You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting: http://www.intel.com/design/literature.htm

Intel, Atom, and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2017, Intel Corporation. All rights reserved.
# Revision History

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>June 2017</td>
<td>001</td>
<td>Initial release of MR2.</td>
</tr>
</tbody>
</table>

§
1.0 Introduction

The software stack to develop an application to capture images through MIPI-CSI camera sensor is highly convoluted and Operating System (OS) dependent. The layers of software could even be different for different versions of the Android* operating system. Layers of software for example, libCameraHal, media-Control, firmware need the right support and implementation complying with underlaying the Camera controller hardware and its kernel driver.

Figure 1. Typical Software Stack for Capturing Image using MIPI-CSI Camera

Developing the complete stack makes sense for commercial high-volume applications using RAW sensors, but too much for applications that are meant for testing, experiments, and other low volume use-cases.

V4L2 is a popular subsystem in the Linux* kernel providing standard interfaces for cameras, TV tuners, videos, and radios and so on for subsystems. It standardizes, to a big extent, the interface between user-applications and controller drivers and can directly interact with simple user-space applications. Being part of the Linux-kernel it is available across all versions of Linux and Android* operation systems. This document serves as a guide to develop simple V4L2 camera applications and takes Intel Atom® x3- C3200RK Processor (Formerly SoFIA 3G R), Camera Interface Controller(CIF) Image Signal Processor (ISP) 2.0 as a reference. Such applications can be highly useful for enabling smart ISP-integrated-sensors that do not require any performance tuning through 3A algorithms.
## 1.1 Terminology

### Table 1. Terminology

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSP</td>
<td>Board Support Package</td>
</tr>
<tr>
<td>CIF</td>
<td>Camera Interface Controller, another name for IPU</td>
</tr>
<tr>
<td>ION</td>
<td>A memory management mechanism to allocate physically contiguous memory pool</td>
</tr>
<tr>
<td>IPU</td>
<td>Image-Processing Unit</td>
</tr>
<tr>
<td>ISP</td>
<td>Image Signal Processor</td>
</tr>
<tr>
<td>MIPI-CSI</td>
<td>Mobile Industry Processor Interface</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>MMU</td>
<td>Memory Management Unit</td>
</tr>
<tr>
<td>RAW</td>
<td>Minimally processed or unprocessed format of image data received from camera sensors</td>
</tr>
<tr>
<td>V4L2</td>
<td>Video For Linux* subsystem</td>
</tr>
</tbody>
</table>
2.0 **CIF ISP 2.0 Hardware Features**

The following items are features of the CIF ISP 2.0 Hardware.

- A fix-function based hardware image-processing solution that processes input images from an image sensor and stores outputs into system memory.
- Supports a maximum of two cameras, but cannot be operated concurrently.
- Maximum horizontal image resolution is limited to 4256 pixel, which is sufficient to cater for up to 13 M Pixel.
- Vertical resolution can be up to 4096 pixel.
- The primary camera's MIPI CSI-2 consists of four data lanes capable of transferring 1Gbit/s each, the secondary camera is limited to one data lane at 1 Gbit/s.
- The ISP's architecture allows the processing of image data without the need of temporary buffers in system memory.
- The ISP comprises an internal full speed JPEG encoder, which allows outputting JPEG data directly to system memory. Alternatively it can generate YCbCr or RGB data.
- Processing data from system memory, used for debugging and verification purposes, is possible through a special read-back path.
- Image input data formats: RAW (Bayer Pattern) RGB, YCbCr.
- Data agnostic mode: (also known as Bypass Mode) in this mode, data is forwarded to system bus without any processing, for example, used with SoC camera modules.
3.0 CIF ISP 2.0 Kernel Driver Architecture

Intel Atom® x3- C3200RK Processor CIF Driver provides V4L2, the V4L2 sub device, and Media controller interfaces towards the user space. The sub device drivers (sensors, lens, and flash) provide V4L2 sub device interface towards the user space as well as towards the CIF driver. The number and purpose of V4L2 device nodes depends on memory I/Os available in the CIF - each of them is a V4L2 device node.

