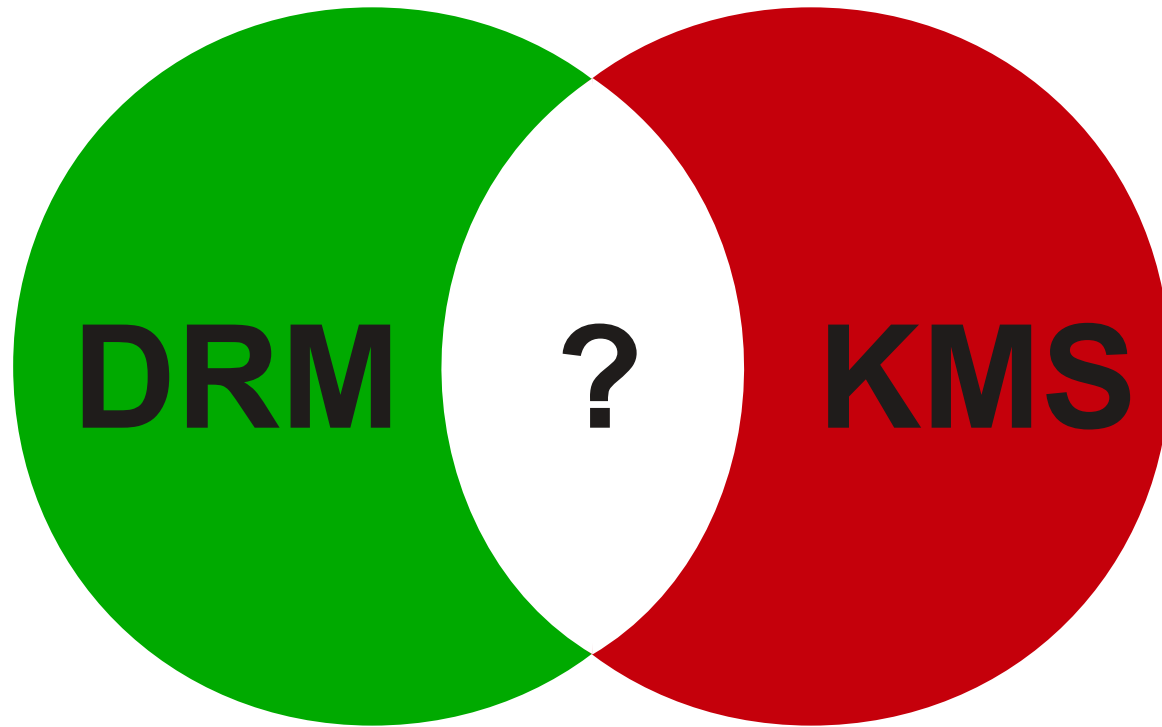


Anatomy of an Embedded KMS Driver

Embedded Linux Conference 2013
San Francisco – 2013/02/20

Laurent Pinchart
laurent.pinchart@ideasonboard.com



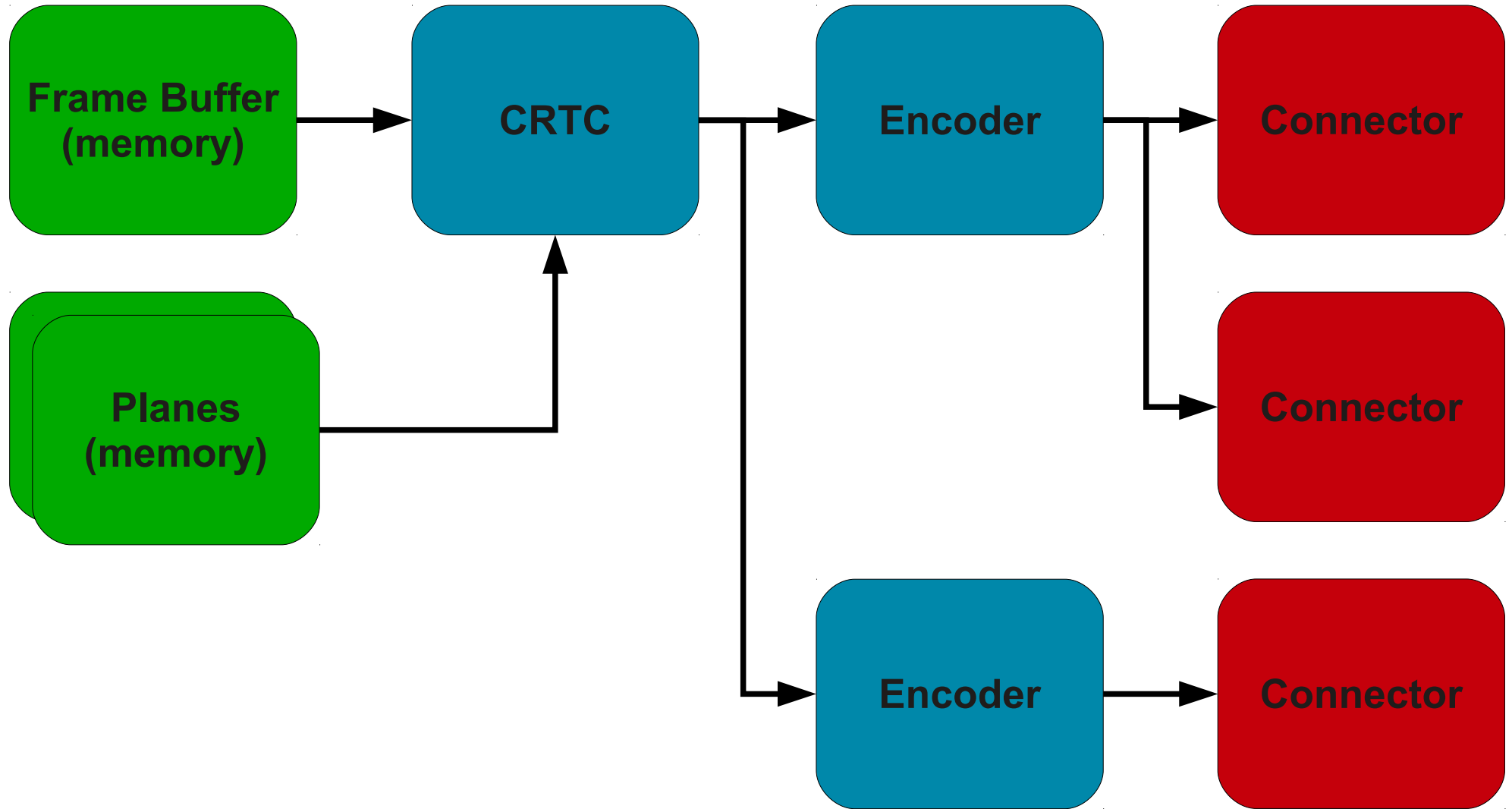
APIs

- Memory Management
- Vertical Blanking
- Version, Authentication, Master, ...

