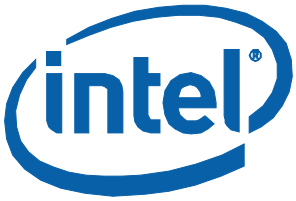


Bruce Liao, Zhou Hua
Platform Application Engineers

Arshad Mehmood
Software Architect

Jia James
Development Manager

Fang Neo, Tang Jun
Senior Software Engineers

Kim HJ
Technical Marketing Engineer

Early Camera Presentation on Intel® Atom™ Processor E38xx Series

Fast Boot, Early Camera Driver, and System Optimization

January 2014

330046-002

Executive Summary

This document introduces a method to minimize the time of every boot stage to meet the very fast boot speed needed to display the rear camera in an In-Vehicle Infotainment (IVI) system. The new method includes steps for optimizing the boot loader, fine-tuning the kernel and camera drivers, and optimizing the Tizen* user space. The goal of this paper is to decrease the boot time for the Intel® Atom™ Processor E38xx Series so that the rear camera is available to the driver as soon as the car is switched into reverse gear. The knowledge and experience used to accomplish this task can be leveraged among the other platforms.

Contents

Figures

Business Challenge

A fundamental requirement for In-Vehicle Infotainment (IVI) devices is that a user must see the rear camera video in the IVI system screen immediately after putting the vehicle in reverse gear, regardless of the current state of the IVI box. In general, this process should occur in less than two seconds. This means that, even in the worst case scenario in which the box is powered off, it should still boot to the operating system (OS) and bring up the camera within a very short time.

Solution

In this white paper, we provide a solution on the Intel® Atom™ Processor E38xx Series for customers requiring very early video camera input. The solution is based on the hardware and software components listed below:

- Intel® Atom™ E38xx series customer reference board
- TW6865/69 PCIe* camera
- Tizen* IVI 3.0 with Linux* kernel 3.8
- Intel® Embedded Media and Graphics Driver (Intel® EMGD)
- Reference Boot Loader from Intel

Optimize the Boot Loader

The BIOS image released from Intel usually is a UEFI BIOS for the Intel® Atom™ Processor E38xx Series. This BIOS image takes 5-10 seconds to start the OS loader, depending on the boot device. As a full-featured boot loader, UEFI BIOS is not the best candidate to implement the early camera feature, which requires minimal boot time. Instead, we propose using the Reference Boot Loader from Intel or other lightweight boot loaders for the boot speed improvement. The biggest task of the Reference Boot Loader from Intel is to initialize the SoC; it does not support set up interface, boot phase user input, and so on.

Memory initialization usually takes a great deal of time, depending on the memory capacity. The Reference Boot Loader from Intel saves the memory parameters in the flash after the first boot, and it no longer needs to train the memory modules in future boots.

The Reference Boot Loader from Intel provides boot media support, such as SD, SATA, USB, and so on. To save the boot speed, only one boot media should be enabled, because in the vehicle user cases, the system configuration is static, and there is only one mass storage device to boot from. Consequently, the initialization of the rest of the parts relies on Linux* kernel drivers.

During the boot process, the Reference Boot Loader from Intel detects the rear gear flag. If this flag is set, the boot loader will bypass the splash screen, saving more than 200 milliseconds (ms).

Fine-Tune the Linux* Kernel and Drivers

Linux* Kernel and Drivers Optimization

To reduce kernel booting time, several techniques are applied to kernel, including the following:

- Customizing Linux* kernel configuration for system requirement
- Asynchronous AHCI port probing
- Reordering initialization of drivers

The following steps are hard coded to further reduce booting time:

- Skip PS2 mouse detection and return immediately
- Skip TSC calibration and assign hard-coded values for Intel® Atom™ Processor E38xx Series

Description of Applied Techniques

1. Customizing Linux* kernel configuration

   Bigger kernel image size results in slower loading time, unnecessary drivers, and options that consume more booting time. Therefore, minimal drivers and options in `menuconfig` are enabled according to the system static configuration.

2. Asynchronous AHCI port probing

   Even though the SATA port is not connected, the kernel scans all SATA ports, which takes time. To avoid this delay, the unconnected SATA port will be specified in the kernel command line and skip detection on the unconnected port.

3. Reordering initialization of drivers

   The camera driver and the Intel® Embedded Media and Graphics Driver (Intel® EMGD) need to be initialized as soon as possible for early camera video. The driver list in `linux/drivers/Makefile` should be reordered to ensure the camera driver and Intel® EMGD driver are at the beginning of the list.

