Trading Fbdev for DRM, no returns accepted

Embedded Linux Conference Europe 2022

Geert Uytterhoeven
geert@linux-m68k.org

http://glider.be/

September 16, 2022

Glider.be

Table of Contents

About Me

- **Freelance Embedded Linux Kernel Hacker @ Glider BV**
- Started with Linux as a hobbyist (1994)
- Amiga, m68k, PPC, MIPS, PS3/Cell, Renesas, ARM, Super-H, RISC-V, ...
- Networking, graphics, IDE, audio, RTC, clock, pin control, ...
- Maintainer of the **Linux Frame Buffer Device subsystem** (1997–2004)
- Member of the **XFree86 Development Team** (1996–2001)
- **Graphics subsystem** for Digital TV products @ Sony (2001–2003)
- **I am no DRM expert!**

Icons created by Freepik and Good Ware — Flaticon

Deprecation of Linux Frame Buffer Device Drivers

- Fbdev was deprecated in 2015
- No new Fbdev drivers accepted
- All new graphics drivers must use DRM/KMS

- Existing Fbdev drivers?
- Out-of-tree drivers?
- What is the fuzz about?

Linux Genesis

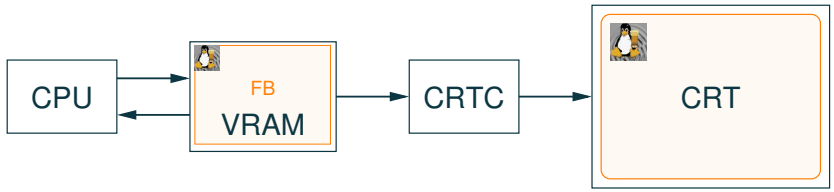
> *I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).*

- EGA/VGA text mode for console in kernelspace
- X server (e.g. XFree86) for graphics in userspace
  - Mode setting
  - Hardware acceleration (optional)

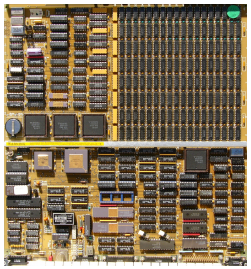
Linux Expansion

- Non-x86 machines may not have VGA
  Amiga, Atari, Mac, SPARC, PowerMac, ...
- Proliferation of platform-specific console implementations
- **Fbdev**: first platform-independent framework for graphical consoles on Linux
- Atari (Martin Schaller, 1995)
- Amiga (1995)
- Abstract console driver (vgacon/mdacon/fbcon/...) (1997)
- ATI Mach64 on PPC (1997)
- VESA on x86 (Gerd Hoffmann, 1998)

Simple Graphics Hardware



Sun-3/50, 1152 × 900 in 128 KiB, 4 MiB RAM

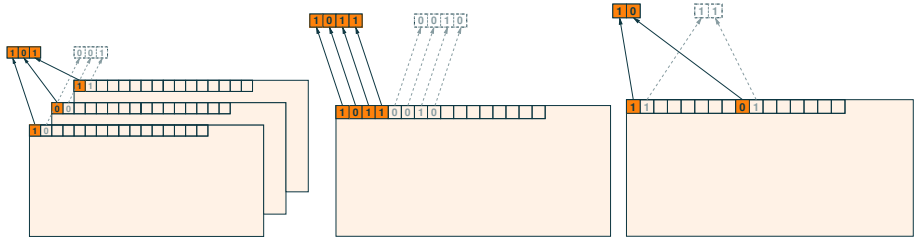


Images © Olaf Püschel and Jef Poskanzer

(More) Color

Add more bits per pixel!

Black/White → Greyscale → CLUT → RGB



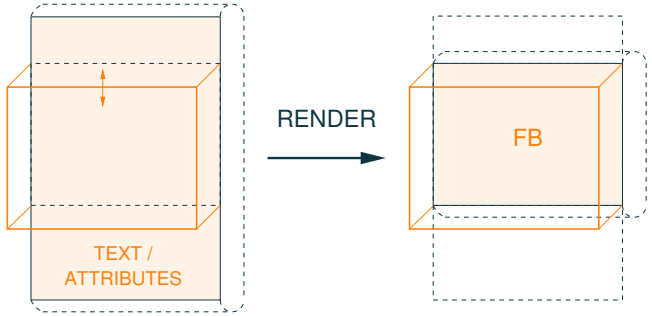
Fbdev API

- Direct access to frame buffer through `mmap()` `/dev/fbX`
  - ✓ Flexible: lots of formats (monochrome, clut, (A)RGB, bitplanes, ...
  - X Each application needs to support all formats (ideally)
- Mode querying/setting, through ioctl()
- Colormap manipulation, through ioctl()