Any driver in the system is independent of the other drivers in the system. Where control is required between different drivers it is achieved through a device-independent interface, V4L2 sub device - this allows any sensor driver to be used with any lens and any ISP driver. It's likely that several ISP drivers are used with a single sensor driver.

The interface of relevance for this paper is only the V4L2 IOCTL-based interface which a simple capture application for a smart sensor can be developed.

Note: V4L2 sub device and media-controller interface are out of scope of this document.

The CIF driver architecture design is presented in Figure 2, where various CIF paths, main path, secondary path, and read-back path, are modeled as different V4L2 video nodes. Image data is handled via the video nodes (green boxes in Figure 2). V4L2 sub-device framework is used to provide the control interface to configure the underlying hardware.
3.1 Video Device Nodes

Table 2 shows the V4L2 device notes, buffer type, and usage.

<table>
<thead>
<tr>
<th>Device Nodes</th>
<th>Buffer Types</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dev/video0 - SP</td>
<td>type=V4L2_BUF_TYPE_VIDEO_OVERLAY</td>
<td>Viewfinder images</td>
</tr>
<tr>
<td>/dev/video1 - ISP</td>
<td>Not applicable</td>
<td>This node is for collecting image data from the sensor for 3A algorithm tuning purpose, which is relevant only for RAW sensors and which is beyond the scope of this document.</td>
</tr>
<tr>
<td>/dev/video2 - MP</td>
<td>type=V4L2_BUF_TYPE_VIDEO_CAPTURE</td>
<td>Still image captures.</td>
</tr>
<tr>
<td>/dev/video3 - RB</td>
<td>type=V4L2_BUF_TYPE_VIDEO_OUTPUT</td>
<td>Images from system memory.</td>
</tr>
</tbody>
</table>
3.2 **V4L2 Sub-Device Nodes**

V4L2 sub device and media-controller interface are out of scope of this document.

3.3 **User Space Interfaces and Functionality Provided by CIF Kernel Driver**

- **V4L2 Interface**

  Video buffer management, including format configuration belongs to V4L2 video device nodes. Cropping, binning, and scaling configuration along the image pipe is done using the V4L2 interface with the standard IOCTLs for the purpose: VIDIOC_SUB_DEVICE_S_FMT and VIDIOC_SUB DEVICE_S_SELECTION.

  When using the V4L2 interface, the input device is specified by the user. All other aspects of the data flow pipe are handled within the driver.

- **V4L2 Sub device and Media Controller:**

  V4L2 Sub device and Media Controller are control interfaces supported by the CIF driver, which are interfaces that abstract access between ISP and component devices such as sensors, lens controllers, and flash controllers. They amend the functionality of the V4L2 interface, but do not replace it.

  **Note:** This guide focuses only on developing application using only the V4L2 interface. V4L2 Sub device and Media Controller Interface are out of scope of this document.
4.0 Memory Management and Buffer Sharing

The CIF ISP 2.0 does not have a Memory Management Unit (MMU) for memory access and therefore allocating physically contiguous memory is required. For allocating contiguous memory locations, an ION mechanism has been implemented. Carve-out memory for the camera reserves the Intel® Architecture (IA) firmware during system start and later used for the camera system framebuffer.

ION manages the CIF ISP 2.0, a generalized memory manager, which supports sharing of this buffer with the GFX/video components. The size of reserved memory is set to a value equal to that of a use-case that requires the largest memory.

This section demonstrates the calculation that went behind deciding the amount of carve-out memory for camera framebuffer. It also acts as a guideline for application to request correct amount of memory from ION memory manager, based on use-case.

