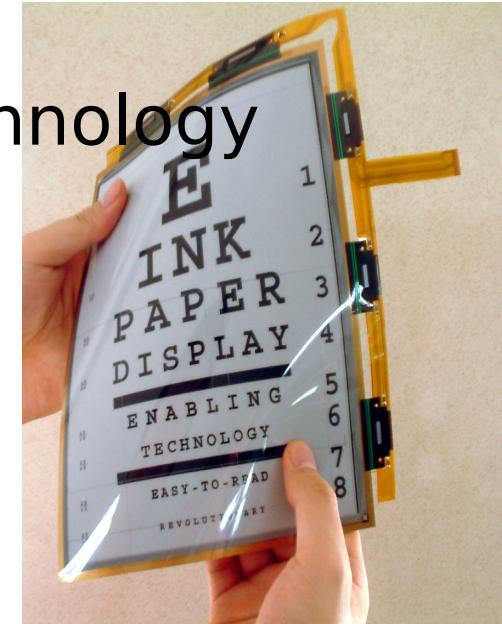




E-Paper Displays

- 5 minutes of Physics & Chemistry :-)
- Linux, Fbdev, & Deferred IO
- Demo
- Future work, controllers, technology



Quick Disclaimer:

- My crystal ball has ECC problems. Some of my claims and predictions about the controllers and technology direction may be wrong.
- I have hearing problems. Please speak loudly and slowly.
- I'm not here speaking on behalf of any E-paper company.
- Just an excited embedded Linux hacker :-)



Why am I excited?

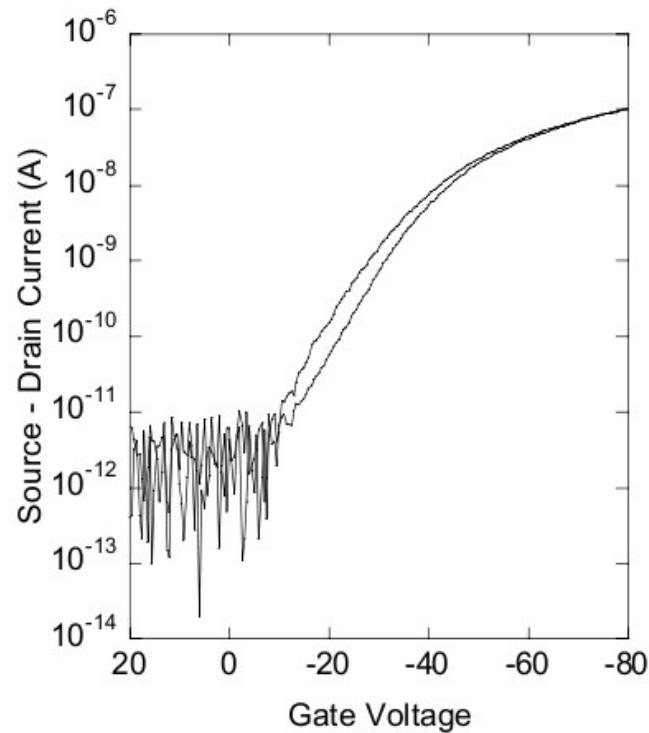


Figure 6. TFT characteristics for a TFT printed on a commodity plastic substrate.

Why am I excited?

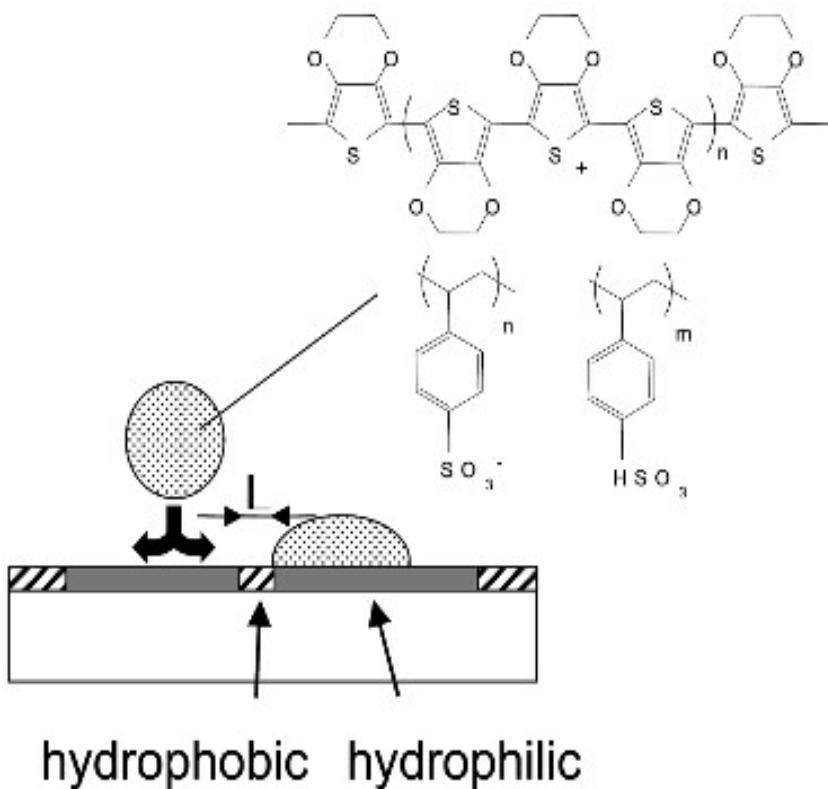


FIGURE 1 — Schematic diagram of high-resolution surface-energy-assisted ink-jet printing.

can be aligned locally with respect to a previously deposited pattern such that high registration accuracy can be main-