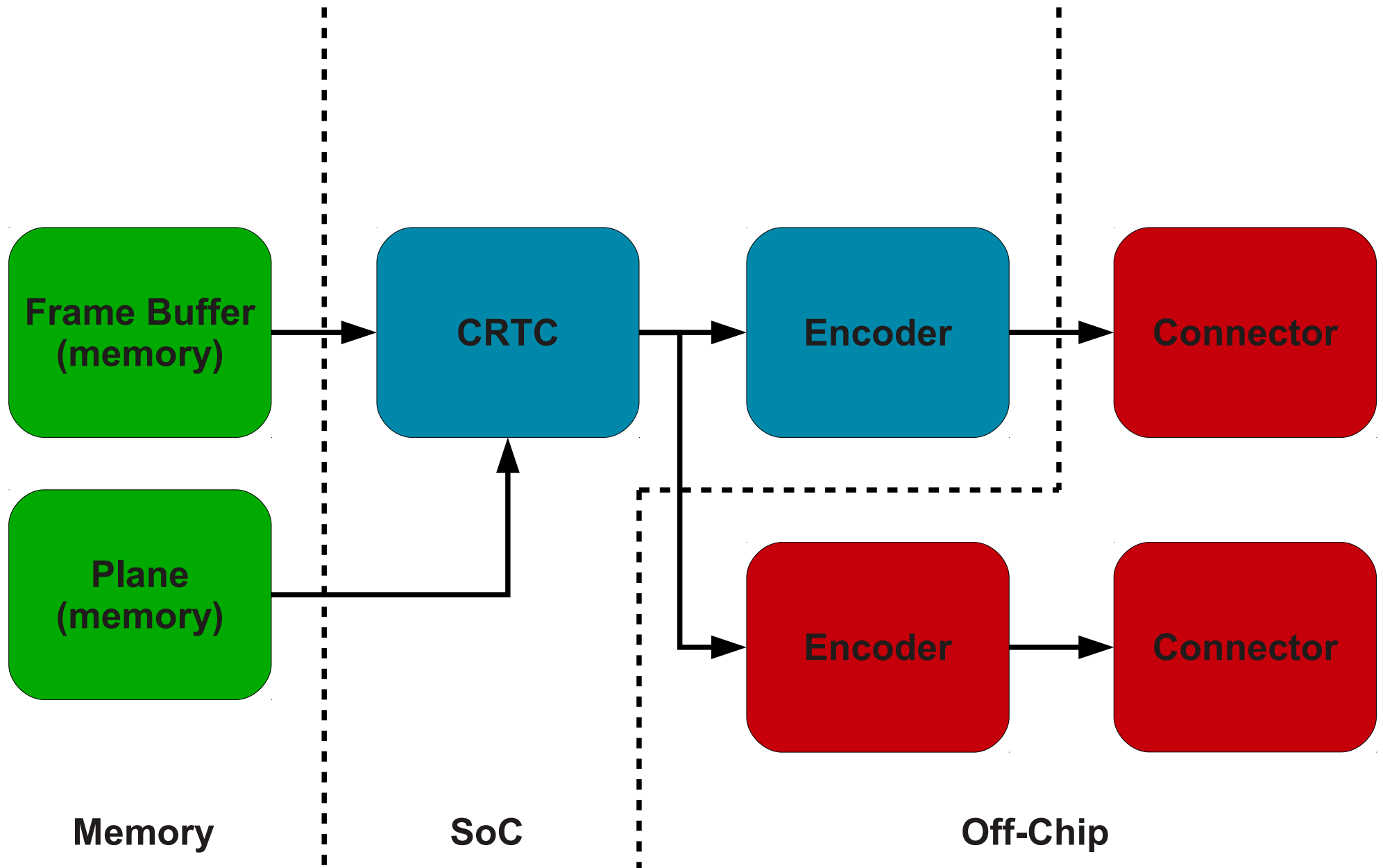


DRM

- Device Model
- Frame Buffer
- Modes
- Page Flip
- Planes
- Cursor, Gamma, ...



