Exposing the Android Camera Stack

Balwinder Kaur, Principal Software Architect

Joe Rickson, Principal Software Engineer

The San Francisco Android User Group

8.28.2012

Agenda

- Camera APIs
  - Overview of android.hardware.Camera
  - What's new in Jelly Bean ?
  - Prominent Camera Use Cases
- Android Media Framework – Camera
  - Native Camera service
  - What's new in Jelly Bean ?
  - Media Subsystem Interactions
- It's all about the Camera hardware !
  - Camera Hardware Abstraction Layer
  - Camera Device Driver
  - Camera Hardware Architecture
- Future Trends
- Q&A

Section I

Android Camera APIs

Overview of android.hardware.Camera

6 Classes

- Camera
- Camera.CameraInfo
- Camera.Parameters
- Camera.Size
- Camera.Face
- Camera.Area

8 Callback Interfaces

- Camera.AutoFocusCallback
- Camera.ErrorCallback
- Camera.FaceDetectionListener
- Camera.OnZoomChangeListener
- Camera.PictureCallback
- Camera.PreviewCallback
- Camera.ShutterCallback
- Camera.AutoFocusMoveCallback

Camera class

Contains all the methods for the Camera Lifecycle

- Open & Release
- Access to the Camera Controls
- Preview
  - Direct Live Preview to the display or a texture
  - Get Preview Frame in a Callback
- Capture
  - Callbacks: Shutter, JPEG, RAW, "Postview"
- Lock & Unlock
- Actions: startAutoFocus, startSmoothZoom & startFaceDetection

Camera.Parameters

Class for Camera Controls

① Mandatory Feature Set

- getSupportedPreviewSizes + set/get

② Optional Feature Set

- isVideoStabilizationSupported + set/get

③ Custom Feature Set

- Camera.Parameters class provides a "dumb" pipe to the hardware for custom controls
- set ("your_param_string", value); get("your__param_string");

Auto White Balance, Scene Modes, Focus Modes, Preview Sizes, Picture Sizes, Thumbnail Sizes, Preview Format, Picture Format, Anti-Banding

… the rest of the Camera Classes

- Camera.CameraInfo
  - For each camera, front or back facing, orientation of the camera image
- Camera.Size
  - width and height of the image
- Camera.Face
  - face-id, co-ordinates for left eye, right eye, mouth, outer bounds of the face
- Camera.Area
  - Rectangular bounds with a weight
  - Metering Regions for 3A : Auto Focus, Auto White Balance, Auto Exposure

What's new in Jelly Bean ?

APIs

- Camera.AutoFocusMoveCallback
  - FOCUS_MODE_CONTINUOUS_PICTURE and FOCUS_MODE_CONTINUOUS_VIDEO allows you to listen for changes to the auto focus movement – starting & stopping
- android.media.MediaActionSound
  - Play an appropriate camera operation sound when implementing a custom still or video recording mechanism, or when implementing some other camera-like function in your application.

What's new in Jelly Bean ?

New System Camera Application

- Source code has two apps
  - packages/apps/Camera
  - packages/apps/LegacyCamera
- Support for swipe gesture been added.
- Flick to the left at any time, and you'll be able to scroll through all the photos you've taken. From there, you can crop, rotate or share, just like in the gallery app.
- Swipe upwards to discard unwanted photos
- Live Preview feed is still running to take still pictures

Android 4.0 Camera Features

Feature	Platform Feature with API	In-built Camera Application Code	Proprietary Solution	API Level
Face Detection	✔			14
Face Recognition			✔	14
Panoramic Stitch		✔		14
Video Snapshot	✔			14
AE & AWB Lock	✔			14
Continuous Focus Mode	✔			14
Region Of Interest (AE, AWB and AF)	✔			14
Zero Shutter Lag*				14
Video Stabilization	✔			15
Live Effects on Images / Video**	✔			14

* There is no API for ZSL. It is a hardware dependent feature.
** android.media.Effect

AE : Auto Exposure

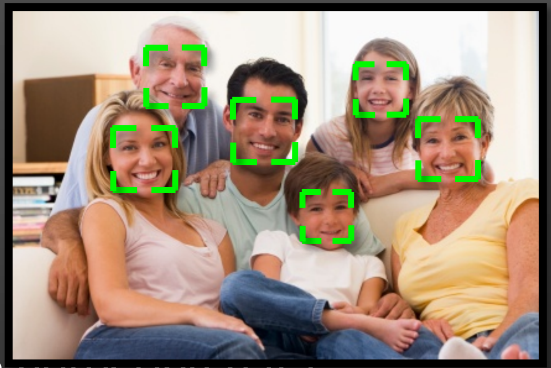
AWB : Auto White Balance

AF : Auto Focus

Prominent Camera Use Cases

- Main Use Cases
  - Live Preview of Camera Stream
    - Live Preview + copy of the Frame returned to the application
  - Capture a frame
  - Video Recording of a Camera Stream
- Secondary Use Cases
  - Configuring the Camera
  - Receiving more than an image back . e.g. face detection data
  - VideoSnapshot
  - Event Callbacks: Shutter Clicked, AutoFocus Achieved
- Note: to use Existing Camera Apps use standard Android Intents

http://developer.android.com/guide/topics/media/camera.html



DEMO

- Preview
- Capture
- Save Picture

Switching to Video Mode

To quickly switch from still to video recording mode, use these steps:

- Open a Camera and startPreview as for still mode
- Call unlock() to allow the media process to access the camera.
- Pass the camera to MediaRecorder.setCamera(Camera).
- Follow MediaRecorder instructions on recording
- When finished recording, call reconnect() to re-acquire and re-lock the camera.
- If desired, restart preview and take more photos or videos.
- Call stopPreview() and release()

Face Detection

- Use Camera.Parameters to see if Face Detection is Supported

```
Camera.Parameters p = mCamera.getParameters();
if (p.getMaxNumDetectedFaces() >0 ) {
 mCamera.startFaceDetection(); }
```

- Face Information is available through the Camera.FaceDetectionListener

```
void onFaceDetection (Face[] faces, Camera camera) {
     // Overlay Green, White or your favorite color squares
     // on the Preview Surface
}
```

Some of the ISPs will overlay

these on the Preview stream directly

Other Applications

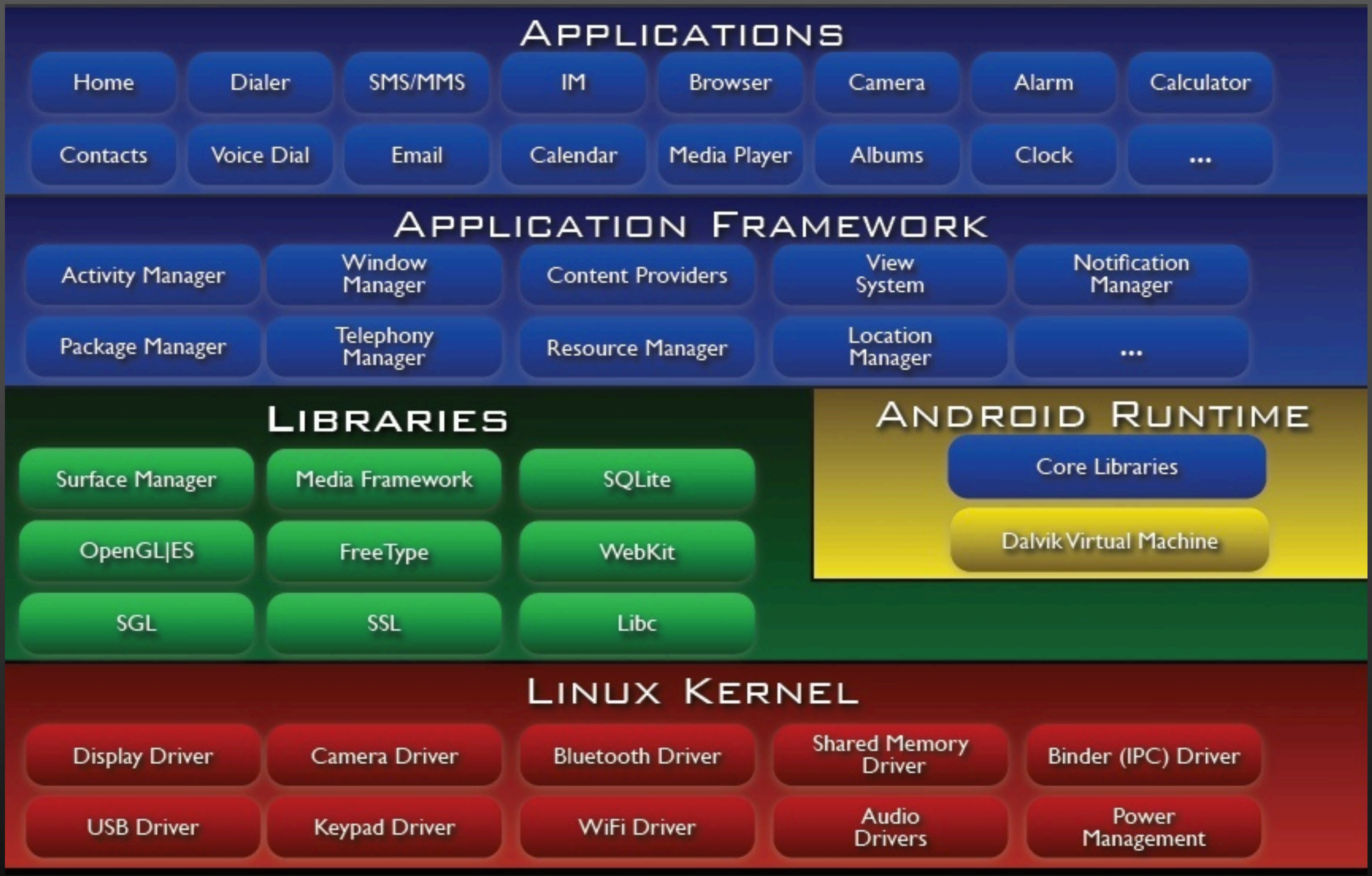
- ZXing: Open Source Library for 1D/2D image processing library
  - uses the Camera.**setOneShotPreviewCallback**
- Processing the live Preview Stream w/o a display
  - In API level 11 (HoneyComb), Camera.setPreviewTexture() call was introduced. With this call, Camera Streams can be processed w/o necessarily needing a display
  - GPU Processing: Need an OpenGL context. SurfaceTexture.updateTexImage will update SurfaceTexture to the latest preview frame from the camera
  - CPU Processing: Don't call updateTexImage. The SurfaceTexture will simply discard all data passed into it by the camera. Set up preview callbacks using setPreviewCallback, and use that data (typically in a YUV format) for CPU processing. Less efficient than GPU processing. No knowledge of OpenGL is needed. OpenCV sample code uses this pattern a lot.
  - Google IO 2011 : http://www.youtube.com/watch?v=OxzucwjFEEs#t=16m30