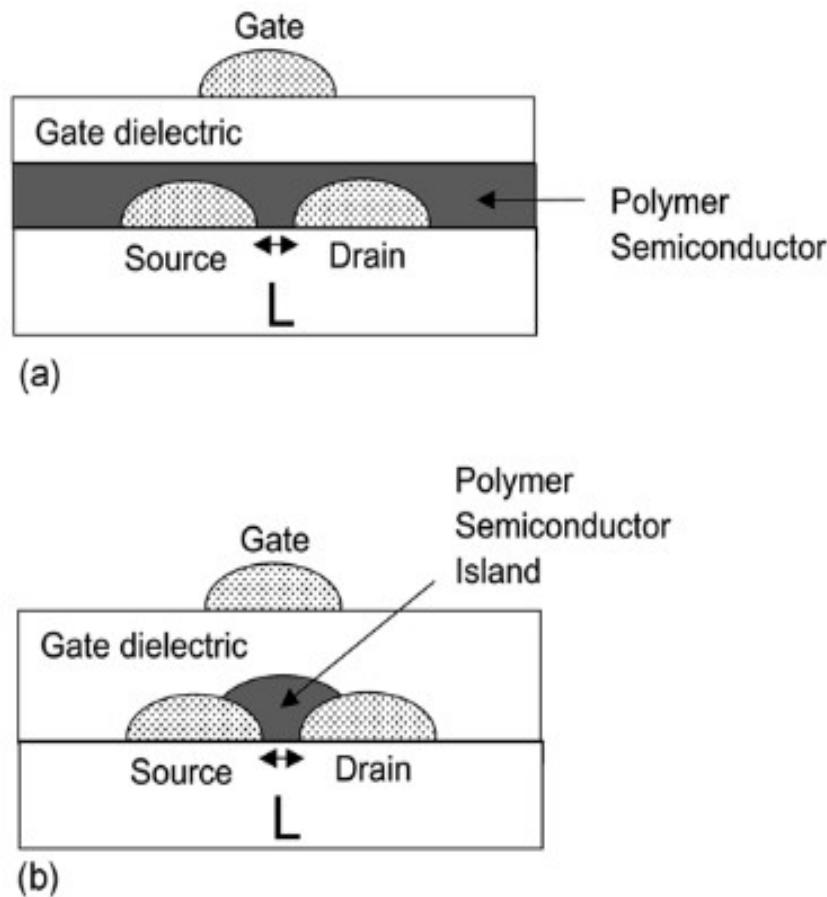
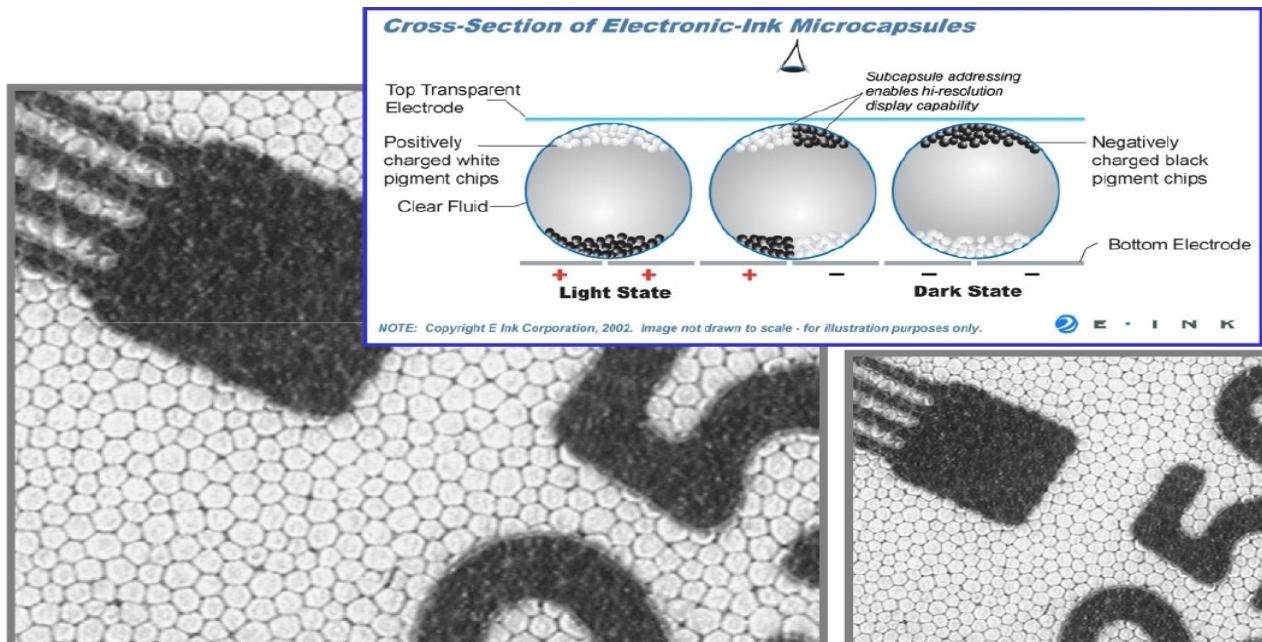


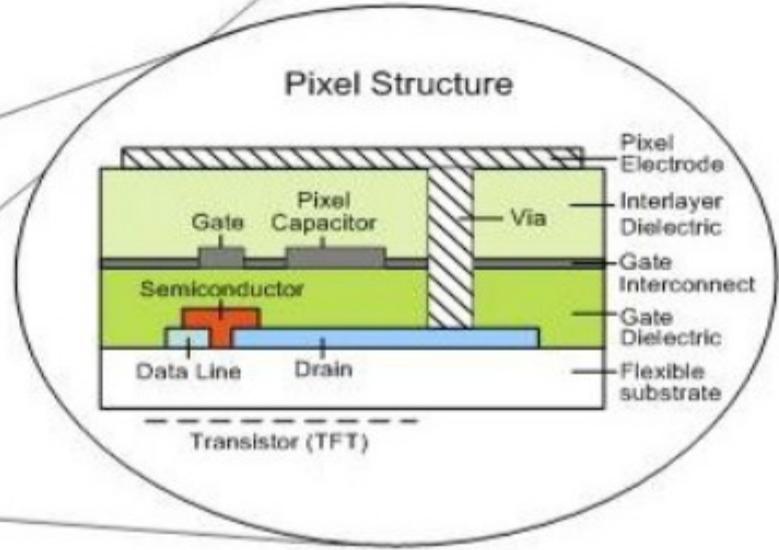
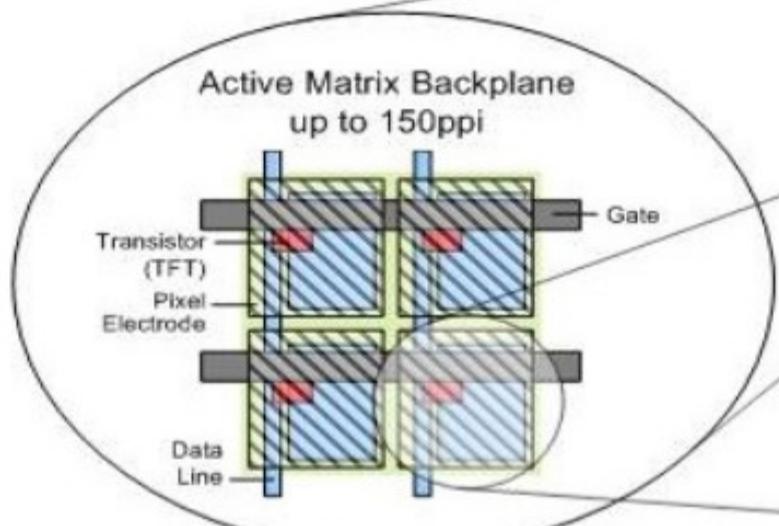
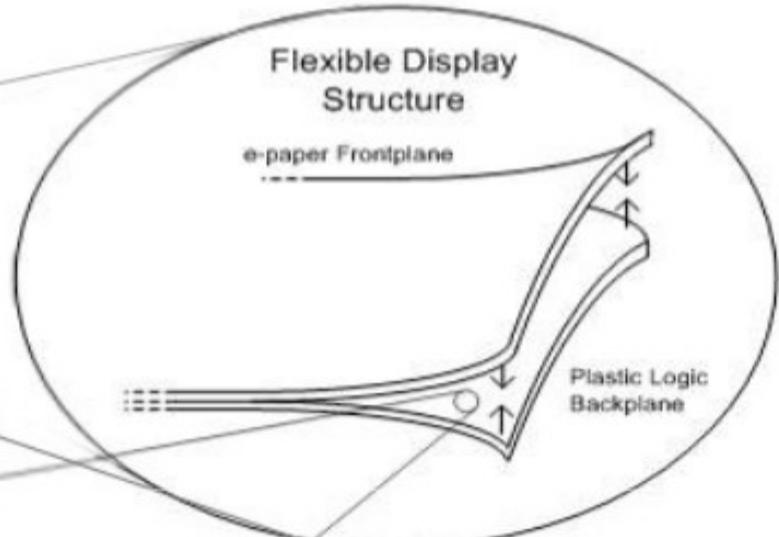
FIGURE 2 — Schematic showing the cross section of a printed transistor. (a) Structure with a continuous film of semiconductor. (b) Structure where the semiconductor is patterned into islands covering the TFT channel only.

