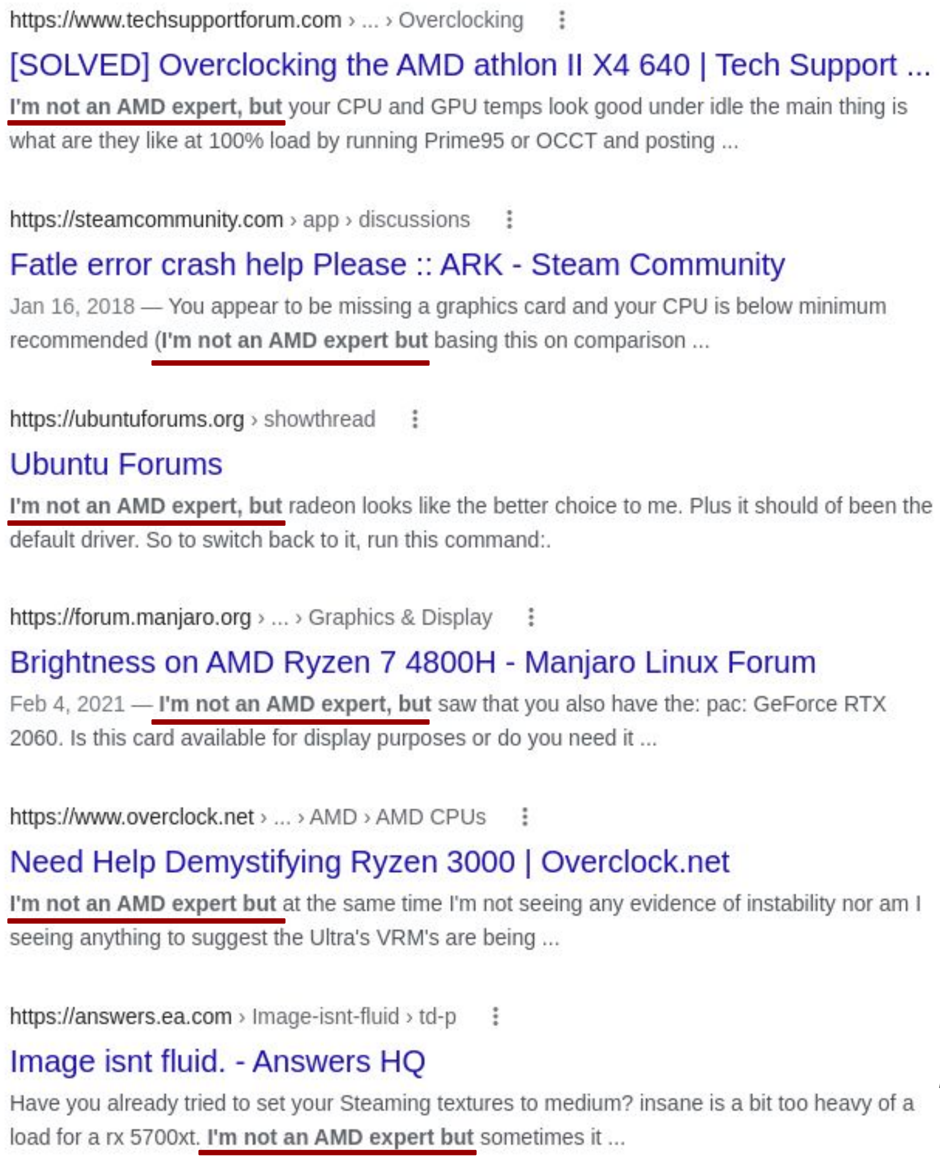
“I’m not an AMD expert, but…”

Melissa Wen @ Igalia

XDC - Oct 2022


https://www.techsupportforum.com › ... › Overclocking

[SOLVED] Overclocking the AMD athlon II X4 640 | Tech Support ...

I'm not an AMD expert, but your CPU and GPU temps look good under idle the main thing is what are they like at 100% load by running Prime95 or OCCT and posting ...

https://steamcommunity.com › app › discussions

Fatle error crash help Please :: ARK - Steam Community

Jan 16, 2018 — You appear to be missing a graphics card and your CPU is below minimum recommended (**I'm not an AMD expert but** basing this on comparison ...

https://ubuntuforums.org › showthread

Ubuntu Forums

I'm not an AMD expert, but radeon looks like the better choice to me. Plus it should of been the default driver. So to switch back to it, run this command:.

https://forum.manjaro.org › ... › Graphics & Display

Brightness on AMD Ryzen 7 4800H - Manjaro Linux Forum

Feb 4, 2021 — **I'm not an AMD expert, but** saw that you also have the: pac: GeForce RTX 2060. Is this card available for display purposes or do you need it ...

https://www.overclock.net › ... › AMD › AMD CPUs

Need Help Demystifying Ryzen 3000 | Overclock.net

I'm not an AMD expert but at the same time I'm not seeing any evidence of instability nor am I seeing anything to suggest the Ultra's VRM's are being ...

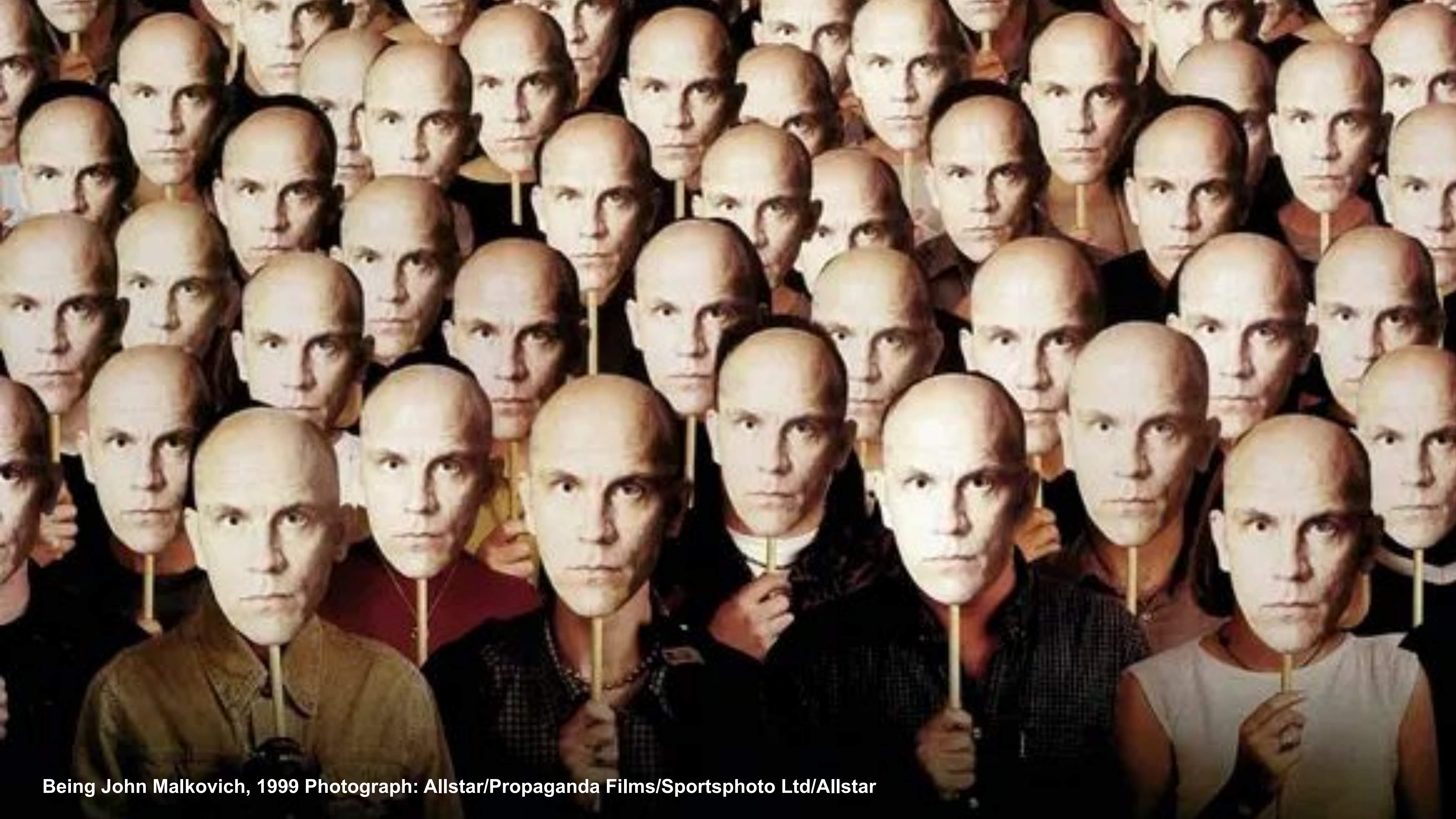
https://answers.ea.com › Image-isnt-fluid › td-p

Image isnt fluid. - Answers HQ

Have you already tried to set your Steaming textures to medium? insane is a bit too heavy of a load for a rx 5700xt. **I'm not an AMD expert but** sometimes it ...