Section II

Android Media Framework - Camera

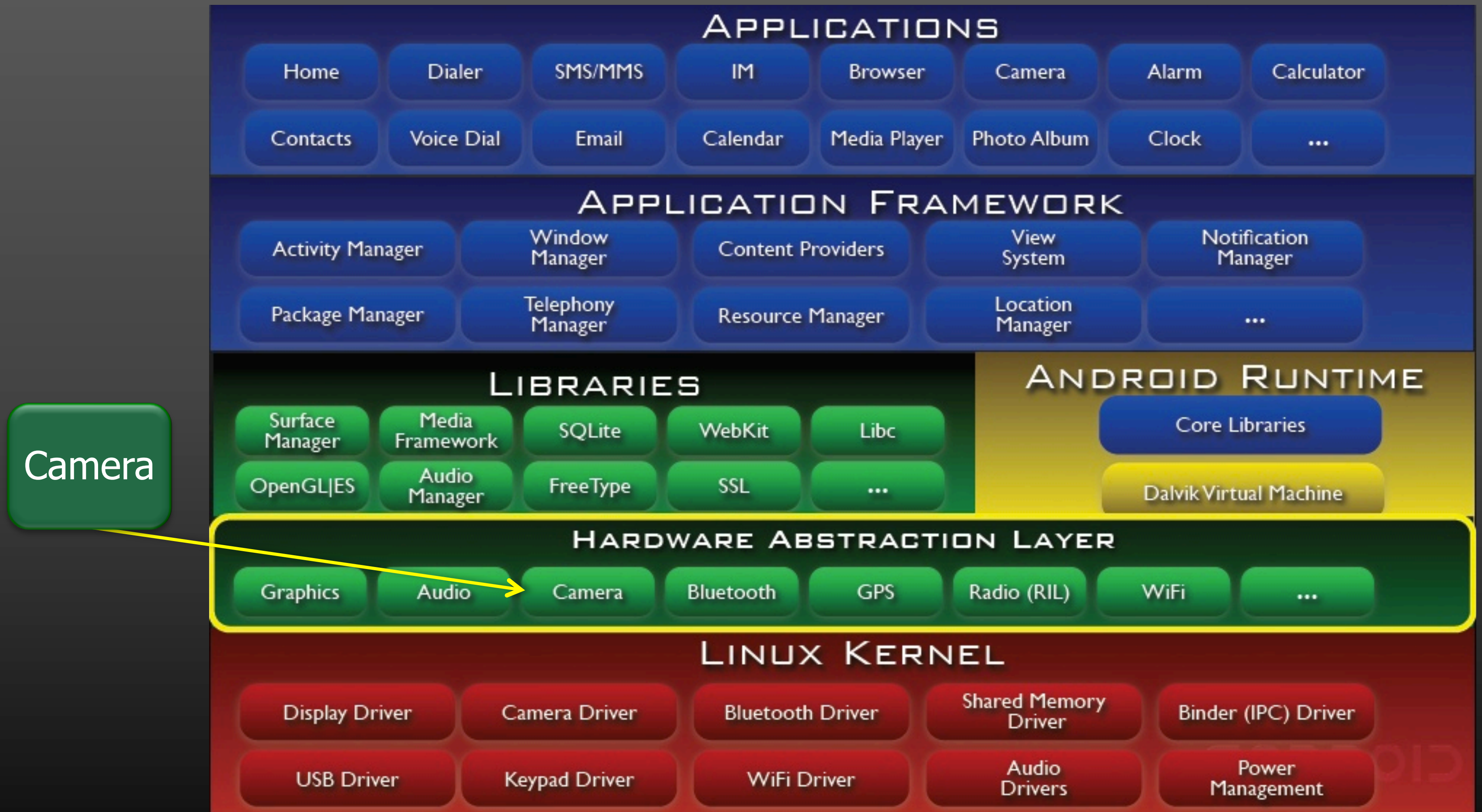
High Level Architecture

Android High Level Architecture



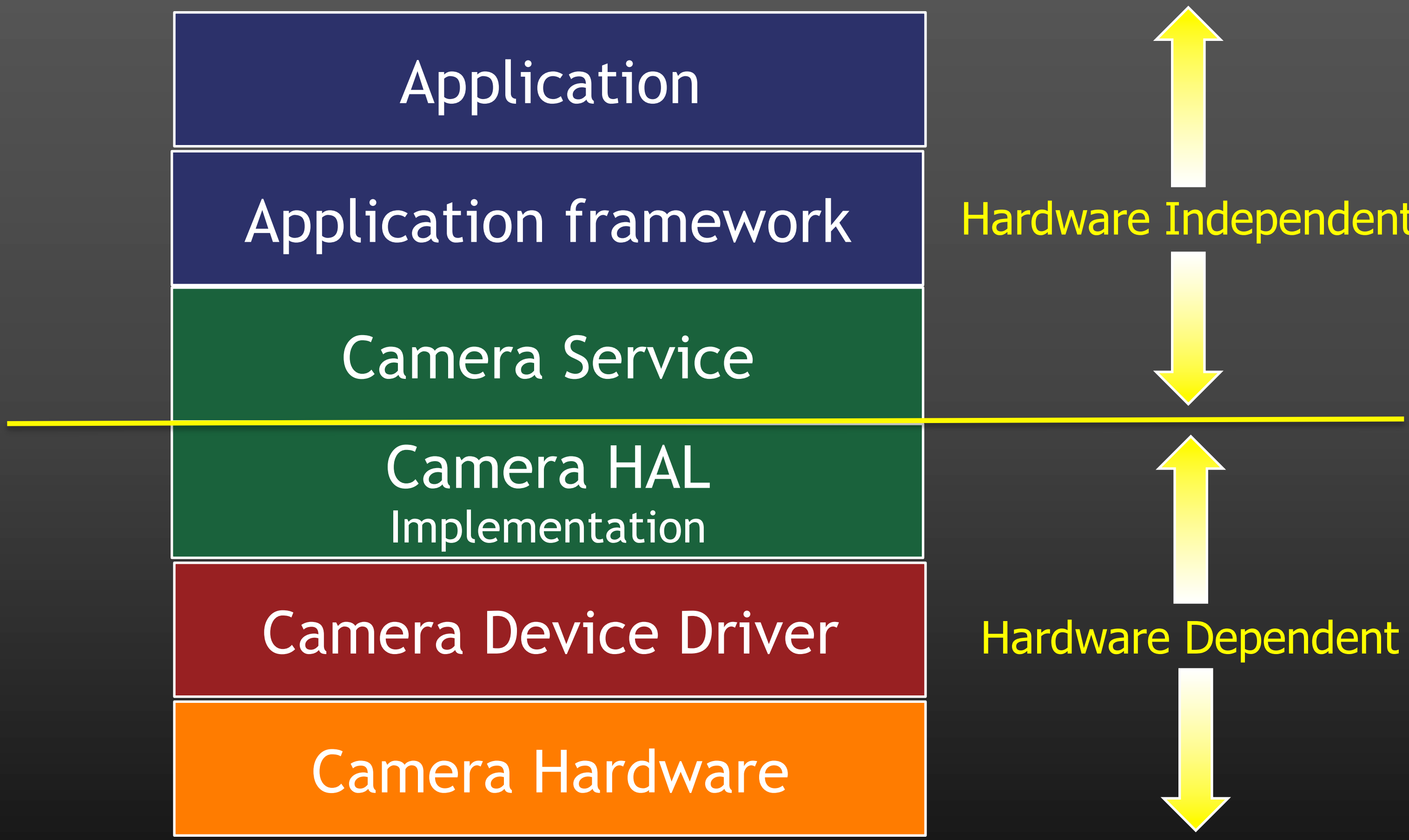
Source: Android Anatomy and Physiology, Google IO 2008

Hardware Abstraction Layer



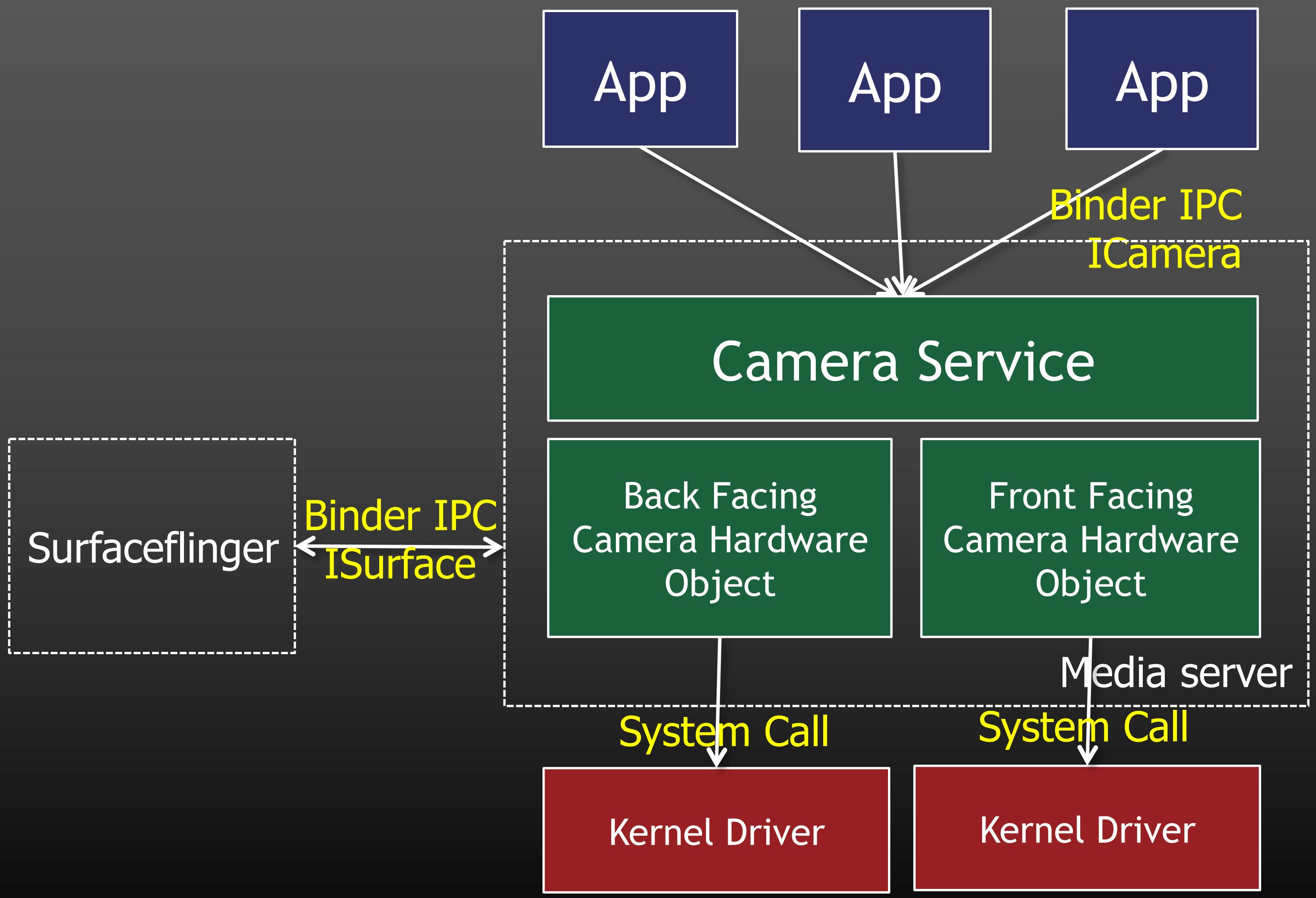
Source: Android Anatomy and Physiology, Google IO 2008