Quick Overview

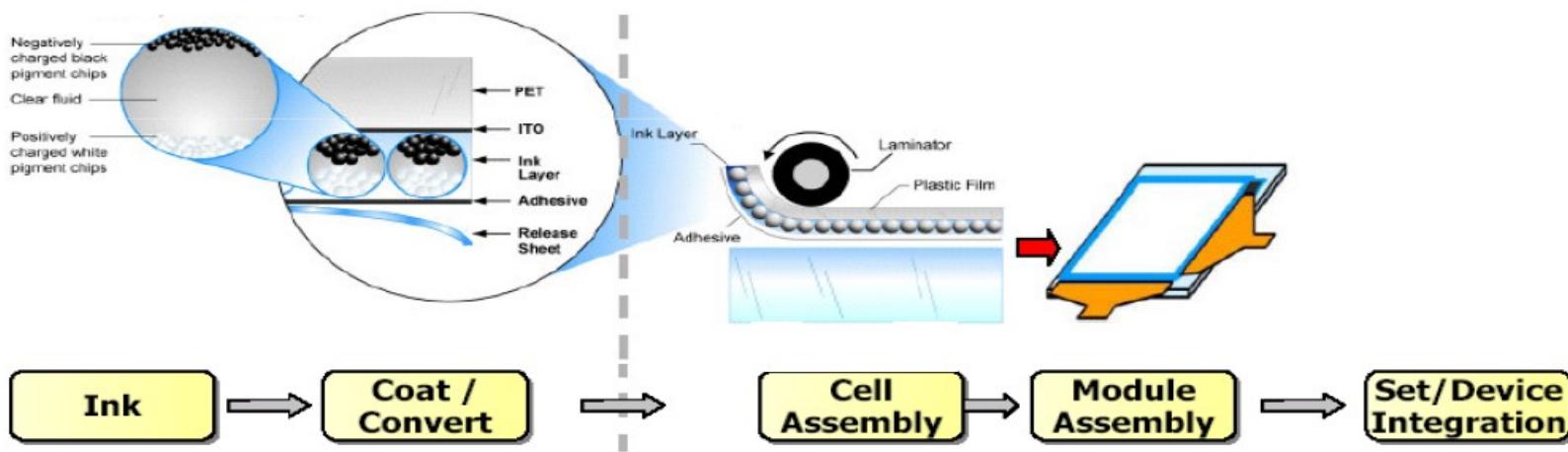
Microencapsulated Electrophoretic Display (EPD)



Why am I excited?



Quick Overview



 E • I N K

 Display modules supplied by:
PRIME VIEW INTERNATIONAL
(and Other Display Manufacturers)

SONY

Why am I excited?



Pretty Pictures. That's why I'm excited.



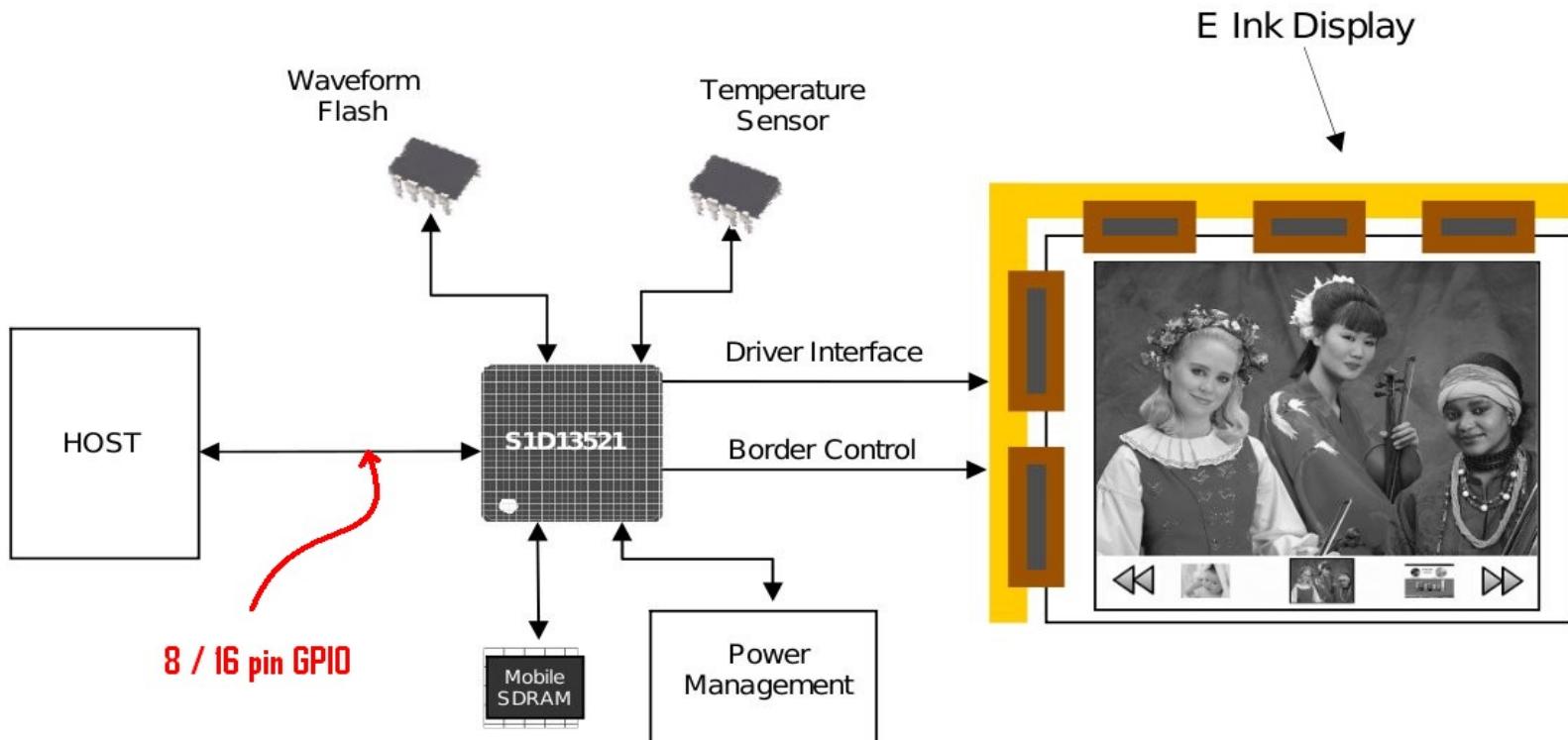
I hope you are excited too.