Device Model



Device Model - SoC

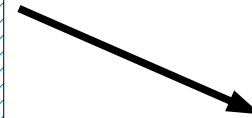


Frame Buffer



KMS – Scanout

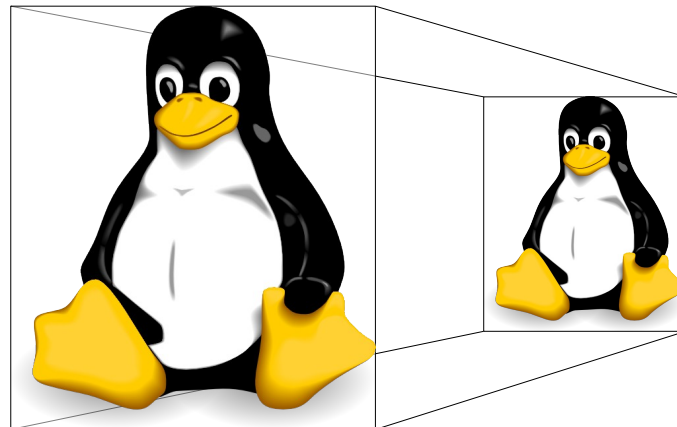
CRTC



Composition

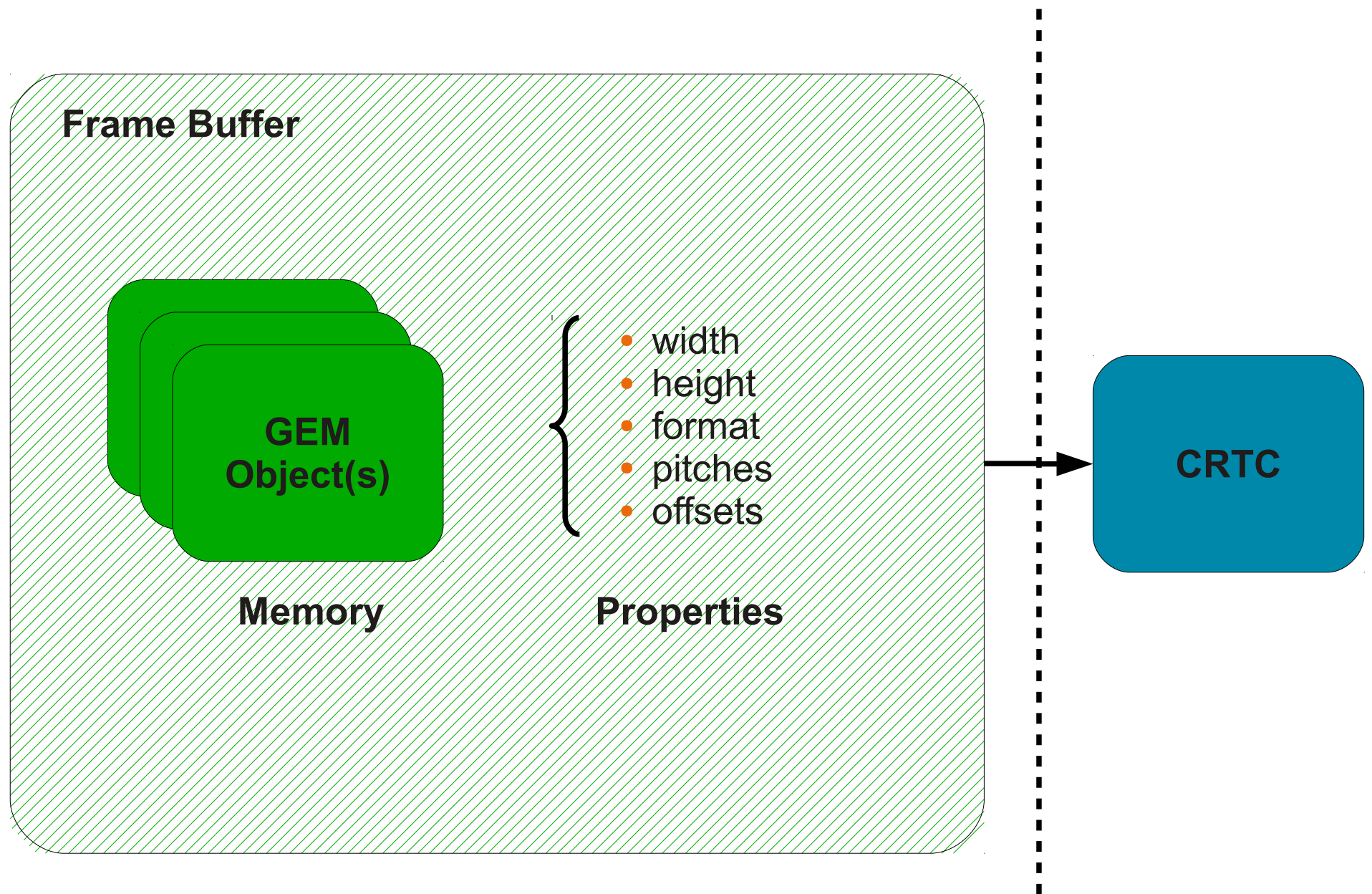


Plane(s)