Camera Subsystem

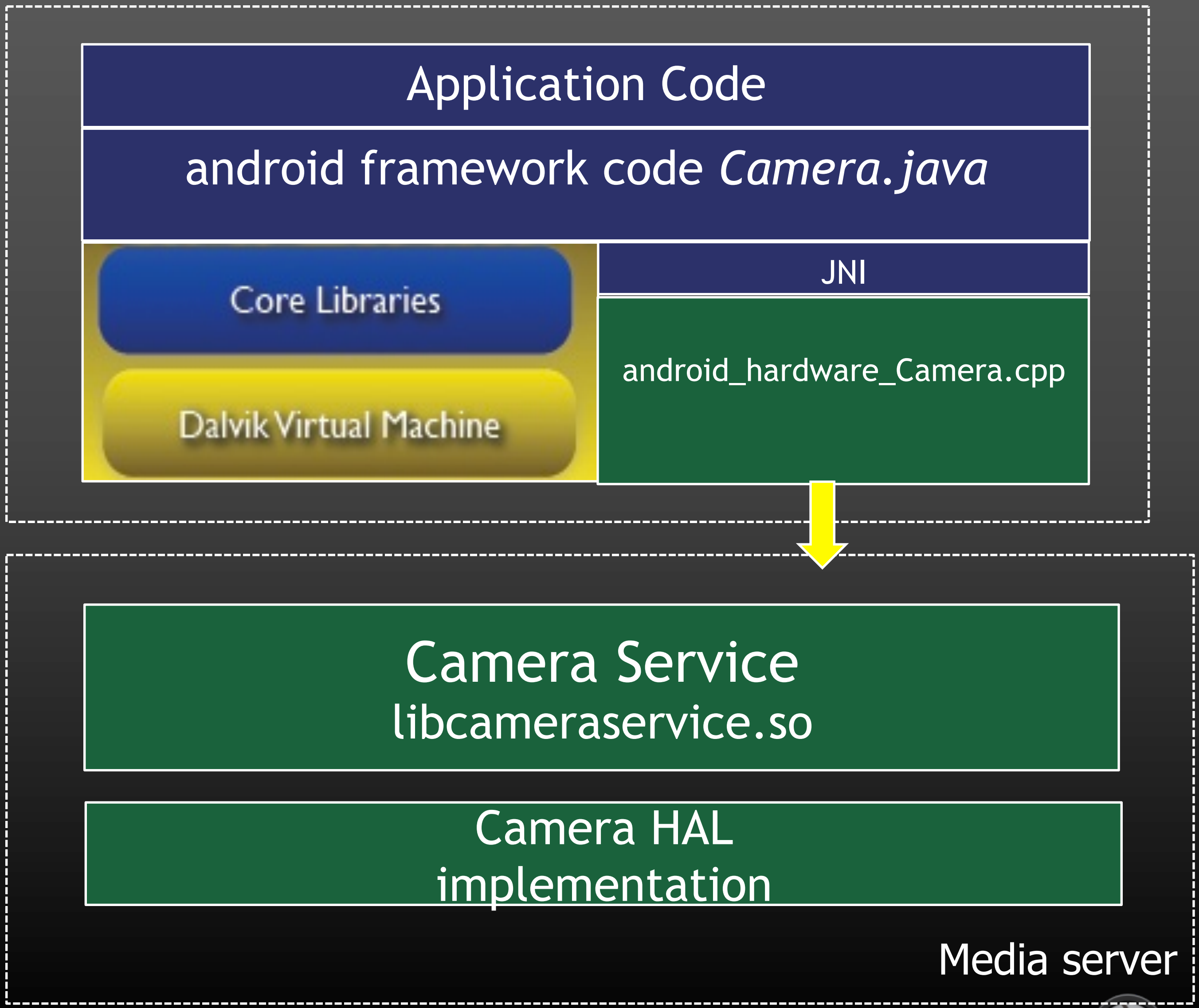


HAL = Hardware Abstraction Layer

Process View

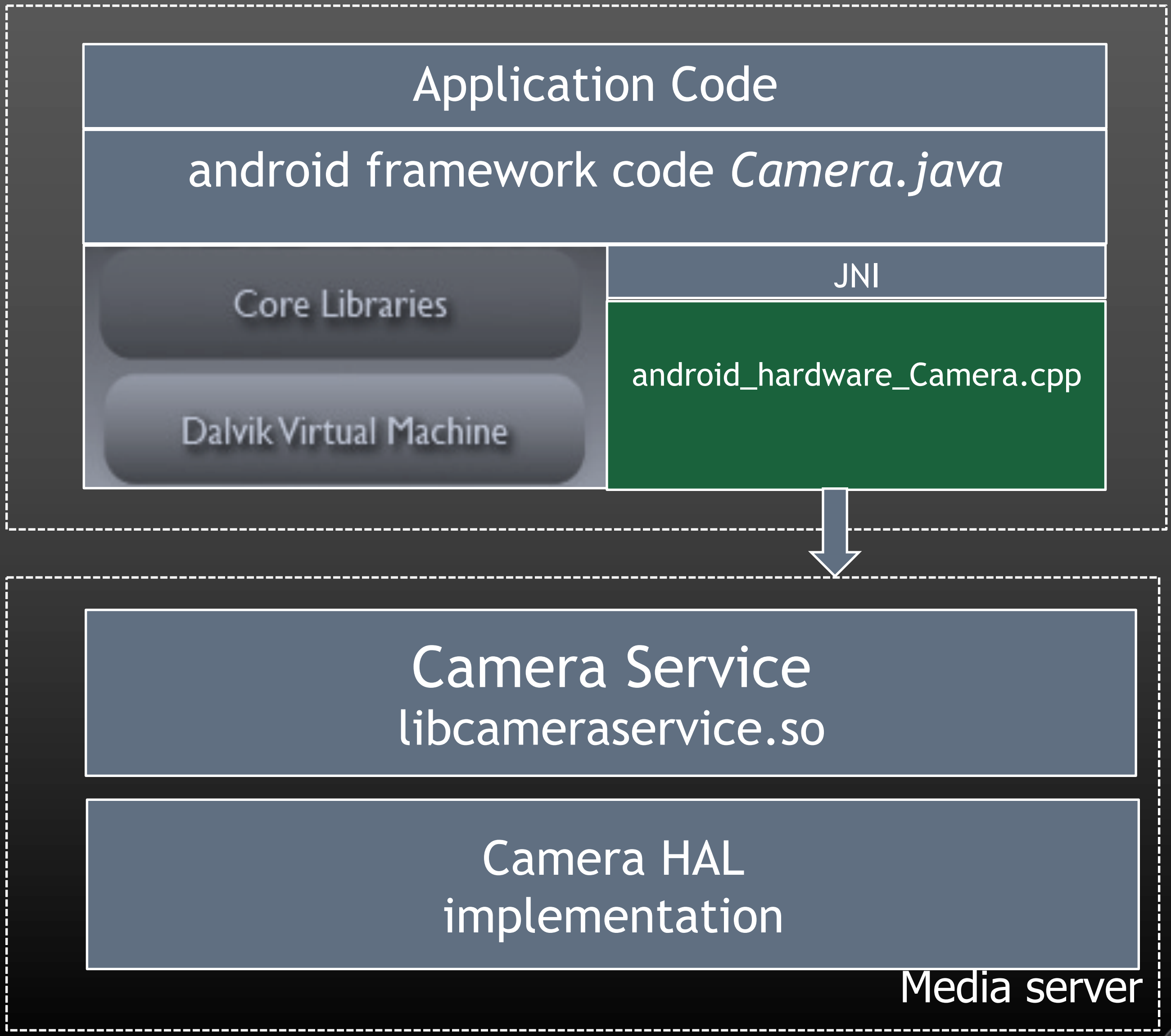


Inside the Camera App



JNI Layer

JNI Layer


Application Code
android framework code Camera.java
Core Libraries
Dalvik Virtual Machine
JNI
android_hardware_Camera.cpp
Camera Service
libcameraservice.so
Camera HAL
implementation
Media server

