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

July 2014
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.


Contents

Background ............................................................................................................. 4
Solution .................................................................................................................. 4
MIPI CSI-2: Interface Specification ............................................................................ 5
Intel® Atom™ Processor E3800 Series Camera Sub System ........................................... 6
Intel® Atom™ Processor E3800 Series ISP Driver Architecture ....................................... 8
Run Time Progression ..................................................................................... 10
Camera Sub System Configuration and Streaming Process ........................................... 11
Acronyms .............................................................................................................. 13
Conclusion ............................................................................................................ 14

Figures

Figure 1. CSI-2 and CCI Transmitter and Receiver Interface ........................................... 5
Figure 2. Customer Reference Board (CRB) Sample Implementation ................................ 7
Figure 3. Image Signal Processor (ISP) Driver Architecture ........................................... 9
Figure 4. ISP Driver Run-Time, Utilizing the Linux Multi-Media Framework ..................... 10

Tables

Table 1. Camera Sensors: Code Sequence ...................................................................... 11
Table 2. Acronyms ...................................................................................................... 13
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 (I²C) 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
The Intel® AtomTM 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 I²C 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

[Diagram showing the integration of OV5640 and MT9M114 sensors with ISP through MIPI-CSI ports]
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:

- `v412_subdev_init`
- `v412_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\textsuperscript{2}C 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\textsuperscript{2}C 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\textsuperscript{2}C 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 I²C 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

```c
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**

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>CCI</td>
<td>Cache Coherent Interconnect</td>
</tr>
<tr>
<td>CRB</td>
<td>Customer Reference Board</td>
</tr>
<tr>
<td>CSI-2</td>
<td>Camera Serial Interface 2</td>
</tr>
<tr>
<td>CSS</td>
<td>Camera Sub System</td>
</tr>
<tr>
<td>DMA</td>
<td>Direct Memory Access</td>
</tr>
<tr>
<td>D-PHY</td>
<td>Digital Physical Layer (MIPI Alliance Specification for display applications)</td>
</tr>
<tr>
<td>GPIO</td>
<td>General Purpose Input/Output</td>
</tr>
<tr>
<td>GTPS</td>
<td>Gigatransfers Per Second</td>
</tr>
<tr>
<td>I/O</td>
<td>Input/Output</td>
</tr>
<tr>
<td>I²C</td>
<td>Inter-Integrated Circuit</td>
</tr>
<tr>
<td>ISP</td>
<td>Image Signal Processor</td>
</tr>
<tr>
<td>LED</td>
<td>Light-emitting Diode</td>
</tr>
<tr>
<td>MIPI</td>
<td>Mobile Industry Processor Interface</td>
</tr>
<tr>
<td>PCI</td>
<td>Peripheral Component Interconnect</td>
</tr>
<tr>
<td>SoC</td>
<td>System on a Chip</td>
</tr>
<tr>
<td>SRAM</td>
<td>Static Random-access Memory</td>
</tr>
<tr>
<td>V4L2</td>
<td>Video for Linux 2</td>
</tr>
</tbody>
</table>
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.


Author

Bruce Liao is a Platform Application Engineer with the Internet of Things (IoTG) Solutions Group at Intel Corporation.
INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL’S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL’S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information. The products described in this document 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.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm%20

Intel, Intel Atom, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

Copyright © 2014 Intel Corporation. All rights reserved. §