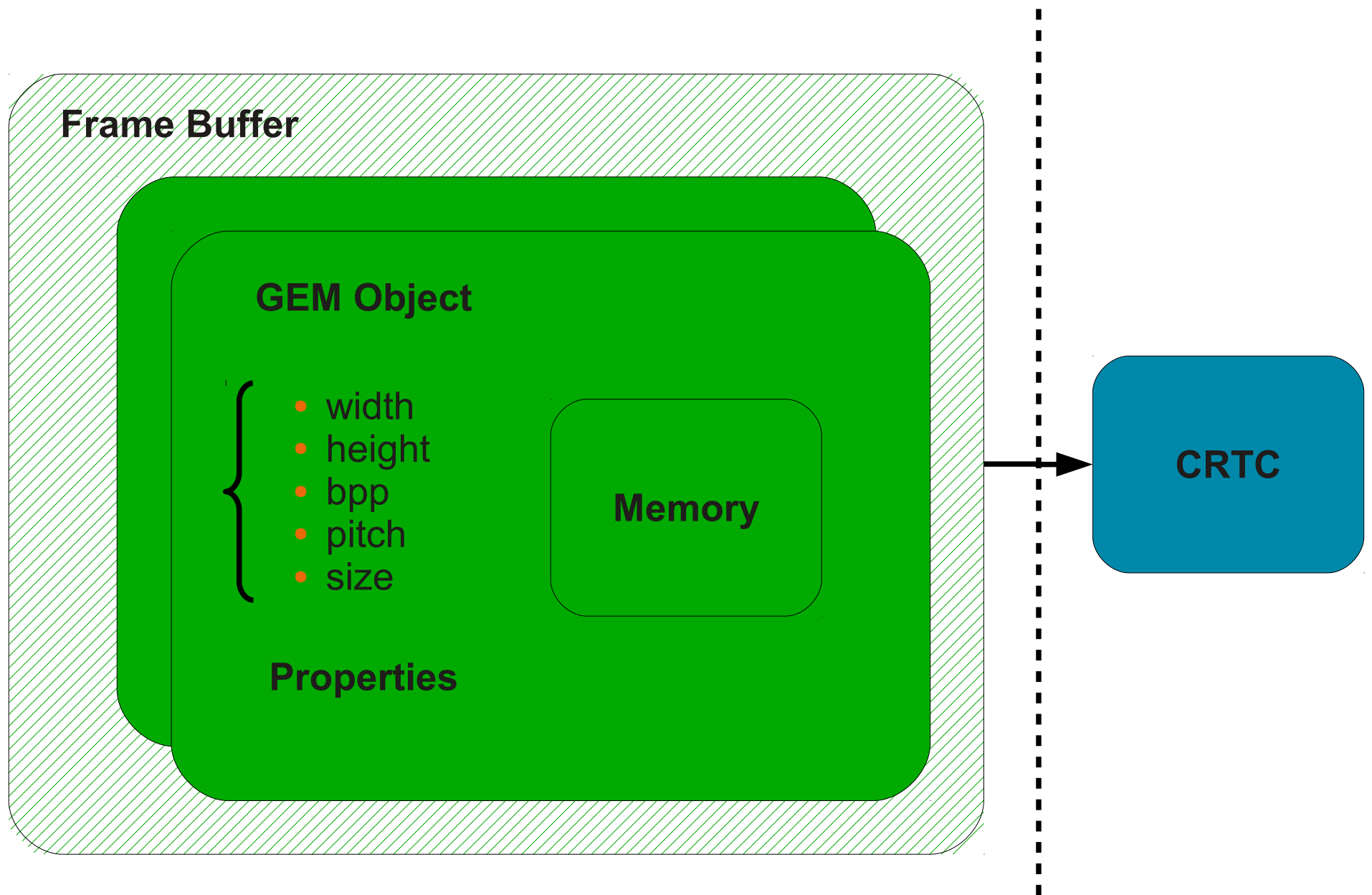


KMS – Composition

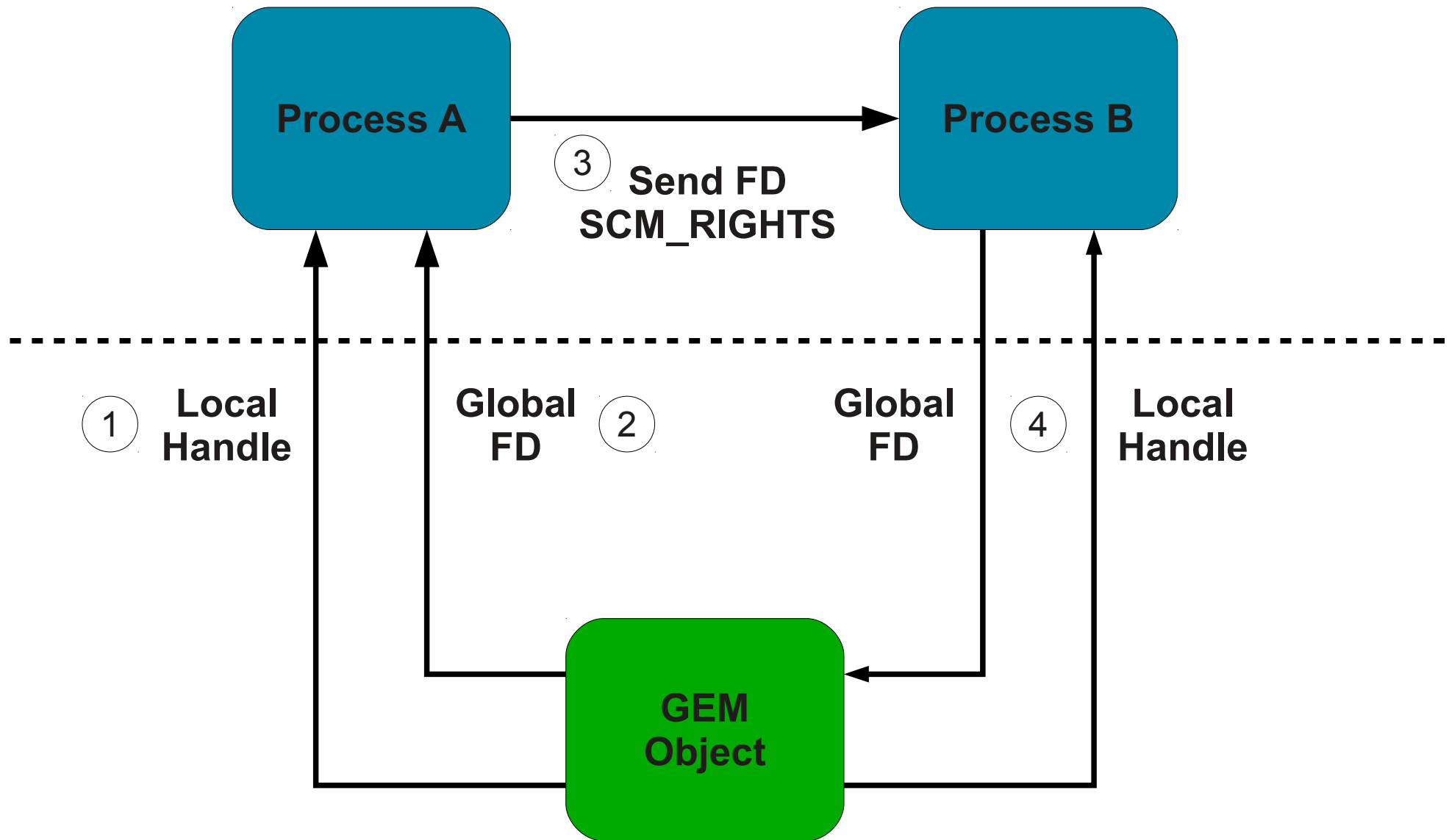




KMS – Frame Buffer

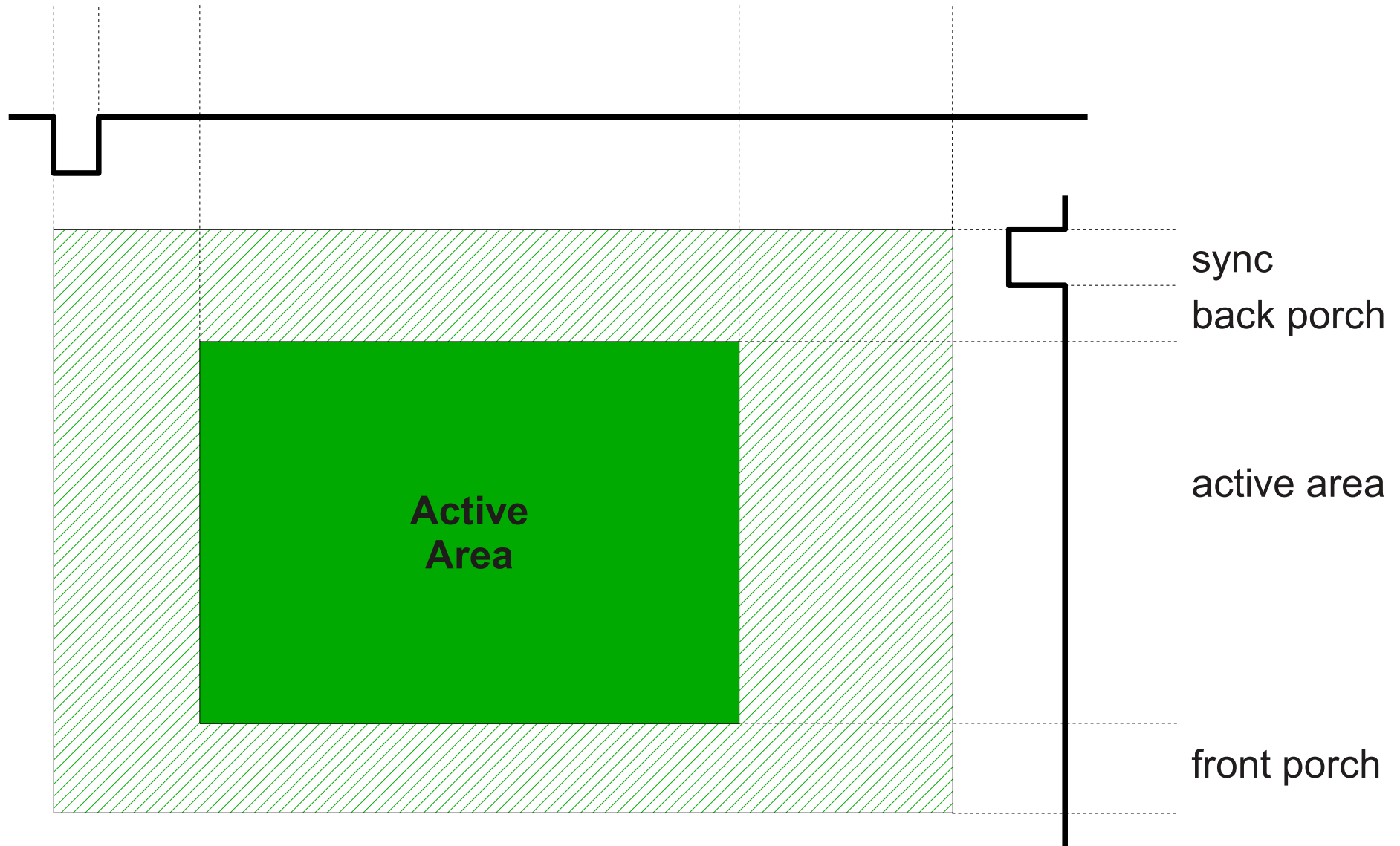