Trend topics not covered by this talk:

- Criticize code statistics
- Complain about the shared code approach of the display driver
- Put a curse on AMD


Being John Malkovich, 1999 Photograph: Allstar/Propaganda Films/Sportsphoto Ltd/Allstar

Play the game you already know…

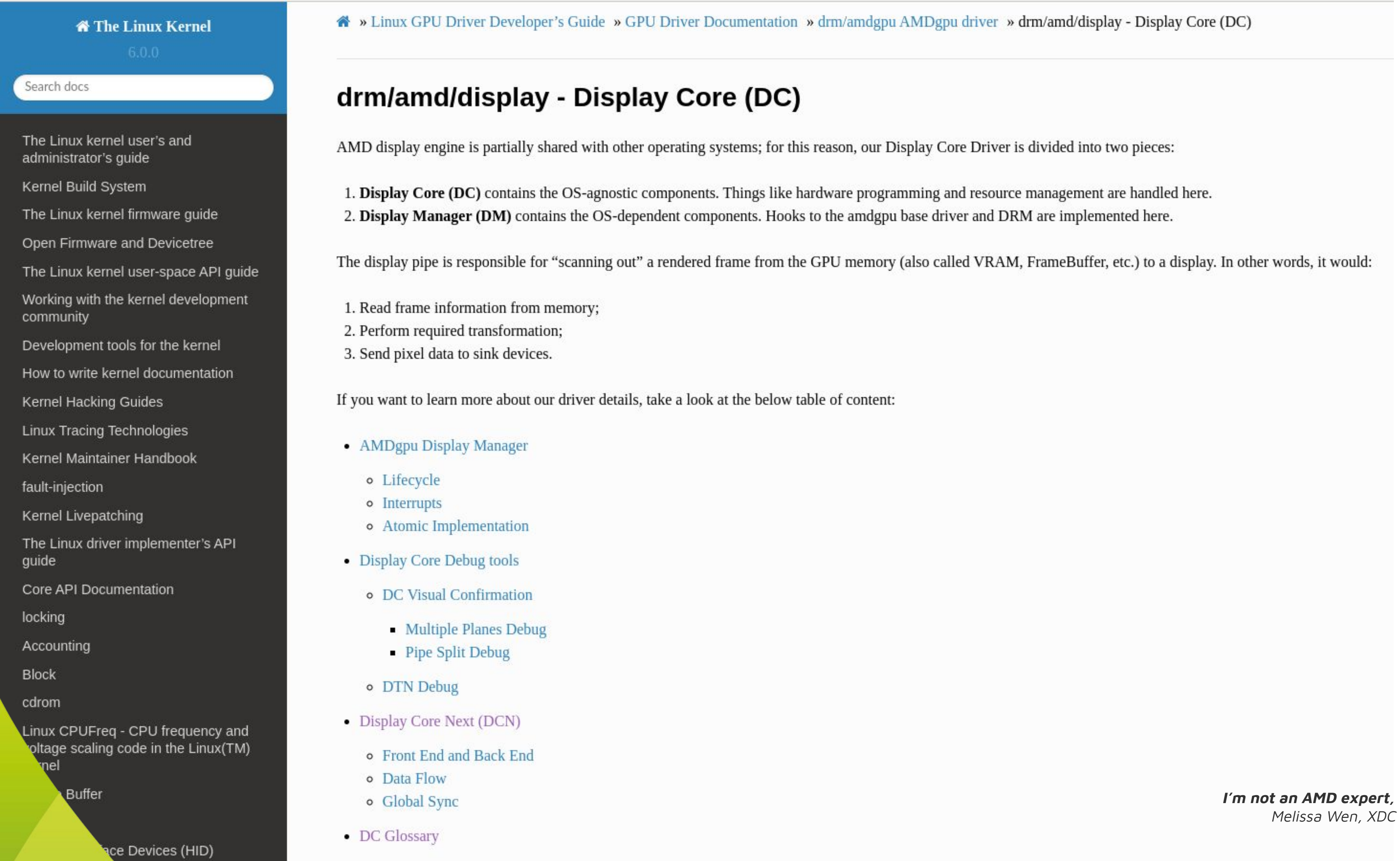
- KMS tests of the IGT testing tools

and have fun fixing bugs

- DRM alpha blend mode

Follow traditional breadcrumbs

- documentation
- mailing list
- git history
- checking drivers from other vendors
- own experience



The Linux Kernel
6.0.0

Search docs

- The Linux kernel user's and administrator's guide
- Kernel Build System
- The Linux kernel firmware guide
- Open Firmware and Devicetree
- The Linux kernel user-space API guide
- Working with the kernel development community
- Development tools for the kernel
- How to write kernel documentation
- Kernel Hacking Guides
- Linux Tracing Technologies
- Kernel Maintainer Handbook
- fault-injection
- Kernel Livepatching
- The Linux driver implementer's API guide
- Core API Documentation
- locking
- Accounting
- Block
- cdrom
- Linux CPUFreq - CPU frequency and oltage scaling code in the Linux(TM) rnel
- Buffer
- ace Devices (HID)

» Linux GPU Driver Developer's Guide » GPU Driver Documentation » drm/amdgpu AMDgpu driver » drm/amd/display - Display Core (DC)

drm/amd/display - Display Core (DC)

AMD display engine is partially shared with other operating systems; for this reason, our Display Core Driver is divided into two pieces:

1. **Display Core (DC)** contains the OS-agnostic components. Things like hardware programming and resource management are handled here.
2. **Display Manager (DM)** contains the OS-dependent components. Hooks to the amdgpu base driver and DRM are implemented here.

The display pipe is responsible for "scanning out" a rendered frame from the GPU memory (also called VRAM, FrameBuffer, etc.) to a display. In other words, it would:

1. Read frame information from memory;
2. Perform required transformation;
3. Send pixel data to sink devices.

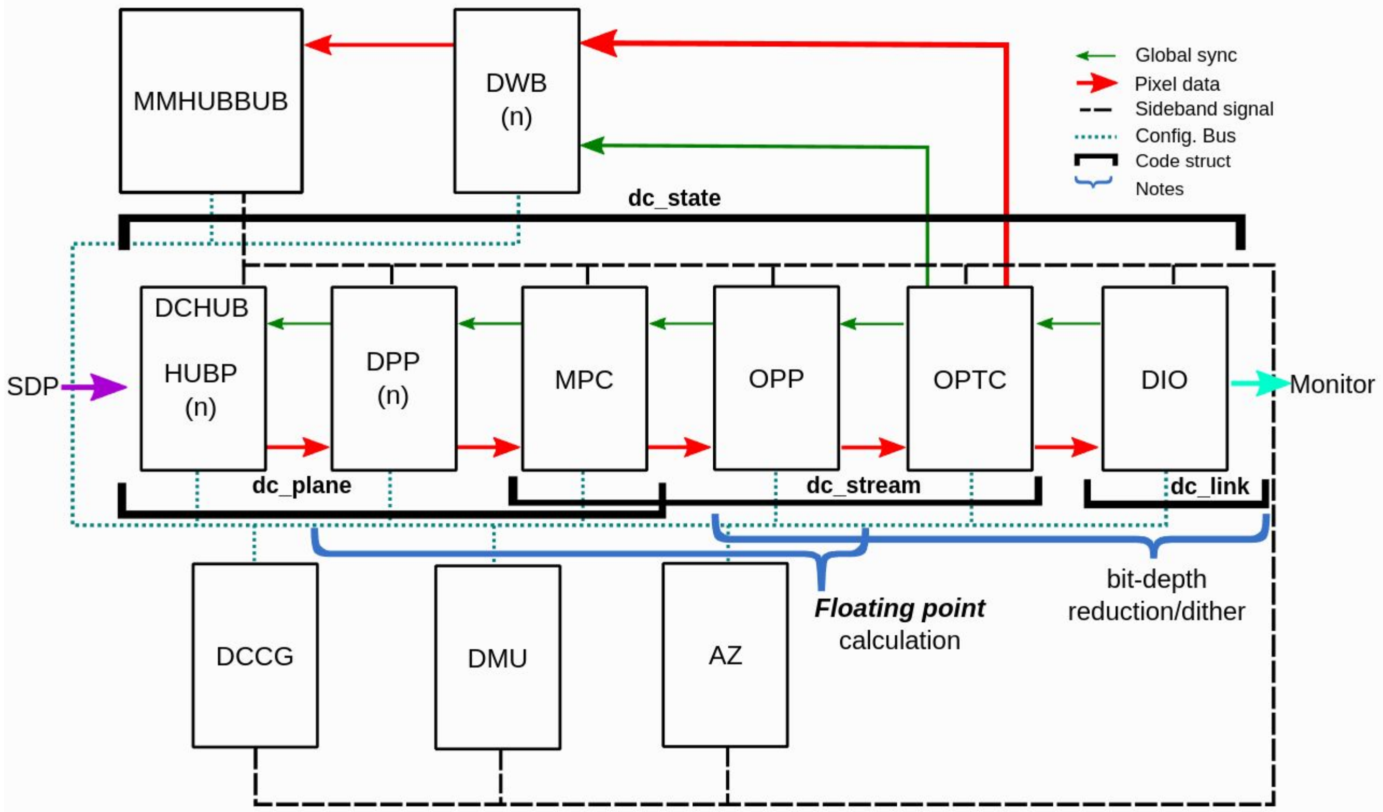
If you want to learn more about our driver details, take a look at the below table of content:

- AMDgpu Display Manager
  - Lifecycle
  - Interrupts
  - Atomic Implementation
- Display Core Debug tools
  - DC Visual Confirmation
    - Multiple Planes Debug
    - Pipe Split Debug
  - DTN Debug
- Display Core Next (DCN)
  - Front End and Back End
  - Data Flow
  - Global Sync
- DC Glossary

I'm not an AMD expert, but...
Melissa Wen, XDC 2022

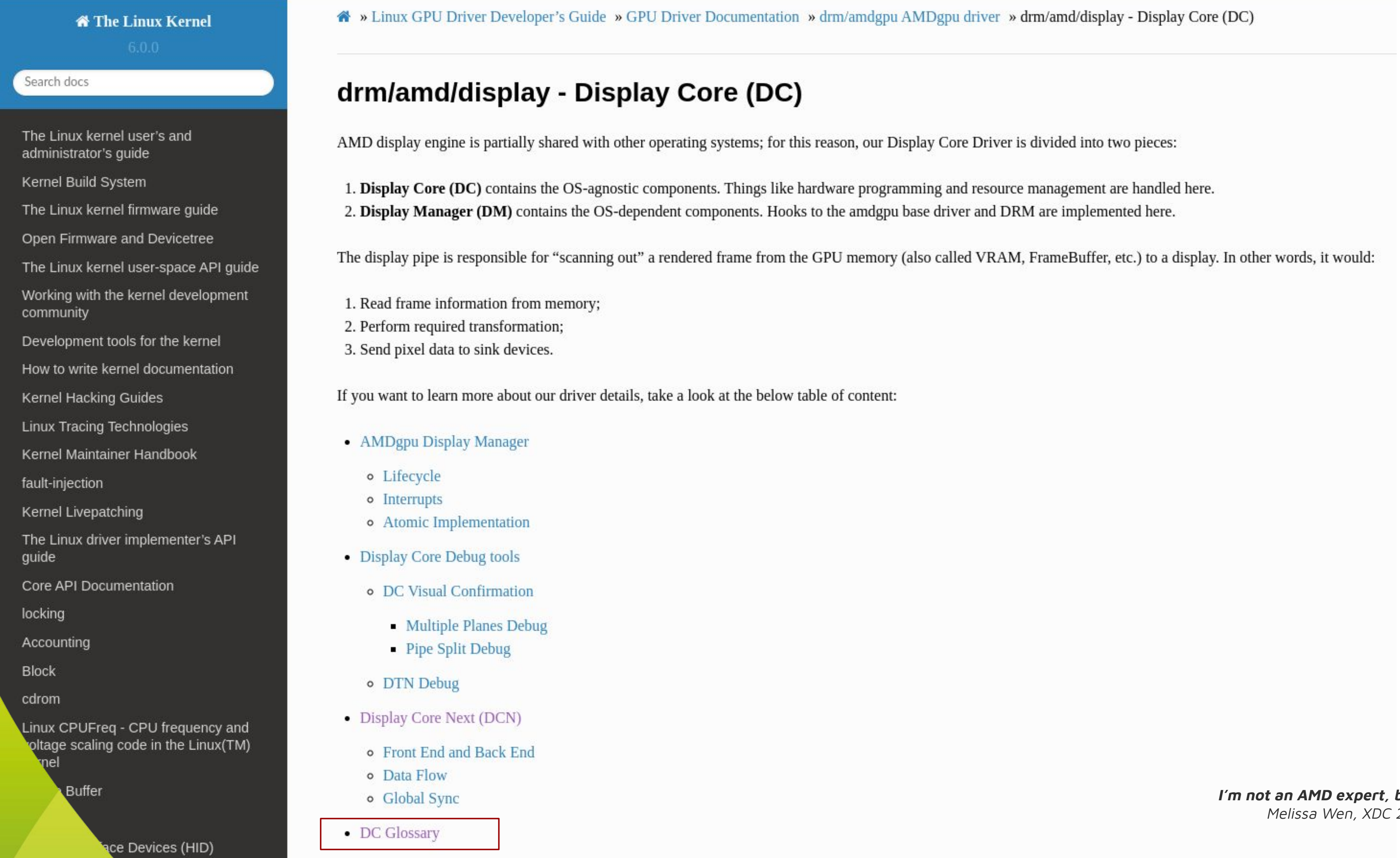
Display Core Next (DCN)

To equip our readers with the basic knowledge of how AMD Display Core Next (DCN) works, we need to start with an overview of the hardware pipeline. Below you can see a picture that provides a DCN overview, keep in mind that this is a generic diagram, and we have variations per ASIC.



Based on this diagram, we can pass through each block and briefly describe them:

I'm not an AMD expert, but...
Melissa Wen, XDC 2022



The Linux Kernel

6.0.0

Search docs

The Linux kernel user's and administrator's guide
Kernel Build System
The Linux kernel firmware guide
Open Firmware and Devicetree
The Linux kernel user-space API guide
Working with the kernel development community
Development tools for the kernel
How to write kernel documentation
Kernel Hacking Guides
Linux Tracing Technologies
Kernel Maintainer Handbook
fault-injection
Kernel Livepatching
The Linux driver implementer's API guide
Core API Documentation
locking
Accounting
Block
cdrom
Linux CPUFreq - CPU frequency and oltage scaling code in the Linux(TM) rnel
Buffer
ace Devices (HID)

» Linux GPU Driver Developer's Guide » GPU Driver Documentation » drm/amdgpu AMDgpu driver » drm/amd/display - Display Core (DC)

drm/amd/display - Display Core (DC)

AMD display engine is partially shared with other operating systems; for this reason, our Display Core Driver is divided into two pieces:

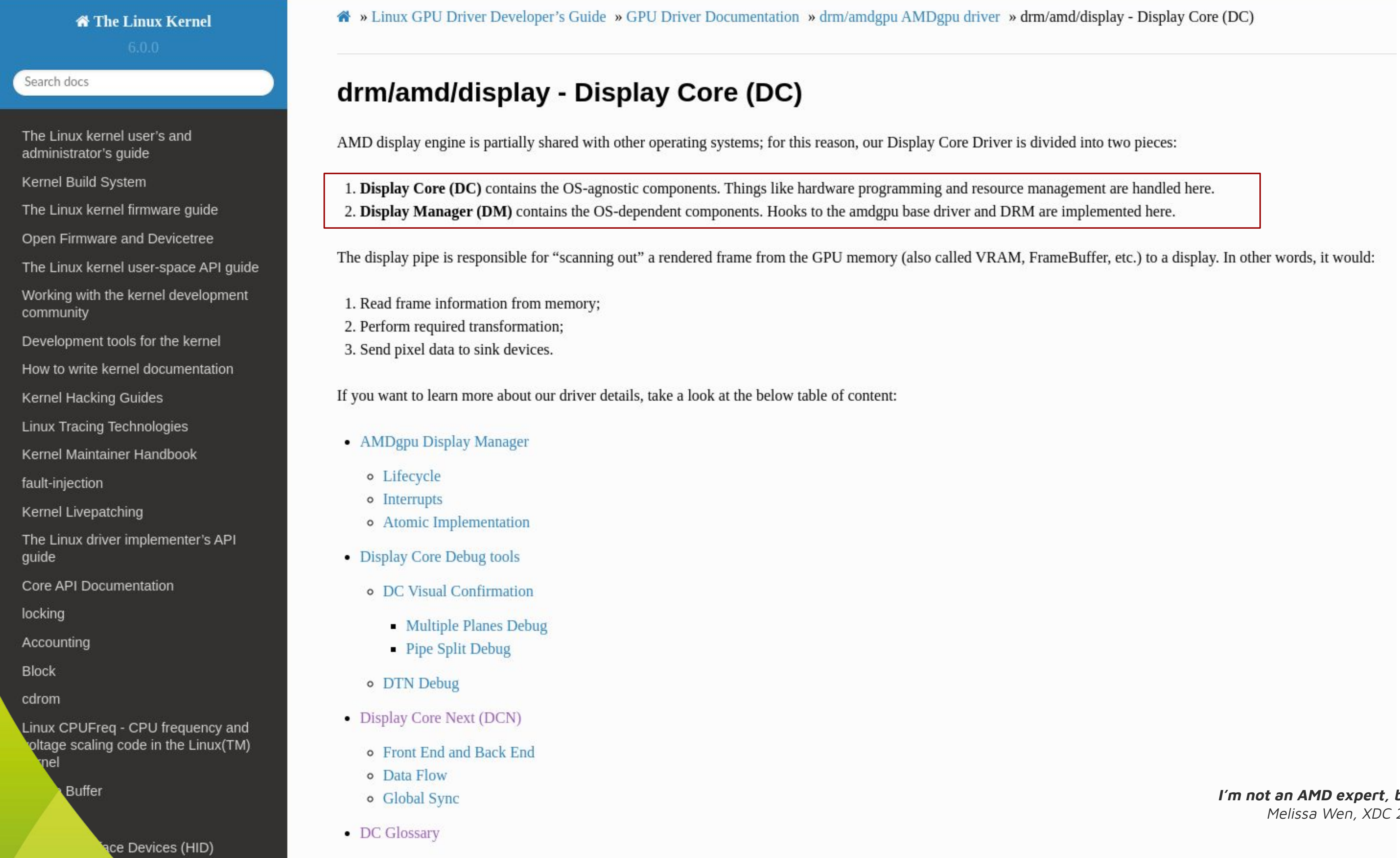
1. **Display Core (DC)** contains the OS-agnostic components. Things like hardware programming and resource management are handled here.
2. **Display Manager (DM)** contains the OS-dependent components. Hooks to the amdgpu base driver and DRM are implemented here.

