Why and How to use KMS as Your Userspace Display API of Choice

LinuxCon Japan 2013
Tokyo – 2013/05/31

Laurent Pinchart
laurent.pinchart@ideasonboard.com

IDEAS ON BOARD

Features

Source: http://xkcd.com/619/

	DRM	FB
Dynamic Allocation	Yes	No
Multiple Buffers	Yes	panning
Import	dmabuf	No
Export	dmabuf mmap	mmap

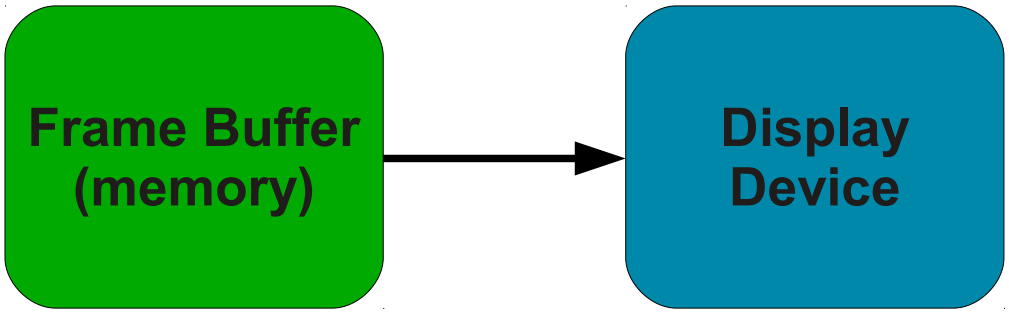
Memory Management

	DRM	FB
Formats	4CC	RGB 4CC
Enumeration	Planes	No
Negotiation	No	No
Atomicity	Yes	No

