which is used in Plan9 and Inferno. It has some limitations as 27 bytes per filename. First I come back to my Mac, inserted SD card and by using Mac fdisk changed type Linux partition to 39 (Plan9). Then as I see size of partition in bytes I created with *dd if=/dev/zero* same length file. Started **emu** (mac hosted inferno) and tried to initialize and mount the file:

```
; mount -c {disk/kfs -r -b 8192 kfs.file} /n/local
```

using 8KB as block size.

Then I just copied whole Inferno folder content to SD partition:

```
; cp -r * /n/local/
cp: cannot create /n/local//os/rpi/labs/lab-03-rpi-booting-process.pdf: name
too long
...
```

Yep, some filenames are too long, but it is okay for now.
Unmount, put SD back into raspberry, boot, again initialize **ctl** with fdisk and try to mount:

```
$ mount -c {disk/kfs /dev/sdM0/plan9} /n/local
mount: can't dial net!{disk/kfs!styx: %q% /net.alt (/net.alt/net/clone)
```

Strange, feel dumb, but we used **tinysh.dis** as our shell and it can not do {}, I guess copied from ipaq spec file with assumption that it needs less dependencies. So fix rpi spec file to use real **sh.dis** as our shell and:

```
; mount -c {disk/kfs -A -n main /dev/sdM0/plan9} /n/local
kfs needs check
kfs: initializing minimal user table
; ls /n/local
/n/local/.DS_Store
/n/local/.hg
/n/local/.hgignore
/n/local/.hgtags
/n/local/CHANGES
/n/local/DragonFly
/n/local/FreeBSD
/n/local/INSTALL
...
```

Wow, we have filesystem mounted!
Later we may need specific procedure to actually create root filesystem with only needed stuff.

This entry was posted in *Blog*, *Inferno OS*, *Raspberry Pi*, *Research*. Bookmark the *permalink*. *Post a comment* or leave a trackback: *Trackback URL*.

« *Lab 22, Usb keyboard*