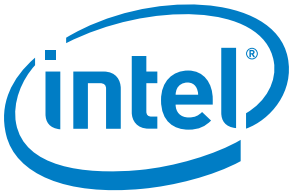
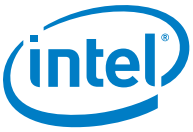


Bruce Liao
Platform Application Engineer
Intel Corporation

Intel® Atom™ Processor E3800 Series: MIPI* CSI-2* Camera Sub System

July 2014

330988-001

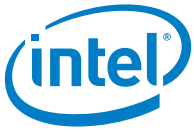


Executive Summary

The Intel® Atom™ Processor E3800 Series leverages the Mobile Industry Processor Interface* (MIPI*) Camera Serial Interface 2* (CSI-2*) technology. This enables the E3800 Processor Series to support many cell phone and tablet cameras.

However, this simultaneously adds a lot of complexity to the Linux Driver*. This white paper ramps up the reader's knowledge of the Intel® Atom™ Processor E3800 Series Image Signal Processor (ISP) Linux Driver.

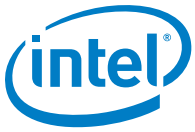
The Intel® Embedded Design Center provides qualified developers with web-based access to technical resources. Access Intel Confidential design materials, step-by step guidance, application reference solutions, training, Intel's tool loaner program, and connect with an e-help desk and the embedded community. Design Fast. Design Smart. Get started today. http://www.intel.com/p/en_US/embedded.



Contents

Figures

Tables



Background

Intel has adopted the Mobile Industry Processor Interface (MIPI) Camera Serial Interface 2 (CSI-2) for the Intel® Atom™ Processor E3800 Series.

MIPI CSI-2 is a standard developed by the MIPI Alliance*. It supports a mobile device feature set that is standard or common on most cell phones and mobile devices.

The MIPI Alliance is a non-profit corporation whose mission is to benefit the entire mobile device industry by establishing standards for hardware and software interfaces.

The mobile handset industry had long had the need for a baseline interface protocol for attaching camera subsystems to the application processors. In response to this need, the MIPI Alliance developed CSI-2.

MIPI CSI-2 provides the mobile industry with a standard benchmark that ensures industry-wide standardization on a robust, scalable, low-power, high-speed, cost-effective mobile camera interface.

The Intel® Atom™ Processor E3800 Series processor's MIPI CSI-2 Interface plays a pivotal role in processor use in mobile devices.

However, inclusion of the required Linux Driver adds considerable complexity to configuration and integration.

Solution

This whitepaper was written to ramp up the reader's knowledge of the Linux* Driver* and to educate the reader on the other integration nuances and intricacies that MIPI CSI-2 brings in to play.

The paper details Intel® Atom™ Processor E3800 Series MIPI CSI-2 integration and configuration, and outlines the key procedures that must occur to ensure successful integration.

It describes, in detail, the software architecture of the Intel® Atom™ Processor E3800 Series Image Signal Processor (ISP) Linux Driver*.

It also delineates the steps required to port a new camera sensor and instructs the reader on how to initiate the streaming process.

This key technical information will ramp up any reader's expertise in this crucial area of Intel® Atom™ Processor E3800 Series MIPI CSI-2 integration and the associated Linux* Driver usage.