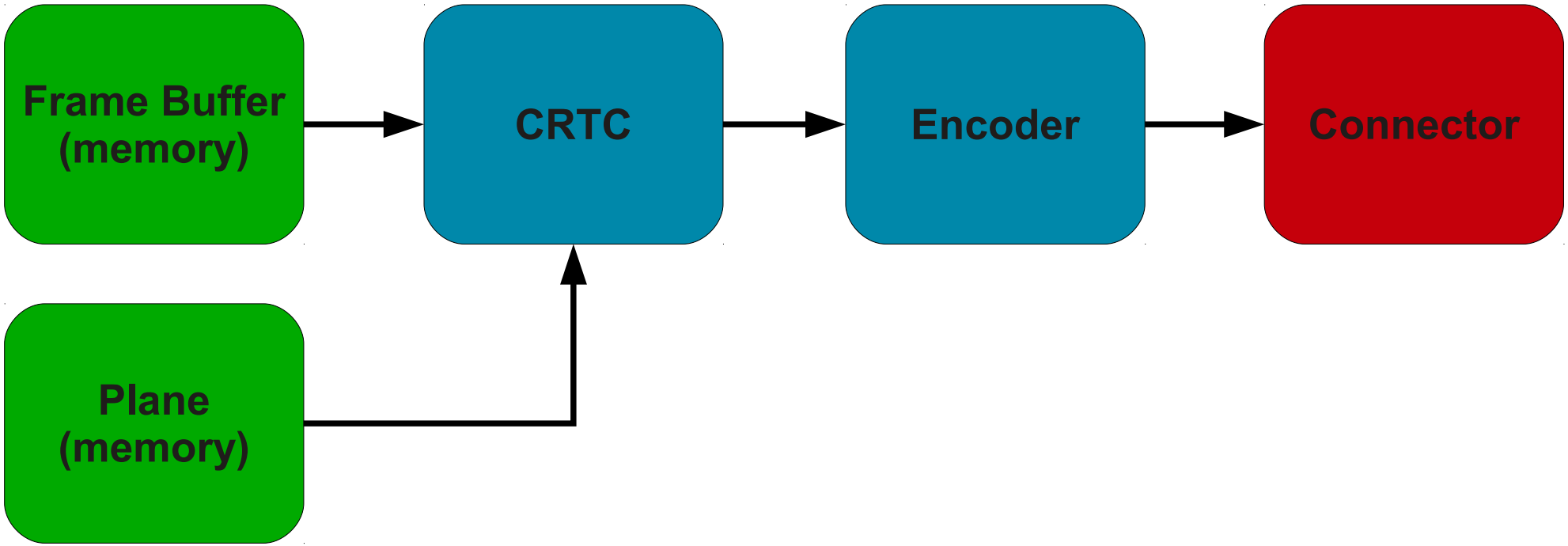
Mode Setting

	DRM	FB
Overlays	Yes	No
Rotation	Yes	No
Scaling	Yes	No
Cropping/Panning	Yes	Yes

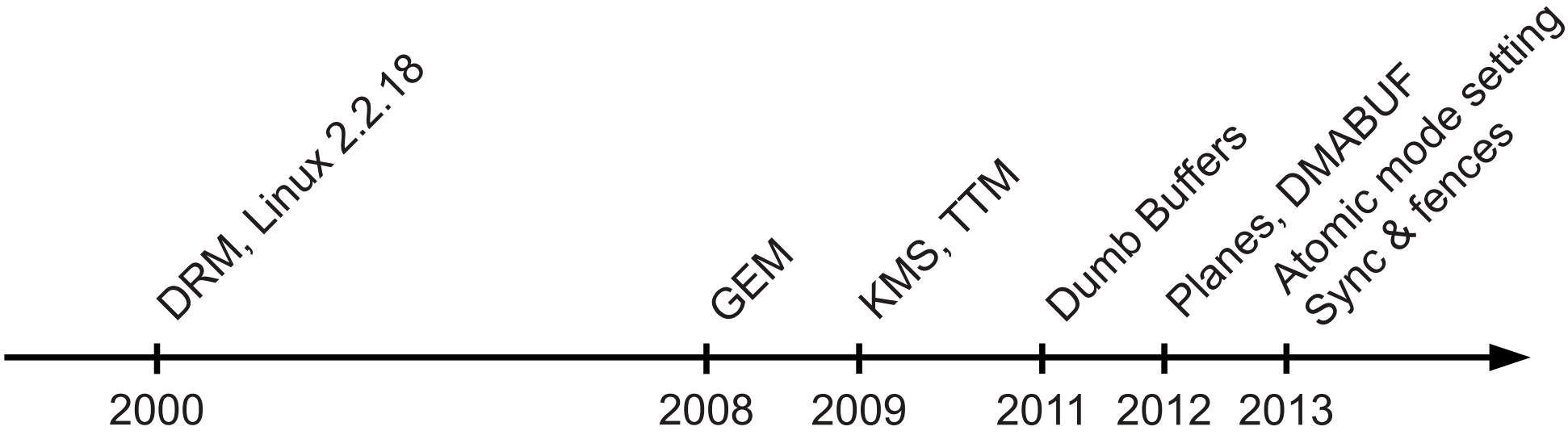
Transformations



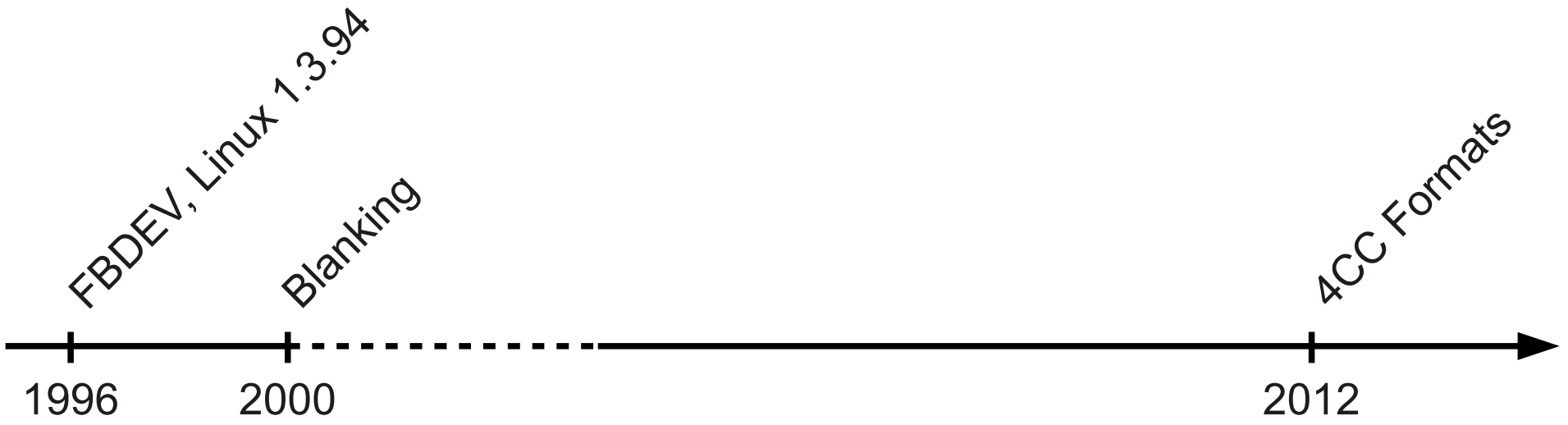
Device Model – FBDEV



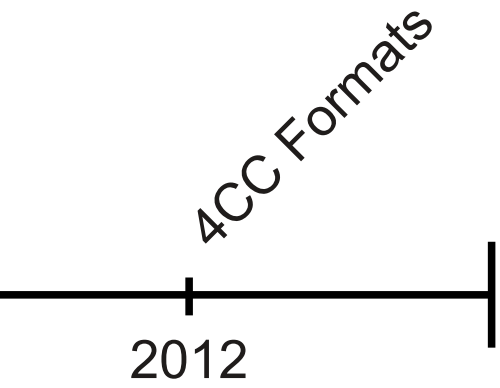
Device Model – DRM/KMS



Activity – DRM/KMS