Software Issues. E-paper's Challenges

- Display latency. (Viscosity)
- Display controllers are unusual. Not PCI, AGP, PCI-E! Not memory mappable.
- GPIO, AMLCD.
- Need specialized waveforms to drive the material.
- Need memory to store the waveforms.
- Temperature sensitive waveforms.

E-Ink Apollo (Hecuba) and Epson/E-Ink Broadsheet EPD Controller



Linux Challenges

How do we memory map a "non memory mappable" IO interface like GPIO?

How do we mitigate the latency associated with display updates?

A possible solution: Deferred IO

What exactly does defio do?

How does it work?

Deferred IO

What?

Framebuffer pages in host
memory

Page entries read-only

App writes to anywhere, say
a particular page

Use page fault to start
delayed workqueue

Leave page as writable. Add
page to pagelist.

App keeps writing to page.
Free of charge.

Workqueue kicks off.

Perform IO to display

Mark page as read-only

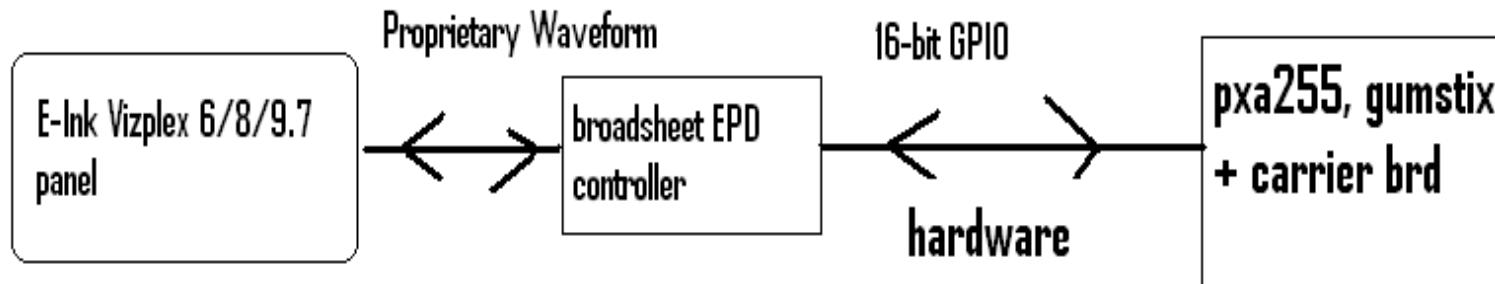
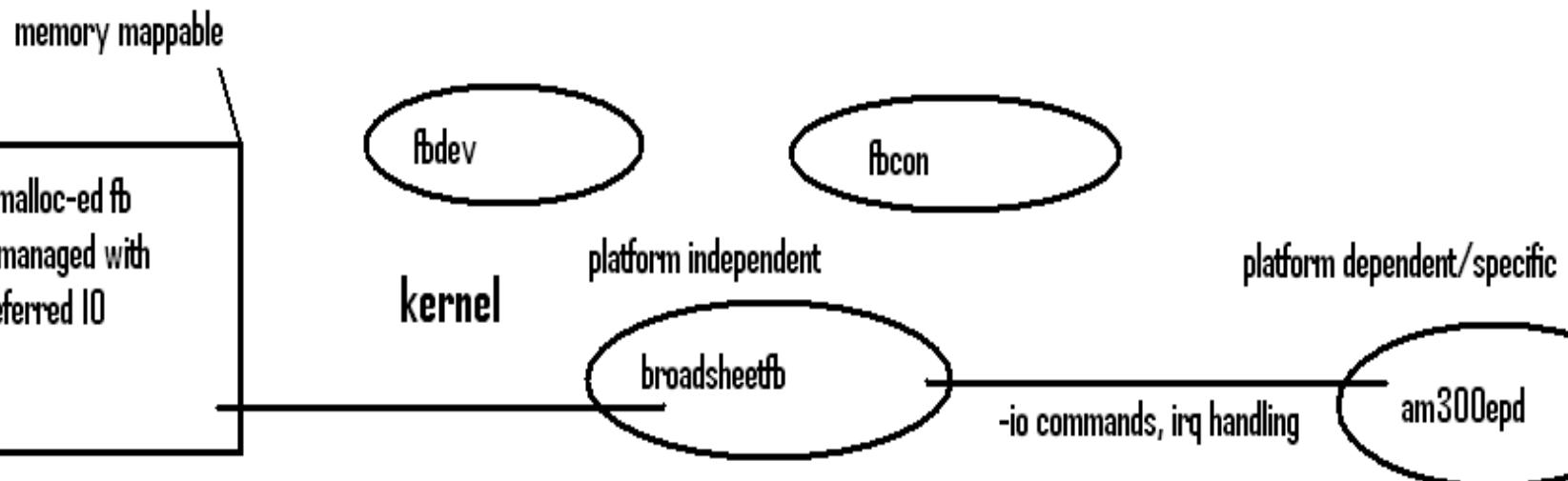
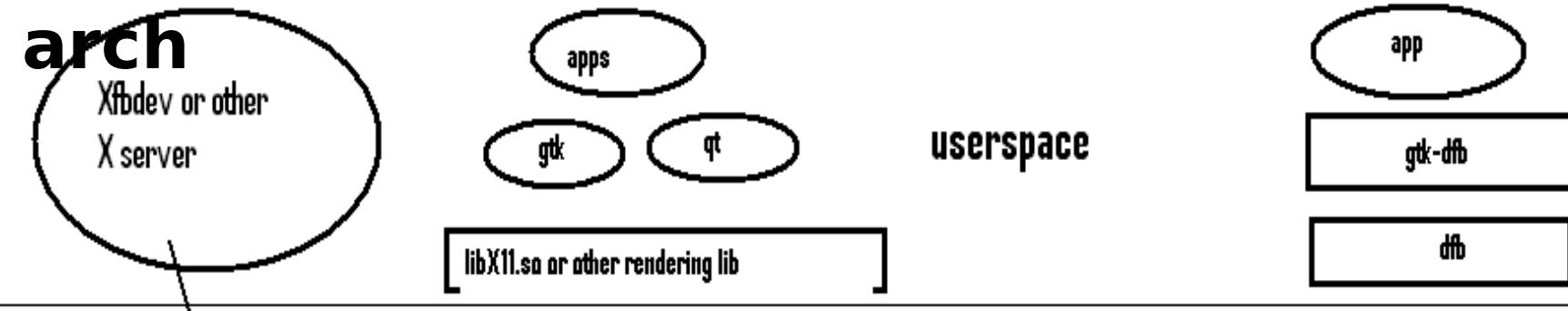
Lather, Rinse and Repeat

What is nice about deferred IO?

