DRM Internals

This chapter documents DRM internals relevant to driver authors and developers working to add support for the latest features to existing drivers.

First, we go over some typical driver initialization requirements, like setting up command buffers, creating an initial output configuration, and initializing core services. Subsequent sections cover core internals in more detail, providing implementation notes and examples.

The DRM layer provides several services to graphics drivers, many of them driven by the application interfaces it provides through libdrm, the library that wraps most of the DRM ioctls. These include vblank event handling, memory management, output management, framebuffer management, command submission & fencing, suspend/resume support, and DMA services.

Driver Initialization

At the core of every DRM driver is a :c:type:`struct drm_driver <drm_driver>` structure. Drivers typically statically initialize a drm_driver structure, and then pass it to drm_dev_alloc() to allocate a device instance. After the device instance is fully initialized it can be registered (which makes it accessible from userspace) using drm_dev_register().

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\gpu\(linux-master)(Documentation)(gpu)drm-internals.rst`, **line 24**); *backlink*
>
> Unknown interpreted text role "c:type".

The :c:type:`struct drm_driver <drm_driver>` structure contains static information that describes the driver and features it supports, and pointers to methods that the DRM core will call to implement the DRM API. We will first go through the :c:type:`struct drm_driver <drm_driver>` static information fields, and will then describe individual operations in details as they get used in later sections.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\gpu\(linux-master)(Documentation)(gpu)drm-internals.rst`, **line 31**); *backlink*
>
> Unknown interpreted text role "c:type".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\gpu\(linux-master)(Documentation)(gpu)drm-internals.rst`, **line 31**); *backlink*
>
> Unknown interpreted text role "c:type".

Driver Information

Major, Minor and Patchlevel

int major; int minor; int patchlevel; The DRM core identifies driver versions by a major, minor and patch level triplet. The information is printed to the kernel log at initialization time and passed to userspace through the DRM_IOCTL_VERSION ioctl.

The major and minor numbers are also used to verify the requested driver API version passed to DRM_IOCTL_SET_VERSION. When the driver API changes between minor versions, applications can call DRM_IOCTL_SET_VERSION to select a specific version of the API. If the requested major isn't equal to the driver major, or the requested minor is larger than the driver minor, the DRM_IOCTL_SET_VERSION call will return an error. Otherwise the driver's set_version() method will be called with the requested version.

Name, Description and Date

char *name; char *desc; char *date; The driver name is printed to the kernel log at initialization time, used for IRQ registration and passed to userspace through DRM_IOCTL_VERSION.

The driver description is a purely informative string passed to userspace through the DRM_IOCTL_VERSION ioctl and otherwise unused by the kernel.

The driver date, formatted as YYYYMMDD, is meant to identify the date of the latest modification to the driver. However, as most drivers fail to update it, its value is mostly useless. The DRM core prints it to the kernel log at initialization time and passes it to userspace through the DRM_IOCTL_VERSION ioctl.

Module Initialization

Module Initialization

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\gpu\(linux-master)(Documentation)(gpu)drm-internals.rst, line 81)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: include/drm/drm_module.h
   :doc: overview
```

Managing Ownership of the Framebuffer Aperture

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\gpu\(linux-master)(Documentation)(gpu)drm-internals.rst, line 87)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: drivers/gpu/drm/drm_aperture.c
   :doc: overview
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\gpu\(linux-master)(Documentation)(gpu)drm-internals.rst, line 90)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: include/drm/drm_aperture.h
   :internal:
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\gpu\(linux-master)(Documentation)(gpu)drm-internals.rst, line 93)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: drivers/gpu/drm/drm_aperture.c
   :export:
```

Device Instance and Driver Handling

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\gpu\(linux-master)(Documentation)(gpu)drm-internals.rst, line 99)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: drivers/gpu/drm/drm_drv.c
   :doc: driver instance overview
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\gpu\(linux-master)(Documentation)(gpu)drm-internals.rst, line 102)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: include/drm/drm_device.h
   :internal:
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\gpu\(linux-master)(Documentation)(gpu)drm-internals.rst, line 105)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: include/drm/drm_drv.h
   :internal:
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\gpu\(linux-master)(Documentation)(gpu)drm-internals.rst, line 108)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: drivers/gpu/drm/drm_drv.c
   :export:
```

Driver Load

Component Helper Usage

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\gpu\(linux-master)(Documentation)(gpu)drm-internals.rst, line 117)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: drivers/gpu/drm/drm_drv.c
   :doc: component helper usage recommendations
```

Memory Manager Initialization

Every DRM driver requires a memory manager which must be initialized at load time. DRM currently contains two memory managers, the Translation Table Manager (TTM) and the Graphics Execution Manager (GEM). This document describes the use of the GEM memory manager only. See ? for details.