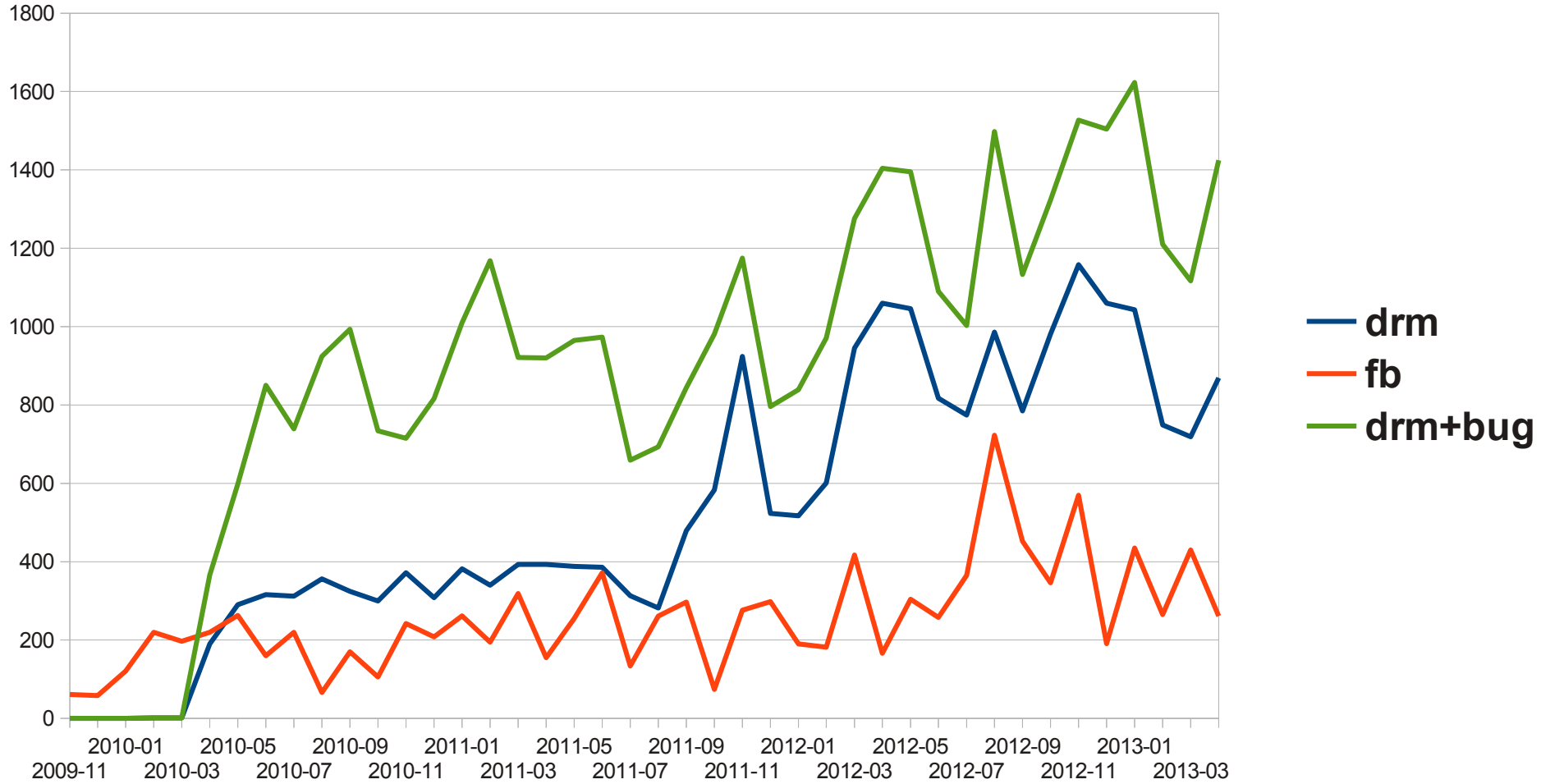
Activity – FBDEV



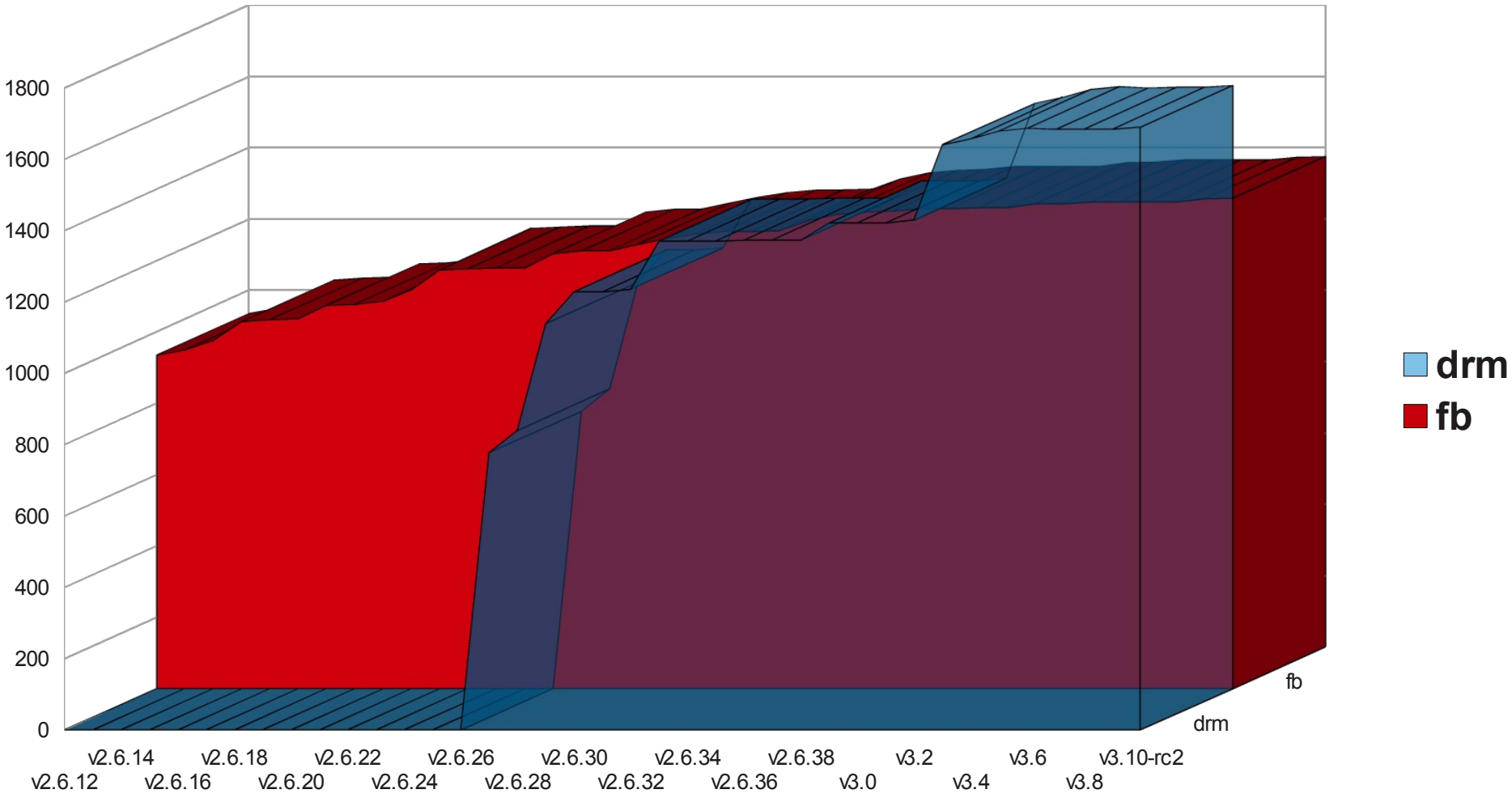
IDEAS ON BOARD

Activity – FBDEV

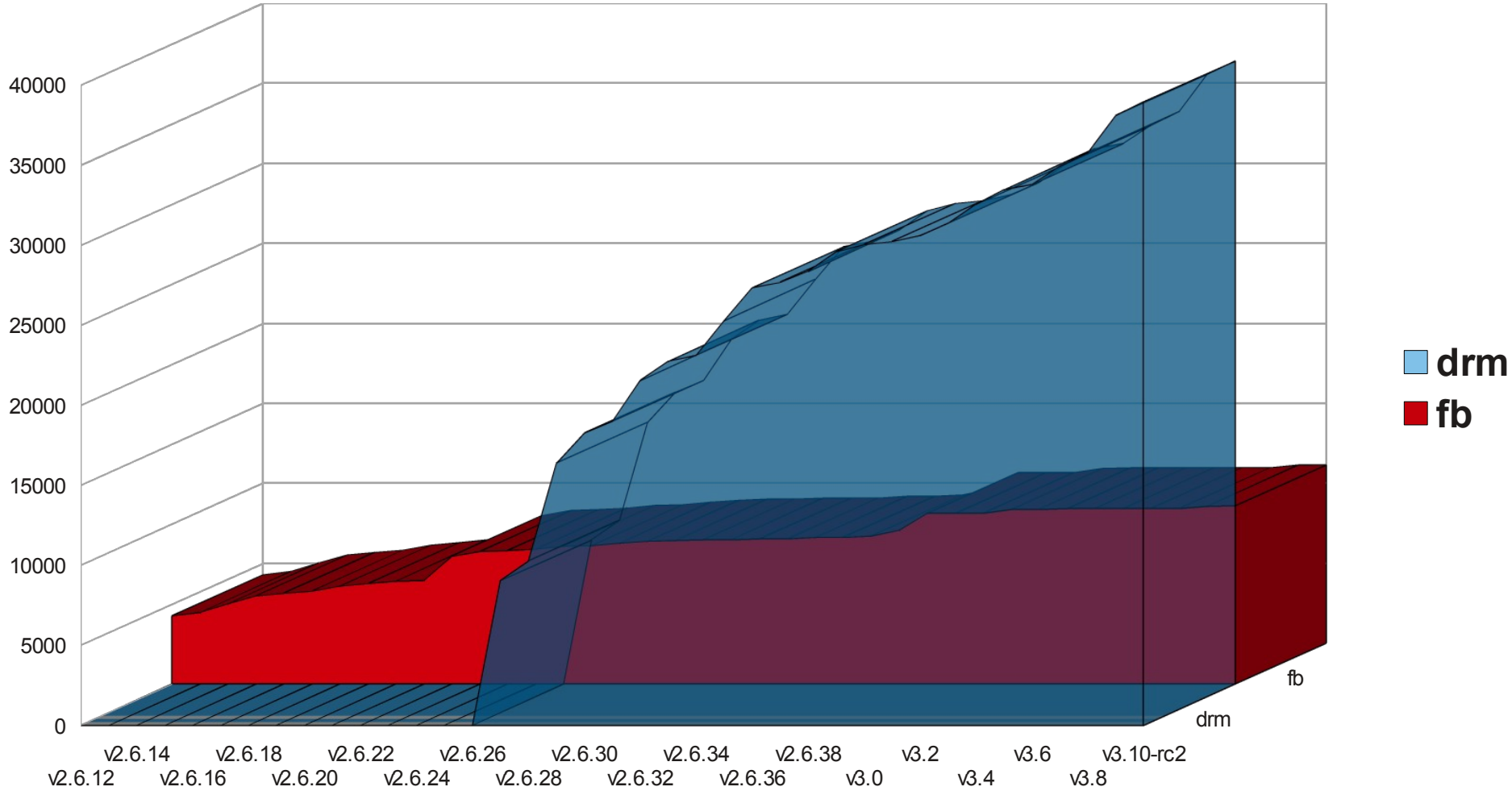
Source: http://valdodge.com/2010/07/



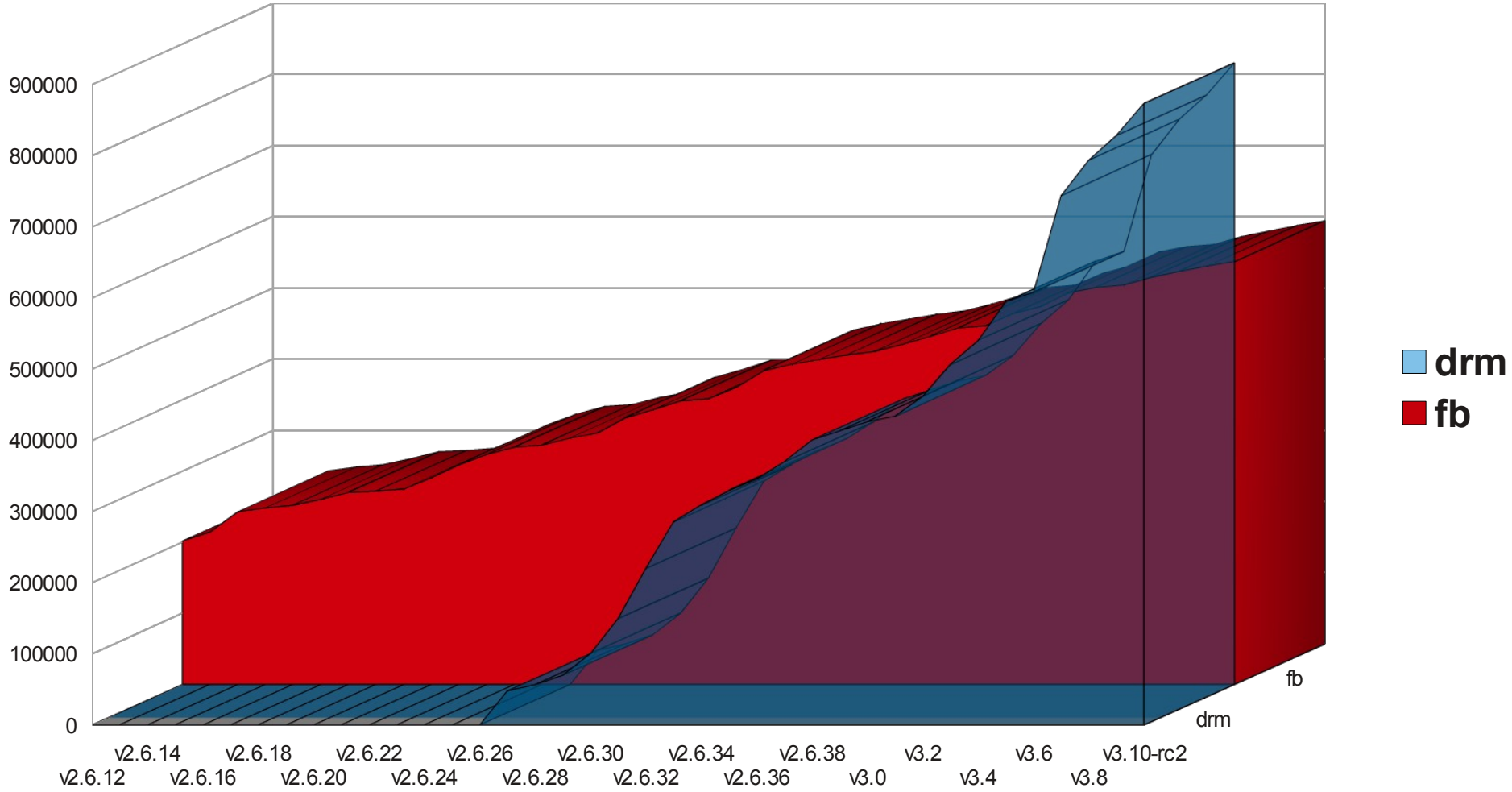
Mailing List Traffic



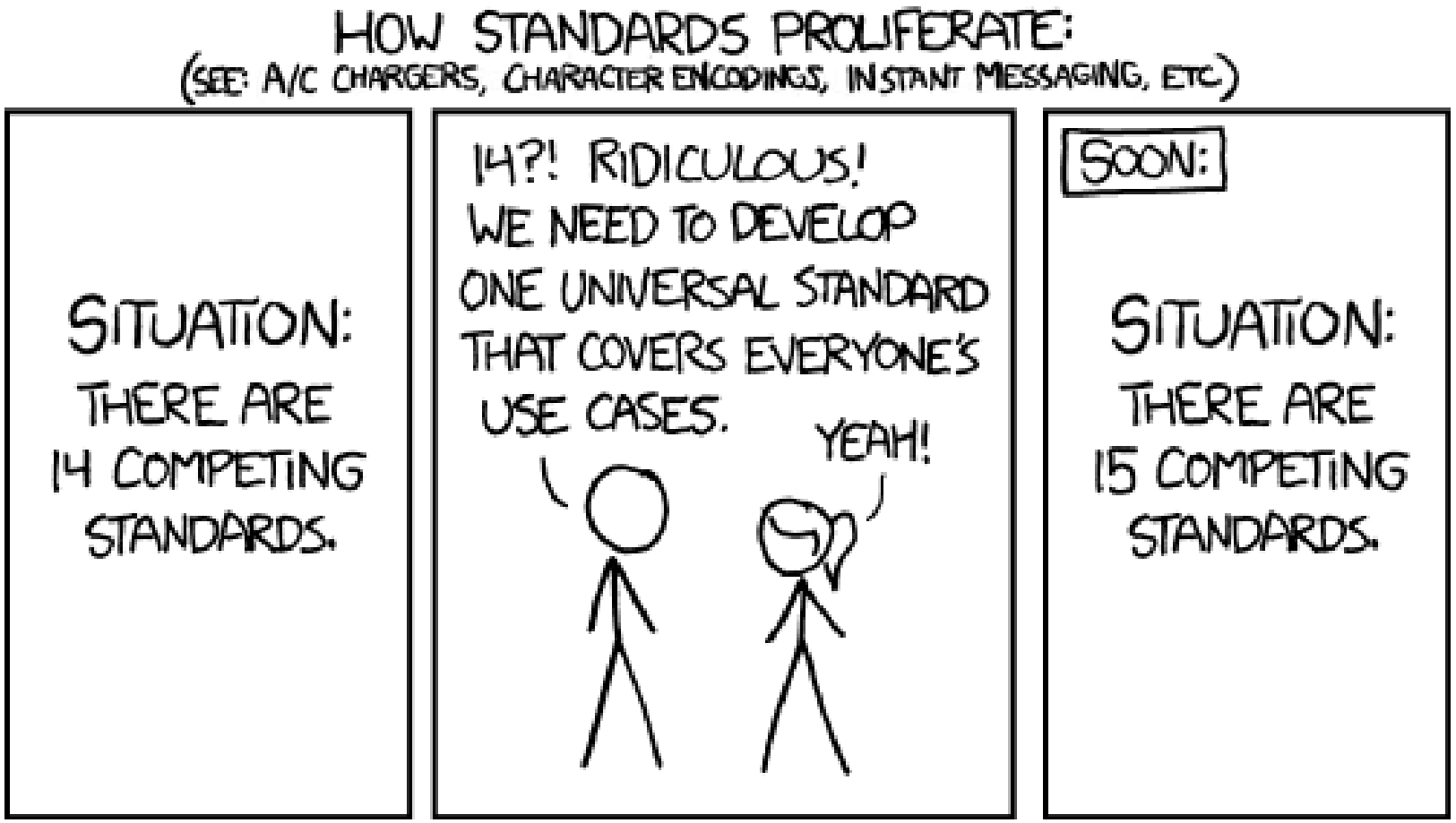
Cumulative Changes - API