The display pipe is responsible for "scanning out" a rendered frame from the GPU memory (also called VRAM, FrameBuffer, etc.) to a display. In other words, it would:

1. Read frame information from memory;
2. Perform required transformation;
3. Send pixel data to sink devices.

If you want to learn more about our driver details, take a look at the below table of content:

I'm not an AMD expert, but...
Melissa Wen, XDC 2022


The Linux Kernel
6.0.0
Search docs

The Linux kernel user's and administrator's guide
Kernel Build System
The Linux kernel firmware guide
Open Firmware and Devicetree
The Linux kernel user-space API guide
Working with the kernel development community
Development tools for the kernel
How to write kernel documentation
Kernel Hacking Guides
Linux Tracing Technologies
Kernel Maintainer Handbook
fault-injection
Kernel Livepatching
The Linux driver implementer's API guide
Core API Documentation
locking
Accounting
Block
cdrom
Linux CPUFreq - CPU frequency and oltage scaling code in the Linux(TM) rnel
Buffer
ace Devices (HID)

» Linux GPU Driver Developer's Guide » GPU Driver Documentation » drm/amdgpu AMDgpu driver » drm/amd/display - Display Core (DC)

drm/amd/display - Display Core (DC)

AMD display engine is partially shared with other operating systems; for this reason, our Display Core Driver is divided into two pieces:

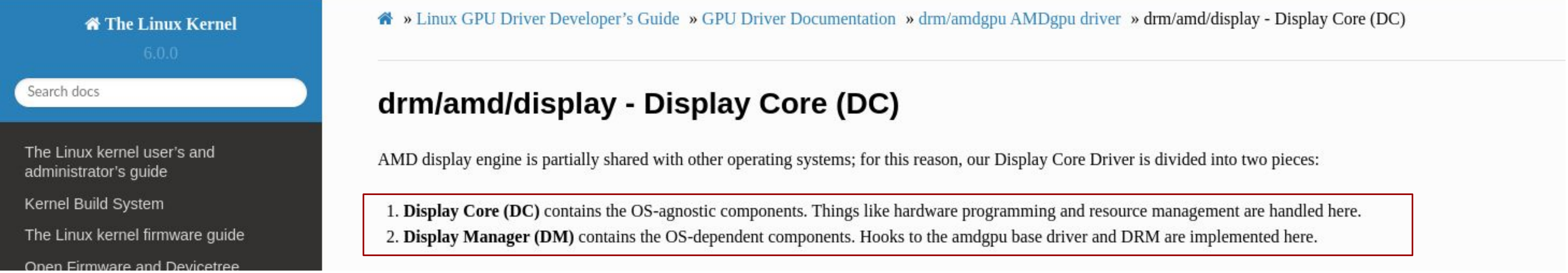
1. **Display Core (DC)** contains the OS-agnostic components. Things like hardware programming and resource management are handled here.
2. **Display Manager (DM)** contains the OS-dependent components. Hooks to the amdgpu base driver and DRM are implemented here.

The display pipe is responsible for "scanning out" a rendered frame from the GPU memory (also called VRAM, FrameBuffer, etc.) to a display. In other words, it would:

1. Read frame information from memory;
2. Perform required transformation;
3. Send pixel data to sink devices.

If you want to learn more about our driver details, take a look at the below table of content:

I'm not an AMD expert, but...
Melissa Wen, XDC 2022



The Linux Kernel
6.0.0
Search docs

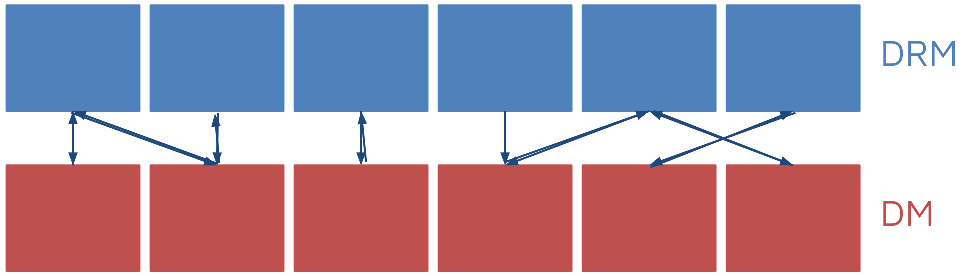
The Linux kernel user's and administrator's guide
Kernel Build System
The Linux kernel firmware guide
Open Firmware and Devicetree

» Linux GPU Driver Developer's Guide » GPU Driver Documentation » drm/amdgpu AMDgpu driver » drm/amd/display - Display Core (DC)

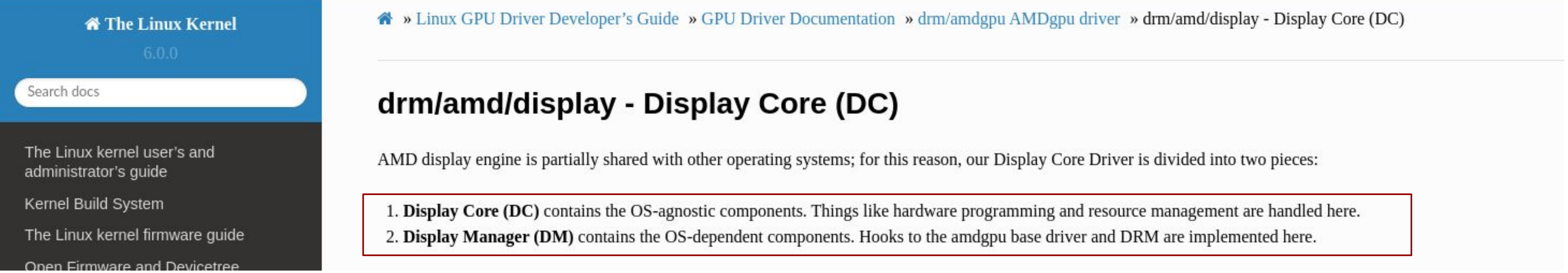
drm/amd/display - Display Core (DC)

AMD display engine is partially shared with other operating systems; for this reason, our Display Core Driver is divided into two pieces:

1. **Display Core (DC)** contains the OS-agnostic components. Things like hardware programming and resource management are handled here.
2. **Display Manager (DM)** contains the OS-dependent components. Hooks to the amdgpu base driver and DRM are implemented here.



I'm not an AMD expert, but...
Melissa Wen, XDC 2022



The Linux Kernel

6.0.0

Search docs

The Linux kernel user's and administrator's guide

Kernel Build System

The Linux kernel firmware guide

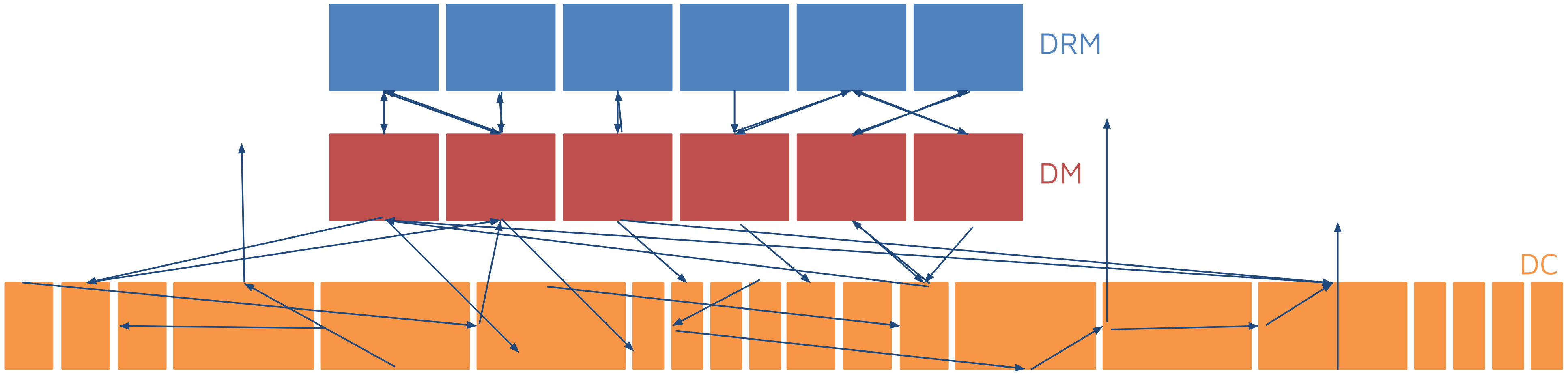
Open Firmware and Devicetree

» Linux GPU Driver Developer's Guide » GPU Driver Documentation » drm/amdgpu AMDgpu driver » drm/amd/display - Display Core (DC)

drm/amd/display - Display Core (DC)