- Acceleration (optional):
  - Kernelspace, text console only
  - Userspace, through `mmap()` `/dev/fbX`
    - Hardware-specific, e.g. some support in XFree86

- Deferred I/O for non-mappable frame buffers

Frame Buffer Console (fbcon)



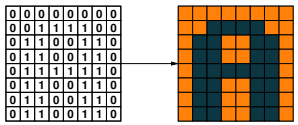
Fast Graphical Text Consoles

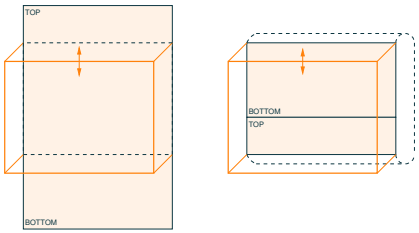
Optional Hardware Acceleration

- Monochrome-to-color expansion
- Rectangle fill/copy
- Pan or Wrap

Scrolling Strategies

- Pan or Wrap
- Copy
- (Smart) Redraw





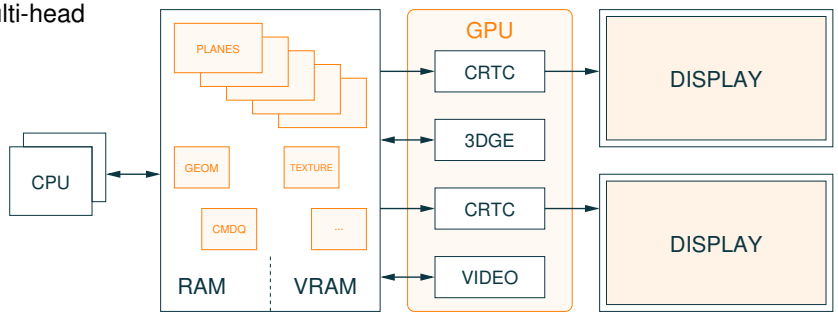
Graphics Stack

	Traditional	Fbdev
Userspace	▸ Mode setting ▸ FB Drawing ▸ HW Accel	▸ FB Drawing ▸ HW Accel
Kernelspace	`/dev/mem`	`/dev/fbX` ▸ Mode setting
Hardware	—	—

Demise of Fbdev

Graphics hardware gained more features and increased complexity:

- Z-Buffer
- 3D acceleration
- Overlays (YCbCr)
- Multiple planes
- Multi-head
- ...



Direct Rendering Infrastructure (DRI/DRM)

3D acceleration for high-end true-color desktop graphics
Precision Insight, Linux Expo 1999)

- DRM: acceleration
  - Userspace driver prepares hardware-specific drawing commands
  - Kernel driver provides mechanism to send them to hardware
- Supported pixel formats:
  - C8 (256 colors)
  - (A)RGB
  - YCbCr (video)
  - ...
- Multiple planes (overlays)
- Memory management
- Complex!
- Zillions of helpers, but which one to use?

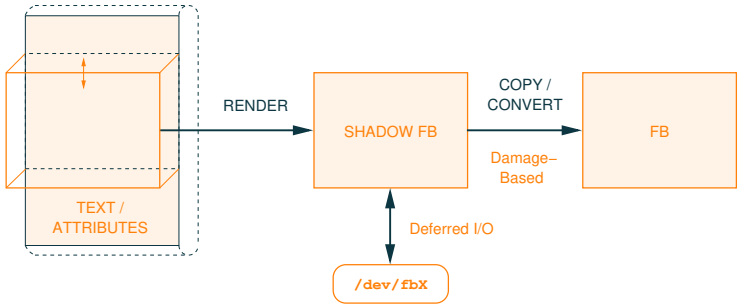
Kernel Mode Setting (DRM/KMS)