Miscellaneous Device Configuration

Another task that may be necessary for PCI devices during configuration is mapping the video BIOS. On many devices, the VBIOS describes device configuration, LCD panel timings (if any), and contains flags indicating device state. Mapping the BIOS can be done using the pci_map_rom() call, a convenience function that takes care of mapping the actual ROM, whether it has been shadowed into memory (typically at address 0xc0000) or exists on the PCI device in the ROM BAR. Note that after the ROM has been mapped and any necessary information has been extracted, it should be unmapped; on many devices, the ROM address decoder is shared with other BARs, so leaving it mapped could cause undesired behaviour like hangs or memory corruption.

Managed Resources

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\gpu\(linux-master)(Documentation)(gpu)drm-internals.rst, line 147)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: drivers/gpu/drm/drm_managed.c
   :doc: managed resources
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\gpu\(linux-master)(Documentation)(gpu)drm-internals.rst, line 150)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: drivers/gpu/drm/drm_managed.c
   :export:
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\gpu\(linux-master)(Documentation)(gpu)drm-internals.rst, line 153)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: include/drm/drm_managed.h
   :internal:
```

Bus-specific Device Registration and PCI Support

A number of functions are provided to help with device registration. The functions deal with PCI and platform devices respectively and are only provided for historical reasons. These are all deprecated and shouldn't be used in new drivers. Besides that there's a few helpers for pci drivers.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\gpu\(linux-master)(Documentation)(gpu)drm-internals.rst, line 165)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: drivers/gpu/drm/drm_pci.c
   :export:
```

Open/Close, File Operations and IOCTLs

File Operations

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\gpu\(linux-master)(Documentation)(gpu)drm-internals.rst, line 176)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: drivers/gpu/drm/drm_file.c
   :doc: file operations
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\gpu\(linux-master)(Documentation)(gpu)drm-internals.rst, line 179)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: include/drm/drm_file.h
   :internal:
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\gpu\(linux-master)(Documentation)(gpu)drm-internals.rst, line 182)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: drivers/gpu/drm/drm_file.c
   :export:
```

Misc Utilities

Printer

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\gpu\(linux-master)(Documentation)(gpu)drm-internals.rst, line 191)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: include/drm/drm_print.h
   :doc: print
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\gpu\(linux-master)(Documentation)(gpu)drm-internals.rst, line 194)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: include/drm/drm_print.h
   :internal:
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\gpu\(linux-master)(Documentation)(gpu)drm-internals.rst, line 197)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: drivers/gpu/drm/drm_print.c
   :export:
```

Utilities

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\gpu\(linux-master)(Documentation)(gpu)drm-internals.rst, line 203)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: include/drm/drm_util.h
   :doc: drm utils
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\gpu\(linux-master)(Documentation)(gpu)drm-internals.rst, line 206)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: include/drm/drm_util.h
   :internal:
```

Legacy Support Code

The section very briefly covers some of the old legacy support code which is only used by old DRM drivers which have done a so-called shadow-attach to the underlying device instead of registering as a real driver. This also includes some of the old generic buffer management and command submission code. Do not use any of this in new and modern drivers.

Legacy Suspend/Resume

The DRM core provides some suspend/resume code, but drivers wanting full suspend/resume support should provide save() and restore() functions. These are called at suspend, hibernate, or resume time, and should perform any state save or restore required by your device across suspend or hibernate states.

int (*suspend) (struct drm_device *, pm_message_t state); int (*resume) (struct drm_device *); Those are legacy suspend and resume methods which *only* work with the legacy shadow-attach driver registration functions. New driver should use the power management interface provided by their bus type (usually through the :c:type:`struct device_driver <device_driver>` dev_pm_ops) and set these methods to NULL.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\gpu\(linux-master)(Documentation)(gpu)drm-internals.rst, line 229); *backlink*

Unknown interpreted text role "c:type".

Legacy DMA Services

This should cover how DMA mapping etc. is supported by the core. These functions are deprecated and should not be used.