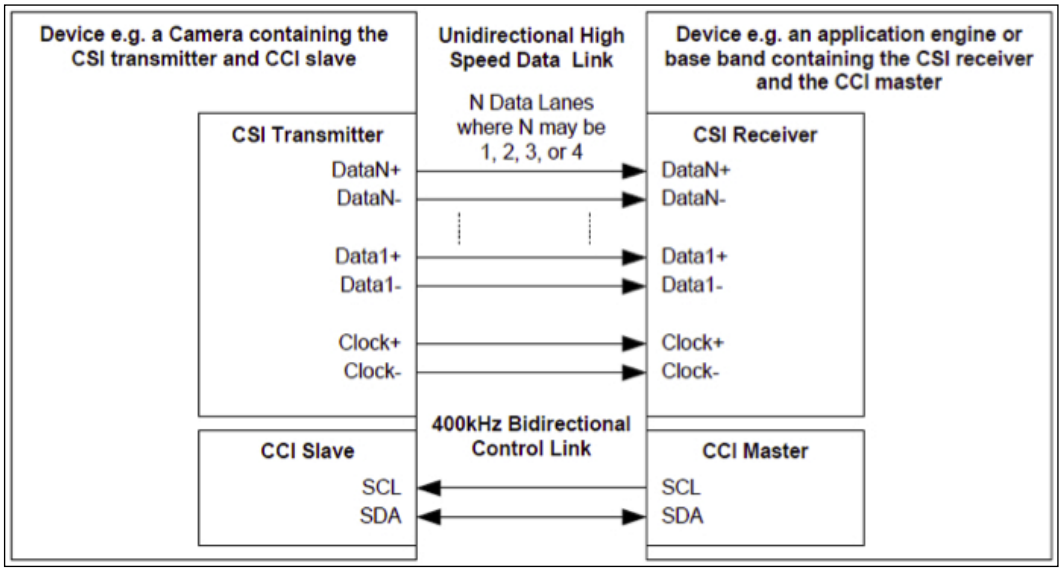
MIPI CSI-2: Interface Specification

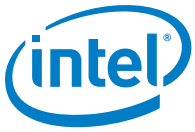
The CSI-2 Specification defines standard data transmission and control interfaces between transmitter and receiver.

Data transmission interfacing occurs via a unidirectional, differential serial interface, utilizing data and clock signals. The physical layer of this interface was designed using the MIPI Alliance standard the Digital Physical Layer (D-PHY*) Specification*. The control interface is bi-directional, and is compatible with the Inter-Integrated Circuit (I^2C) standard.

Figure 1. illustrates the CSI-2 and Cache Coherent Interconnect (CCI) Transmitter and Receiver Interface.

Figure 1. CSI-2 and CCI Transmitter and Receiver Interface





Intel® Atom™ Processor E3800 Series Camera Sub System

The Intel® Atom™ Processor E3800 Series integrates the MIPI-CSI 2.0 Interface and the ISP via Direct Memory Access (DMA) and local Static Random-access Memory (SRAM). The System on a Chip (SoC) has three MIPI clock lanes and five MIPI data lanes. These operate at up to 1 Gigatransfers per Second (GTPS). D-PHY utilizes these lanes and connects them to three virtual ports. The three virtual ports support:

- 3 Sensors (x2 lanes, x2 lanes, x1 lane or x3 lanes, x1 lane, x1 lane), or,
- 2 Sensors (x4 lanes, x1 lane)

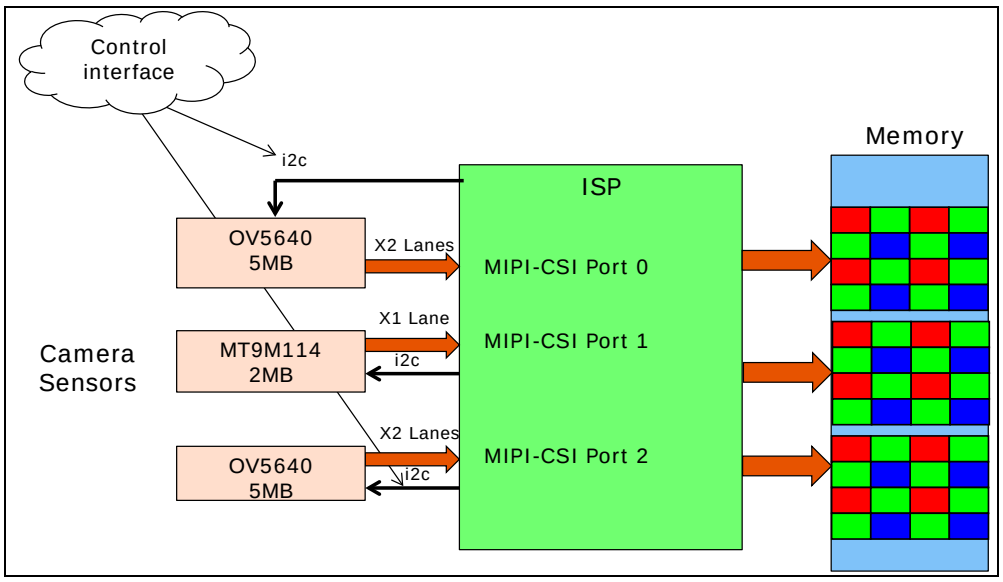
This white paper uses the Intel® Atom™ Processor E3800 Customer Reference Board (CRB) Implementation for reference purposes. This CRB is displayed in Figure 2.