4. Skip PS2 mouse detection

   In the latest kernel, the PS2 port is detected on the Intel® architecture system even though the PS2 mouse option is disabled in the kernel. This step will consume several hundred ms. The `drivers/input/serio/i8042-x86ia64io.h` must be changed to skip the PS2 detection and return immediately.

5. Skip TSC calibration and assign hard-coded value

   TSC calibration is not needed for system initialization, so the TSC value can be hard coded in `arch/x86/kernel/tsc.c`.

6. Disable the kernel log

   The log print takes too much time during boot up (typically log printing will consume more than 1.5 seconds), and so the “quiet” kernel parameter should be added in the kernel argument.

Camera Driver Optimization

Early camera functionality can meet the following requirements:

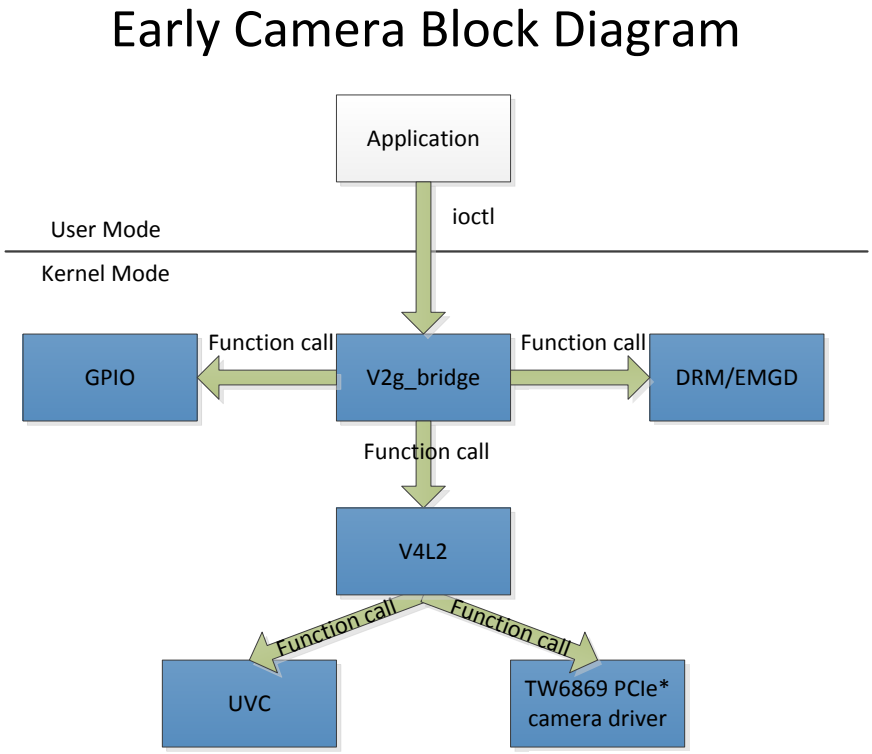
1. Put the camera input onto the display panel in 2 seconds from power on.
2. Auto-detect the reverse gear by the GPIO to enable or disable the camera output.
3. Use the user mode application to terminate kernel camera application.

It consists of several components:

1. `V2g_bridge` is the newly created kernel mode component. It plays the role of camera application in the kernel mode.
2. The other four components include the following:
   - Direct rendering manager (DRM) framework: DRM and Intel® EMGD
   - Video for Linux* framework: V4L2
   - Camera driver: UVC or TW6865 PCIe* camera
   - GPIO