Isn't that what Fbdev does? (Non-)Atomic mode setting

	Traditional	Fbdev	DRM+KMS
Userspace	▸ Mode setting ▸ FB Drawing ▸ HW Accel	▸ FB Drawing ▸ HW Accel	▸ Buffer Drawing ▸ HW Accel frontend
Kernelspace	`/dev/mem`	`/dev/fbX` ▸ Mode setting	`/dev/dri/` ▸ Mode setting ▸ Buffer Management ▸ HW Accel backend
Hardware	**Hardware** —	**Hardware** —	**Hardware** —

Fbdev Emulation

- Exposes a frame buffer device (`/dev/fbX`)
  - Drawing only
  - No mode setting
- Mainly used for text console



Deprecation of Fbdev

2012: A Call For Deprecating The Linux Frame-Buffer FBDEV (Laurent Pinchart)
`https://www.phoronix.com/news/MTE3MjI`

2015: No more new fbdev drivers, please (Tomi Valkeinen)
`https://lore.kernel.org/all/5603EC15.9090605@ti.com`

Why?

- Not suitable for contemporary graphics hardware
- Endless stream of security issues detected by fuzzing
- Essentially unmaintained

Maintainers:

- James Simmons (2002–2004)
- Antonino Daplas (2004–2009)
- Paul Mundt (2011)
- Florian Tobias Schandinat (2011–2013)
- Jean-Christophe Plagniol-Villard (2013–2016)
- Tomi Valkeinen (2013–2016)
- Bartlomiej Zolnierkiewicz (2017–2020)
- dri-devel (2017–)
- Helge Deller (2022–)

Thanks a lot!

Erosion of Fbdev Features

- Fbdev (used to be) de-facto maintained by DRM maintainers
  - Mostly because of fbcon
  - Conflict of interest?
- Focus on scrolling by redraw
- Removal of scrollback (v5.9)
- Removal of *unused* hardware acceleration (v5.11) (reverted in v5.17)

DRM is a bad Fbdev driver?

DRM's Fbdev emulation violates Fbdev rules
(rounding rules, validating parameters) → being fixed

It's so easy to fall in love. . . with DRM

Since 2015: No new Fbdev drivers accepted

- Discussions about helpers and simpledrm

  It's very easy to write a simple drm driver

  cfr. the `tiny` DRM drivers (+50% larger than simple Fbdev)
  cfr. simpledrm (2015)

- *Show me the code?*
- Simpledrm first mentioned and posted in 2013?
- Simpledrm merged in 2021 (v5.14), for driving firmware frame buffers (EFI, OF, . . .)
- Existing drivers? +100!
- Out-of-tree drivers in vendor BSPs?

Converting Fbdev Drivers to DRM Drivers

Why not?

- No time
- No hardware
- No understanding of DRM (= no time?)
- Zillions of helpers to choose from
- Missing features in DRM
- Fbconv (Thomas Zimmermann, 2019) is not yet upstream, except for some helpers
- Zillions of helpers to choose from

Icon created by Freepik — Flaticon

Case Study: Atari DRM

Converting Atafb into Ataridrm

Why Ataridrm?

- Developing on ARAnyM emulator
  - Faster turn-around time than real hardware (which I don't have)
- Variety of supported modes:
  - Monochrome (actually 2 colors)
  - Color-mapped (2, 4, 16, 256 colors), using interleaved bitplanes
  - RGB (big-endian RGB565)
- Started for real in 2020, unsuccesful
- Regained interest in 2022
- Still not ready for submission

Low-color Frame Buffer Formats

DRM supported only color-indexed frame buffer formats with 256 colors (C8)