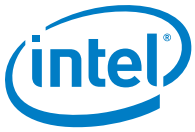
The 3 MIPI-CSI ports connect to these camera sensors:

- OV5640 (uses 2 lanes)
- MT9M114 (uses 1 lanes)
- OV564 (uses 2 lanes)

For the camera interface, the SoC supports three I^2C ports. These ports are used to control the camera sensors and the camera peripherals like the Flash Light-emitting Diode (LED) and the Lens Motor.

Figure 2. Customer Reference Board (CRB) Sample Implementation





Intel® Atom™ Processor E3800 Series ISP Driver Architecture

This document utilizes Linux Kernel 3.10*, and the Intel® Atom™ Processor E3800 CRB implementation for procedural reference. The core of the Intel® Atom™ Processor E3800 Series ISP is a vector processor; thus, software is a component element. This software is called ISP firmware. The ISP firmware is loaded into the ISP during driver initialization.

The capability of the camera sub-system is highly dependent on the ISP firmware. The feature functions of the ISP firmware include (but aren't limited to):

- Scaling
- Black Level Correction
- Automatic Exposure
- Auto White Balancing, and
- Auto Focus

Usually the ISP firmware is released as a binary, as there is not enough available ISP hardware information. An abstraction layer, the ISP Camera Sub System (CSS) Application Programming Interface (API) Layer, is laid above the ISP firmware layer.

The APIs defined in the abstraction layer list are:

- **`ia_css_init`** (for camera sub system initialization)
- **`ia_css_suspend/ia_css_resume`** (for camera sub system power management events)
- **`ia_css_stream_start`** (to initiate photo imaging/video recording)

For the Intel® Atom™ Processor E3800 Series processor to be able to leverage and fully utilize Linux* camera applications, the ISP driver must be integrated with the Linux* Video for Linux* 2 (V4L2*) Framework.

The entry point for the ISP driver is via the Linux* Peripheral Component Interconnect (PCI) Device Driver* probe function. Using the probe function, the ISP driver searches for and then recognizes all of the connected camera sensors and MIPI-CSI ports. It then records all required sensor and port data, using the following V4L2 APIs:

- `v4l2_subdev_init`
- `v4l2_device_register_subdev`
- `video_register_device`

After this, the Linux* user applications or middleware (example: Linux* Android Framework*) open, configure, and start the camera sensors, via the standard V4L2 Interfaces.

Several camera sensors connect to the ISP via the MIPI CSI and I^2C Interfaces. These sensors have a variety of capabilities, such as:

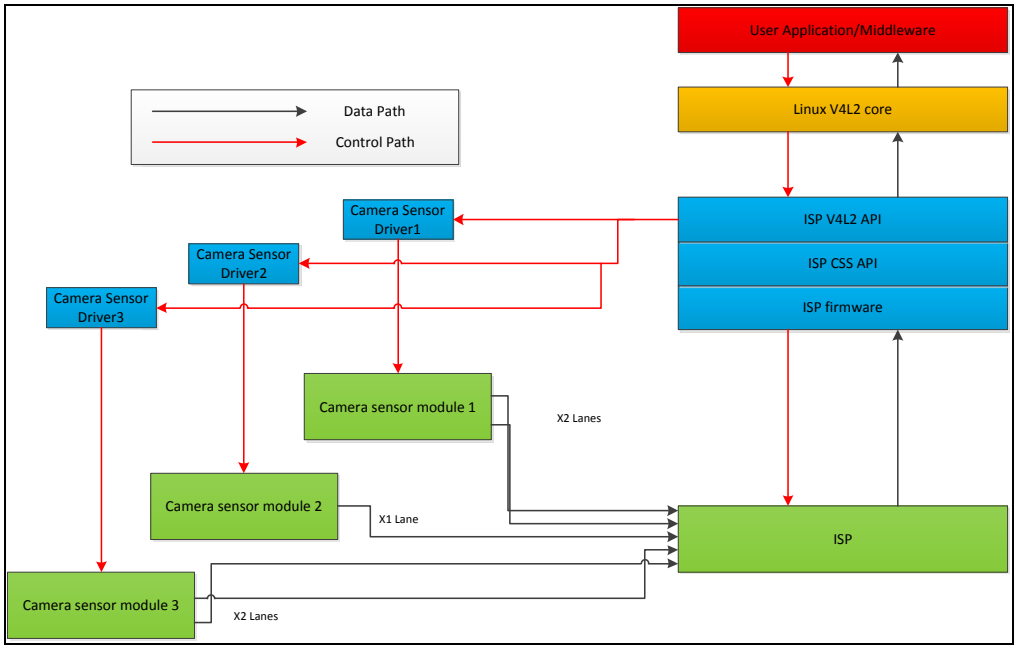
- Resolution
- Format support
- Night Mode support
- Auto Focus
- Continuous Focus

For each camera sensor, there is a specific, corresponding I^2C Driver. Once the Linux Camera Application* or middleware issues the standard V4L2 control instruction (example: `VIDIOC_G_FMT`), the ISP driver intercepts the instruction and invokes the camera sensor driver.