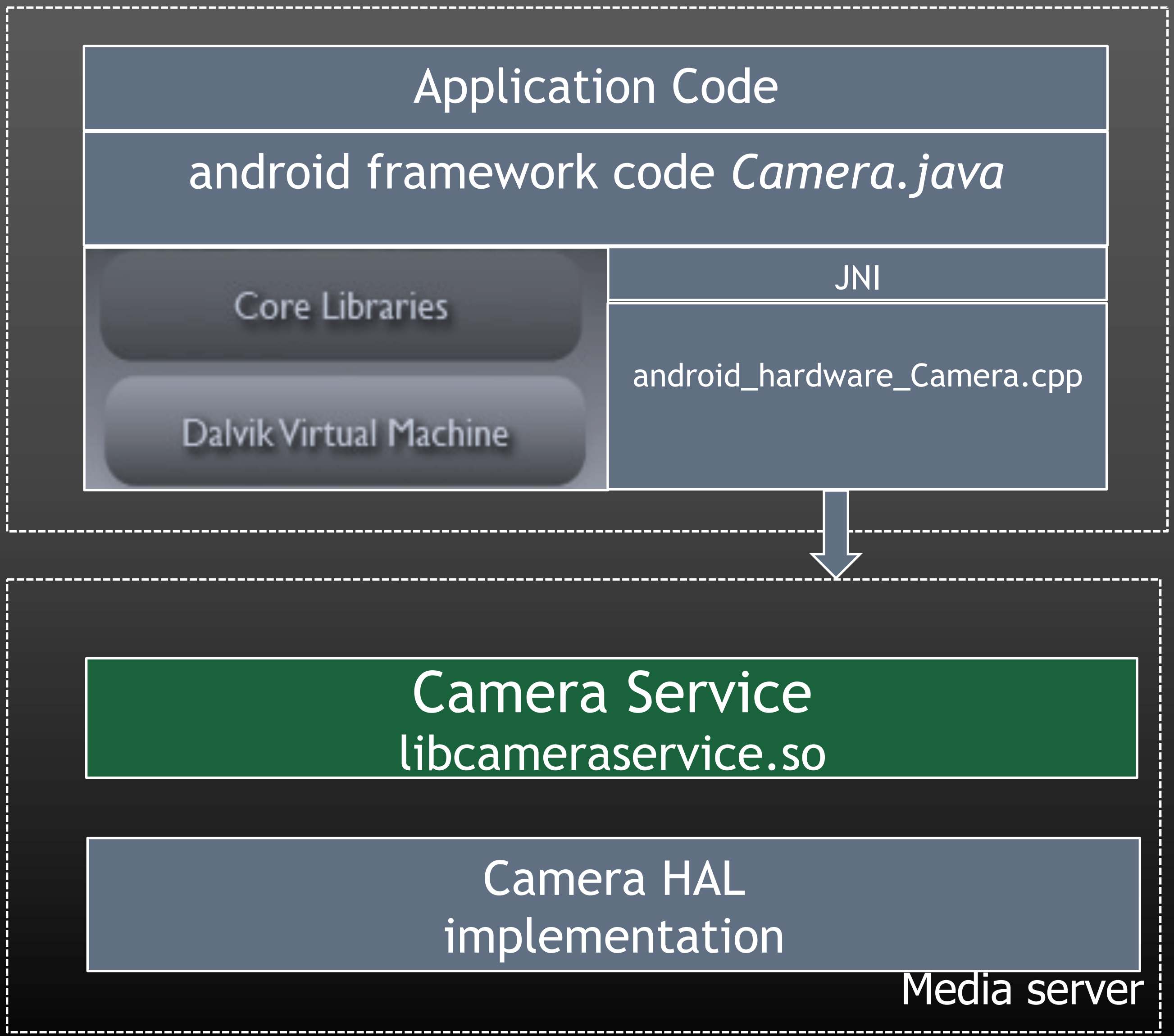
 |

android_hardware_Camera

- Creates a persistent context for callbacks from native code to Java (JNICameraContext)
- Holds references to the Java Camera, Face and Area objects.
- If a Copy of the Preview Frame is requested by the app, then the copy from native to java buffers is done here.
- Allocates Memory from the Java memory heap for JPEG images.

Camera Service

Camera Service



Camera Service

- Resource Manager for the Camera Hardware Asset
- Runs in the media server process
- It is a shared library libcameraservice.so
- Main Functions:

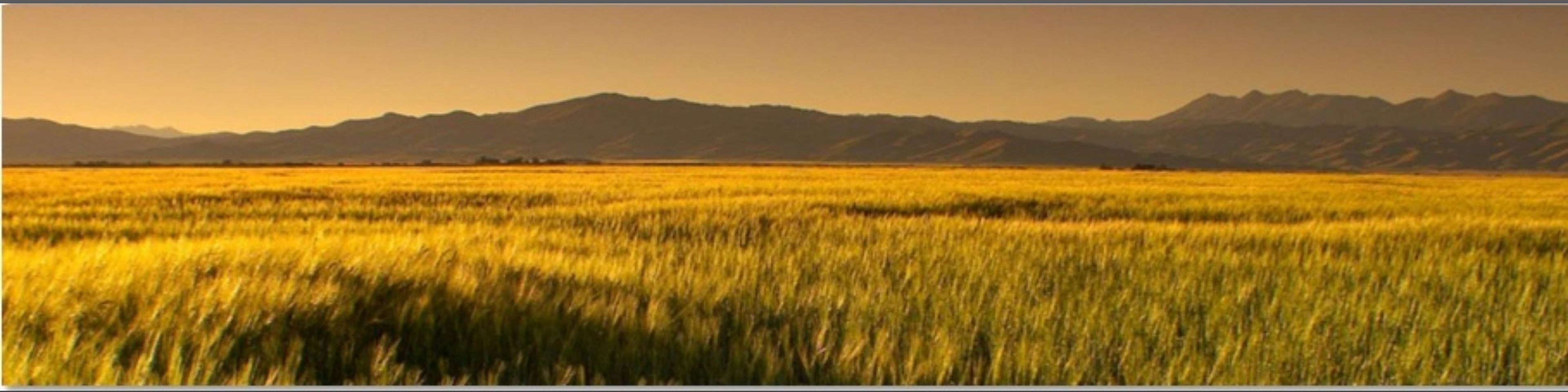
Permission check android.permission.CAMERA
Ensures only one Client connects to a Camera Hardware Object
Ensures each Process connects to a single Camera Hardware Object
Redirects callbacks back to the app layer
Accessed over IBinder Interface
Number of Cameras Available
CameraInfo Details

Android Open Source Project (AOSP) Structure
Android 4.0 (ICS)

- Android Framework
  - Java: frameworks/base/core/java/android/hardware
  - JNI: frameworks/base/core/jni
- Camera Service
  - frameworks/base/services/camera/libcameraservice/
- IBinder Interfaces
  - *frameworks/base/include/camera/ICamera.h*
- IBinder Implementation
  - *frameworks/base/libs/camera/ICamera.cpp etc.*
- Camera HAL Interface
  - *frameworks/base/services/camera/libcameraservice/CameraHardwareInterface.h*
- Camera HAL
  - hardware/<vendor>/camera (typically)

What's changed in Jelly Bean ?
Android Open Source Project (AOSP) Structure

- Android Framework
  - Java: frameworks/base/core/java/android/hardware
  - JNI: frameworks/base/core/jni
- Camera Service
  - frameworks/**av**/services/camera/libcameraservice/
- IBinder Interfaces
  - *frameworks/**av**/include/camera/ICamera.h etc.*
- IBinder Implementation
  - *frameworks/**av**/camera/ICamera.cpp etc.*
- Camera HAL Interface
  - *frameworks/**av**/services/camera/libcameraservice/CameraHardwareInterface.h*
- Camera HAL
  - hardware/<vendor>/camera (typically)



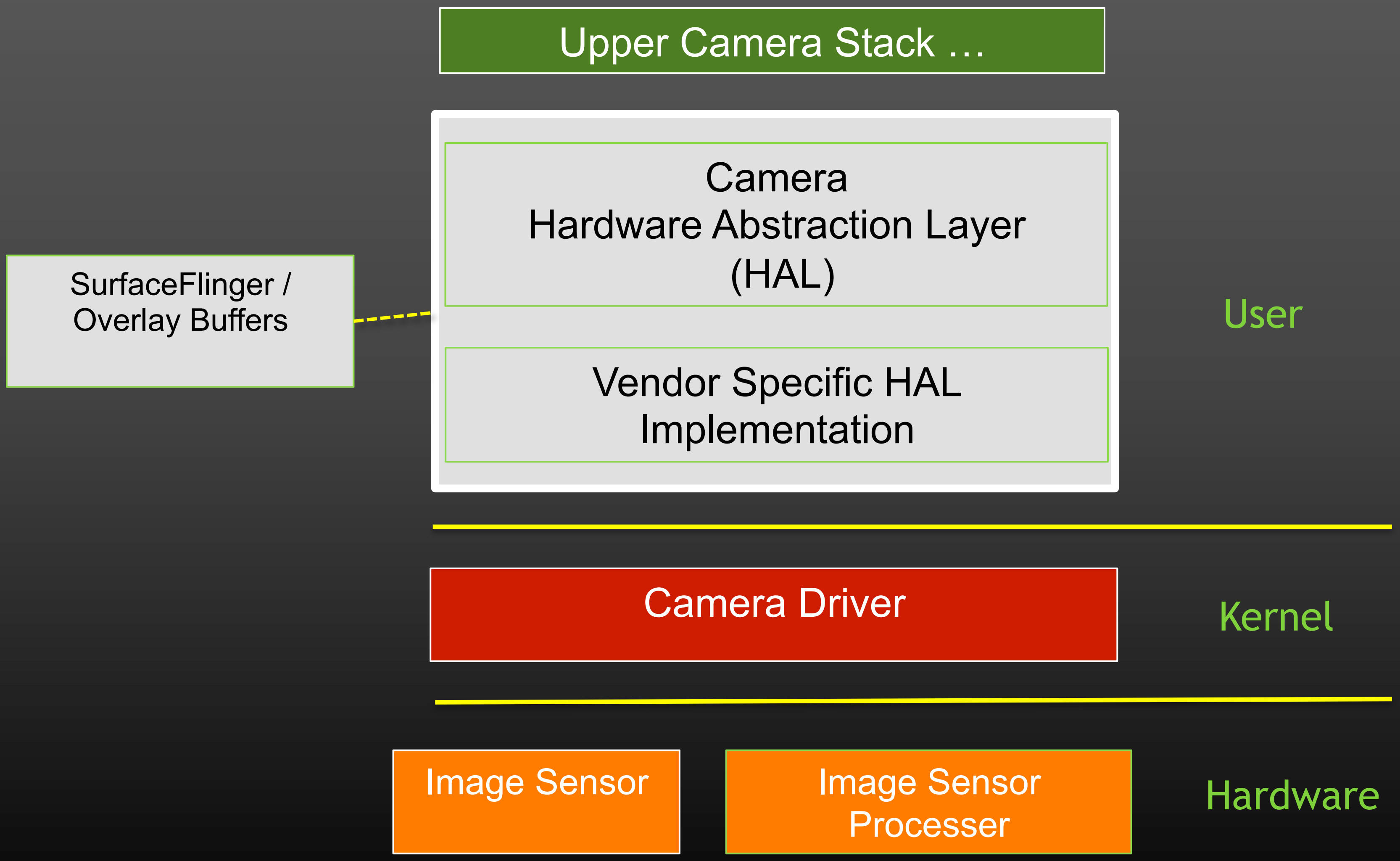
Section III

Its all about the Camera Hardware !