Cumulative Changes - Core



Cumulative Changes - Drivers



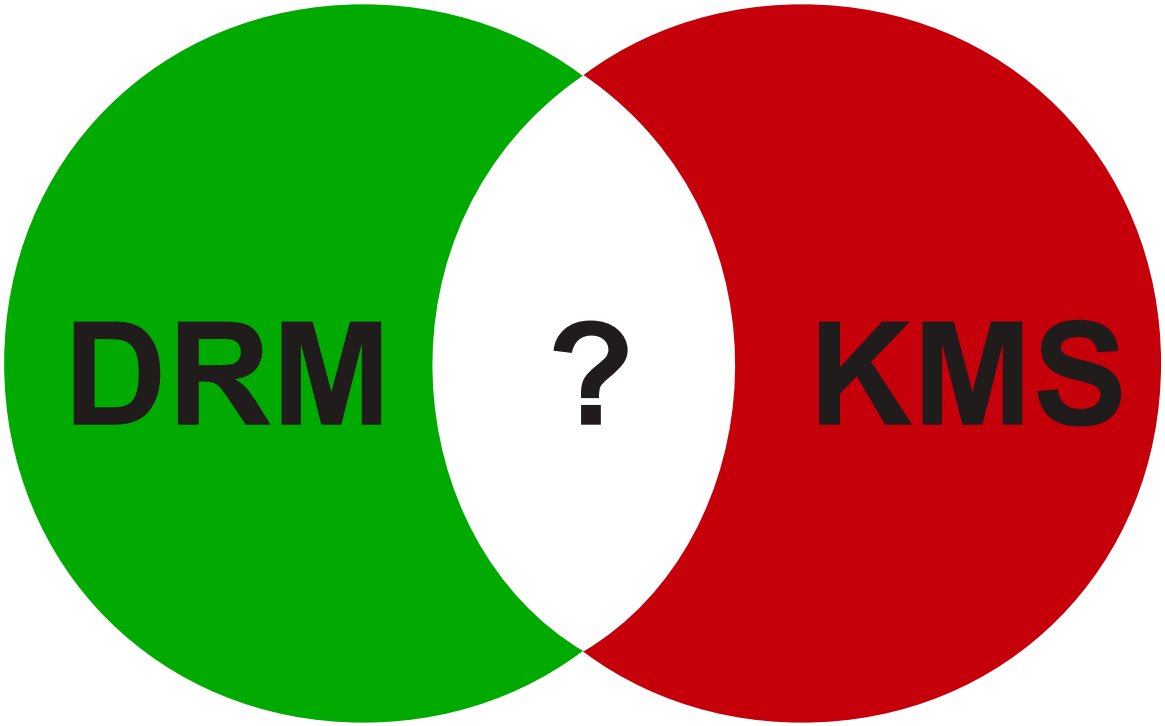
Use Cases

Source: http://xkcd.com/927/

Use Cases - FBDEV

(that's it...)

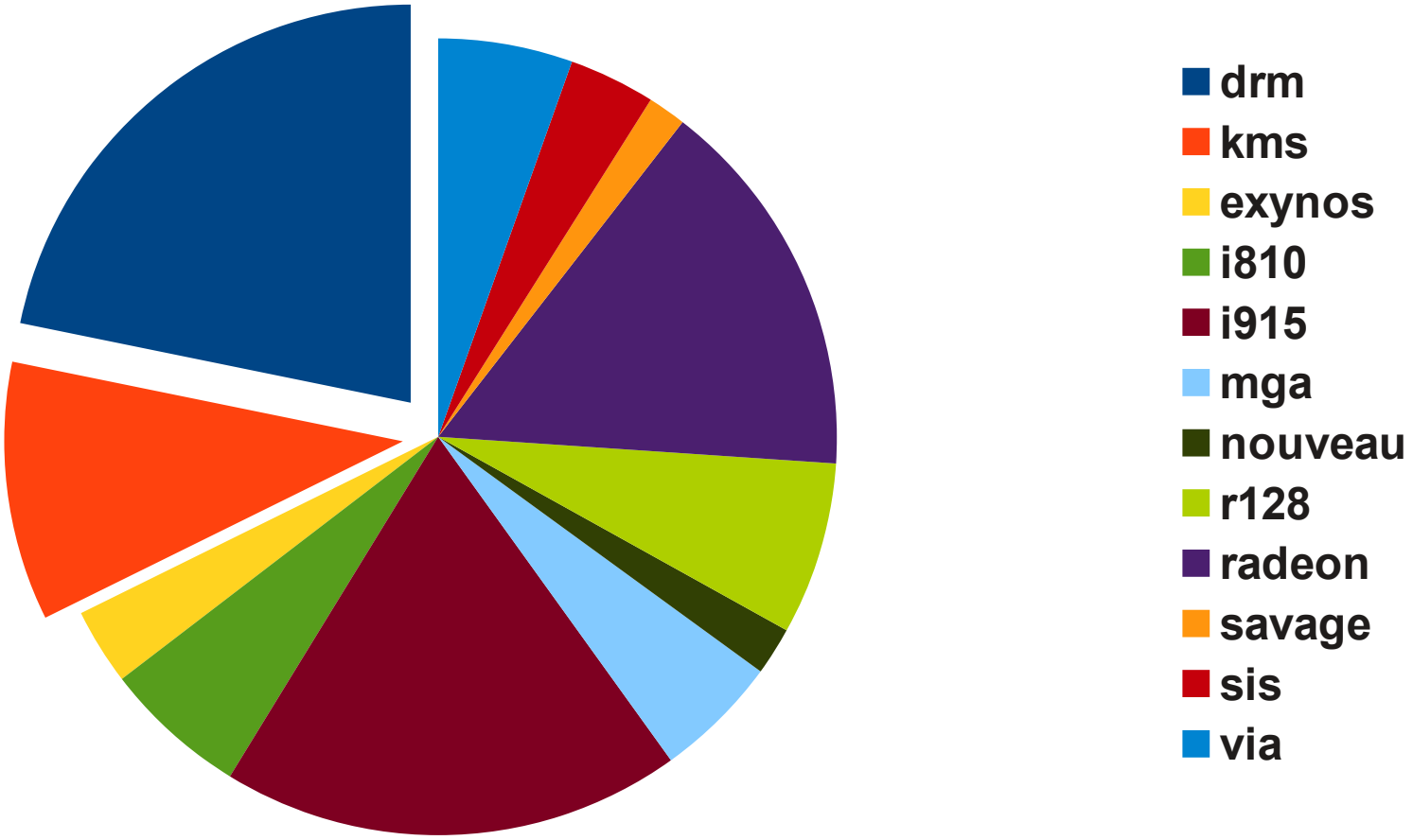
Everything else



APIs

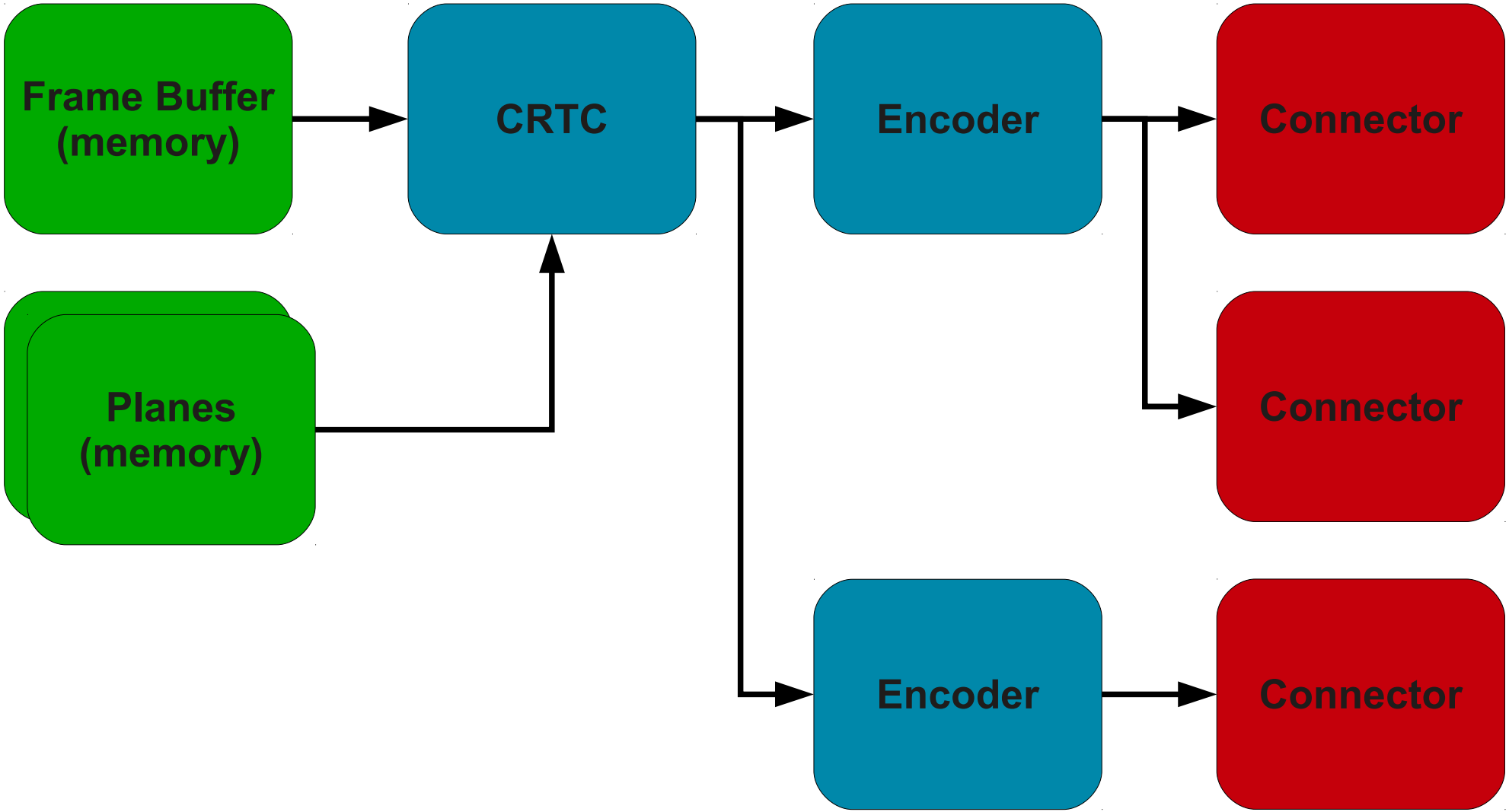
- Memory Management
- Vertical Blanking
- Version, Authentication, Master, ...