The camera sensor driver then communicates with the camera sensor via the standard I^2C API `I2C_transfer` for configuration and streaming operation (Start/Stop).

To learn how to initialize and configure the camera sensors, refer to your vendor's specification documentation.

Figure 3. Image Signal Processor (ISP) Driver Architecture



Run Time Progression

To understand exactly how the run time sequence progresses, in terms of the underlying technical mechanics, one needs to first understand that the Linux Multi-Media Framework* is structured in a very particular way, quite deliberately.

The Linux Multi-Media Framework makes use of a nuanced algorithmic modeling schema.

This schema was designed with the primary goal of enabling Linux to facilitate, in the simplest and most direct method possible, within any given set of technological circumstances, the most rapid completion of any run time maneuver.

The Linux Media Framework utilizes an oriented graph of building blocks. These blocks are called entities. These entities are connected through pads.

So, at runtime, to locate the camera sub system, and to configure it for use, the system is modeled as the oriented graph of the building blocks.

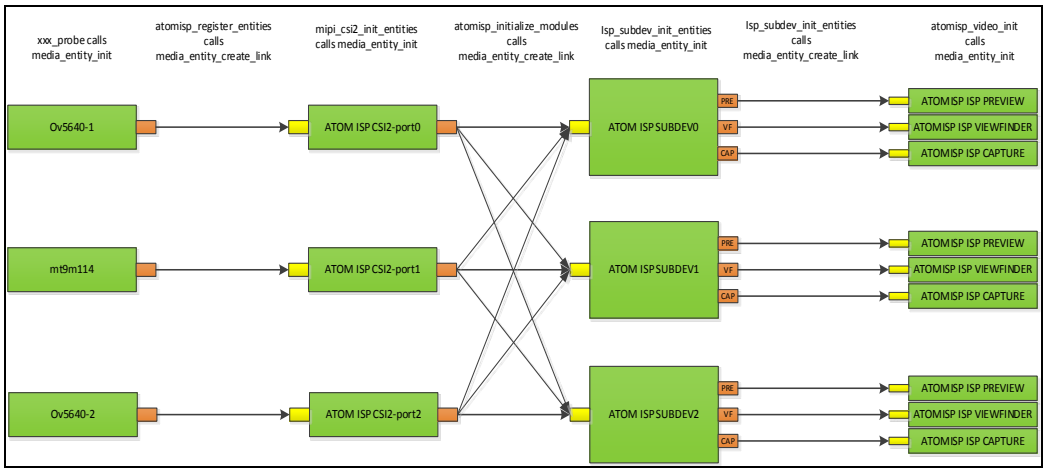
`Media_device structure` is embedded into `atomisp_device structure`.

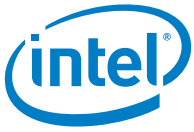
For this, the APIs `media_entity_init`, `media_device_register`, and `media_entity_create_link` are used.

Upon execution of the PCI driver's probe function, the oriented graph is built.

The oriented graph output is shown in Figure 4.

Figure 4. ISP Driver Run-Time, Utilizing the Linux Multi-Media Framework





Camera Sub System Configuration and Streaming Process

All data about the camera and MIPI CSI ports are recorded in `platform_camera.c`.

The Kernel* runs the corresponding code very early, as it is defined using `arch_initcall`.

At the end of the code flow, the camera configuration information link list is built. The link head for the list is `v4l2_subdev_table_head`.

After this, during the PCI device driver's probe function, `atomisp_subdev_probe` traverses the entire link list and invokes `v4l2_i2c_new_subdev_board` for all nodes. The corresponding I^2C drivers then execute their probe functions.

Figure 2. illustrates these connections.

The first camera sensor, OV5640, is connected via the primary MIPI CSI port: It is named MIPI CSI Port 0. MIPI CSI Port 0 has 2 lanes.

Camera sensor MT9M114 is connected via the secondary MIPI CSI port, MIPI CSI Port 1. MIPI CSI Port 1 only has 1 lane.