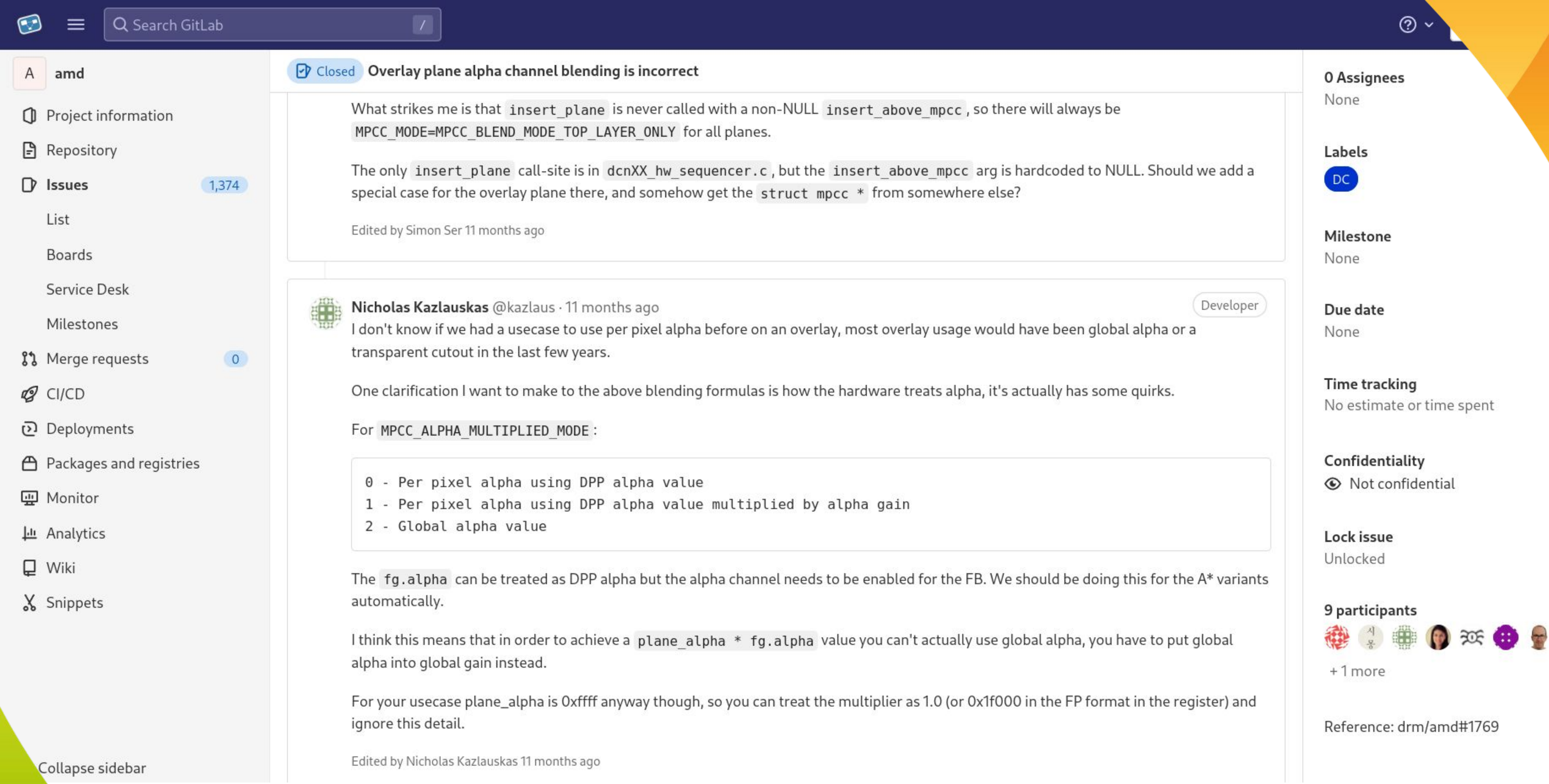
AMD display engine is partially shared with other operating systems; for this reason, our Display Core Driver is divided into two pieces:

1. **Display Core (DC)** contains the OS-agnostic components. Things like hardware programming and resource management are handled here.
2. **Display Manager (DM)** contains the OS-dependent components. Hooks to the amdgpu base driver and DRM are implemented here.



Forget the Traditional Breadcrumbs

Perspective-Taking



Search GitLab

amd

Project information
Repository
Issues 1,374
List
Boards
Service Desk
Milestones
Merge requests 0
CI/CD
Deployments
Packages and registries
Monitor
Analytics
Wiki
Snippets
Collapse sidebar

Closed **Overlay plane alpha channel blending is incorrect**

What strikes me is that `insert_plane` is never called with a non-NULL `insert_above_mpcc`, so there will always be `MPCC_MODE=MPCC_BLEND_MODE_TOP_LAYER_ONLY` for all planes.

The only `insert_plane` call-site is in `dcnXX_hw_sequencer.c`, but the `insert_above_mpcc` arg is hardcoded to NULL. Should we add a special case for the overlay plane there, and somehow get the `struct mpcc *` from somewhere else?

Edited by Simon Ser 11 months ago

Nicholas Kazlauskas @kazlaus · 11 months ago Developer

I don't know if we had a usecase to use per pixel alpha before on an overlay, most overlay usage would have been global alpha or a transparent cutout in the last few years.

One clarification I want to make to the above blending formulas is how the hardware treats alpha, it's actually has some quirks.

For `MPCC_ALPHA_MULTIPLIED_MODE`:

```
0 - Per pixel alpha using DPP alpha value
1 - Per pixel alpha using DPP alpha value multiplied by alpha gain
2 - Global alpha value
```

The `fg.alpha` can be treated as DPP alpha but the alpha channel needs to be enabled for the FB. We should be doing this for the A* variants automatically.

I think this means that in order to achieve a `plane_alpha * fg.alpha` value you can't actually use global alpha, you have to put global alpha into global gain instead.

For your usecase plane_alpha is 0xffff anyway though, so you can treat the multiplier as 1.0 (or 0x1f000 in the FP format in the register) and ignore this detail.