DRM/KMS – GEM Object

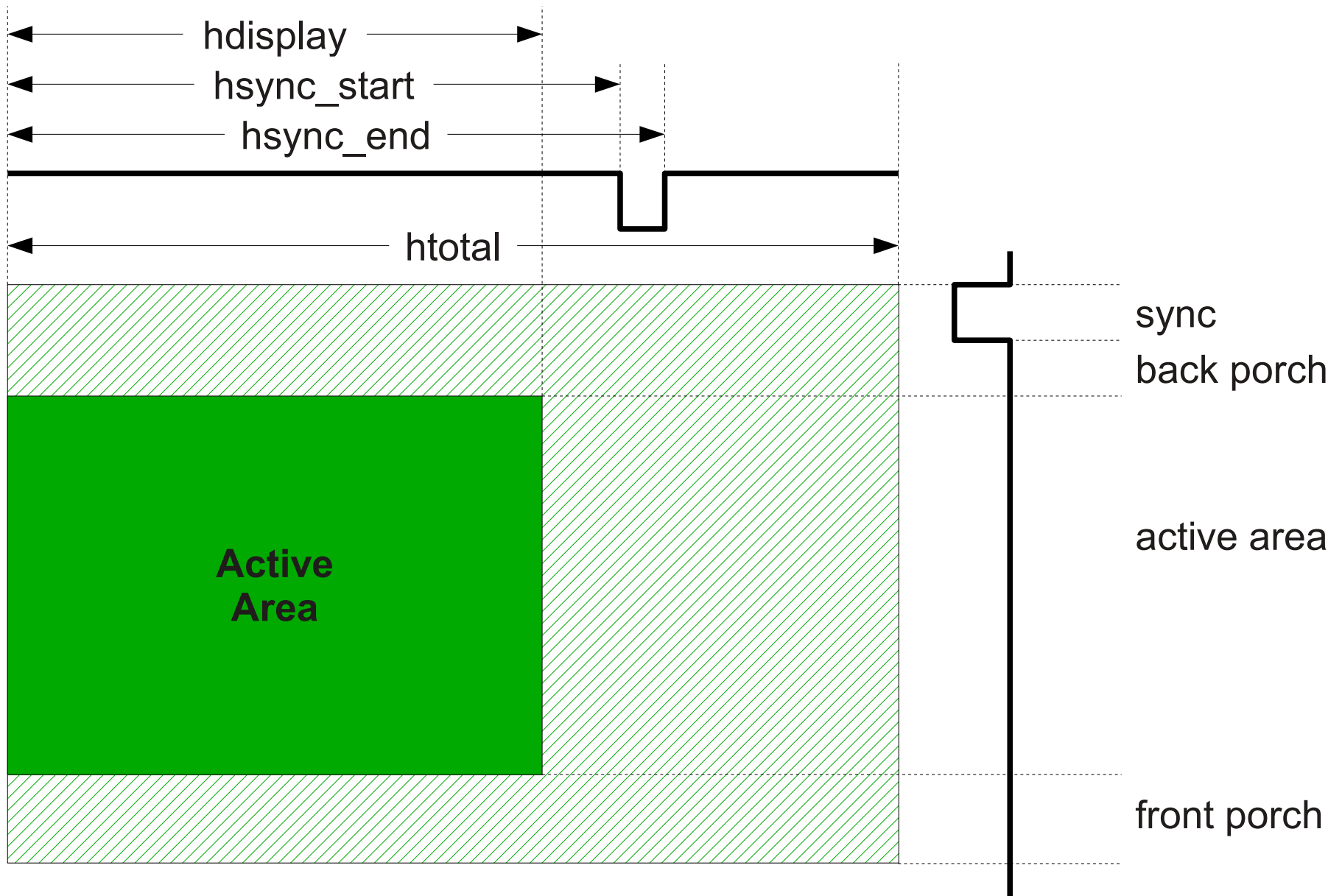


DRM – Handles

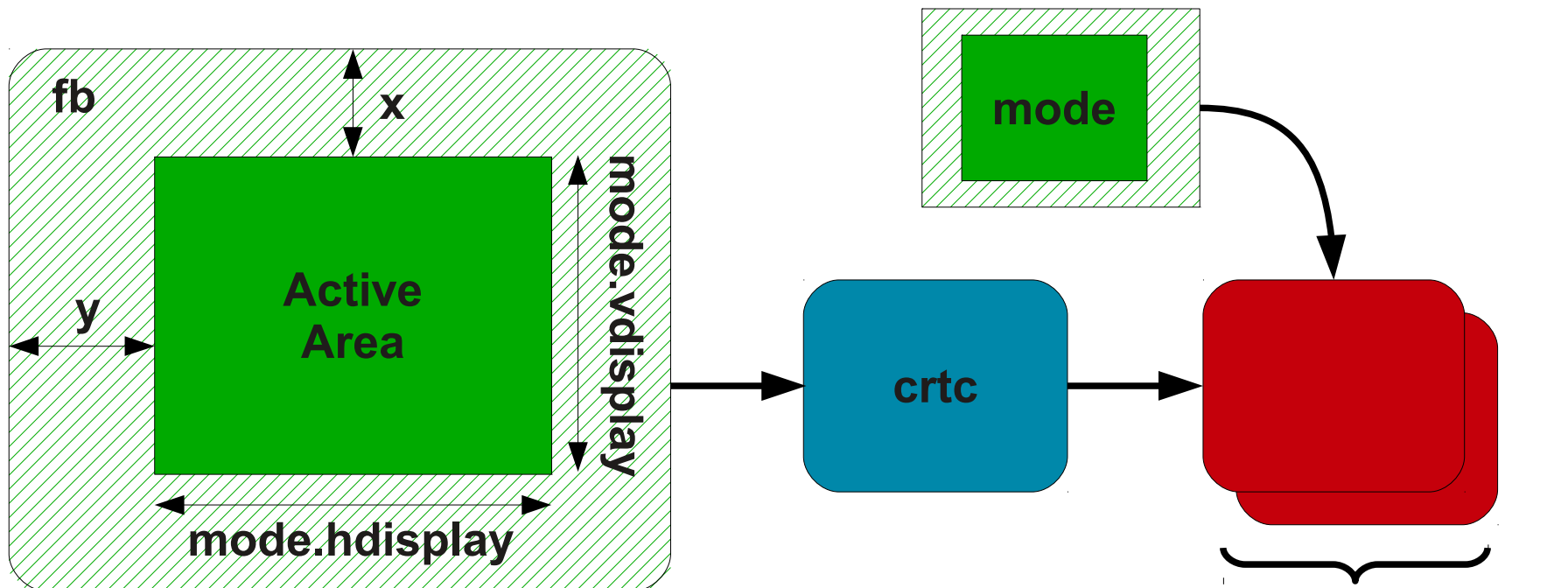




KMS – Modes (1/2)