The second instance of camera sensor OV5640 is connected via the third MIPI CSI port, MIPI CSI Port 1. MIPI CSI Port 1 has 2 lanes.

This code sequence is thus documented as such, within v4l2_ids in `platform_camera.c`, as shown in Table 1.

Table 1. Camera Sensors: Code Sequence

```
static const struct intel_v4l2_subdev_id v4l2_ids[] = {
    {"mt9m114", SOC_CAMERA, ATOMISP_CAMERA_PORT_SECONDARY, 1},
    {"ov5640-2", SOC_CAMERA, ATOMISP_CAMERA_PORT_THIRD, 2},
    {"ov5640-1", SOC_CAMERA, ATOMISP_CAMERA_PORT_PRIMARY, 2},
   {},
 };
```

Note: Ensure that your code, in exacting detail and utilizing all proper nomenclature, precisely describes the actual configuration of your platform.

The Intel® Atom™ Processor E3800 CRB Schematic illustrates how the Camera Sensor Add-on Board is connected to I²C3, the 4th Intel® Atom™ Processor E3800 Series I²C Controller.

The corresponding software I²C bus number is 0x3. This is defined in `i2c-designware-pcidrv.c`.

Thus, the bus number parameter of intel_ignore_i2c_device_register in `platform_camera.c` is 0x3.

Ensure that your definitions for the above exactly reflect your particular schematic design.

You need to update the gpio_table in `platform_camera.c` in accordance with your own platforms. This is because, based on the platform, different General Purpose Input/Output GPIO) pins may be connected to the each camera's reset and power signal.

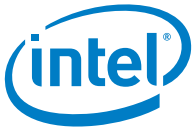
When the Linux camera application opens the camera shutter, the callback function `atomisp_open` executes. This wakes up the ISP unit, with `pm_runtime_get_sync`.

Note: As part of the PCI device driver's probe function, the application calls `pm_runtime_allow`, and the ISP unit goes to sleep.

The application calls `ISP CSS API ia_css_init` to initialize the camera sub system. Once the API is executed, the firmware is loaded into the ISP and the ISP pipeline is initialized.

Even after the Linux camera application has turned the camera on and enabled use of the camera features, the application continues to define the camera input, formats, and parameters, via the I/O control interface. Correspondingly, `atomisp_s_input`, `atomisp_s_fmt_cap`, and `atomisp_s_parm`, are continuously executed.

In the last step, the Linux camera application calls qbuf and streamon, via the I/O control interface. Correspondingly, `atomisp_qbuf` and `atomisp_streamon` are executed.



Acronyms

Table 2. Acronyms

Acronym	Definition
API	Application Programming Interface
CCI	Cache Coherent Interconnect
CRB	Customer Reference Board
CSI-2	Camera Serial Interface 2
CSS	Camera Sub System
DMA	Direct Memory Access
D-PHY	Digital Physical Layer (MIPI Alliance Specification for display applications)
GPIO	General Purpose Input/Output
GTPS	Gigatransfers Per Second
I/O	Input/Output
I^2C	Inter-Integrated Circuit
ISP	Image Signal Processor
LED	Light-emitting Diode
MIPI	Mobile Industry Processor Interface
PCI	Peripheral Component Interconnect
SoC	System on a Chip
SRAM	Static Random-access Memory
V4L2	Video for Linux 2

Conclusion

The MIPI CSI-2 interface of the Intel® Atom™ Processor E3800 Series enables the SoC to support a wide array of camera sensors on a multitude of cell phones and mobile products.

However, the MIPI CSI-2 also adds complexity to the SoC.

This white paper describes in detail the underlying implications of these intricacies, and provides extensive instructional and procedural reference information.

It educates System Integrators, developers, and other end-users about the key technical realities of MIPI CSI-2 and Linux Driver usage with the Intel® Atom™ Processor E3800 Series.

It will equip readers with the requisite knowledge needed to navigate the configuration and interface requirements of MIFI CSI-2.

The Intel® Embedded Design Center provides qualified developers with web-based access to technical resources. Access Intel Confidential design materials, step-by step guidance, application reference solutions, training, Intel's tool loaner program, and connect with an e-help desk and the embedded community. Design Fast. Design Smart. Get started today. http://www.intel.com/p/en_US/embedded.

Author

Bruce Liao is a Platform Application Engineer with the Internet of Things (IoTG) Solutions Group at Intel Corporation.