Edited by Nicholas Kazlauskas 11 months ago

0 Assignees
None

Labels
DC

Milestone
None

Due date
None

Time tracking
No estimate or time spent

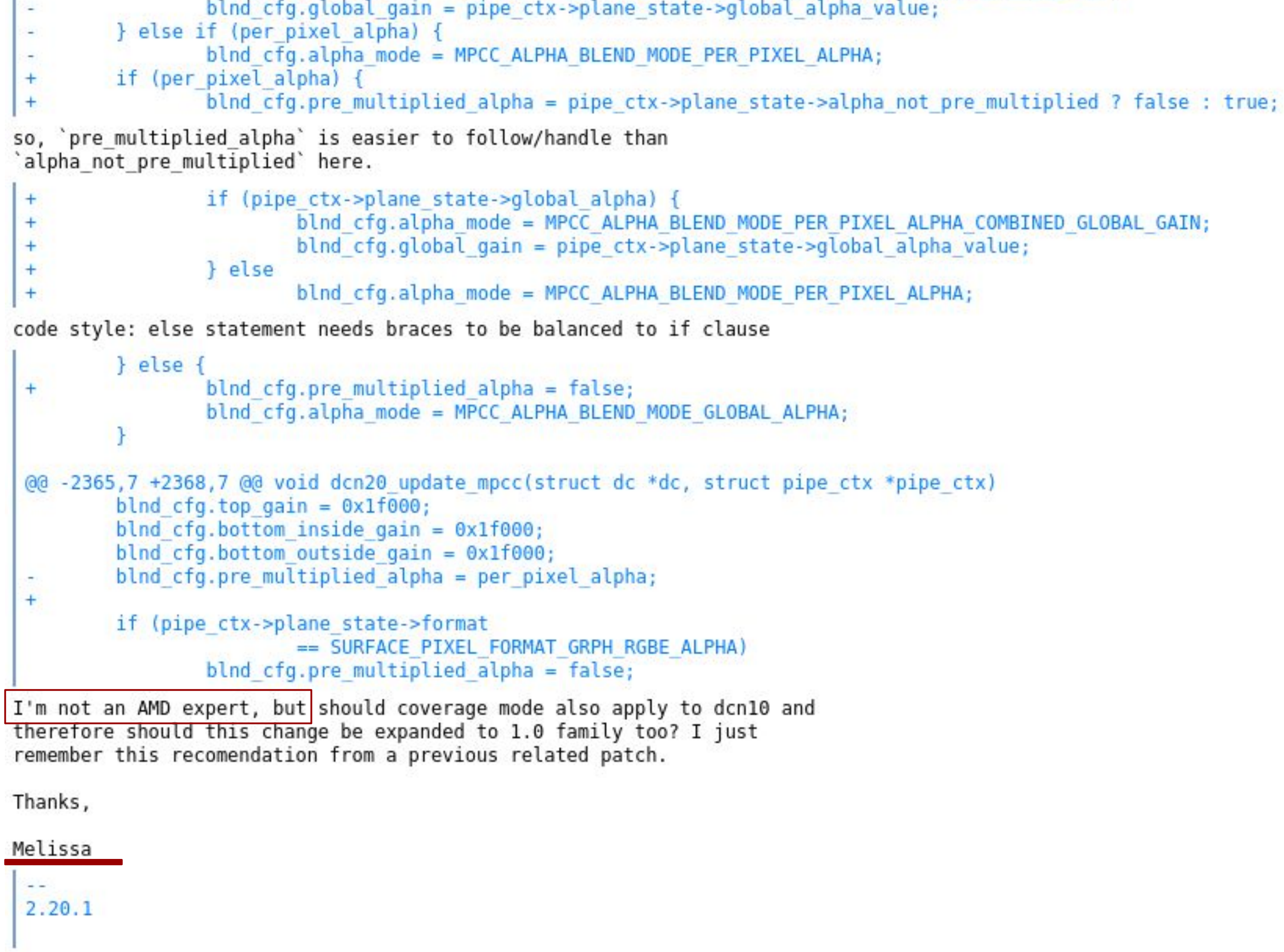
Confidentiality
Not confidential

Lock issue
Unlocked

9 participants
+ 1 more

Reference: drm/amd#1769

I'm not an AMD expert, but...
Melissa Wen, XDC 2022



```
-                       blnd_cfg.global_gain = pipe_ctx->plane_state->global_alpha_value;
-               } else if (per_pixel_alpha) {
-                       blnd_cfg.alpha_mode = MPCC_ALPHA_BLEND_MODE_PER_PIXEL_ALPHA;
+               if (per_pixel_alpha) {
+                       blnd_cfg.pre_multiplied_alpha = pipe_ctx->plane_state->alpha_not_pre_multiplied ? false : true;
```

so, `pre_multiplied_alpha` is easier to follow/handle than
`alpha_not_pre_multiplied` here.

```
+                       if (pipe_ctx->plane_state->global_alpha) {
+                               blnd_cfg.alpha_mode = MPCC_ALPHA_BLEND_MODE_PER_PIXEL_ALPHA_COMBINED_GLOBAL_GAIN;
+                               blnd_cfg.global_gain = pipe_ctx->plane_state->global_alpha_value;
+                       } else
+                               blnd_cfg.alpha_mode = MPCC_ALPHA_BLEND_MODE_PER_PIXEL_ALPHA;
```

code style: else statement needs braces to be balanced to if clause

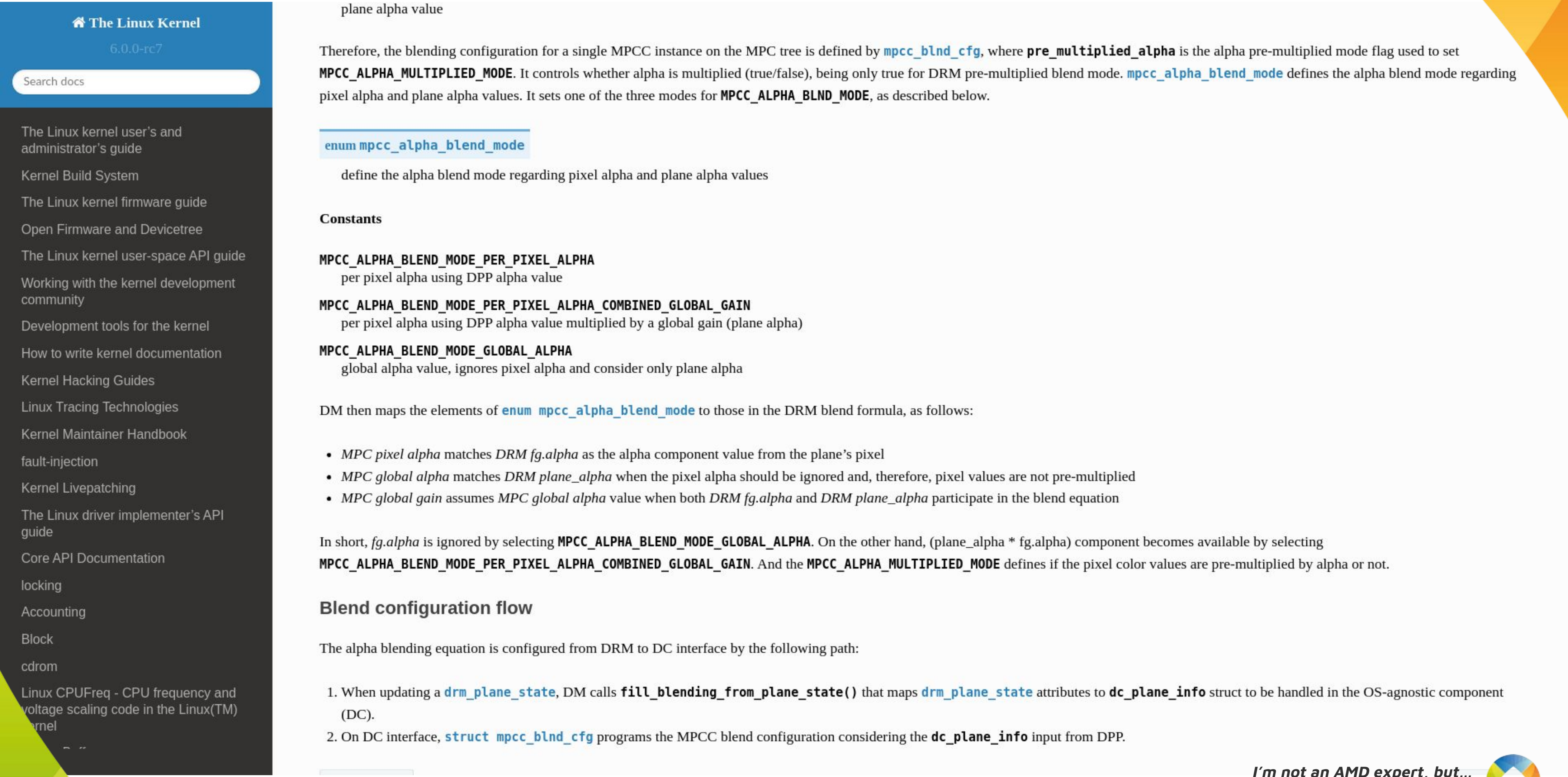
```
                } else {
+                       blnd_cfg.pre_multiplied_alpha = false;
                        blnd_cfg.alpha_mode = MPCC_ALPHA_BLEND_MODE_GLOBAL_ALPHA;
                }

@@ -2365,7 +2368,7 @@ void dcn20_update_mpcc(struct dc *dc, struct pipe_ctx *pipe_ctx)
                blnd_cfg.top_gain = 0x1f000;
                blnd_cfg.bottom_inside_gain = 0x1f000;
                blnd_cfg.bottom_outside_gain = 0x1f000;
-               blnd_cfg.pre_multiplied_alpha = per_pixel_alpha;
+
                if (pipe_ctx->plane_state->format
                                == SURFACE_PIXEL_FORMAT_GRPH_RGBE_ALPHA)
                        blnd_cfg.pre_multiplied_alpha = false;
```

I'm not an AMD expert, but should coverage mode also apply to dcn10 and
therefore should this change be expanded to 1.0 family too? I just
remember this recomendation from a previous related patch.

Thanks,

Melissa

```
--
2.20.1
```



plane alpha value

Therefore, the blending configuration for a single MPCC instance on the MPC tree is defined by `mpcc_blnd_cfg`, where **`pre_multiplied_alpha`** is the alpha pre-multiplied mode flag used to set **`MPCC_ALPHA_MULTIPLIED_MODE`**. It controls whether alpha is multiplied (true/false), being only true for DRM pre-multiplied blend mode. `mpcc_alpha_blend_mode` defines the alpha blend mode regarding pixel alpha and plane alpha values. It sets one of the three modes for **`MPCC_ALPHA_BLND_MODE`**, as described below.

enum `mpcc_alpha_blend_mode`

define the alpha blend mode regarding pixel alpha and plane alpha values

Constants

`MPCC_ALPHA_BLEND_MODE_PER_PIXEL_ALPHA`
per pixel alpha using DPP alpha value

`MPCC_ALPHA_BLEND_MODE_PER_PIXEL_ALPHA_COMBINED_GLOBAL_GAIN`
per pixel alpha using DPP alpha value multiplied by a global gain (plane alpha)

`MPCC_ALPHA_BLEND_MODE_GLOBAL_ALPHA`
global alpha value, ignores pixel alpha and consider only plane alpha

DM then maps the elements of `enum mpcc_alpha_blend_mode` to those in the DRM blend formula, as follows:

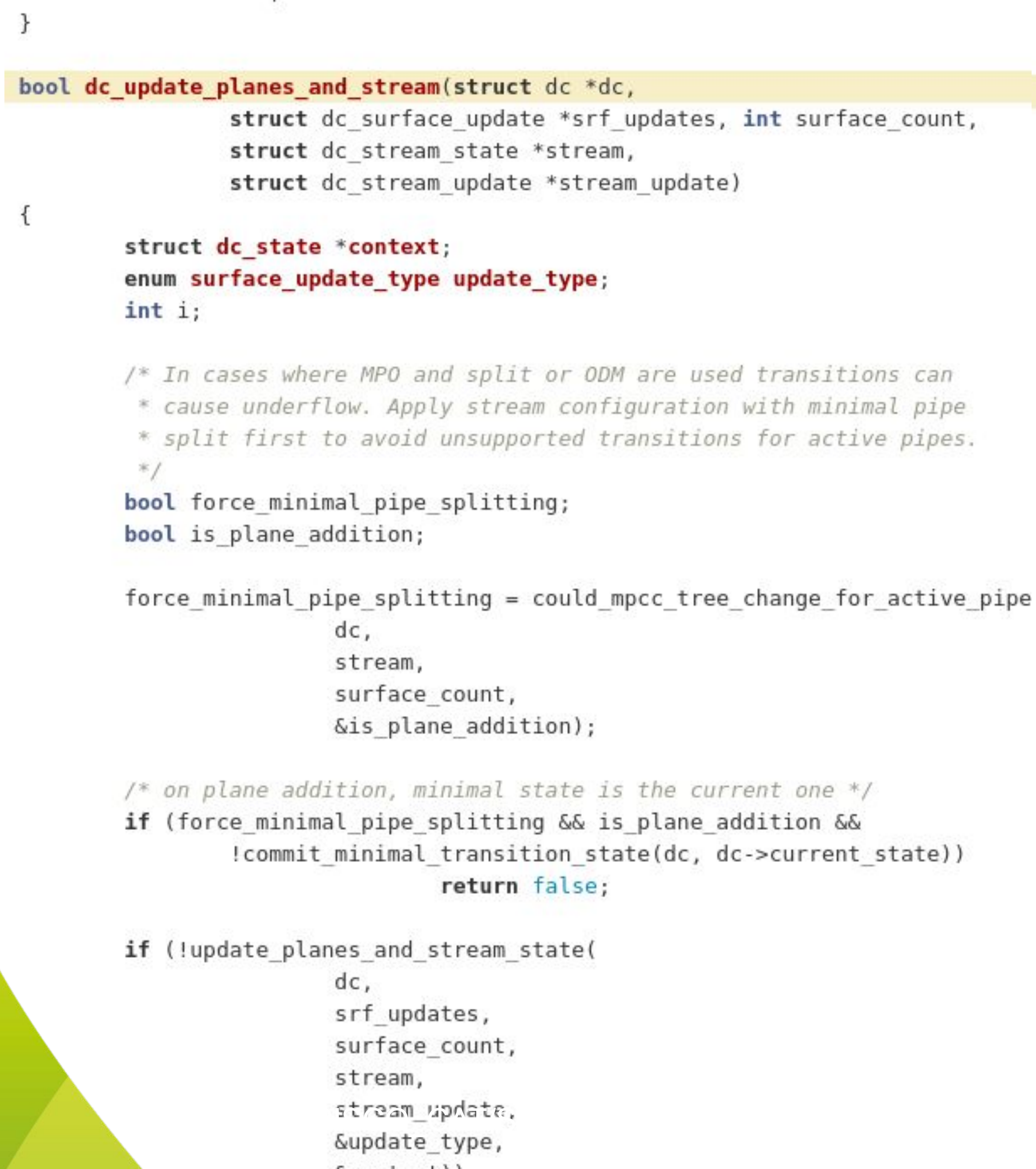
- *MPC pixel alpha* matches *DRM fg.alpha* as the alpha component value from the plane's pixel
- *MPC global alpha* matches *DRM plane_alpha* when the pixel alpha should be ignored and, therefore, pixel values are not pre-multiplied
- *MPC global gain* assumes *MPC global alpha* value when both *DRM fg.alpha* and *DRM plane_alpha* participate in the blend equation

In short, *fg.alpha* is ignored by selecting **`MPCC_ALPHA_BLEND_MODE_GLOBAL_ALPHA`**. On the other hand, (plane_alpha * fg.alpha) component becomes available by selecting **`MPCC_ALPHA_BLEND_MODE_PER_PIXEL_ALPHA_COMBINED_GLOBAL_GAIN`**. And the **`MPCC_ALPHA_MULTIPLIED_MODE`** defines if the pixel color values are pre-multiplied by alpha or not.

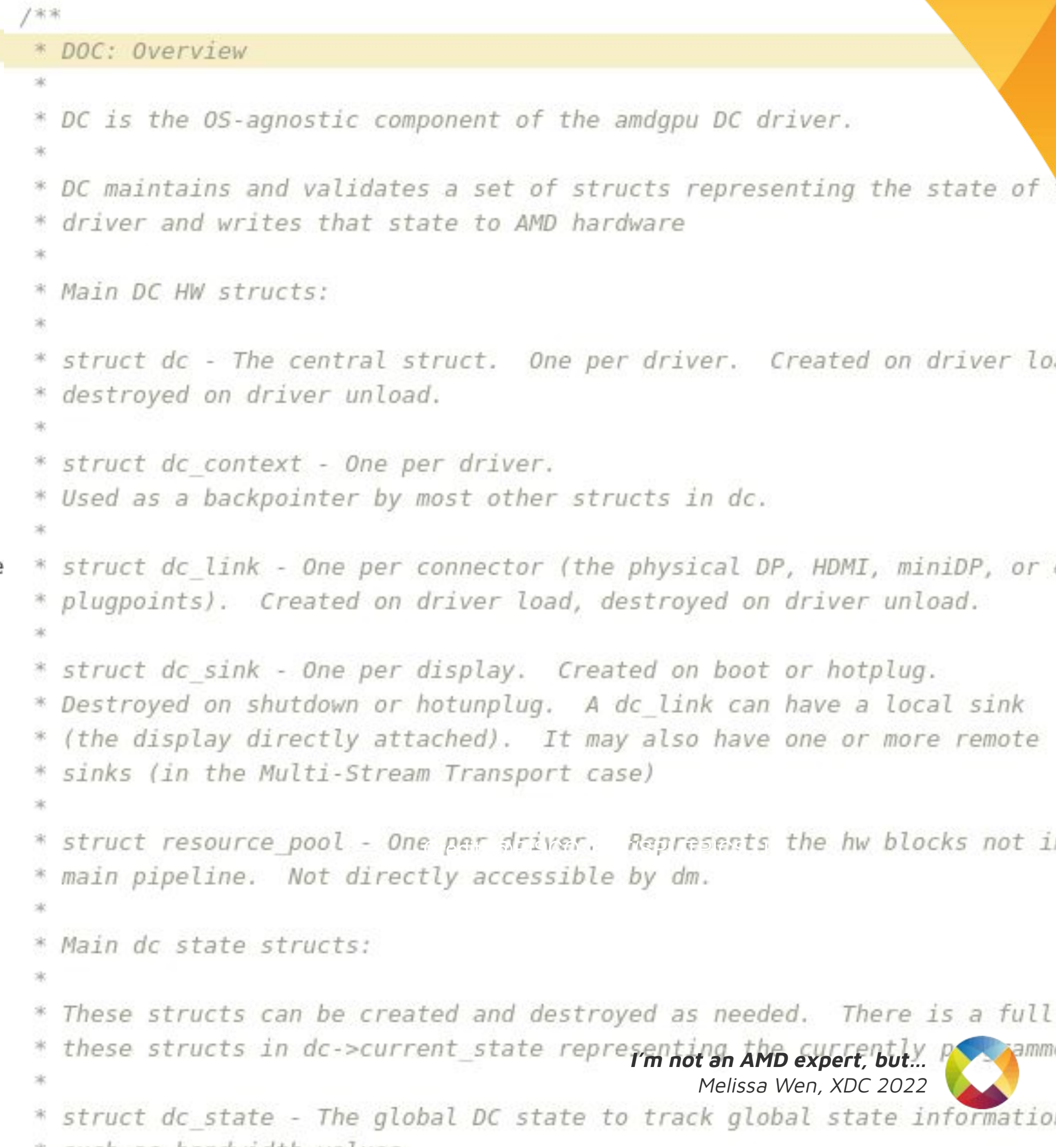
Blend configuration flow

The alpha blending equation is configured from DRM to DC interface by the following path:

1. When updating a `drm_plane_state`, DM calls **`fill_blending_from_plane_state()`** that maps `drm_plane_state` attributes to **`dc_plane_info`** struct to be handled in the OS-agnostic component (DC).
2. On DC interface, `struct mpcc_blnd_cfg` programs the MPCC blend configuration considering the **`dc_plane_info`** input from DPP.