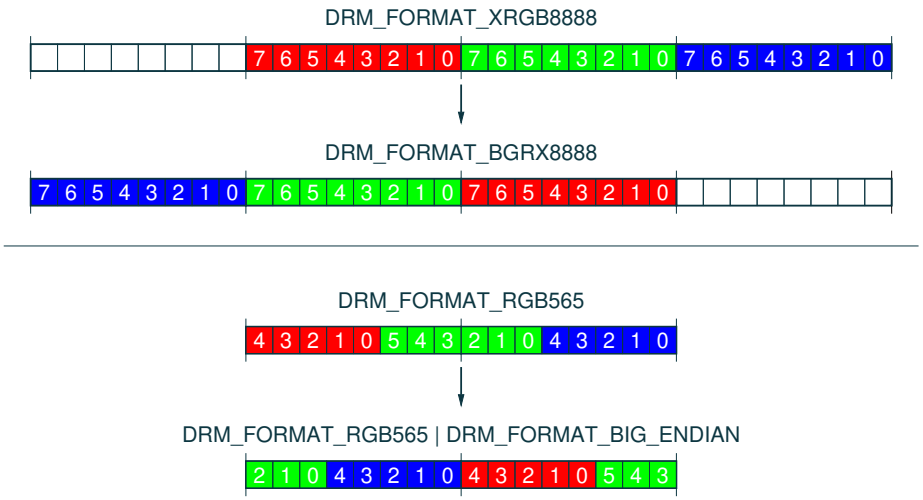
- Added new C1/C2/C4 formats (2, 4, and 16 colors)
- Added support for C1/C2/C4 to DRM and `modetest` (SMPTE test pattern)
- Fixed assumption in DRM that pixels are at least one byte
- Added new formats R1/R2/R4/R8 and D1/D2/D4/D8 (direct resp. inverse relationship between channel value and brightness)
- CLUT size still fixed to 256 (gamma table)
- **Your userspace** still needs to gain support for the new low-color formats, too!
- Linux support queued in drm-misc for ~~v5.20~~ ~~v6.0~~ v6.1
- Libdrm support TBD

Endianness

DRM formats are defined to be little-endian, unless bit 31 is set in the fourcc code.

- Caveat: old drivers (on PowerPC?) may use native endianness
- New drivers running on big-endian must set the `quirk_addfb_prefer_host_byte_order` flag
- XRGB8888 and ARGB8888 are translated
- XRGB1555 and RGB565 get bit 31 set

Endianness



Endianness

- XRGB8888 and ARGB8888 are translated
- XRGB1555 and RGB565 get bit 31 set
- Big-endian XRGB1555 and RGB565 were not recognized
- Most `drm_fb_*_to_*()` conversion helpers lacked endiannes handling

- Fixed `modetest` on big-endian, too
- Toolkits like Cairo always uses native byte order for rendering, so you have to do byte swapping after rendering :-(
- Other software?? X? Wayland?

- Conversion helper fixes queued in drm-misc for v6.1
- Libdrm support TBD

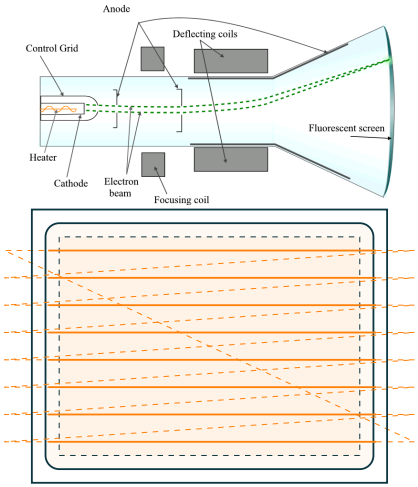
Video Mode Selection

- Analog video modes can be very device-specific
- Added support for driver-specific named modes, incl. dashes (e.g. `tt-mid`)

- Mixed reception
- Slightly-related Analog TV Improvements by Maxime Ripard

Analog Displays

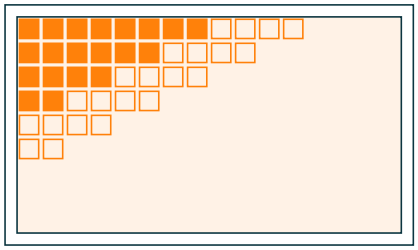
- Fixed number of lines by display standard (or multiscan)
- Variable horizontal resolution depends on bandwidth/pixel clock
- Overscan
  Don't use for text, do use for video
- Knobs to control image size/position
- May predate DDC
- ⇒ Infinite number of modes



CRT image © Theresa Knott

Digital Displays

- Fixed number of pixels
  Fuzzy if scaling
- Back-channel to advertise capabilities
- Desktop: interface is digital variant of old component/VGA analog
- Embedded: interface is packet based, changes only
- Preferred (standard) modes



Modus Francus

All drivers expose at least **XR24** aka **XRGB8888** (RGB, 8-bit per color component)

- ✓ Works with everything
- ✓ Suitable for desktop graphics
- X May be overkill for lesser systems
- X Overhead of copying and conversion
  - Not every device has +10 GiB/s bandwidth
- X Application is not aware of the real display properties
  - E.g. GUI assumes continuous tone RGB, which becomes useless on monochrome, or very ugly on a low-color display

Atari DRM: Status

What works?