Camera Hardware Abstraction Layer

Review of a Typical Implementation

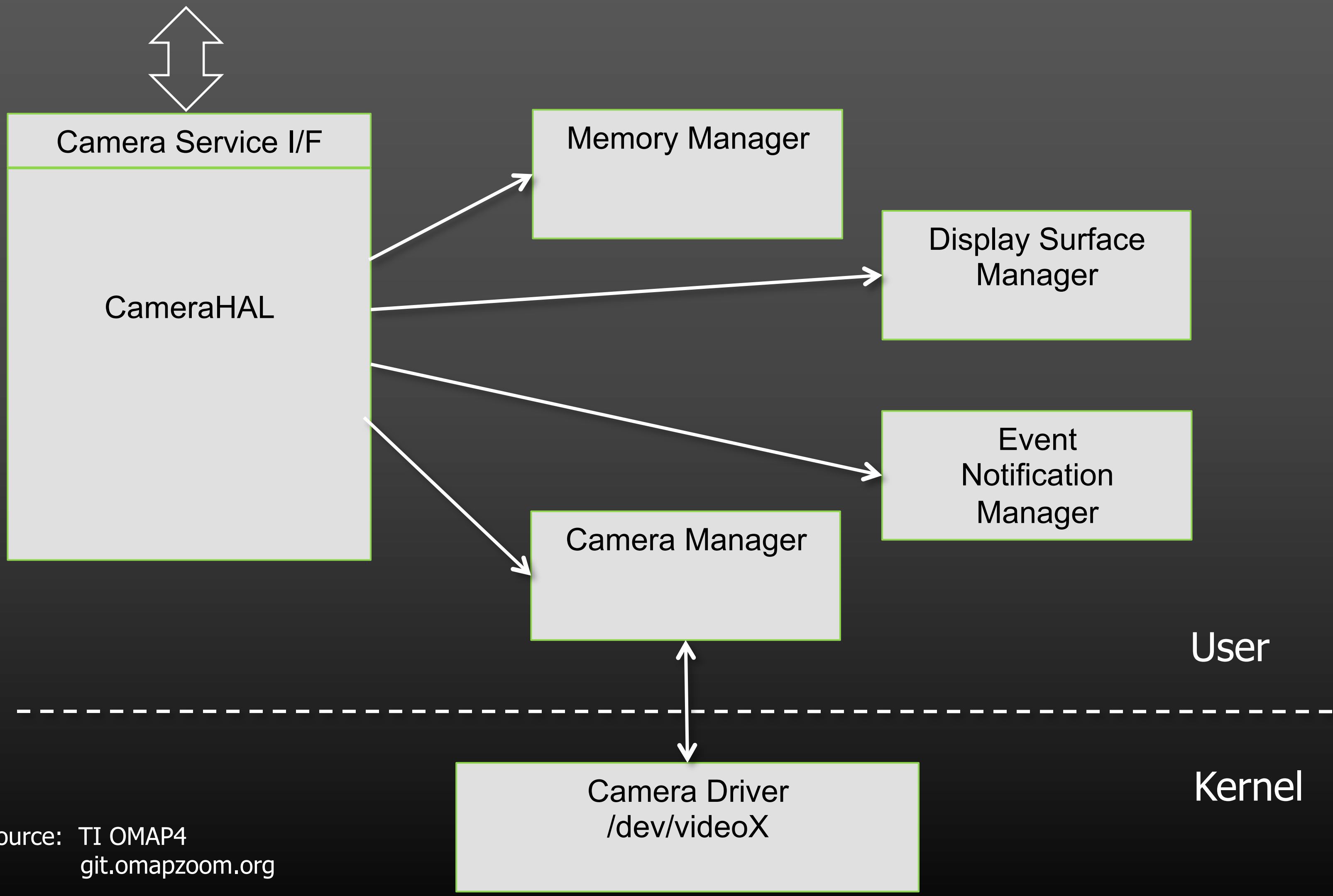
Camera Stack – Camera HAL



Android CameraHAL Library

- The Camera Hardware Abstraction Layer (HAL) is a library that is specific to the camera hardware platform
  - Written by hardware vendors (Qualcomm, TI, others)
- CameraHAL maps Android Camera Service calls to driver functions
  - Android Froyo uses CameraHardwareInterface.h wrapper
  - Ice Cream Sandwich (ICS) and above use camera.h
- CameraHAL low level interface communicates with the kernel level driver
  - It can support interfaces including Video for Linux 2 (V4L2) or OpenMax (OMX)
  - Communicates with the driver through file I/O calls (open, close, input/output controls (IOCTL), etc)

Sample CameraHAL Functional Diagram



Source: TI OMAP4
git.omapzoom.org

CameraHAL Block Diagram Discussion (1)

- Parts of the previous block diagram are hardware vendor specific
  - May be different for each vendor and target platform
- CameraHAL
  - Initialization – initialize the CameraHAL block and the target device driver
  - Camera Services interface – Handle each Camera Service request, dispatch requests to the appropriate functional block
  - Camera State machine – maintain the camera state through different API calls (e.g., preview, capture, recording, focus enable, etc).
- Memory Manager
  - Cameras are memory intensive devices
  - On request, allocate buffers for preview, capture and other functions
- Display Surface Manager
  - Controls preview and video displaying - helps to coordinate with the camera manager block

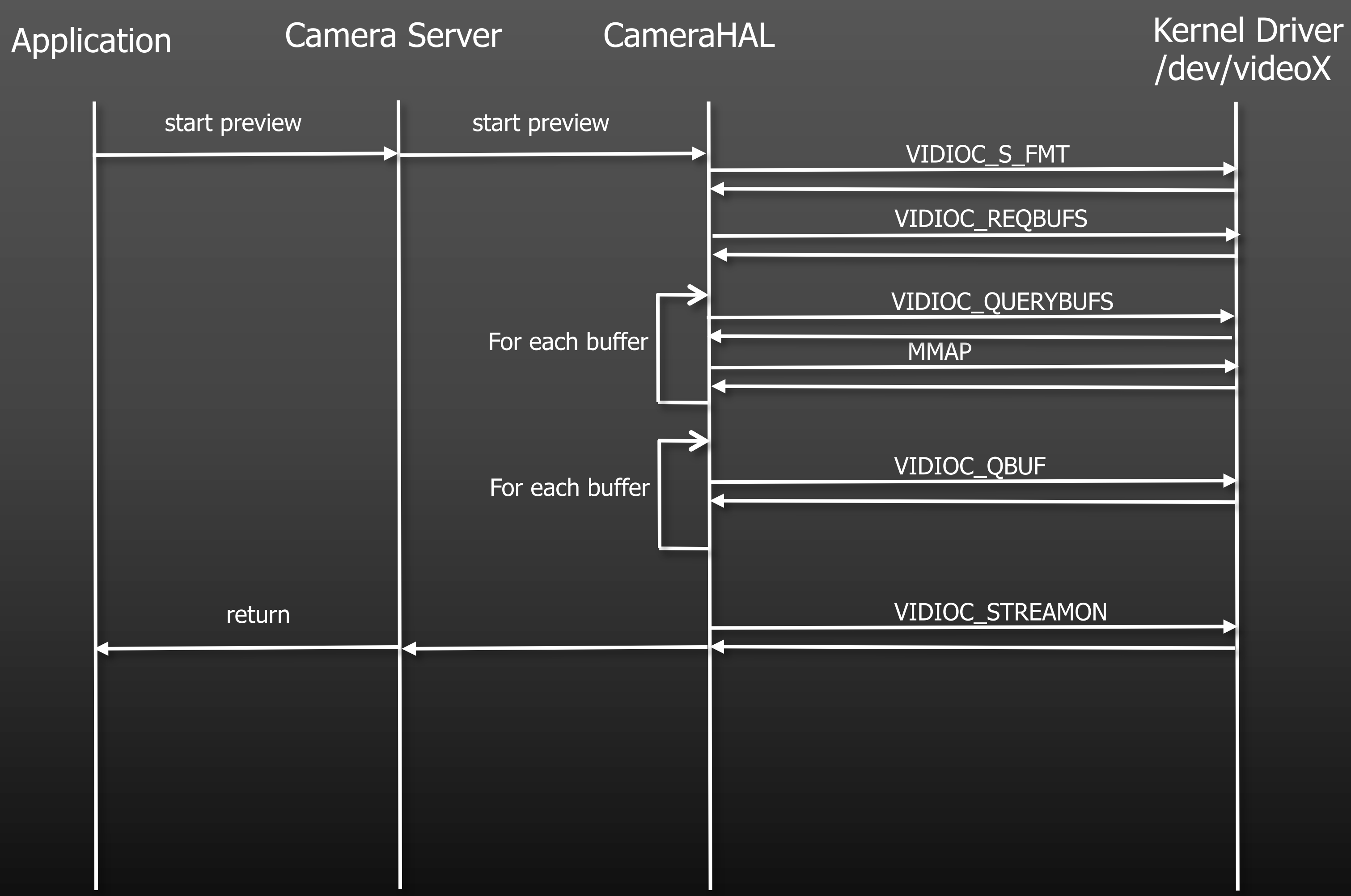
CameraHAL Block Diagram Discussion (2)

- Display Surface Manager (cont)
  - Communicates to the display when a frame is ready for preview
  - Signals to the Camera Manager when the image buffer can be re-queued
- Event Notification Manager
  - Supported callbacks include notify, data and timestamp
    - Notify – call on camera error, shutter, focus, zoom events or raw image notify event
    - Timestamp – call on video frame event
    - Data – call on preview, postview, compressed image, and other capture events
  - Call backs types are separated at the Camera Service level
- Camera Manager
  - Handle camera activities
    - Setting parameters
    - Preview and snapshot callback
  - Interface with kernel driver