KMS – Modes (2/2)



```
struct drm_mode_set {
    struct drm_framebuffer *fb;
    struct drm_crtc *crtc;
    struct drm_display_mode *mode;
    uint32_t x;
    uint32_t y;
    struct drm_connector **connectors;
    size_t num_connectors;
};
```

KMS – Mode Setting



Documentation/ DocBook/drm.tmpl

Please contribute



Documentation

Code Ahead

Locking and error
handling omitted for
readability



Disclaimer

drm_* = core
rcar_du_* = driver



Cheat Sheet

```
struct drm_driver rcar_du_driver = {  
    ...  
};  
  
int rcar_du_probe(struct platform_device *pdev)  
{  
    return drm_platform_init(&rcar_du_driver,  
                             pdev);  
}  
  
int rcar_du_remove(struct platform_device *pdev)  
{  
    drm_platform_exit(&rcar_du_driver, pdev);  
    return 0;  
}
```



Driver Initialization

```
struct drm_driver rcar_du_driver = {  
    .driver_features =  
        DRIVER_HAVE_IRQ | DRIVER_GEM |  
        DRIVER_MODESET | DRIVER_PRIME,  
    .name    = "rcar-du",  
    .desc    = "Renesas R-Car Display Unit",  
    .date    = "20130110",  
    .major   = 1,  
    .minor   = 0,  
    .patchlevel = 0,  
    ...  
};
```



Driver Information

```
struct file_operations rcar_du_fops = {  
    .owner          = THIS_MODULE,  
    .open           = drm_open,  
    .release        = drm_release,  
    .unlocked_ioctl = drm_ioctl,  
    .compat_ioctl   = drm_compat_ioctl,  
    .poll           = drm_poll,  
    .read           = drm_read,  
    .fasync         = drm_fasync,  
    .llseek         = no_llseek,  
    .mmap           = ...,  
};
```

```
struct drm_driver rcar_du_driver = {  
    .fops           = &rcar_du_fops,  
};
```



File Operations

```
int rcar_du_load(struct drm_device *dev,
                 unsigned long flags)
{
    struct platform_device *pdev =
        dev->platformdev;
    struct rcar_du_device *rcdu;

    rcdu = devm_kzalloc(&pdev->dev,
                       sizeof(*rcdu), GFP_KERNEL);
    dev->dev_private = rcdu;

    /* Memory, clocks, regulators, ... */
}

struct drm_driver rcar_du_driver = {
    .load = rcar_du_load,
};
```



Driver Load

```
int rcar_du_load(struct drm_device *dev,
                 unsigned long flags)
{
    ...
    drm_irq_install(dev);
    /* Behind the scene:
     * request_irq(platform_get_irq(..., 0))
     */
    ...
}

struct drm_driver rcar_du_driver = {
    /* .irq_preinstall */
    .irq_handler = rcar_du_irq,
    /* .irq_postinstall */
};
```



IRQ Installation

```
struct drm_mode_config_funcs rcar_du_modecfg_funcs =
{
    .fb_create = ...,
};

int rcar_du_load(struct drm_device *dev,
                unsigned long flags)
{
    ...
    drm_mode_config_init(dev);

    dev->mode_config.min_width = 0;
    dev->mode_config.min_height = 0;
    dev->mode_config.max_width = 4095;
    dev->mode_config.max_height = 2047;
    dev->mode_config.funcs =
        &rcar_du_modecfg_funcs;
    ...
}
```



Mode Config

```
struct file_operations rcar_du_fops = {  
    .mmap                = drm_gem_cma_mmap,  
};  
  
struct drm_driver rcar_du_driver = {  
    .gem_free_object      = drm_gem_cma_free_object,  
    .gem_vm_ops           = &drm_gem_cma_vm_ops,  
};
```



GEM


```
struct drm_driver rcar_du_driver = {  
    .dumb_create      = drm_gem_cma_dumb_create,  
    .dumb_map_offset  = drm_gem_cma_dumb_map_offset,  
    .dumb_destroy     = drm_gem_cma_dumb_destroy,  
};
```



GEM – Dumb Objects

```
struct drm_driver rcar_du_driver = {  
    .prime_handle_to_fd = drm_gem_prime_handle_to_fd,  
    .prime_fd_to_handle = drm_gem_prime_fd_to_handle,  
    .gem_prime_import    = drm_gem_cma_dmabuf_import,  
    .gem_prime_export    = drm_gem_cma_dmabuf_export,  
};
```



GEM – PRIME

```

drm_framebuffer *
rcar_du_fb_create(struct drm_device *dev,
                  struct drm_file *file_priv,
                  struct drm_mode_fb_cmd2 *mode_cmd)
{
    /* Validate the pixel format, size and pitches */
    ...
    return drm_fb_cma_create(dev, file_priv,
                             mode_cmd);
}

struct drm_mode_config_funcs rcar_du_modecfg_funcs =
{
    .fb_create = rcar_du_fb_create,
};

```



Frame Buffer

```
struct drm_crtc_funcs crtc_funcs = {  
    .destroy = drm_crtc_cleanup,  
    .set_config = ...,  
    .page_flip = ...,  
};  
  
int rcar_du_load(struct drm_device *dev,  
                unsigned long flags)  
{  
    struct drm_crtc *crtc;  
    ...  
    drm_crtc_init(dev, crtc, &crtc_funcs);  
    ...  
}
```



CRTC

```
struct drm_encoder_funcs encoder_funcs = {  
    .destroy = drm_encoder_cleanup,  
};  
  
int rcar_du_load(struct drm_device *dev,  
                unsigned long flags)  
{  
    struct drm_encoder *encoder;  
    ...  
    encoder->possible_crtcs = 1 << crtc;  
    drm_encoder_init(dev, encoder, &encoder_funcs,  
                    DRM_MODE_ENCODER_DAC);  
    ...  
}
```



Encoder

```

struct drm_connector_funcs connector_funcs = {
    ...
};

int rcau_load(struct drm_device *dev,
               unsigned long flags)
{
    struct drm_connector *connector;
    ...
    connector->display_info.width_mm = ...;
    connector->display_info.height_mm = ...;

    drm_connector_init(dev, connector,
                       &connector_funcs, DRM_MODE_CONNECTOR_VGA);
    drm_sysfs_connector_add(connector);
    drm_mode_connector_attach_encoder(connector,
                                       encoder);
}

```



Connector

```
struct drm_crtc_funcs crtc_funcs = {  
    .set_config = ...,  
};  
  
int (*set_config)(struct drm_mode_set *set);  
  
struct drm_mode_set {  
    struct drm_framebuffer *fb;  
    struct drm_crtc *crtc;  
    struct drm_display_mode *mode;  
    uint32_t x;  
    uint32_t y;  
    struct drm_connector **connectors;  
    size_t num_connectors;  
};
```