- Solves latency
- App is unimpeded by display latency.
- Solves IO interface
- Use any IO you want. AMLCD. USB. GPIO. Works with mmap.
- Solves userspace write detection. No polling. No timers.
- You know exactly which pages have been written to.

Broadsheet

sw arch



Deferred IO: How is it used?

Setup your struct:

```
static struct fb_deferred_io myfb_defio =  
{  
    .delay          = HZ, // 1 second  
    .deferred_io    = myfb_defio_handler,  
};
```

Deferred IO: How to use?

Setup your defio handler:

```
static void myfb_defio_handler(struct  
fb_info *info, struct list_head  
*pagelist)  
{  
    list_for_each_entry(cur, pagelist...) {  
        do_io_for_fb_page(cur);  
    }  
}
```

Deferred IO

Where? When?

In mainline Linux today

Added in 2.6.22

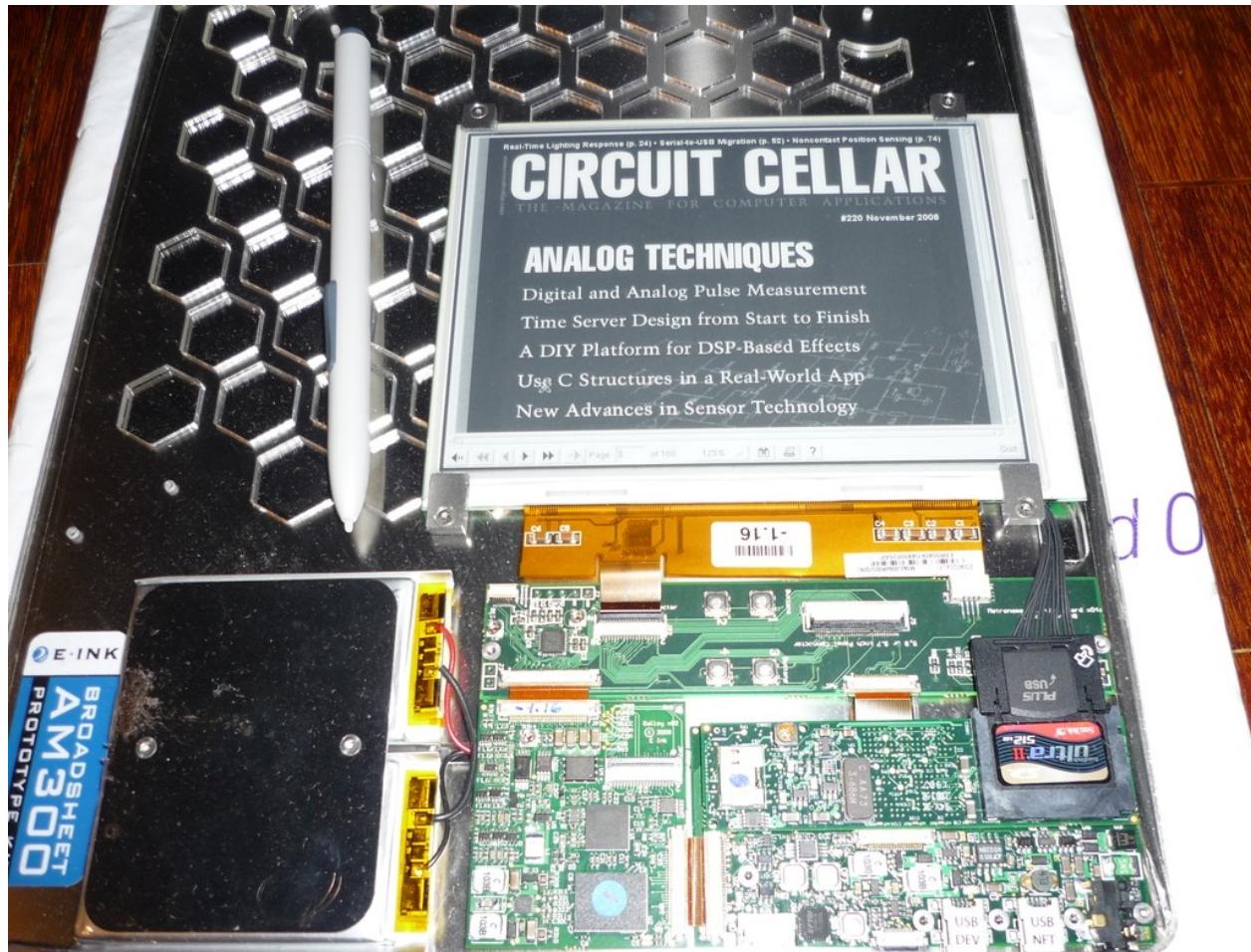
Usable since 2.6.25

Who's using it?