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


drm
kms
exynos
i810
i915
mga
nouveau
r128
radeon
savage
sis
via

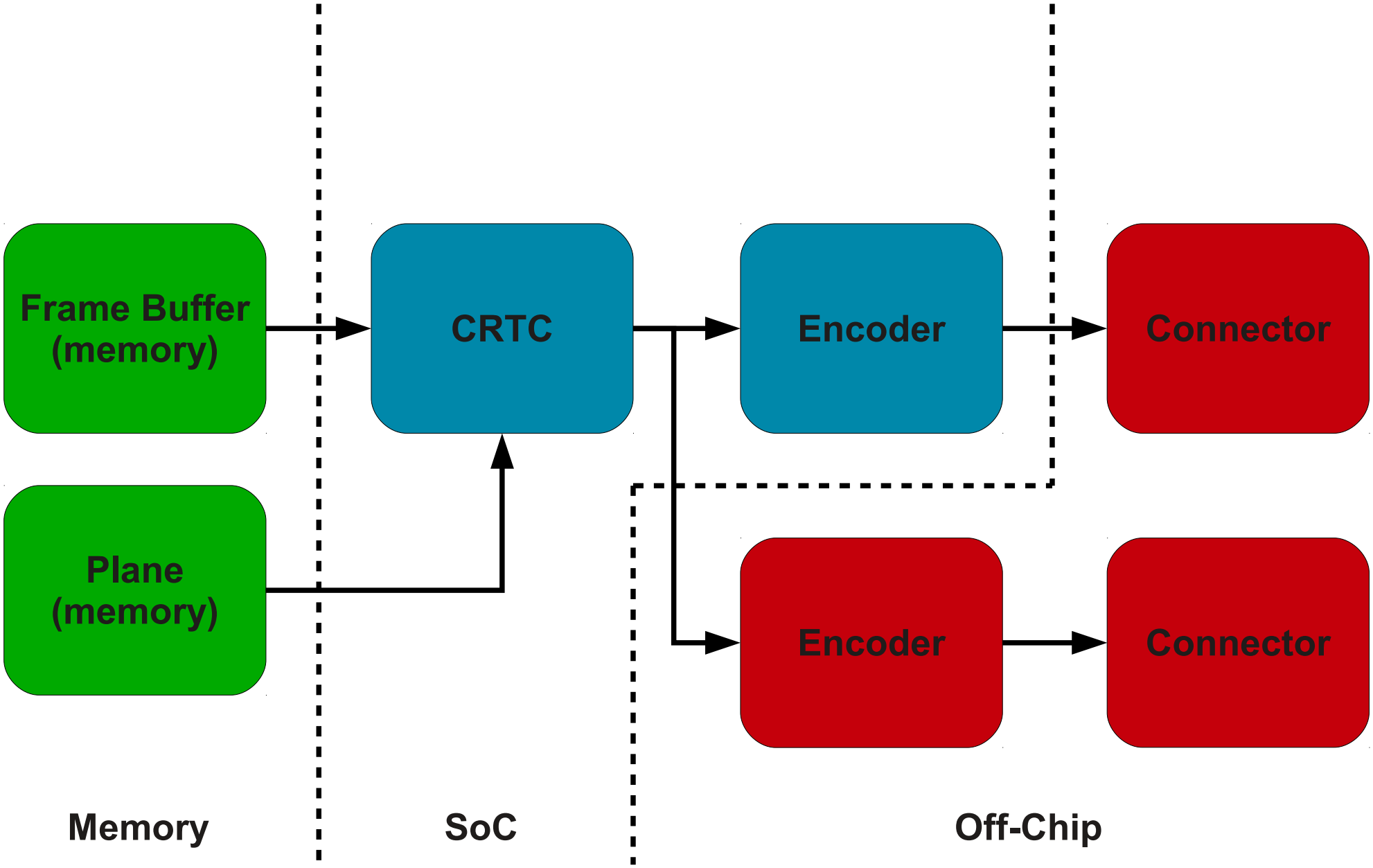

IDEAS
ON BOARD

DRM/KMS API



IDEAS ON BOARD

Device Model



Device Model - SoC

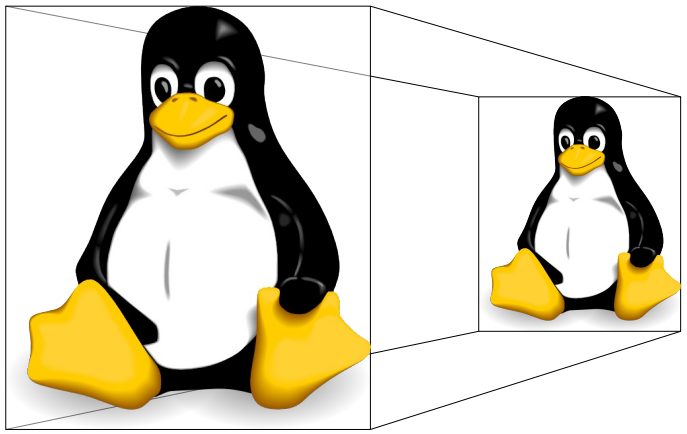
Frame Buffer

KMS – Scanout

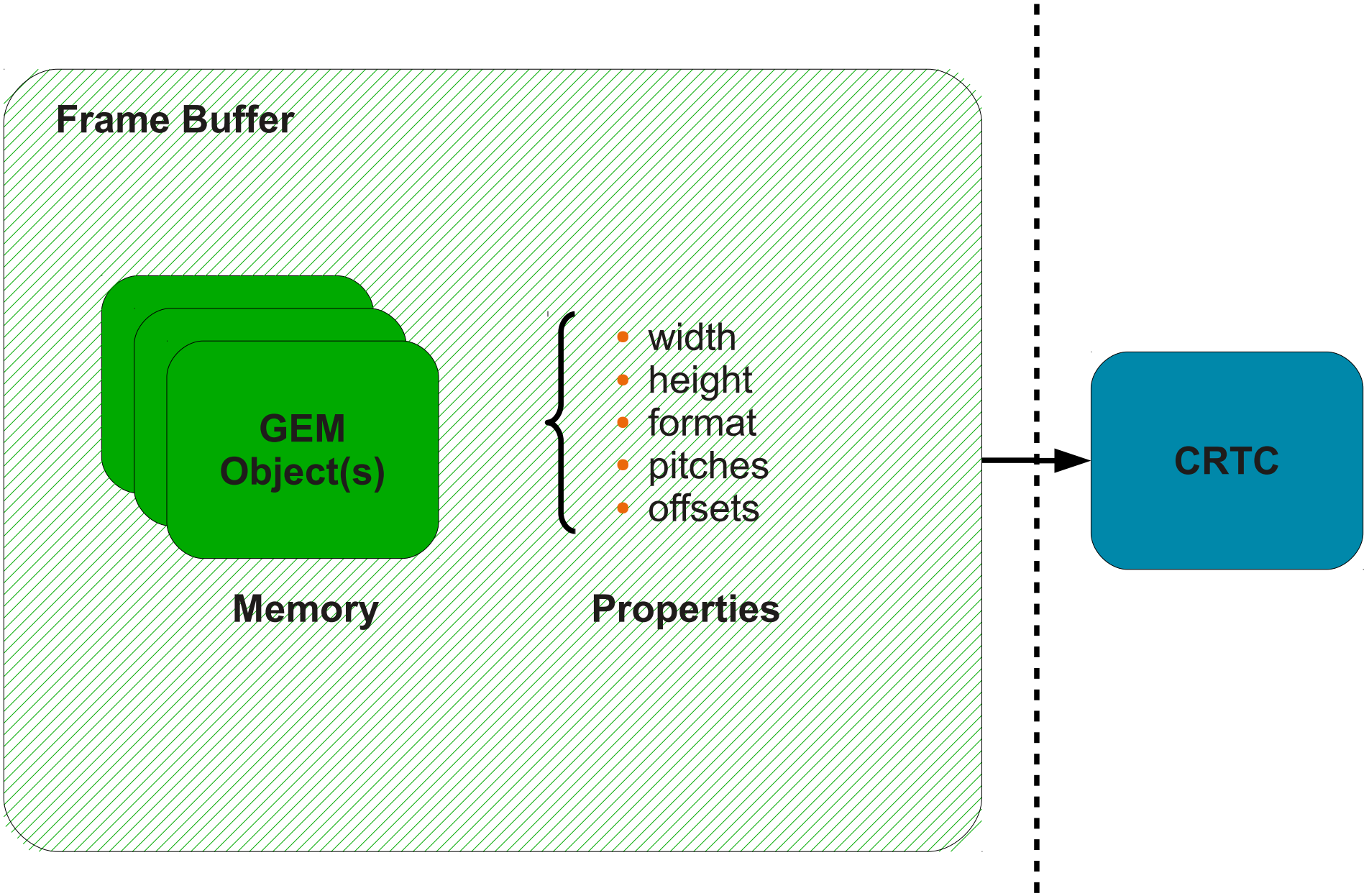
CRTC

Composition

Plane(s)