4.1 Calculate Memory Required for Various Use-Cases

The size of required memory can be calculated as below for various use cases:

1. Preview (720p):
   - **Viewfinder**: (1280x720 pixels) * 1.5 byte/pixel for YUV420 format * 8 buffers’ depth = 10.8MB

2. Still image capture (13MP output)
   - **Image**: 13M * 1.0 = 13MB (1.0 byte/pixel is assumed to be maximum size for JPEG)
   - **Postview**: 0.9M * 1.5 = 1.35MB (1.5 byte/pixel is for YUV420 format)

3. Video Capture (1080p)
   - **Video**: (1920x1080 pixels) * 1.5 byte/pixel for YUV420 format * 8 buffers’ depth = 24MB
   - **Viewfinder**: 0.9M * 1.5 * 8 = 10.8MB (1.5 byte/pixel is for YUV420 format) Burst still image capture (13MP, burst length to 5)

4. Burst still image capture (13MP, burst length to 5)
   - **Image**: 13M * 1.0 * 5 = 80MB (1.0 byte/pixel is assumed to be maximum size for JPEG)
   - **Postview**: 0.9M * 1.5 * 5 = 6.75MB (1.5 byte/pixel is for YUV420 format)

5. Zero shutter lag (8MP, ring buffer size 10)
   - **Image**: 8M * 1.0 * 10 = 80MB (1.0 byte/pixel is assumed to be maximum size for JPEG)
   - **Viewfinder**: 0.9M * 1.5 * 8 = 10.8MB (1.5 byte/pixel is for YUV420 format)
6. HDR (13MP, composition of 3 frames):
   Image: 13M * 1.5 * 3 = 58.5MB (1.5 byte/pixel is for YUV420 format)

7. HDR (13MP, composition of 3 frames):
   Image: 13M * 1.5 * 3 = 58.5MB (1.5 byte/pixel is for YUV420 format)

Note: The amount of carve-out memory should match the camera use case that requires the most memory. Based on the calculation above, the size of carve-out is decided to be at least 91 MB.

4.2 ION-Based Memory Management

ION manages the carve-out memory, which is a generalized memory manager introduced by Google* in the Android* 4.0 ICS (Ice Cream Sandwich) release to address the issue of fragmented memory management interfaces across different Android devices.

https://lwn.net/Articles/480055/

int ion_open();
int ion_close(int fd);
int ion_alloc(int fd, size_t len, size_t align, unsigned int heap_mask,
   unsigned int flags, ion_user_handle_t *handle);
int ion_alloc_fd(int fd, size_t len, size_t align, unsigned int heap_mask,
   unsigned int flags, int *handle_fd);
int ion_sync_fd(int fd, int handle_fd);
int ion_free(int fd, ion_user_handle_t handle);
int ion_map(int fd, ion_user_handle_t handle, size_t length, int prot,
   int flags, off_t offset, unsigned char **ptr, int *map_fd);
int ion_share(int fd, ion_user_handle_t handle, int *share_fd);
int ion_import(int fd, int share_fd, Ion_user_handle_t *handle)

---

1 This content is based on an article “The Android ION memory allocator”, February 8, 2012 contributed by Thomas M. Zeng.
4.3 Usage Example

A CRC_ion block is created inside the ION kernel framework to manage the carve-out memory for camera as shown in Figure 3. It provides function like allocate, free.

**Figure 3. ION Memory Management Usage**

Besides reserved memory, ION can also be used to support buffer sharing with GFX/Video components.

Memory allocation and buffer sharing in SoFIA 3G-R operates in user pointer mode, where memory is allocated in the user space and passed to the kernel. In this mode, user-space applications and HALs can accept external buffers to avoid memory copies between components.

In general, components in user space need to call ion_alloc with flag `ION_HEAP_TYPE_CAREOUT` to allocate physically contiguous memory. CRC_ion in kernel space receives this call and allocate memory from carve-out heap and return an `fd` to user space as the handle of this memory. Camera/GFX/Video components share this `fd` and pass it into their kernel space drivers.
In kernel space, each driver imports the physical address from ION with the shared fd and callback `ion_import_fd`. Then HW registers (or MMU if applicable) are programmed based on the physical address.

On SoFIA 3G-R, as shown in Figure 4, for buffers shared between Camera, Video and GFX, GFX initiates the buffer allocation.

Figure 4. Memory Management and Buffer Sharing with ION
5.0 Sample V4L2 Application Flow

5.1 Choose the Device-node Based on Functionality
1. /dev/video0 being more suitable for streaming/preview kind of use-cases and /dev/video2 being suitable for all other capture purposes. /dev/video3 is meant for read-back offline images from memory and processing them. /dev/video3 use-case is out of scope of this document.
2. V4L2 IOCTLs supported by different video device nodes /dev/video0, /dev/video2, /dev/video3 are listed in Table 2.