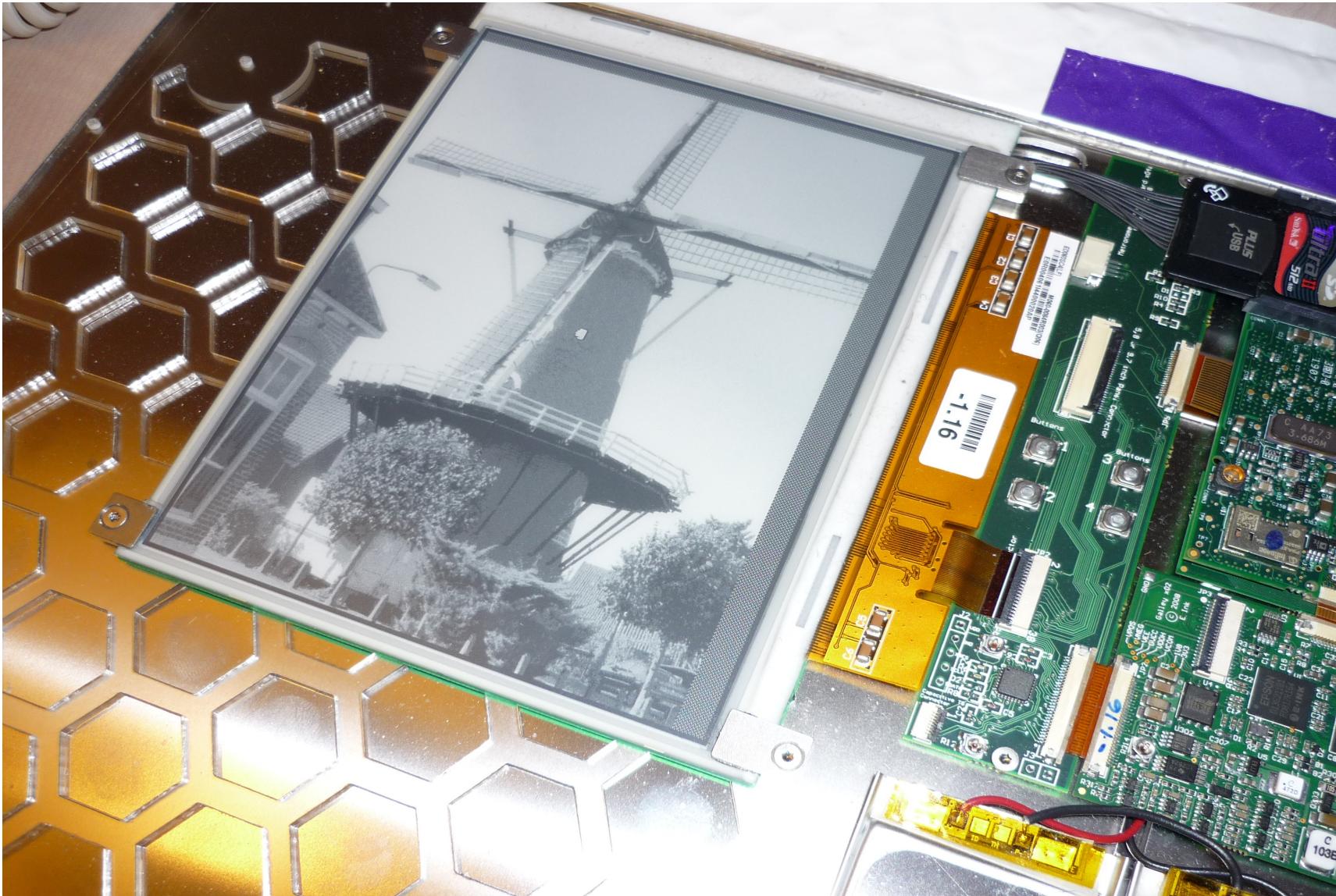
hecubafb, metronomefb, broadsheetfb,
xen_pvfb

other out of tree drivers for matrix and usb
framebuffer devices.

Lets watch some video clips



xloadimage

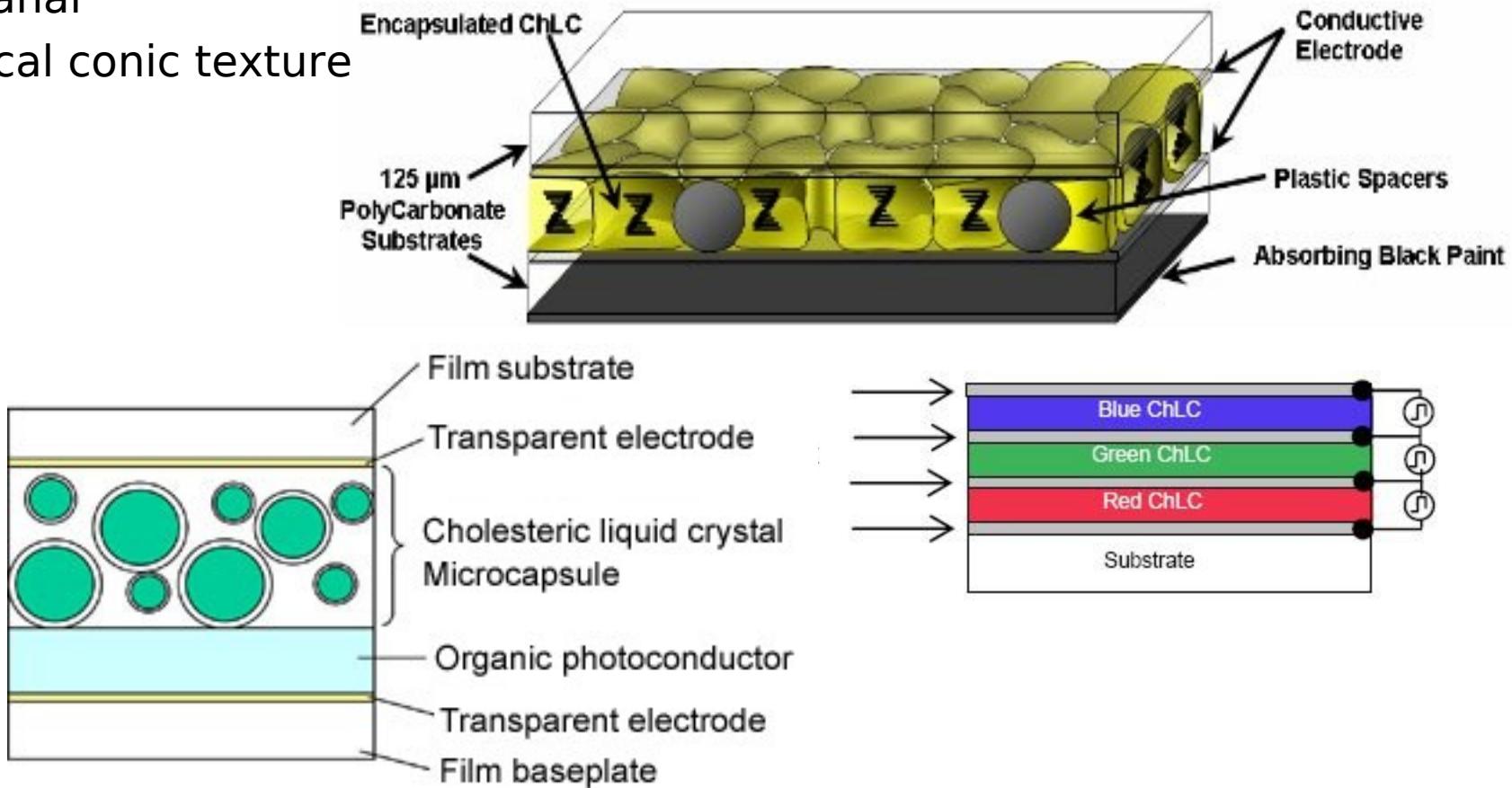


Other Technologies

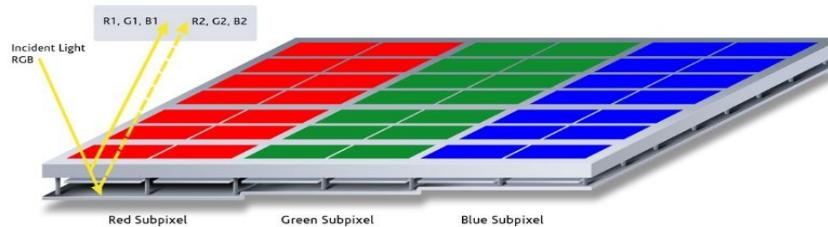
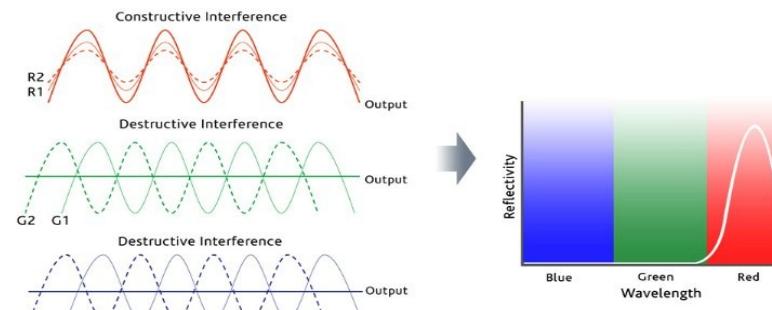