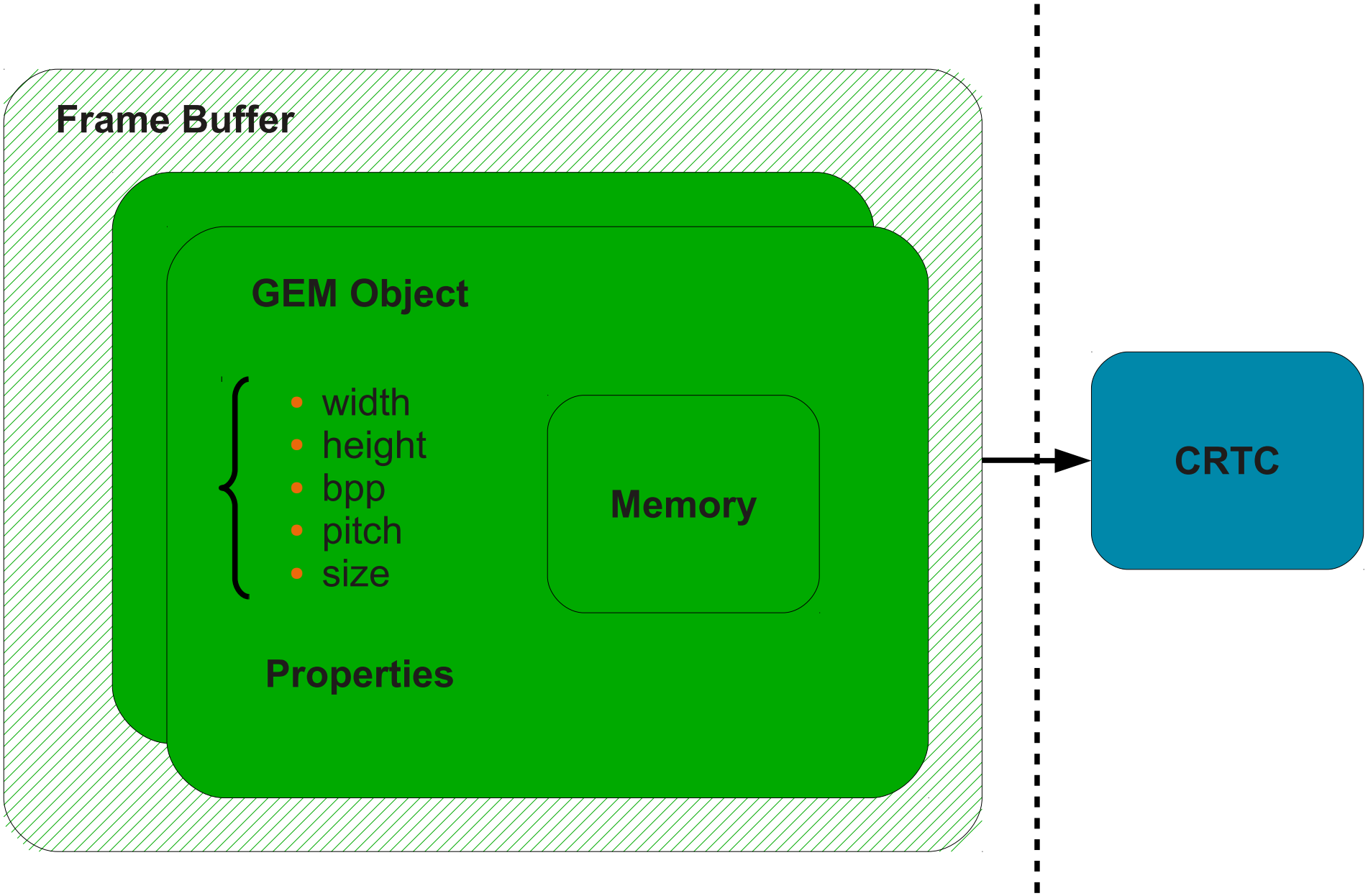


KMS – Composition

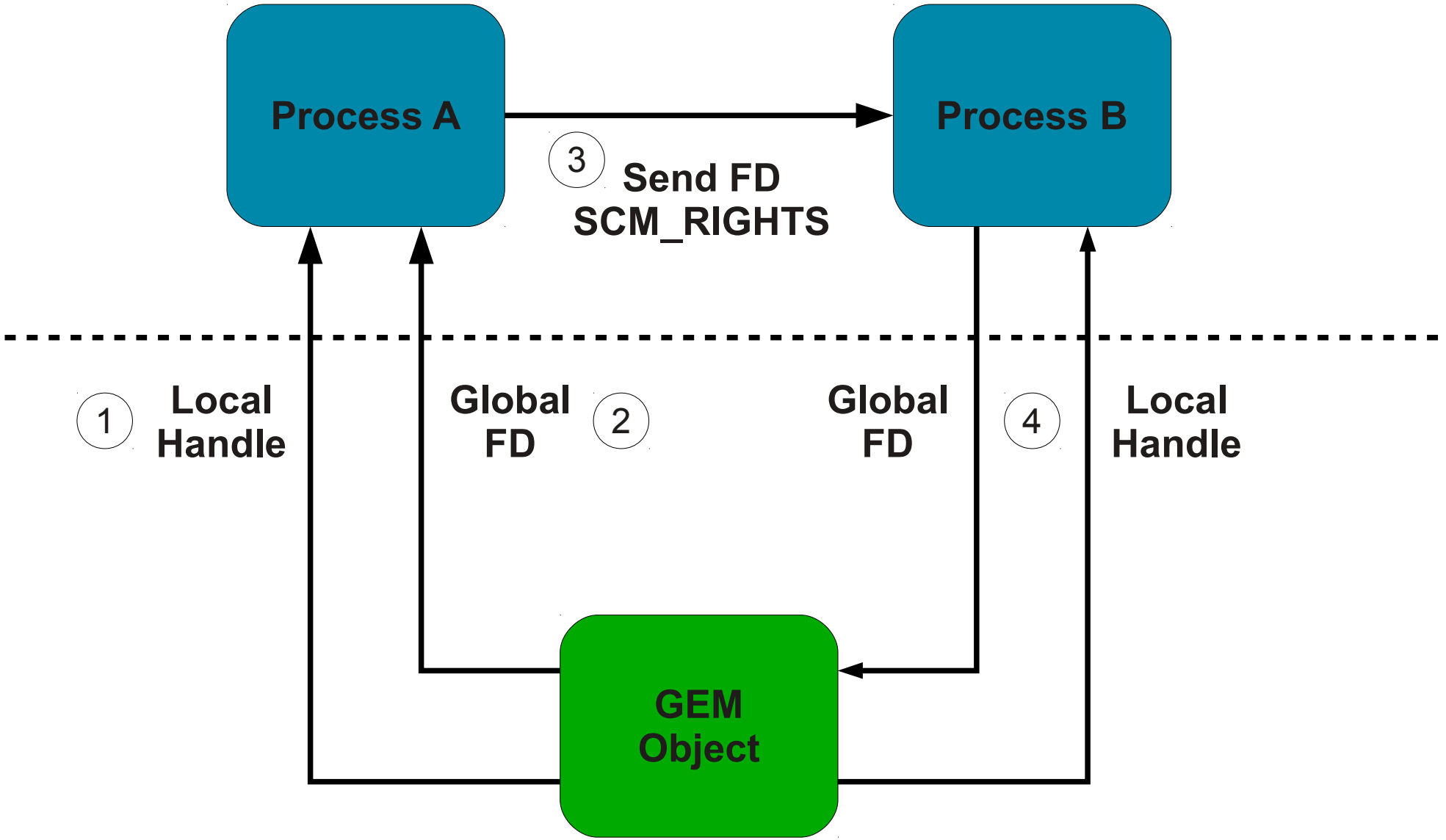


IDEAS ON BOARD

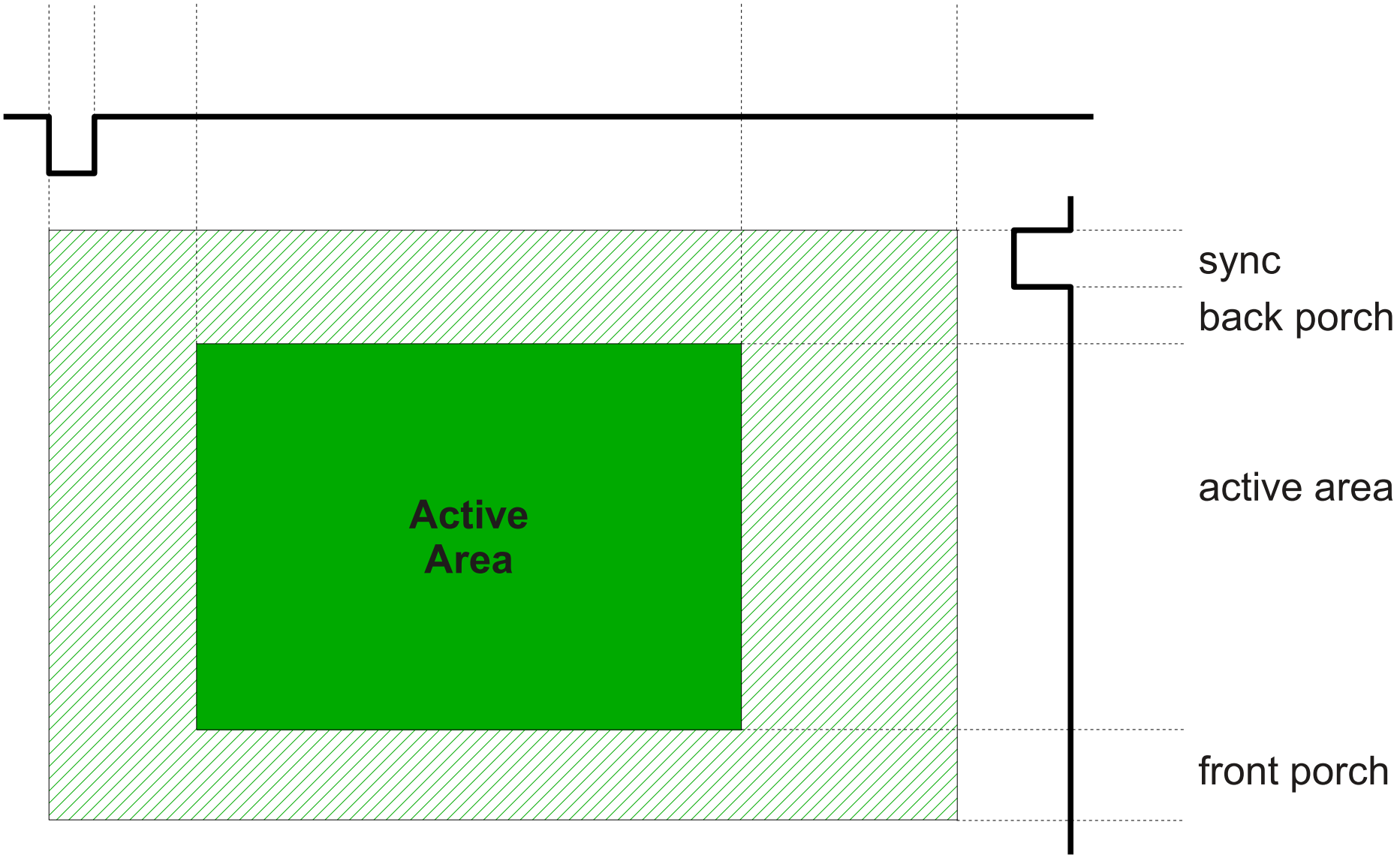
KMS – Frame Buffer



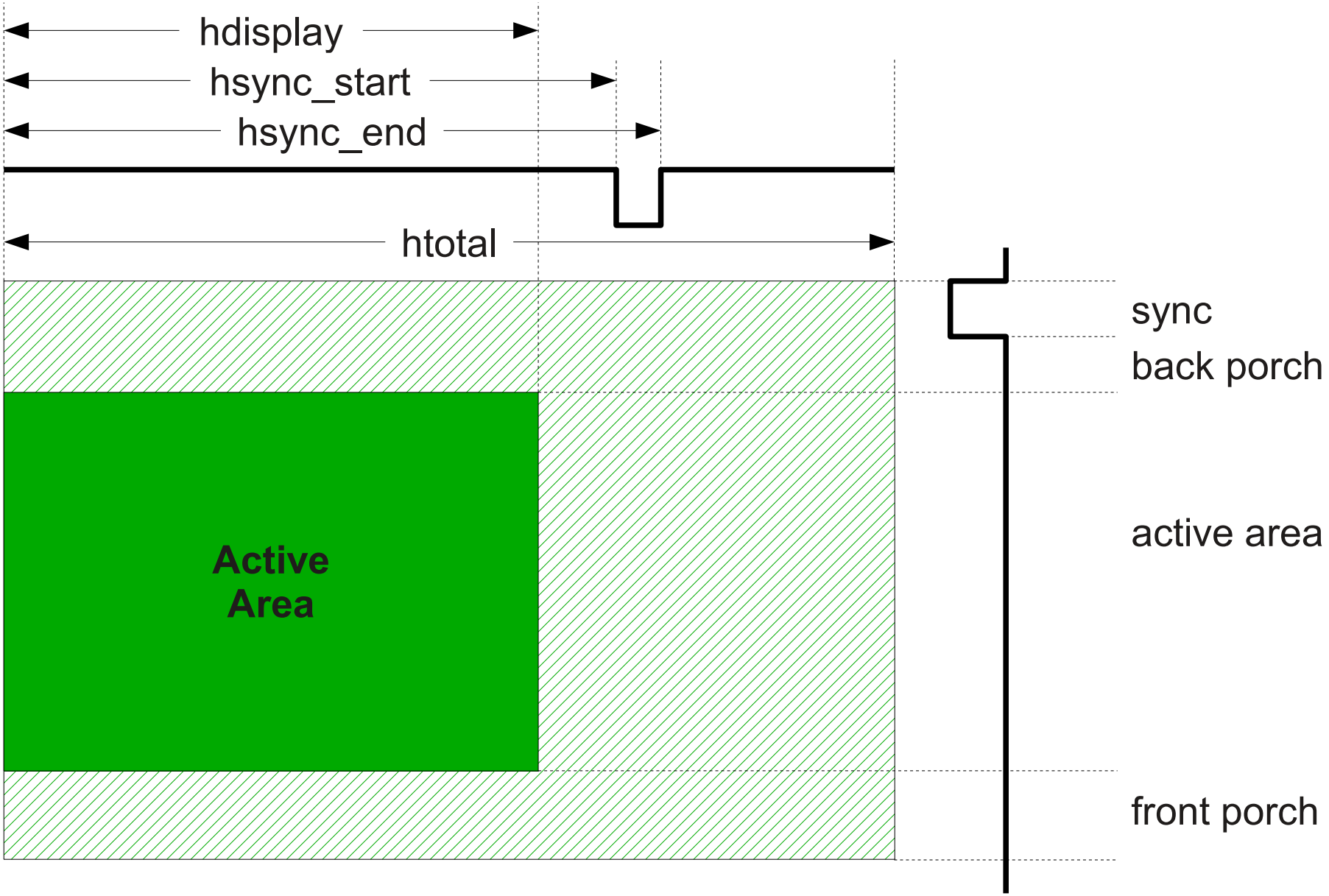
DRM/KMS – GEM Object


Process A
Process B
3
Send FD
SCM_RIGHTS
1
Local
Handle
Global
FD
2
Global
FD
4
Local
Handle
GEM
Object

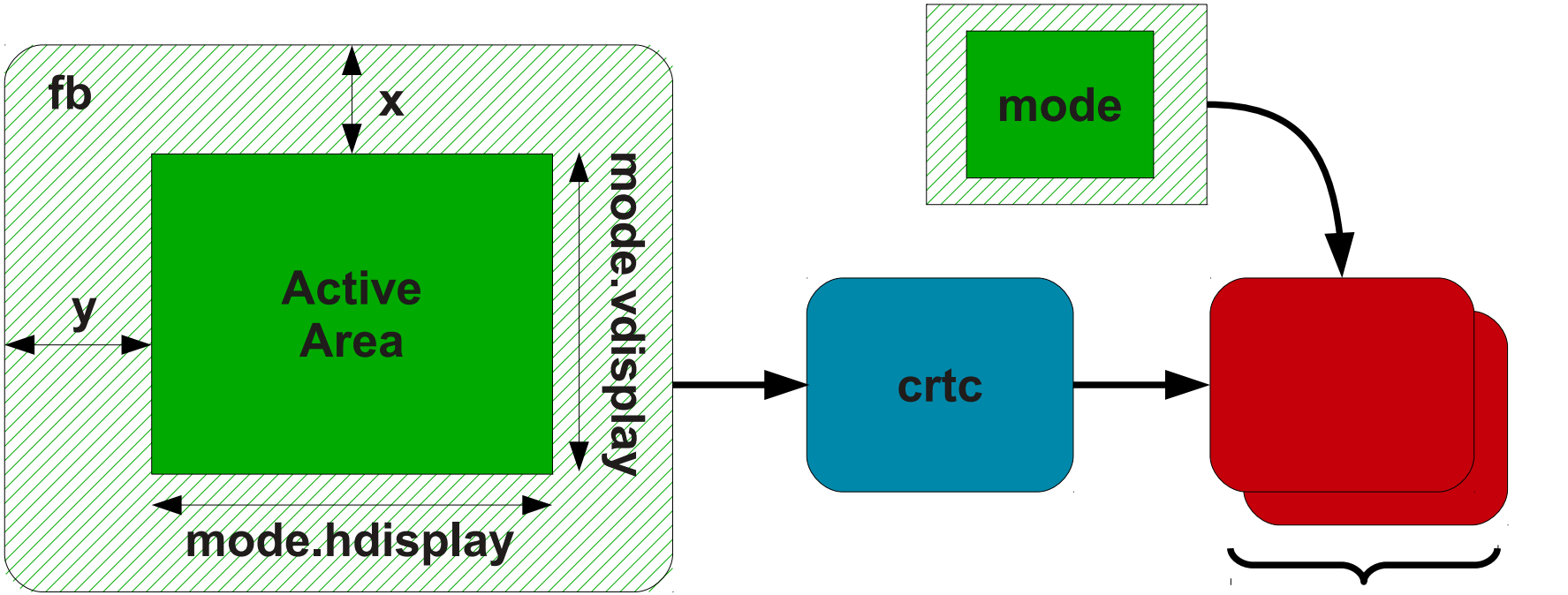
DRM – Handles



KMS – Modes (1/2)


hdisplay
hsync_start
hsync_end
htotal
Active
Area
sync