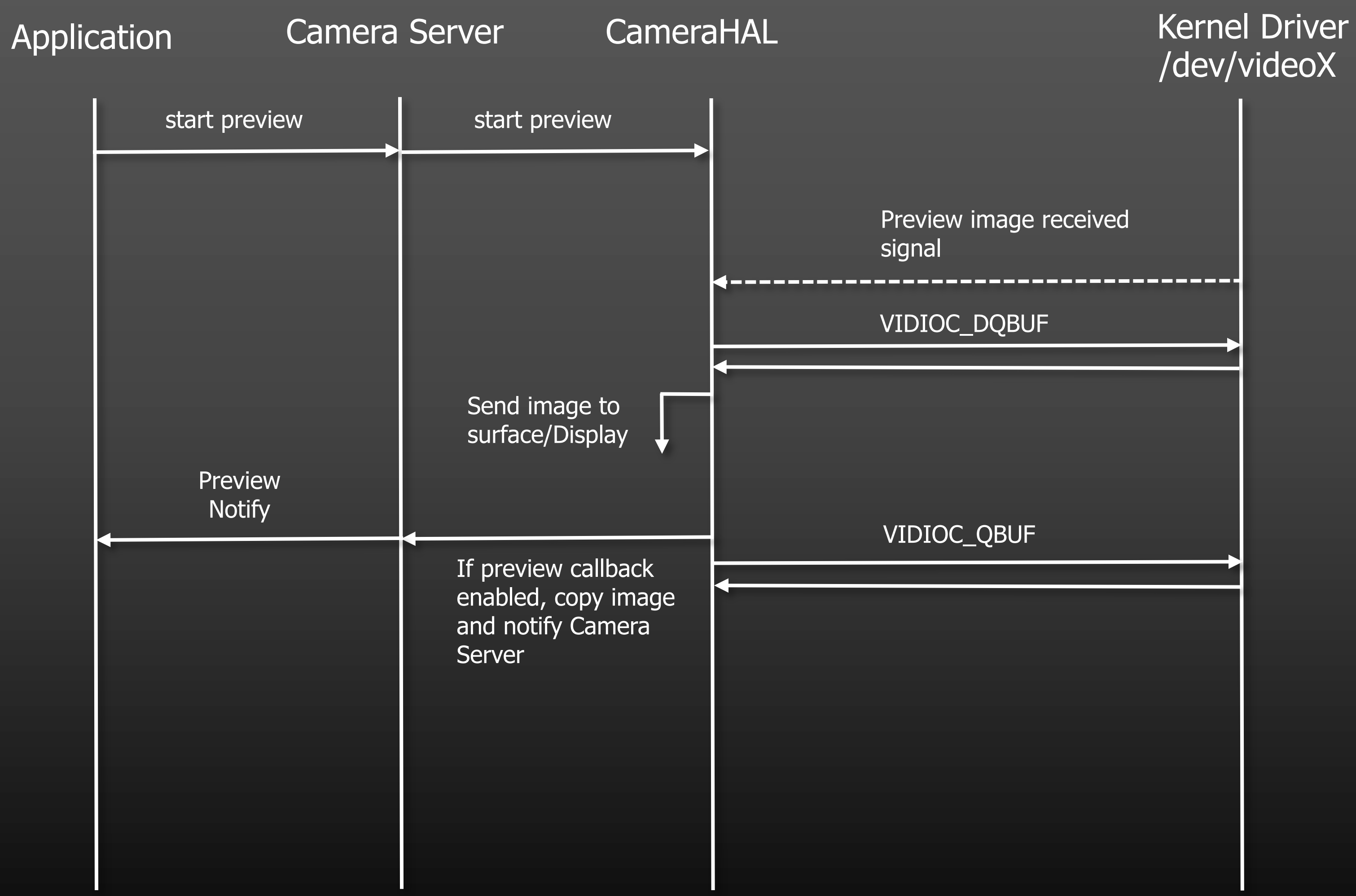
CameraHAL Preview Discussion

- The following slides discuss the preview use case
- Preview – displaying the camera image on the device display in real time
- The startPreview application call initiates image preview
  - A single application level call results in a chain of CameraHAL and driver events
- Preview continues until the stopPreview() application call
  - During preview, no application interaction unless a preview callback is registered

Preview Start Up Sequence Diagram (V4L2)



Preview Operation Sequence Diagram (V4L2)

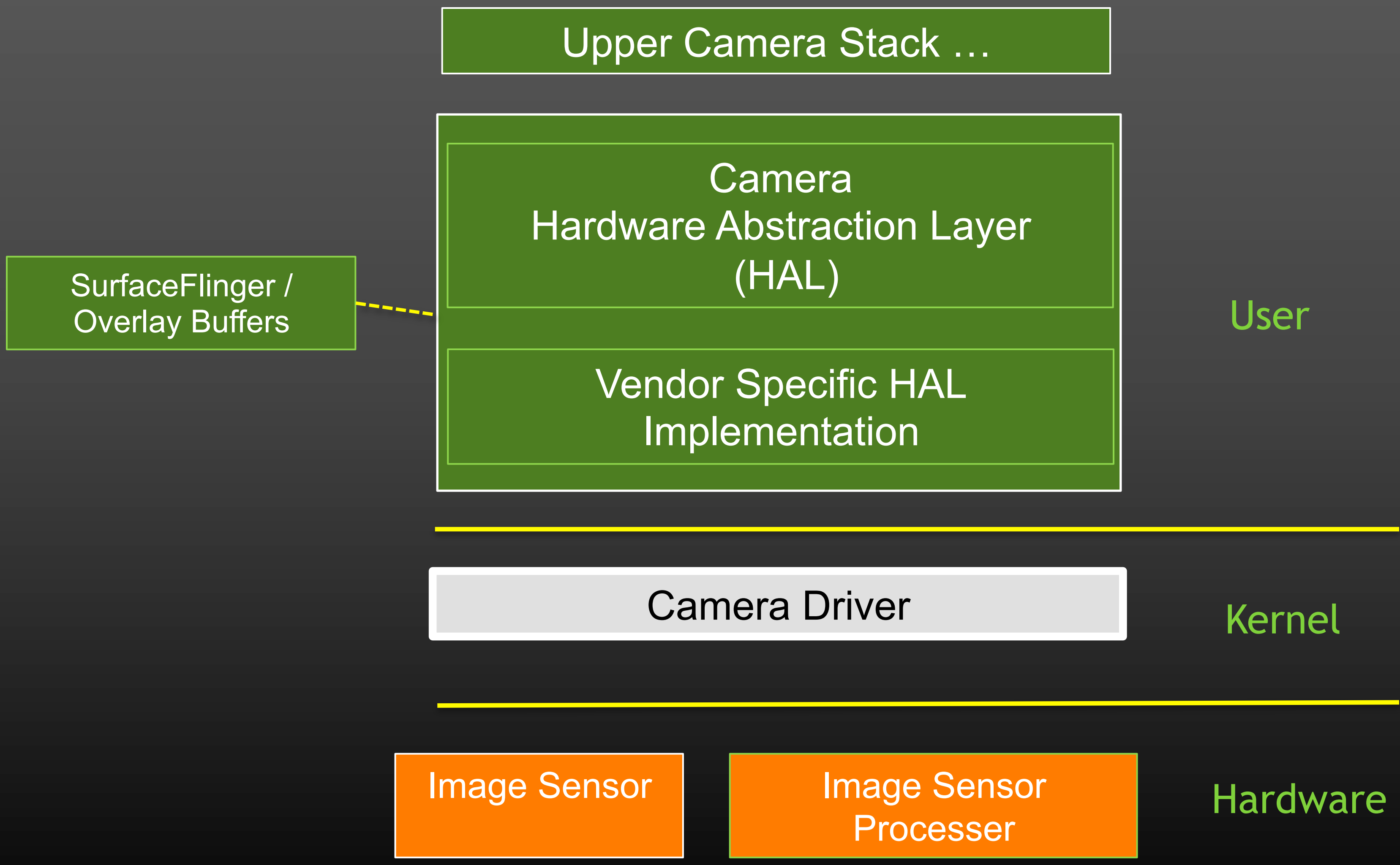


Camera Preview Interaction with the Display Subsystem

- Matching the timing of 2 events
  - Preview frames arrive asynchronously from the camera
  - The display subsystem refreshes the display at regular intervals
  - Potential mismatch between these 2 system
- Sending the preview image to the display subsystem
  - The preview frame is removed from the V4L2 queue of buffers
  - The frame is sent to the display subsystem
    - The frame memory is shared by the display subsystem
    - Or the frame is copied to a buffer for display subsystem use
  - The preview frame may be copied to a user space buffer if preview callback is enabled
  - The frame is returned to the V4L2 queue of buffers when done

Camera Device Driver

Camera Stack – Camera Driver

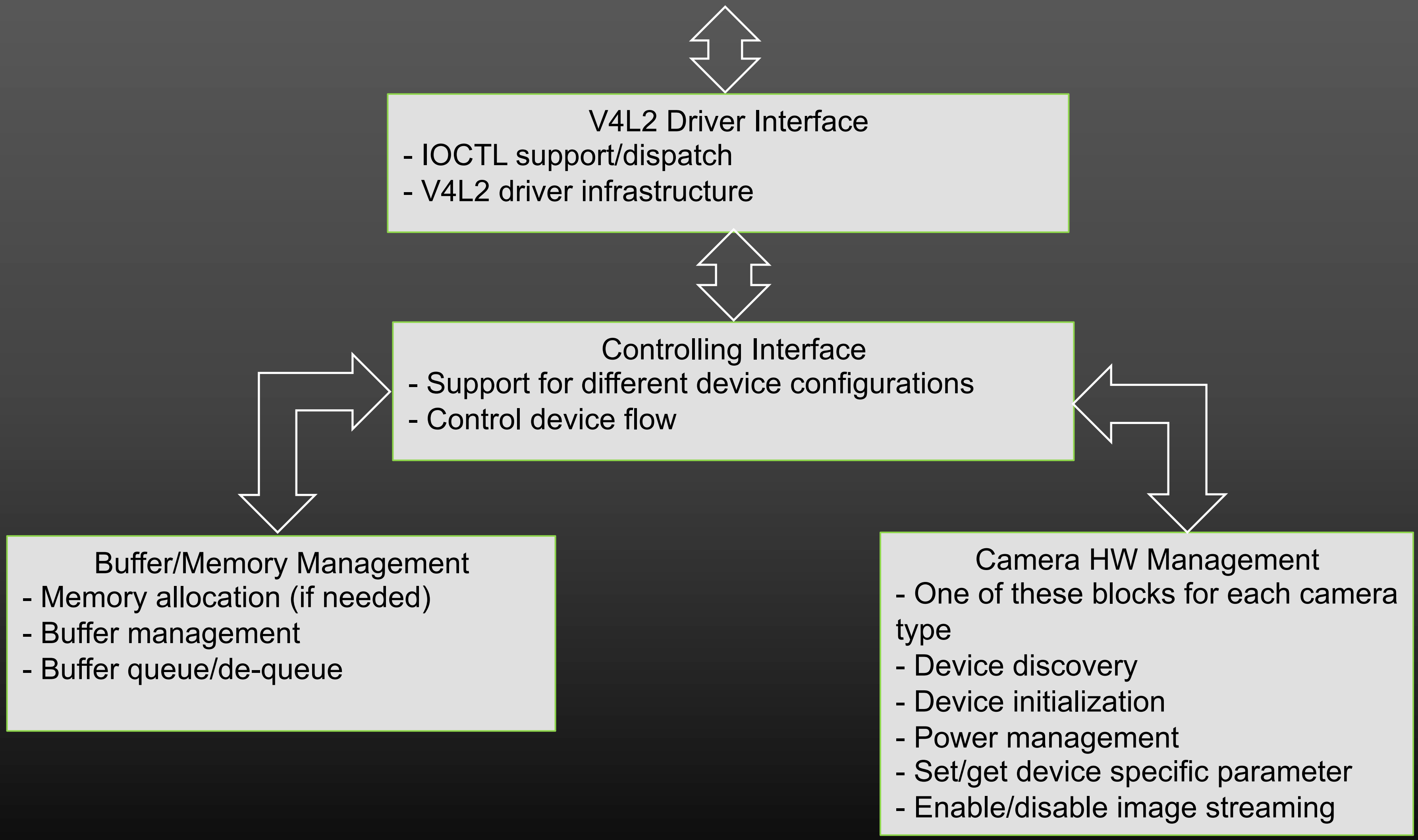


 |

Android Kernel Camera Driver

- The kernel driver presents a standard interface for different types of camera hardware
  - Camera hardware specific attributes are handled by the low level kernel driver
  - Image Sensor Processor (ISP) vs. SOC (smart) sensor - differences are handled at the driver level
- For Android, Video for Linux 2 (V4L2) is used in many implementations
  - V4L2 has been in existence for many years
  - OpenMax (OMX) is also used for a low level driver interface by some vendors.