Cholesteric LCD

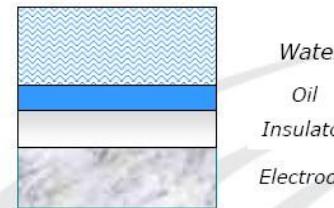
- planar
- focal conic texture



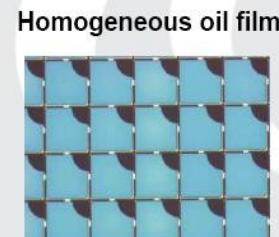
Others:
QR LPD
Bi-Stable LCD
IMOD
Electrowetting



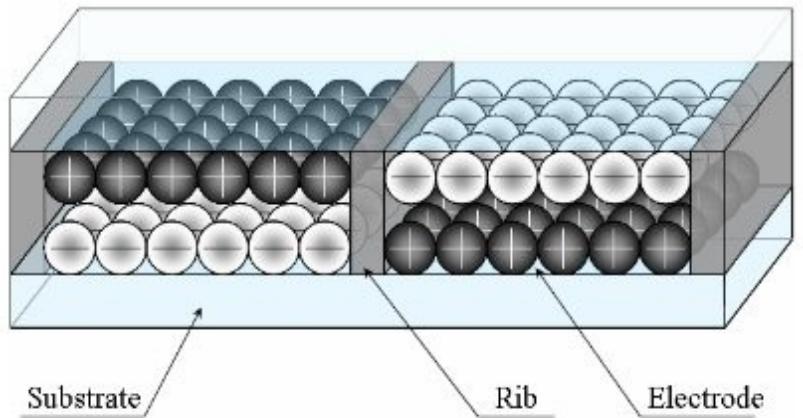
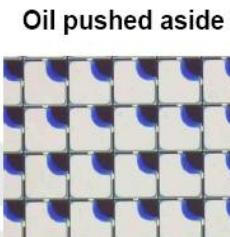
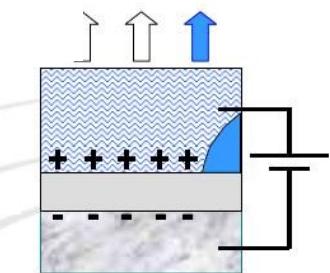
(b)



(c)



(d)



Future Work

- Still a lot to do
- Lots of optimizations to be made
- Partial Update & Coalescing
- Sub-page detection
- Buffering
- Cleaning up update sequences

Future Work

Controller supports changing display source address. Ie:
can change equivalent of smem_base dynamically.

Expose via ypan? Like double buffering? Or overlay?

Controller supports different “usage mode” impulse
waveform modes. Eg: fast but buildup, slow and flash but
clean, automode? App hints? Tracking history?

Multi buffering? Prerendered pages for e-book readers

Graphical Framework awareness of refresh rate. Qt. Gtk.
etc

Future Work: Controller Quirks

Controller supports queued updates. Ie: can queue (fixed queue size) a series of updates that don't need to wait for completion (ie: don't need to use sync). How do we expose this?

Or do we go the traditional way? Ie: assume app doesn't want sync() by default and so all updates use the async queue by default unless app imposes otherwise? How do we manage the queue size?

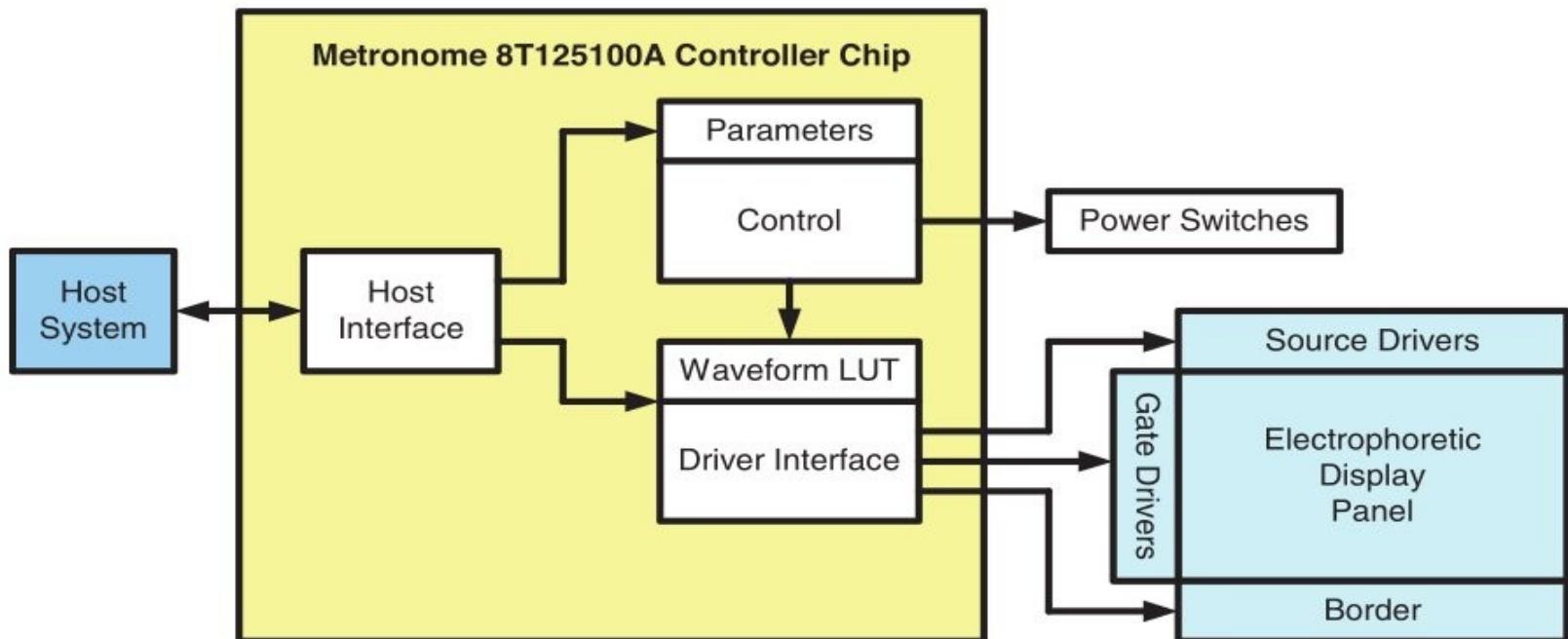
Thanks!