back porch
active area
front porch

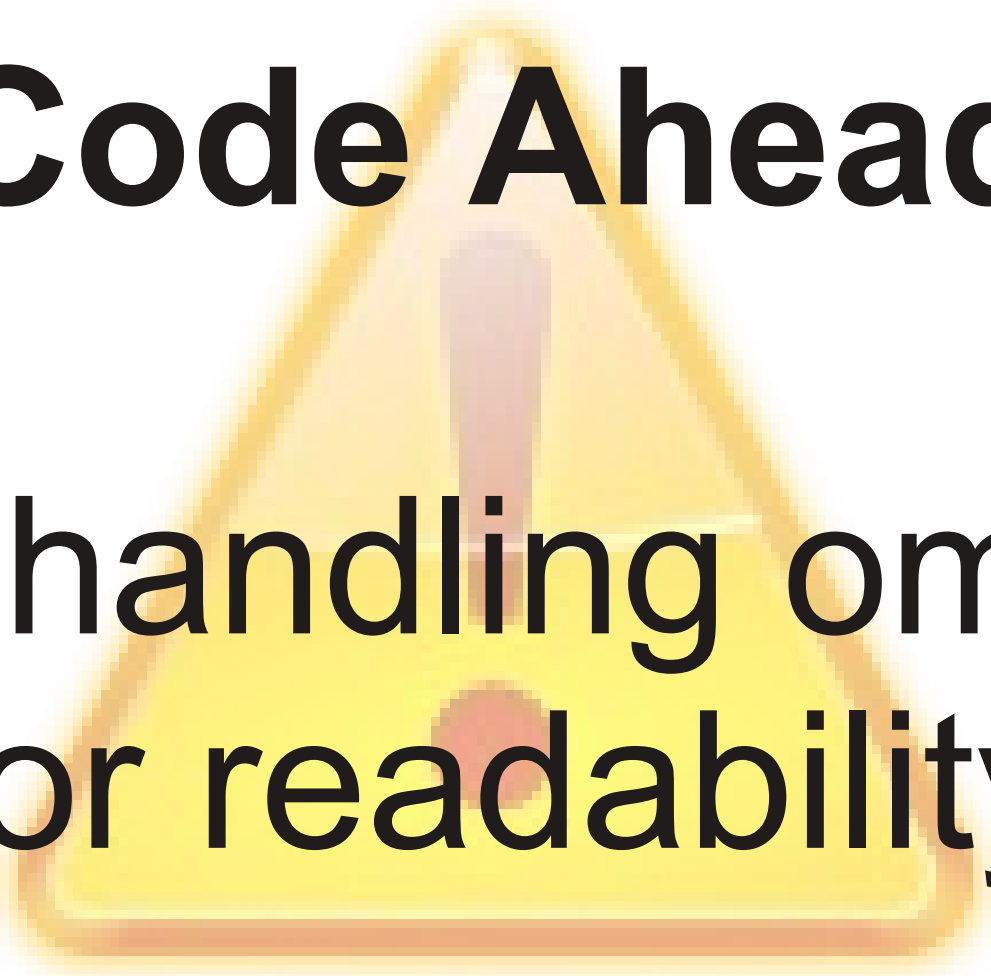
KMS – Modes (2/2)




```
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

Code Ahead



Error handling omitted
for readability

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>

int main(int argc, char **argv)
{
        return 0;
}
```

Skeleton

```
/*
 * Open
 *
 * #include <xf86drm.h>
 *
 * int drmOpen(const char *name, const char *busid);
 */
int fd;

fd = drmOpen("rcar-du", NULL);
```