Mode Setting (1/4)

```
struct drm_crtc_funcs crtc_funcs = {  
    .set_config = drm_crtc_helper_set_config,  
};
```

```
int rcar_du_load(struct drm_device *dev,  
                unsigned long flags)  
{  
    ...  
    drm_crtc_helper_add(crtc, &crtc_helper_funcs);  
    drm_connector_helper_add(connector,  
                             &connector_helper_funcs);  
    drm_encoder_helper_add(encoder,  
                           &encoder_helper_funcs);  
    ...  
}
```



Mode Setting (2/4)


```
struct drm_crtc_helper_funcs crtc_helper_funcs = {  
    .mode_fixup = rcar_du_crtc_mode_fixup,  
    .prepare = rcar_du_crtc_mode_prepare,  
    .commit = rcar_du_crtc_mode_commit,  
    .mode_set = rcar_du_crtc_mode_set,  
    .mode_set_base = rcar_du_crtc_mode_set_base,  
    .disable = rcar_du_crtc_disable,  
};
```



Mode Setting (3/4)

```
struct drm_encoder_helper_funcs encoder_helper_funcs = {  
    .mode_fixup = rcar_du_vga_encoder_mode_fixup,  
    .prepare = rcar_du_vga_encoder_mode_prepare,  
    .commit = rcar_du_vga_encoder_mode_commit,  
    .mode_set = rcar_du_vga_encoder_mode_set,  
};  
  
struct drm_connector_helper_funcs connector_helper_funcs =  
{  
    .best_encoder = rcar_du_vga_connector_best_encoder,  
};
```



Mode Setting (4/4)

```
struct drm_connector_funcs connector_funcs = {  
    .fill_modes = ...,  
};  
  
int (*fill_modes)(struct drm_connector *connector,  
    uint32_t max_width, uint32_t max_height);
```



Modes Discovery (1/2)

```
struct drm_connector_funcs connector_funcs = {  
    .fill_modes = drm_helper_probe_single_connector_modes,  
};
```

Core

```
struct drm_connector_helper_funcs connector_helper_funcs =  
{  
    .get_modes = rcar_du_vga_connector_get_modes,  
    .mode_valid = rcar_du_vga_connector_mode_valid,  
};
```

Helpers



Modes Discovery (2/2)

```
struct drm_connector_funcs connector_funcs = {  
    .dpms = ...,  
};
```

```
void (*dpms)(struct drm_connector *connector,  
             int mode);
```

```
#define DRM_MODE_DPMS_ON          0  
#define DRM_MODE_DPMS_STANDBY    1  
#define DRM_MODE_DPMS_SUSPEND    2  
#define DRM_MODE_DPMS_OFF        3
```



DPMS (1/2)

```
struct drm_connector_funcs connector_funcs = {  
    .dpms = drm_helper_connector_dpms,  
};
```

```
void rcar_du_crtc_dpms(struct drm_crtc *crtc, int mode)  
{  
    ...  
}
```

```
struct drm_crtc_helper_funcs crtc_helper_funcs = {  
    .dpms = rcar_du_crtc_dpms,  
};
```

```
void rcar_du_vga_encoder_dpms(struct drm_encoder *encoder,  
                               int mode)  
{  
    ...  
}
```

```
struct drm_encoder_helper_funcs encoder_helper_funcs = {  
    .dpms = rcar_du_vga_encoder_dpms,  
};
```



DPMS (2/2)

```
int rcar_du_load(struct drm_device *dev,  
                unsigned long flags)  
{  
    ...  
    drm_vblank_init(dev, 1);  
    ...  
}  
  
irqreturn_t rcar_du_irq(int irq, void *arg)  
{  
    struct drm_device *dev = arg;  
  
    drm_handle_vblank(dev, 0);  
}
```



Vertical Blanking (1/2)

```
int rcar_du_enable_vblank(struct drm_device *dev,
                          int crtc)
{
    /* Enable the vblank interrupt for the CRTC */
    return 0;
}

void rcar_du_disable_vblank(struct drm_device *dev,
                            int crtc)
{
    /* Disable the vblank interrupt for the CRTC */
}

struct drm_driver rcar_du_driver = {
    .get_vblank_counter = drm_vblank_count,
    .enable_vblank      = rcar_du_enable_vblank,
    .disable_vblank     = rcar_du_disable_vblank,
};
```



Vertical Blanking (2/2)


```
int rcar_du_crtc_page_flip(struct drm_crtc *crtc,
                           struct drm_framebuffer *fb,
                           struct drm_pending_vblank_event *event)
{
    struct rcar_du_crtc *rcrtc = to_rcar_crtc(crtc);

    if (rcrtc->event != NULL)
        return -EBUSY;

    crtc->fb = fb;
    rcar_du_crtc_update_base(rcrtc);

    if (event) {
        event->pipe = 0;
        rcrtc->event = event;
        drm_vblank_get(crtc->dev, 0);
    }
    return 0;
}
```



Page Flipping (1/2)

```
void rcar_du_crtc_finish_page_flip(struct rcar_du_crtc *rcrtc)
{
    struct drm_pending_vblank_event *event;
    struct timeval vblanktime;

    event = rcrtc->event;
    rcrtc->event = NULL;
    if (event == NULL)
        return;

    event->event.sequence =
        drm_vblank_count_and_time(dev, 0, &vblanktime);
    event->event.tv_sec = time.tv_sec;
    event->event.tv_usec = time.tv_usec;

    list_add_tail(&event->base.link,
                  &event->base.file_priv->event_list);
    wake_up_interruptible(&event->base.file_priv->event_wait);

    drm_vblank_put(dev, 0);
}
```



Page Flipping (2/2)

```
struct drm_plane_funcs rcar_du_plane_funcs = {  
    .update_plane = rcar_du_plane_update,  
    .disable_plane = rcar_du_plane_disable,  
    .destroy = drm_plane_cleanup,  
};  
  
uint32_t formats[] = {  
    DRM_FORMAT_RGB565, ...  
};  
  
int rcar_du_load(struct drm_device *dev,  
                unsigned long flags)  
{  
    struct drm_plane *plane;  
    ...  
    drm_plane_init(dev, plane, 1,  
                   &rcar_du_plane_funcs, formats,  
                   ARRAY_SIZE(formats), false);  
}
```



Planes (1/2)

```
int rcar_du_plane_update(struct drm_plane *plane,  
    struct drm_crtc *crtc,  
    struct drm_framebuffer *fb,  
    int crtc_x, int crtc_y,  
    unsigned int crtc_w, unsigned int crtc_h,  
    uint32_t src_x, uint32_t src_y,  
    uint32_t src_w, uint32_t src_h)  
{  
    ...  
}  
  
int rcar_du_plane_disable(struct drm_plane *plane)  
{  
    ...  
}
```



Planes (2/2)

- Properties
- FBDEV Emulation
- Connector Status Poll
- Atomic Page Flip
- ...



And Much More...