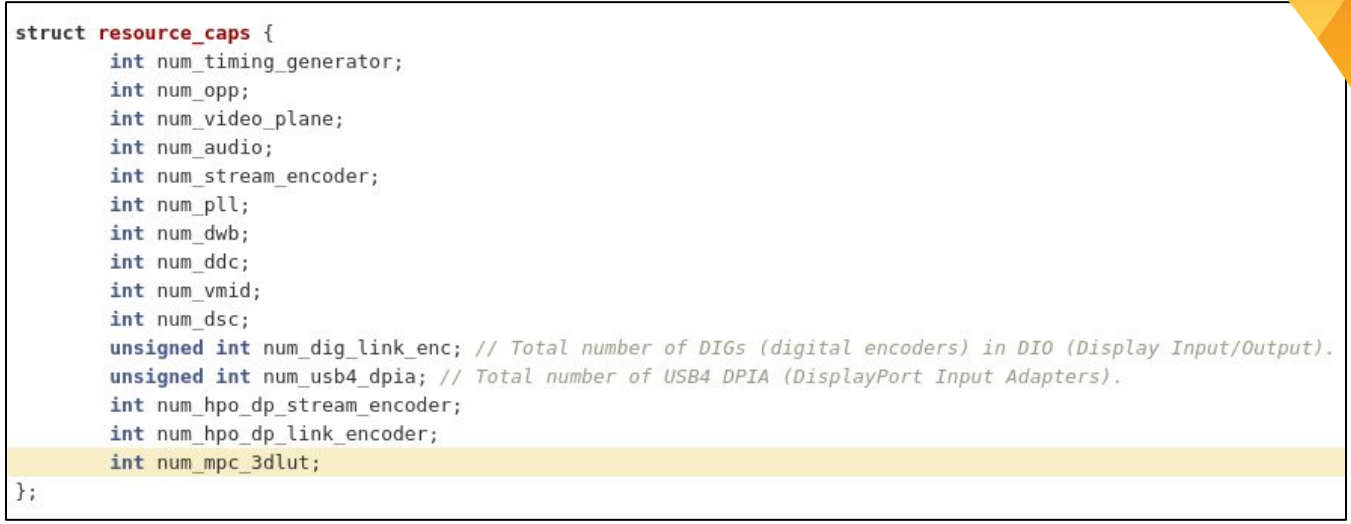


```
}

bool dc_update_planes_and_stream(struct dc *dc,
                struct dc_surface_update *srf_updates, int surface_count,
                struct dc_stream_state *stream,
                struct dc_stream_update *stream_update)
{
        struct dc_state *context;
        enum surface_update_type update_type;
        int i;

        /* In cases where MPO and split or ODM are used transitions can
         * cause underflow. Apply stream configuration with minimal pipe
         * split first to avoid unsupported transitions for active pipes.
         */
        bool force_minimal_pipe_splitting;
        bool is_plane_addition;

        force_minimal_pipe_splitting = could_mpcc_tree_change_for_active_pipe
                        dc,
                        stream,
                        surface_count,
                        &is_plane_addition);

        /* on plane addition, minimal state is the current one */
        if (force_minimal_pipe_splitting && is_plane_addition &&
                !commit_minimal_transition_state(dc, dc->current_state))
                                return false;

        if (!update_planes_and_stream_state(
                        dc,
                        srf_updates,
                        surface_count,
                        stream,
                        stream_update,
                        &update_type,
```

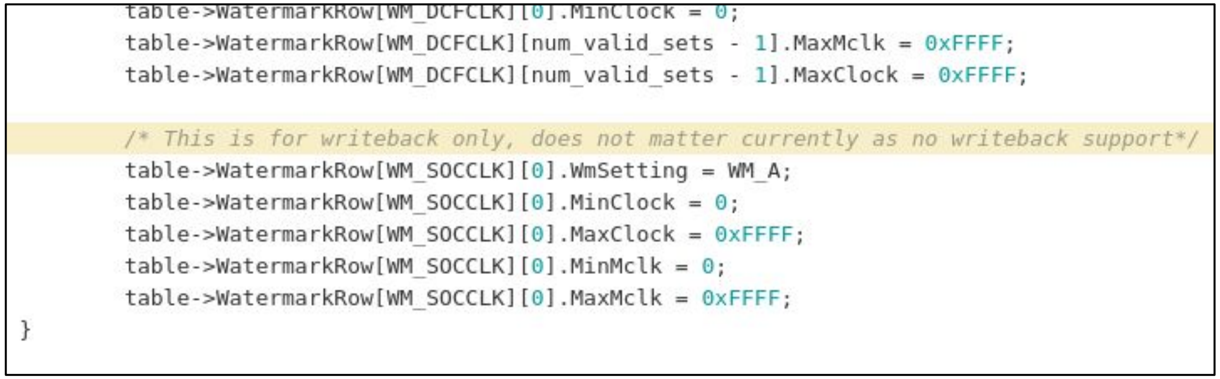


```
/**
 * DOC: Overview
 *
 * DC is the OS-agnostic component of the amdgpu DC driver.
 *
 * DC maintains and validates a set of structs representing the state of
 * driver and writes that state to AMD hardware
 *
 * Main DC HW structs:
 *
 * struct dc - The central struct.  One per driver.  Created on driver lo
 * destroyed on driver unload.
 *
 * struct dc_context - One per driver.
 * Used as a backpointer by most other structs in dc.
 *
 * struct dc_link - One per connector (the physical DP, HDMI, miniDP, or
 * plugpoints).  Created on driver load, destroyed on driver unload.
 *
 * struct dc_sink - One per display.  Created on boot or hotplug.
 * Destroyed on shutdown or hotunplug.  A dc_link can have a local sink
 * (the display directly attached).  It may also have one or more remote
 * sinks (in the Multi-Stream Transport case)
 *
 * struct resource_pool - One per driver.  Represents the hw blocks not i
 * main pipeline.  Not directly accessible by dm.
 *
 * Main dc state structs:
 *
 * These structs can be created and destroyed as needed.  There is a full
 * these structs in dc->current_state representing the currently p
 *
 * struct dc_state - The global DC state to track global state informatio
```

START CROSSING
Watch For Vehicles
DON'T START
Finish Crossing If Started
FLASHING
08
TIMER
TIME REMAINING To Finish Crossing
STEADY
DON'T CROSS
PUSH BUTTON
TO CROSS



```
struct resource_caps {
	int num_timing_generator;
	int num_opp;
	int num_video_plane;
	int num_audio;
	int num_stream_encoder;
	int num_pll;
	int num_dwb;
	int num_ddc;
	int num_vmid;
	int num_dsc;
	unsigned int num_dig_link_enc; // Total number of DIGs (digital encoders) in DIO (Display Input/Output).
	unsigned int num_usb4_dpia; // Total number of USB4 DPIA (DisplayPort Input Adapters).
	int num_hpo_dp_stream_encoder;
	int num_hpo_dp_link_encoder;
	int num_mpc_3dlut;
};
```



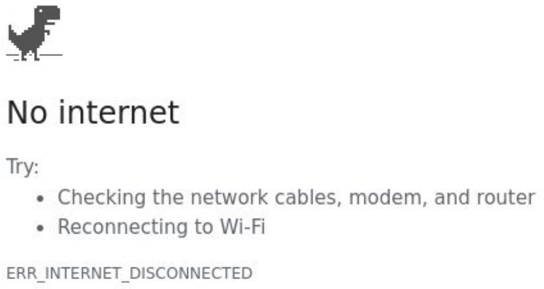
```
	table->WatermarkRow[WM_DCFCLK][num_valid_sets - 1].MaxMclk = 0xFFFF;
	table->WatermarkRow[WM_DCFCLK][num_valid_sets - 1].MaxClock = 0xFFFF;

	/* This is for writeback only, does not matter currently as no writeback support*/
	table->WatermarkRow[WM_SOCCLK][0].WmSetting = WM_A;
	table->WatermarkRow[WM_SOCCLK][0].MinClock = 0;
	table->WatermarkRow[WM_SOCCLK][0].MaxClock = 0xFFFF;
	table->WatermarkRow[WM_SOCCLK][0].MinMclk = 0;
	table->WatermarkRow[WM_SOCCLK][0].MaxMclk = 0xFFFF;
}
```

No internet

Try:

- Checking the network cables, modem, and router
- Reconnecting to Wi-Fi

ERR_INTERNET_DISCONNECTED



No internet

Try:

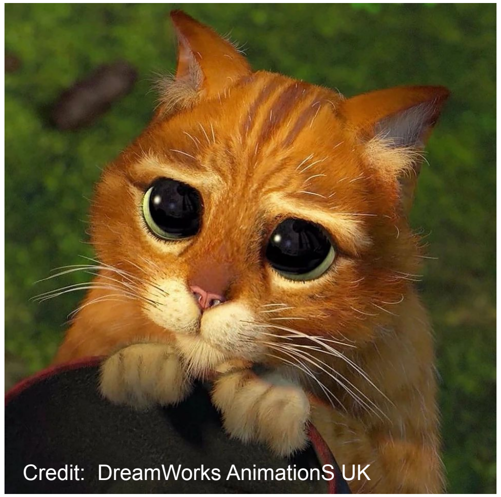
- Checking the network cables, modem, and router
- Reconnecting to Wi-Fi

ERR_INTERNET_DISCONNECTED

ATLANTIC OCEAN
MOROCCO
WESTERN SAHARA
CABO VERDE
PRAIA
MAURITANIA
MALI
SENEGAL
THE GAMBIA
GUINEA-BISSAU
GUINEA
SIERRA LEONE
LIBERIA
ATLANTIC OCEAN
© Encyclopædia Britannica, Inc.

"I want it all"

"I want it all"



More transparency

- Merging changes to DC (the shared/OS-agnostic part)
  - Change limitations in some part of the code
  - Workarounds to attend DRM/KMS framework
  - DC implementations with no Linux support yet
  - Public CI (?)

- Upstream to DRM new features implementation

- More documentation is always welcome

Complaints? Questions?