V4L2 Kernel Driver Block Diagram



Android Linux Kernel Functionality

- Support for multiple camera types
  - Camera specific code is localized to one file
  - Compile time option to add other cameras (one driver can support many different camera hardware)
  - More cameras means longer start up times since a camera's initialization can be time consuming
- The driver manages the underlying hardware topology (e.g., ISP + sensor, smart sensor)
- For two or more cameras, the V4L2 driver creates additional device nodes
  - Devices show up as /dev/video0 (primary), /dev/video1 (secondary), …

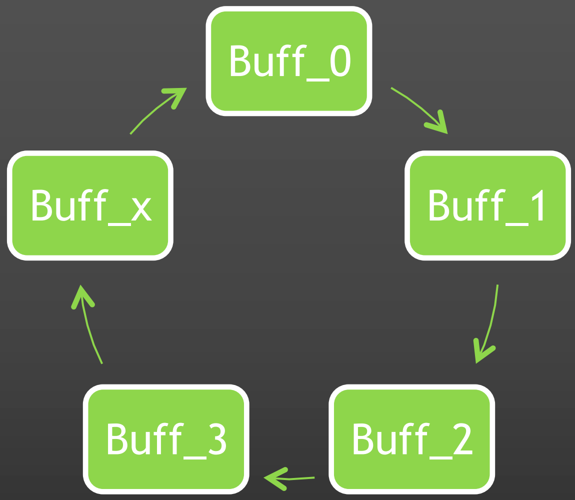
V4L2 Kernel Driver Resources

- Memory
  - Memory can be either driver-allocated or user-provided
  - The image transfers from the camera to memory through hardware Direct Memory Access (DMA)
  - Hardware memory management may be used to avoid contiguous memory requirement
- Interrupts
  - Camera ports support for interrupts on events such as frame start, finish, focus events, etc.
- Camera Control: I2C/SPI
  - I2C (Inter-Integrated Control) is used for writing or reading camera registers
  - SPI (Serial Peripheral Interface) is a faster alternative to I2C
- Control Signals/GPIO
  - All controlled by the low level driver
- Power
  - Sensor power management is critical to embedded device operation
  - Sensors support standby mode where settings are maintained while power usage is reduced

 |

V4L2 Driver Buffer Management

- One or more buffers are supported
- User buffers or kernel-allocated buffers are supported
- Buffers are treated the same for preview, capture, video (output resolution does not matter)
- Buffers are queued to a circular list
- Buffer filling starts when the V4L2 Stream_On command is executed
- Once filled, the CameraHAL de-queues a buffer, processes the buffer, then re-queues the buffer
- The Stream_Off command causes all buffer to be released



Typical V4L2 Preview Sequence (1)

- V4L2 preview start up sequence is given below

V4L2 Call	Driver Events	Hardware Events
VIDIOC_S_FMT – set format	Set image format and size	Set both resolution and output pixel format
VIDIOC_G_PARM – get parameter	Get a camera driver or hardware parameter	Read camera parameter
VIDIOC_S_PARM – set parameter	Set a camera driver or hardware parameter	Write camera parameter
VIDIOC_CROPCAP – get cropping capabilities	Return camera cropping capabilities	None
VIDIOC_S_CROP – set cropping	Set cropping rectangle	Set camera cropping rectangle
VIDIOC_REQBUFS – request camera buffer	Request buffer support from the driver (user vs. kernel)	None
Loop: VIDIOC_QUERYBUF – query buffer caps	For kernel allocated buffers, return buffer characteristics	None
V4L2_MMAP – map buffers to user space	For kernel allocated buffers, memory map to user space	None

Typical V4L2 Preview Sequence (2)

- V4L2 preview start up sequence (cont)

V4L2 Call	Driver Events	Hardware Events
Loop: VIDIOC_QBUF – queue buffers	Queue buffers in the circular queue	none
VIDIOC_STREAM_ON – start streaming	Start image capture state	Enable image output

- V4L preview shut-down sequence

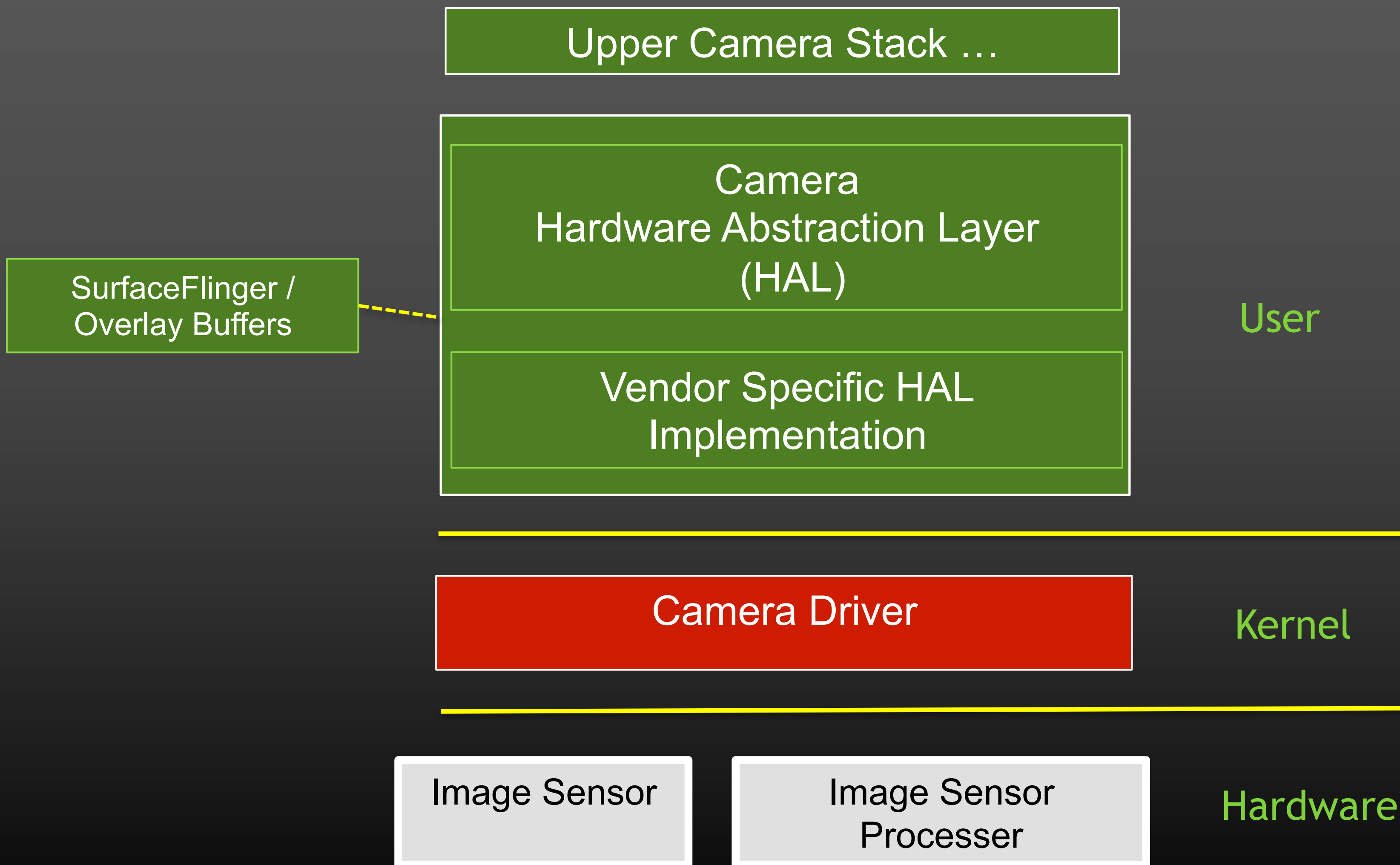
V4L2 Call	Driver Events	Hardware Events
VIDIOC_STREAM_OFF – stop streaming	Stop streaming, deallocate buffer	Disable image output

V4L2 Driver Directions

- Other Topics
  - V4L2 Media Controller Architecture
    - Exposing the hardware image processor to the calling application
    - Allows for greater programmer control
    - Supported only on open source architectures
  - Proprietary ISP software moves to user space
    - Many ISP providers wish to hide their hardware
    - Moving ISP code to user space handles this (avoid kernel open source issues)
- Driver source code location:
  - {kernel sources}/drivers/media/video

Camera Hardware Overview

Camera Stack – Camera Hardware



 |

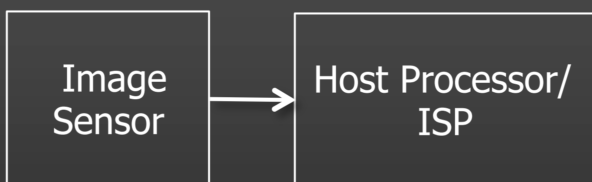
Camera Hardware Introduction

- Types of Image Sensor Hardware
  - Raw or Bayer Sensor
    - Outputs a Bayer image
    - Limited or no Image processing capability
    - Requires host ISP
    - Simple controls from the host system
  - Smart or System On a Chip (SOC) Sensor
    - Outputs a processed image
    - Image processing occurs on-chip (built in ISP)
    - No host ISP required
    - Complex controls from the host system

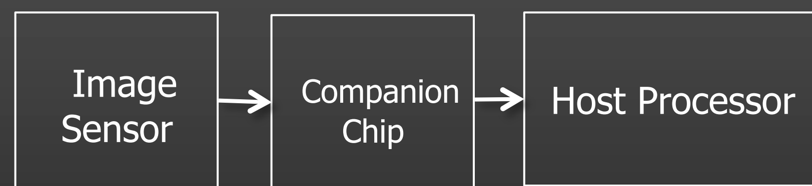
Bayer Sensor Discussion

G1	R	G1	R
B	G2	B	G2

- Outputs a Bayer (unprocessed) image
- Used with internal or external ISP
  - Internal ISP – System Processor and ISP bundled together
  - External ISP – External companion chip