- Thanks to fbdev people: Tony, Geert, Kryzstof, James
- Thanks to all code reviewers
- Special thanks to CELF, NLUUG, Ruud, Armijn and organizers!

Questions/Answers
or
Additional Slides if we have
time.

I welcome your feedback to jayakumar.lkml@gmail.com

E-Ink Metronome EPD Controller



Deferred IO: How to use?

Setup init and cleanup:

```
_init() {
    info->fbdefio = &myfb_defio;
    fb_deferred_io_init(info);

_exit() {
    fb_deferred_io_cleanup(info);
```

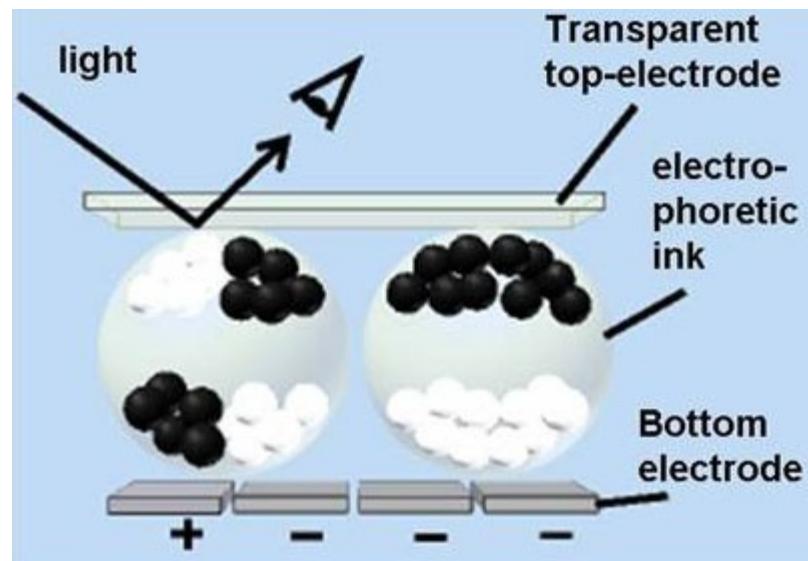
Electronic Paper

Quick Overview

Electrophoretic Displays

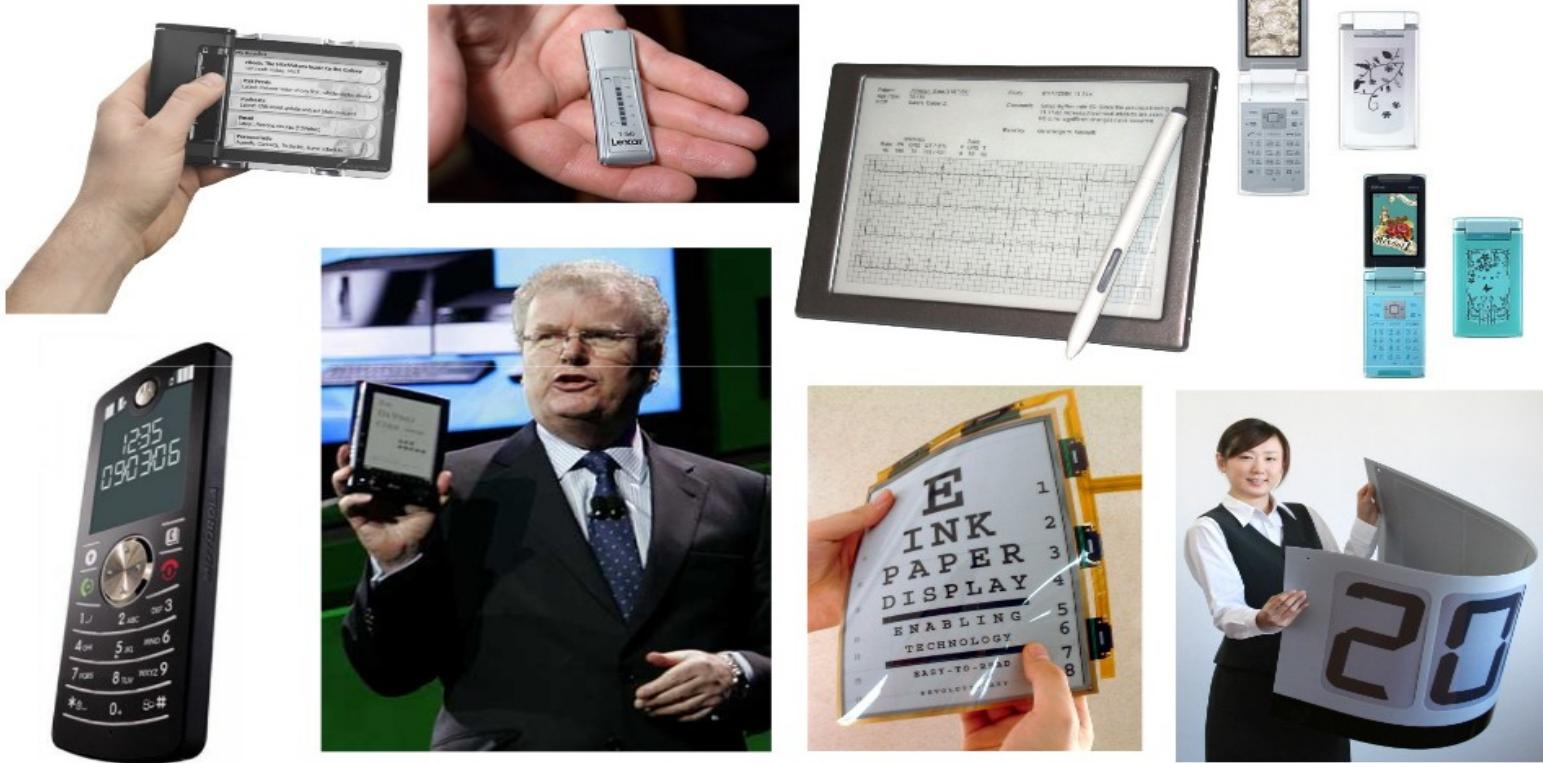
Reflective

Bi-stable



I like pretty pictures:

E Ink: The Most Convenient Way to Read Digital Information



Sources & References:

Figures:

E-Ink
Kent Displays
Qualcomm
Bridgestone
Plastic Logic
Polymer Vision
Liquavista
PVI
Sipix
Sony
Amazon
Epson
Nemoptic

Data:

EE Times