5.2 V4L2 API Call Flow for Capture Application
1. Choose desired input device from available sensor devices.
2. Enumerate formats supported.
3. Choose desired format from the list of sensor's supported resolutions.
4. Allocate memory accordingly using ION technique.
5. Trigger V4L2 IOCTLs in sequence:
   a. VIDIOC_S_INPUT (select desired input sensor device)
   b. VIDIOC_S_FMT (set chosen format/resolution)
   c. VIDIOC_REQBUFS (with the memory allocated through ION mechanism, initiate memory mapping)
   d. VIDIOC_S_CTRL (set any desired control parameters)
   e. VIDIOC_QUERYBUF (query buffer status when required)
   f. VIDIOC_QBUF (Queue buffer to driver)
   g. VIDIOC_STREAMON (start streaming)
   h. VIDIOC_DQBUF (Dequeue buffer from driver)
   i. VIDIOC_STREAMOFF (turn stream off when done)
Figure 5. V4L2 Call Sequence Chart
## Appendix A  Supported V4L2 IOCTLs

Table 3 provides a list of supported V4L2 IOCTLs.

### Table 3. Supported V4L2 IOCTLs

<table>
<thead>
<tr>
<th>Device Node</th>
<th>Supported IOCTLs</th>
</tr>
</thead>
</table>
| /dev/video0 (Secondary Path)     | VIDIOC_REQBUFS  
VIDIOC_QUERYBUF  
VIDIOC_QBUF  
VIDIOC_DQBUF  
VIDIOC_STREAMON  
VIDIOC_STREAMOFF  
VIDIOC_S_INPUT  
VIDIOC_ENUM_INPUT  
VIDIOC_G_CTRL  
VIDIOC_S_CTRL  
VIDIOC_S_FMT_VID_OVERLAY  
VIDIOC_G_FMT_VID_OVERLAY  
VIDIOC_S_EXT_CTRL  
VIDIOC_QUERYCAP  
VIDIOC_CROPCAP  
VIDIOC_S_CROP  
VIDIOC_G_CROP  
VIDIOC_SUBSCRIBE_EVENT  
VIDIOC_UNSUBSCRIBE_EVENT  
VIDIOC_DEFAULT |
| /dev/video2 (Main Path)          | VIDIOC_REQBUFS  
VIDIOC_QUERYBUF  
VIDIOC_QBUF  
VIDIOC_DQBUF  
VIDIOC_STREAMON  
VIDIOC_STREAMOFF  
VIDIOC_S_INPUT  
VIDIOC_ENUM_INPUT  
VIDIOC_G_CTRL  
VIDIOC_S_CTRL  
VIDIOC_S_FMT_VID_CAP  
VIDIOC_G_FMT_VID_CAP  
VIDIOC_ENUM_FMT_VID_CAP  
VIDIOC_ENUM_FRAMESIZES  
VIDIOC_S_PARM  
VIDIOC_CROPCAP  
VIDIOC_S_CROP  
VIDIOC_G_CROP |
## Supported V4L2 IOCTLs

<table>
<thead>
<tr>
<th>Device Node</th>
<th>Supported IOCTLs</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dev/video3 (Read-Back/DMA)</td>
<td>VIDIOC_REQBUFS, VIDIOC_QUERYBUF, VIDIOC_QBUF, VIDIOC_DQBUF, VIDIOC_STREAMON, VIDIOC_STREAMOFF, VIDIOC_S_FMT_VID_OUT, VIDIOC_G_FMT_VID_OUT, VIDIOC_CROP, VIDIOC_S_CROP, VIDIOC_G_CROP</td>
</tr>
</tbody>
</table>

## A.1 Reference

The firmware and OS combined package for Intel Atom® x3-C3200RK Processor platforms, and all information/procedure related to its flashing and so forth, can be found here: [https://www.intel.com/content/www/us/en/embedded/products/sofia-3g-r/software.html](https://www.intel.com/content/www/us/en/embedded/products/sofia-3g-r/software.html)