- All hardware pixel formats (2, 4, 8, 16 bpp)
- Text console using any of the above
- Conversion from XR24 to C8 (RGB332) or big-endian RGB565
- Conversion from RGB565 to big-endian RGB565
- Video mode programming (boot + modetest)
- Fbtest

What can be improved?

- Video mode programming cleanup
- Allocate native RGB565 buffers from ST-RAM
- Tested on ARAnyM only
- Benchmarks on real hardware

Performance

Memory Usage

- Kernel size: 295 KiB more
  DRM includes everything and the kitchen sink (HDMI, I2C, IRQ_DOMAIN, . . .)
- Memory available (kernel): 392 KiB less
- Memory available (user): 648 KiB less

Text Console Performance

- Preliminary (ARAnyM): almost 10× slower scrolling
- Can this be improved? E.g. by operating directly on the native frame buffer?

Is all of this still relevant?

- Lots of handheld devices have advanced 3D graphics hardware (Hi, smartphone!)
- Legacy hardware is. . . obsolete
- Low-end embedded platforms?
  - Small or low-color displays
  - Limited amount of RAM
  - Limited CPU processing power
  - Limited power budget (battery, small solar panel)

Example: 1 Mpixel e-Ink Display

1. Monochrome buffer $= 1\times$ Sun-3/50 VRAM

2. XR24 shadow buffer $= 1\times$ Sun-3/50 RAM
3. Greyscale buffer $= 1/4\times$ Sun-3/50 RAM

Contemporary e-Reader $\approx 256\times$ Sun-3/50 RAM $\longrightarrow$

Images © Olaf Püschel and John Weaver

Example: 7-color e-Paper Display

- 4 bits per pixel
- Model as C4 with fixed palette?
  - Fbdev has static pseudocolor
  - Not yet supported by DRM

Image © Waveshare Electronics

Conclusion

- Fbdev drivers can be converted to DRM drivers
- Most missing functionality has been identifed and fixed

What do we gain?

- Common userspace
  - DRM driver converts from standard to exotic pixel formats
  - Amiga Hold-and-Modify could be implemented (convert from RGB)
- One less subsystem to maintain

What do we give up?

- Low memory consumption
- Performance

Thanks & Acknowledgements

- The **Linux Foundation**, for organizing this conference and giving me the opportunity to present here,
- The **Linux Kernel Community**, for having so much fun working together towards a common goal,
- **Renesas Electronics Corporation**, for contracting me for upstream Linux kernel work.

Questions & Answers

Appendix

Patch Series

- *[PATCH v2 00/15] DRM fbconv helpers for converting fbdev drivers* by Thomas Zimmermann
  `https://lore.kernel.org/r/20191014140416.28517-10-tzimmermann@suse.de`
- *[PATCH v2 00/41] drm: Analog TV Improvements* by Maxime Ripard
  `https://lore.kernel.org/r/20220728-rpi-analog-tv-properties-v2-0-459522d653a7@cerno.tech`
- *[PATCH v3 00/10] drm: Add support for low-color frame buffer formats*
  `https://lore.kernel.org/r/cover.1657294931.git.geert@linux-m68k.org`
- *[PATCH 0/3] drm: Endianness fixes*
  `https://lore.kernel.org/r/cover.1657300532.git.geert@linux-m68k.org`
- *[PATCH v2 0/5] drm/modes: Command line mode selection fixes and improvements*
  `https://lore.kernel.org/r/cover.1657788997.git.geert@linux-m68k.org`
- *[PATCH libdrm v2 00/10] Add support for low-color frame buffer formats*
  `https://lore.kernel.org/r/cover.1657302034.git.geert@linux-m68k.org`
- *[PATCH libdrm v2 00/10] Big-endian fixes*
  `https://lore.kernel.org/r/cover.1657302103.git.geert@linux-m68k.org`