Source: <http://www.flickr.com/photos/buckarooobay/3721809183/>

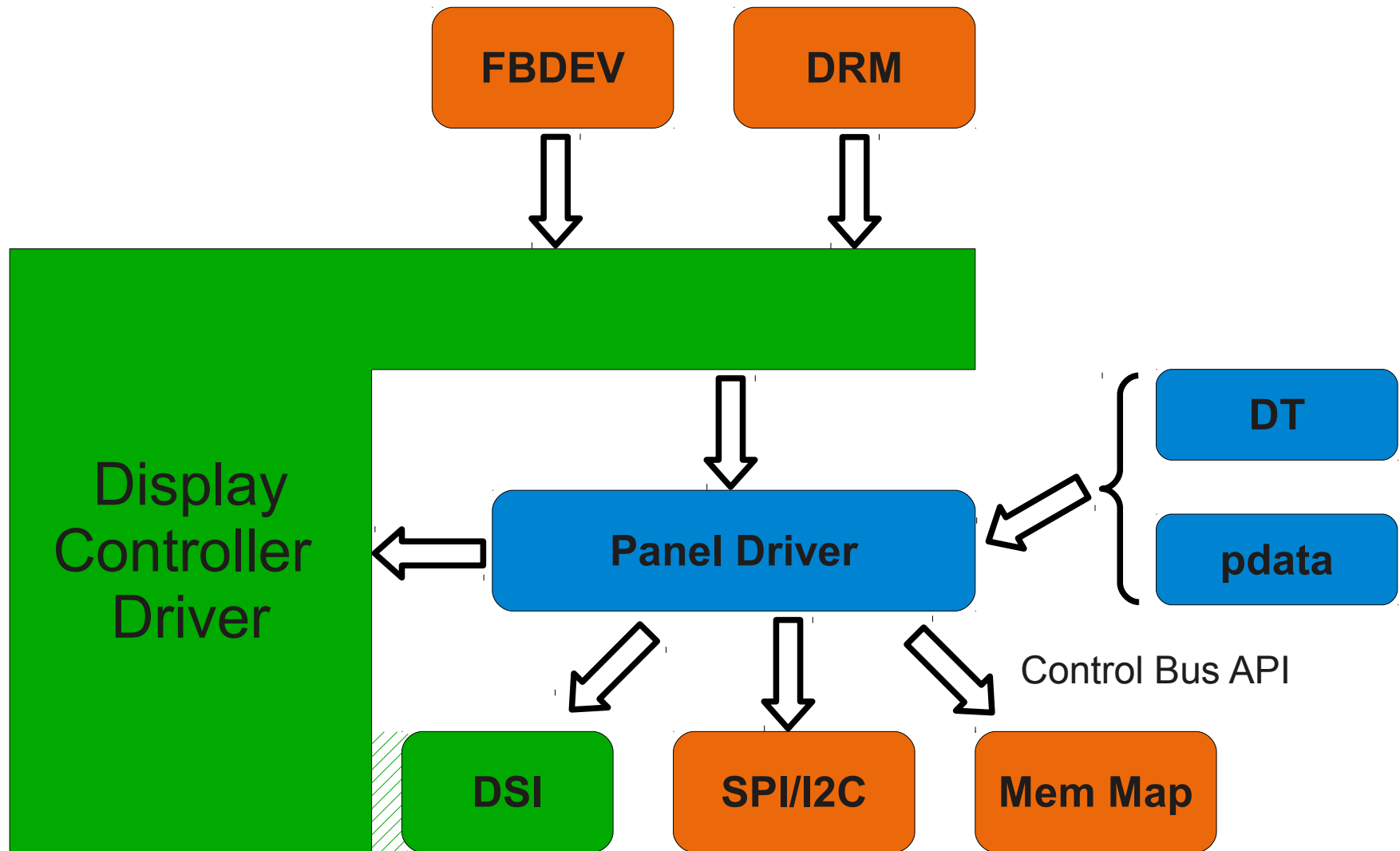
Generic ~~Panel~~ Display Framework

<http://lwn.net/Articles/512363/>

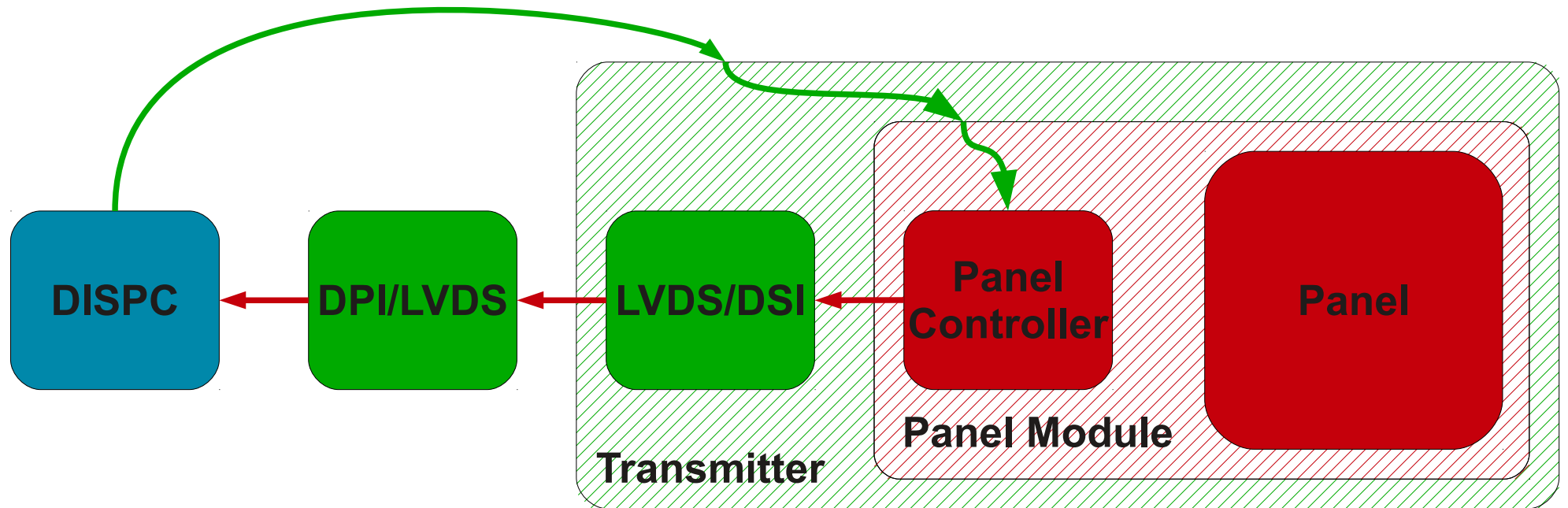
! BoF session tomorrow @4PM !



WIP – Display Framework



WIP – Display Framework



WIP – Display Framework

- 
- dri-devel@listsfreedesktop.org
 - laurent.pinchart@ideasonboard.com



Contact

?

!

thx.