Host Internal ISP

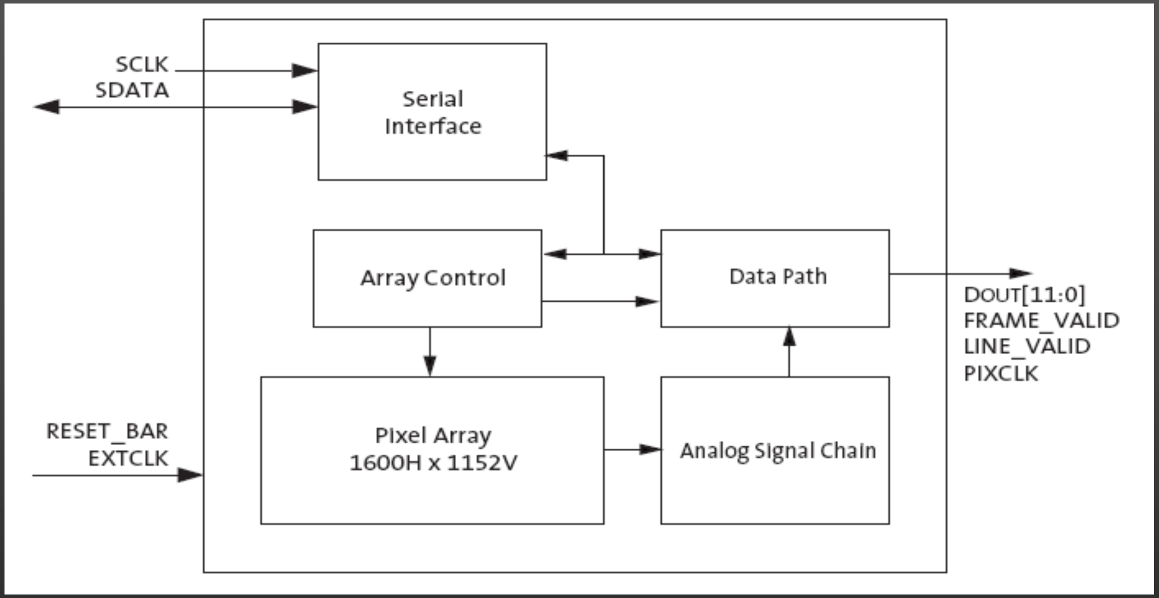


External ISP + Host

- Sensor controls include exposure time and analog/digital gains
- ISP controls high level parameters (exposure, white balance, lens shading, noise filtering, resize/zoom, others)

Bayer Sensor Block Diagram

- Example - MT9M032 – 1.6MP Image Sensor



- Uses an Electronic Rolling Shutter for image readout
  - One line of the pixel array read at time
- Pixels are output from the sensor one pixel at a time
  - 8/10/12/14 bits per pixel

SOC Sensor Discussion

- Outputs a processed image such as YUV, RGB or JPEG

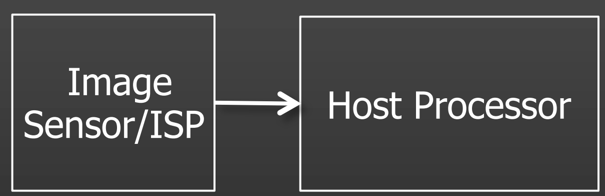
Cb_i	Y_i	Cr_i	Y_{i+1}

Ycbcr Pattern

$R_7R_6R_5R_4R_3G_7G_6G_5$	$G_4G_3G_2B_7B_6B_5B_4B_3$

RGB565 Pattern

- Does not require a host ISP – ISP is built into the sensor

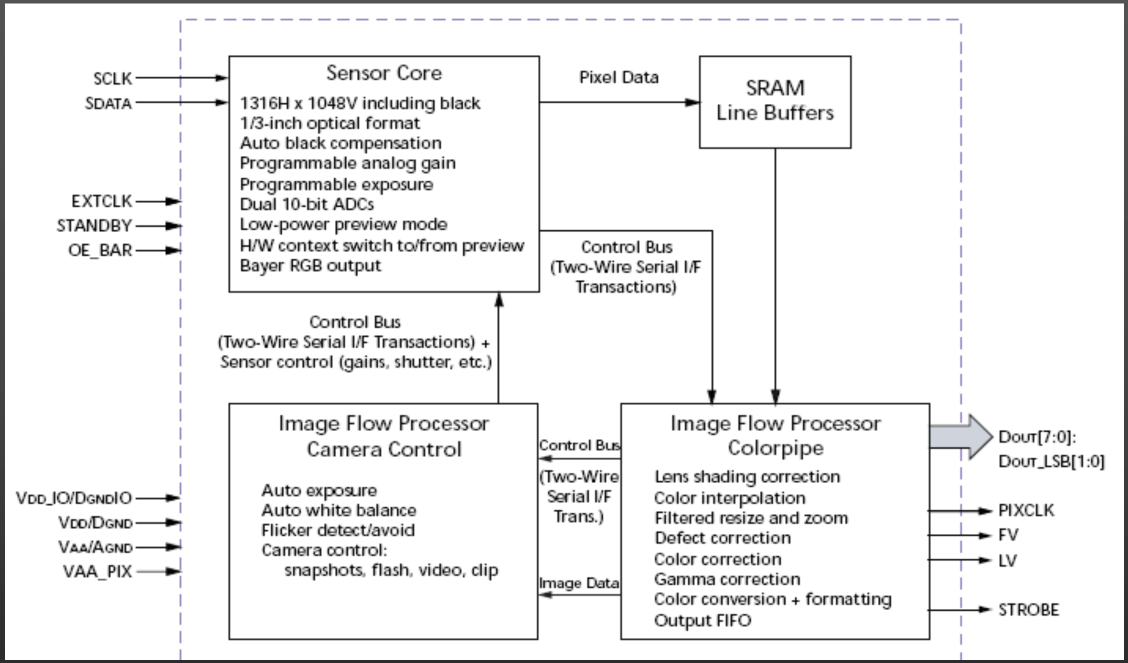


SOC + Host

- Sensor controls include exposure, white balance, lens shading, noise filtering, resize/zoom, others

SOC Sensor Block Diagram

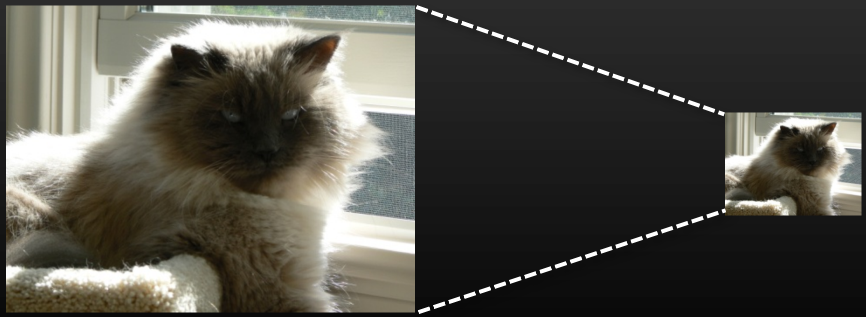
- Example - MT9M131 – 1.3 MP Image Sensor



- Also uses an ERS for image readout
  - One line of the pixel array read at time
- Pixels are output from the sensor one pixel at a time
  - 8/16 bits per pixel

Preview at the Camera Level

- Several methods used at the hardware level to go from a full sized image to a preview sized image
  - Skipping – skipping 1 or more rows and columns to reduce image size (good for power savings)
  - Binning – combining several pixels into one pixel
  - Scaling – reduce the image size using hardware scaling algorithms (best for image quality)



A Peek into the Future

Camera Application Trends

- Android Applications – memory limitation 16MB ~ 24MB
  - Higher pixel sizes and Bursty modes put a strain on the system
- Computer Vision Applications go mainstream
  - APIs on Object Tracking, Gesture Recognition become more common place
- Computation Photography application
  - Developers get fine grained control of flash and camera

Camera Hardware Trends

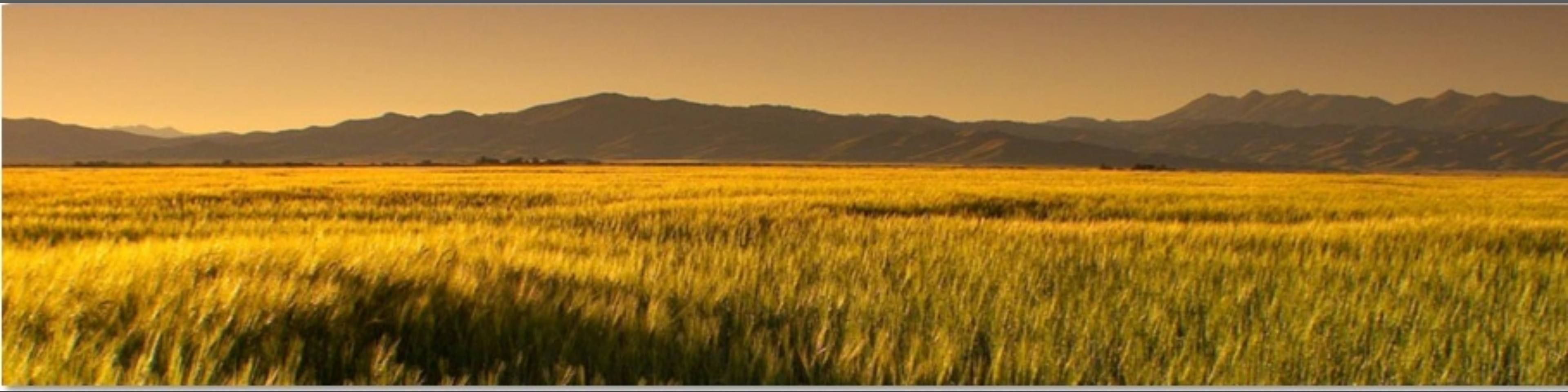
- Back Side Illumination(BSI) vs. Front Side Illumination(FSI)
  - BSI can add up to 30% more light gathering capability
- Smaller Pixels
  - Constant push to reduce pixel and sensor package sizes
- Faster data output rates, higher clock speeds
  - 1080p30, 1080p60
  - Serial data interfaces enable increased sensor output speeds
- High Dynamic Range
  - Ability to capture larger exposure range
- 3D Imaging
  - Use of 2 cameras to generate a 3D image

Q&A

References

- http://developer.android.com/
- http://www.codeaurora.org
- http://omappedia.org
- http://source.android.com
- http://stackoverflow.com/questions/10775942/android-sdk-get-raw-preview-camera-image-without-displaying-it
- http://www.cjontechnology.com/blog/?p=14

Aptina™
IMAGING



Backup Slides

Interaction with the Media Subsystem

- ICameraRecordingProxy and ICameraRecordingProxyListener were introduced in Android 4.0
- Allow apps to use the camera subsystem while the MediaRecorder is recording the video frames.
- ICameraRecordingProxy is a proxy of Icamera
  - startRecording
  - stopRecording
  - releaseRecordingFrame
- ICameraRecordingProxyListener is an interface that allows the recorder to receive video frames during recording.
  - dataCallbackTimestamp

More on Camera Service (ICS)

- Android.mk file
  - frameworks/base/media/mediaserver/Android.mk

```
LOCAL_SHARED_LIBRARIES := \
        libaudioflinger \
        libcameraservice \
        libmediaplayerservice \
        libutils \
        libbinder
```

- Gets instantiated as along with other components of the media server

```
AudioFlinger::instantiate();
MediaPlayerService::instantiate();
CameraService::instantiate();
AudioPolicyService::instantiate();
```