Open

```
/*
 * Get resources
 *
 * #include <xf86drmMode.h>
 *
 * drmModeResPtr drmModeGetResources(int fd);
 * void drmModeFreeResources(drmModeResPtr ptr);
 */
drmModeResPtr resources;
uint32_t crtc_id;
uint32_t connector_id;

resources = drmModeGetResources(fd);

crtc_id = resources->crtcs[0];
connector_id = resources->connectors[0];
```

Get Resources

```
/*
 * Get modes
 *
 * #include <xf86drmMode.h>
 *
 * drmModeConnectorPtr drmModeGetConnector(int fd,
 *             uint32_t connectorId);
 * void drmModeFreeConnector(drmModeConnectorPtr ptr);
 */
drmModeConnectorPtr connector;
drmModeModeInfo mode;
uint32_t width;
uint32_t height;

connector = drmModeGetConnector(fd, connector_id);

mode = connector->modes[0];
width = mode.hdisplay;
height = mode.vdisplay;
```

Get Modes

```
/*
 * Initialize libkms
 *
 * #include <libkms.h>
 *
 * int kms_create(int fd, struct kms_driver **out);
 * int kms_destroy(struct kms_driver **kms);
 */
struct kms_driver *kms;

kms_create(fd, &kms);
```

Initialize libkms

```
/*
 * Create buffer
 *
 * #include <libkms.h>
 *
 * int kms_bo_create(struct kms_driver *kms,
 *             const unsigned *attr, struct kms_bo **out);
 * int kms_bo_destroy(struct kms_bo **bo);
 */
unsigned bo_attribs[] = {
   KMS_WIDTH,   width,
   KMS_HEIGHT,  height,
   KMS_BO_TYPE, KMS_BO_TYPE_SCANOUT_X8R8G8B8,
   KMS_TERMINATE_PROP_LIST
};
struct kms_bo *bo;

kms_bo_create(kms, bo_attribs, &bo);
```

Create Buffer

```
/*
 * Get buffer handle and pitch
 *
 * #include <libkms.h>
 *
 * int kms_bo_get_prop(struct kms_bo *bo, unsigned key,
 *                     unsigned *out);
 */
uint32_t handles[4];
uint32_t pitches[4];
uint32_t offsets[4];

kms_bo_get_prop(bo, KMS_HANDLE, &handles[0]);
kms_bo_get_prop(bo, KMS_PITCH, &pitches[0]);
offsets[0] = 0;
```

Get Buffer Handle and Pitch

```
/*
 * Fill buffer
 *
 * #include <libkms.h>
 *
 * int kms_bo_map(struct kms_bo *bo, void **out);
 * int kms_bo_unmap(struct kms_bo *bo);
 */
void *plane;

kms_bo_map(bo, &plane);
fill_pattern(DRM_FORMAT_ARGB8888, plane, width,
             height, pitches[0]);
kms_bo_unmap(bo);
```

Fill Buffer

```
/*
 * Create frame buffer
 *
 * #include <drm_fourcc.h>
 * #include <xf86drmMode.h>
 *
 * int drmModeAddFB2(int fd, uint32_t width,
 *       uint32_t height, uint32_t pixel_format,
 *       uint32_t bo_handles[4], uint32_t pitches[4],
 *       uint32_t offsets[4], uint32_t *buf_id,
 *       uint32_t flags);
 */
uint32_t fb_id;

drmModeAddFB2(fd, width, height, DRM_FORMAT_ARGB8888,
              handles, pitches, offsets, &fb_id, 0);
```

Create Frame Buffer

```
/*
 * Set the mode
 *
 * #include <xf86drmMode.h>
 *
 * int drmModeSetCrtc(int fd, uint32_t crtcId,
 *                    uint32_t bufferId, uint32_t x, uint32_t y,
 *                    uint32_t *connectors, int count,
 *                    drmModeModeInfoPtr mode);
 */
drmModeSetCrtc(fd, crtc_id, fb_id, 0, 0,
               &connector_id, 1, &mode);
```

Set the Mode

```
/*
 * Get plane resources
 *
 * #include <xf86drmMode.h>
 *
 * drmModePlaneResPtr drmModeGetPlaneResources(int fd);
 * void drmModeFreePlaneResources(drmModePlaneResPtr ptr);
 */
drmModePlaneResPtr planes;
uint32_t plane_id;

planes = drmModeGetPlaneResources(fd);
plane_id = planes->planes[0];
```

Get Plane Resources

```
/*
 * Set the plane
 *
 * #include <xf86drmMode.h>
 *
 * int drmModeSetPlane(int fd,
 *         uint32_t plane_id, uint32_t crtc_id,
 *         uint32_t fb_id, uint32_t flags,
 *         uint32_t crtc_x, uint32_t crtc_y,
 *         uint32_t crtc_w, uint32_t crtc_h,
 *         uint32_t src_x, uint32_t src_y,
 *         uint32_t src_w, uint32_t src_h);
 */
drmModeSetPlane(fd, plane_id, crtc_id, fb_id, 0,
          width / 4, height / 4, width / 2, height / 2,
          0, 0, (width / 2) << 16, (height / 2) << 16);
```

Set the Plane

```
/*
 * List plane properties
 *
 * #include <drm_mode.h>
 * #include <xf86drmMode.h>
 *
 * drmModeObjectPropertiesPtr drmModeObjectGetProperties(
 *       int fd, uint32_t object_id, uint32_t object_type);
 * void drmModeFreeObjectProperties(
 *       drmModeObjectPropertiesPtr ptr);
 */
drmModeObjectPropertiesPtr properties;

properties = drmModeObjectGetProperties(fd, plane_id,
                                        DRM_MODE_OBJECT_PLANE);
```

List Plane Properties

```
/*
 * Find color keying property
 *
 * #include <xf86drmMode.h>
 *
 * drmModePropertyPtr drmModeGetProperty(int fd,
 *                                       uint32_t propertyId);
 * void drmModeFreeProperty(drmModePropertyPtr ptr);
 */
drmModePropertyPtr property;
unsigned int i;

for (i = 0; i < properties->count_props; ++i) {
        property = drmModeGetProperty(fd,
                                      properties->props[i]);
        if (!strcmp(property->name, "colorkey"))
                break;
}
```

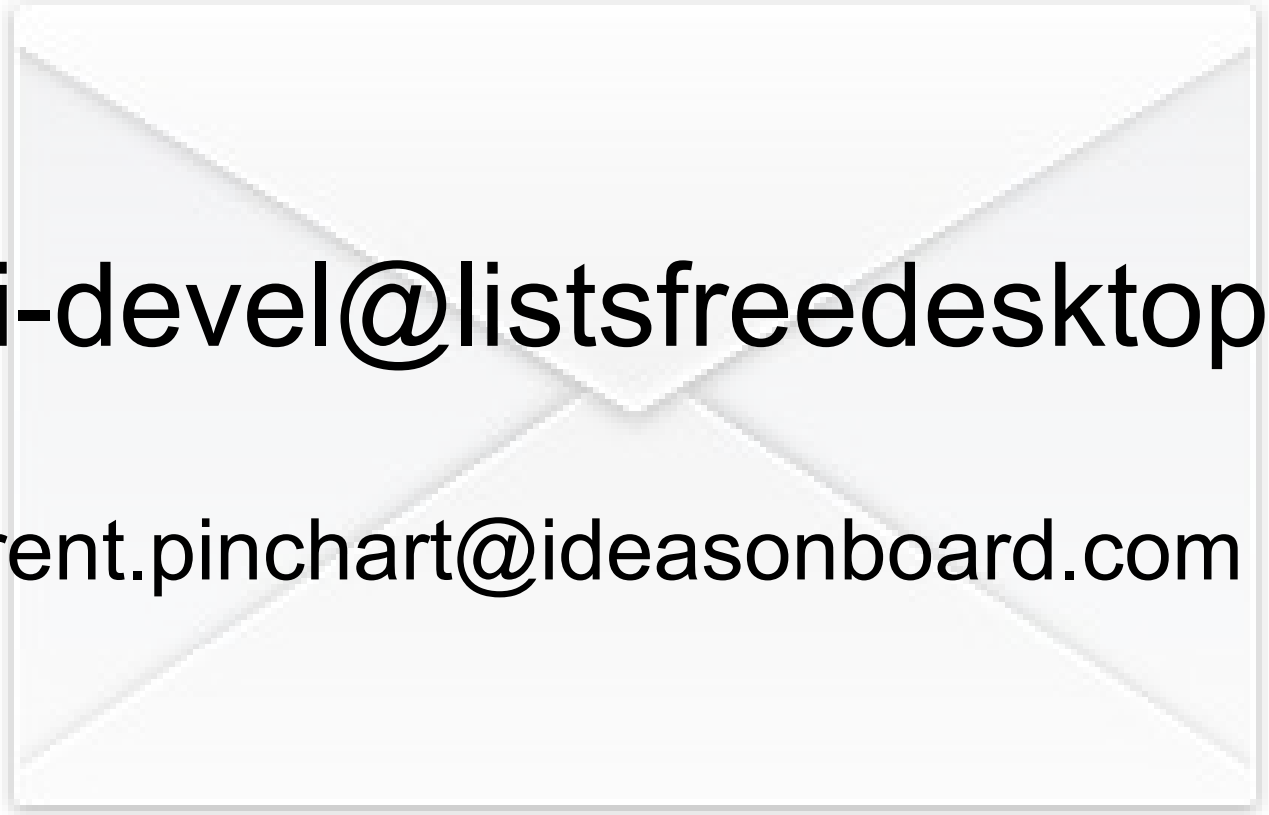
Find Color Keying Property

```
/*
 * Turn color keying on
 *
 * #include <xf86drmMode.h>
 *
 * int drmModeObjectSetProperty(int fd, uint32_t object_id,
 *              uint32_t object_type, uint32_t property_id,
 *              uint64_t value);
 */
drmModeObjectSetProperty(fd, plane_id,
                         DRM_MODE_OBJECT_PLANE,
                         property->prop_id, 0x0100c0c0);
```

Turn Color Keying On

- Page Flip
- Cursor
- Events
- ...

And Much More...



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

Contact

thx.