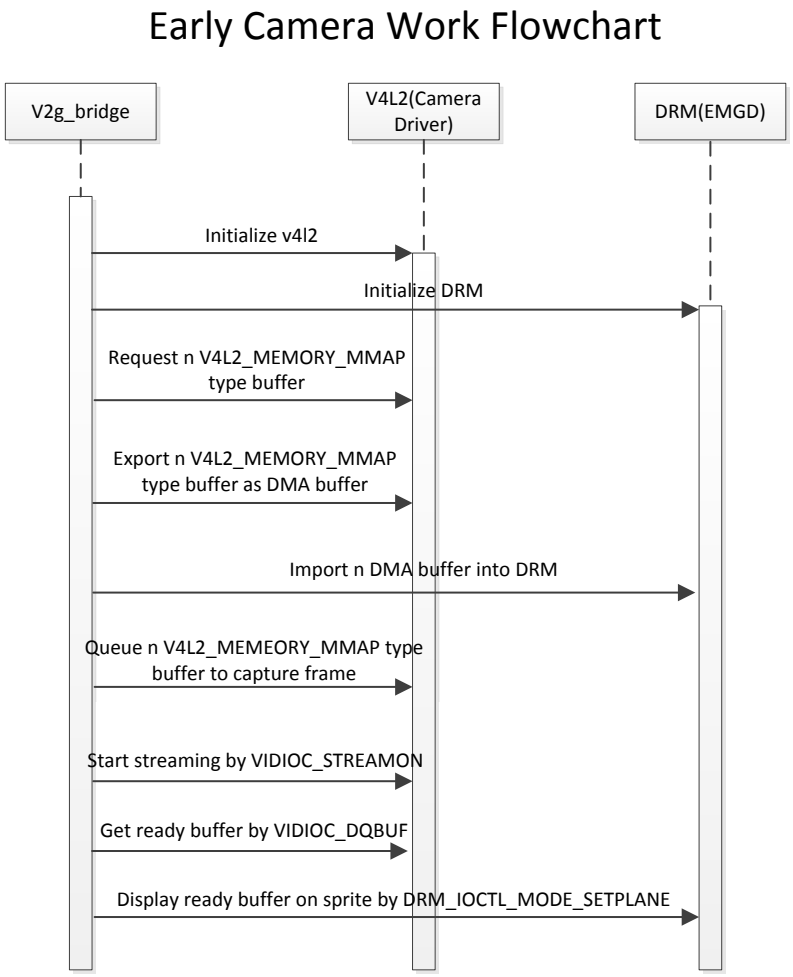
They are all kernel drivers. New APIs are added in these modules to support early camera in kernel mode. Here is a general block diagram:

Figure 1. Early Camera Block Diagram



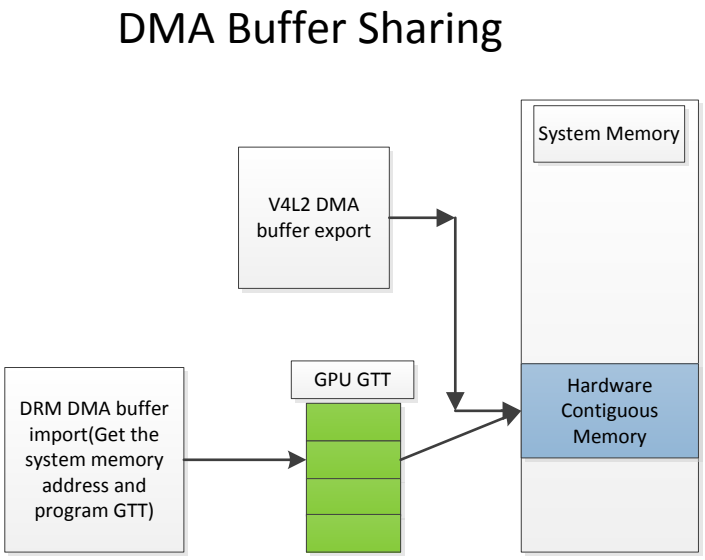
V2g_bridge module work flow is shown below:

Figure 2 Early Camera Work Flowchart



V2g_bridge implementation takes advantage of DMA-buffer sharing introduced by the latest 3.8 Linux* kernel. With this technology, the DRM module can directly import the buffer exported by the V4L2 module to avoid buffer copy between these two modules.

Figure 3. DMA Buffer Sharing



To reduce the early camera-related components boot time, several techniques are applied in the Intel® EMGD and PCIe* camera driver:

Intel® EMGD:

- Enable Intel® EMGD quick boot feature
- Configure proper timing for display device
- Remove unused port driver initialization
- Port driver optimizations

PCIe* camera driver:

- Support DMA-buffer sharing to avoid extra buffer copy
- Remove audio support

Description of Applied Techniques

- Enable the Intel® EMGD quick boot feature in `default_config.c`.

Disable unused ports from port order list.	Only enabling the port that is to be used saves lots of time.
Disable display detect flag (remove `IGD_DISPLAY_DETECT`).	Intel® EMGD will not check whether the display port is connected or not.
Set quickboot flag to 0x3.	Intel® EMGD will skip GTT clear, save, and restore, and skip time consuming initialization for ports not in port order list.
Set current port's `edia_not_avail` flag as `0x0`.	Intel® EMGD will use standard timing directly. Save time to read EDID from other places such as the display devices.

- Set the proper Embedded DisplayPort* (eDP*) panel timing.

  Default eDP* timing configuration wastes some time, so you must reduce the amount of time in eDP* timing as much as the specific panel will allow.

- Remove the unused port driver in `cfg/config.h`.

  Intel® EMGD will call port driver initialization functions in `pi_init_all`, even if it is not included in the port order list. Remove the unused port configuration flag in `cfg/config.h`, and keep only `CONFIG_PD_DP` for eDP* panel.

- The eDP* port driver in Intel® EMGD sets power on state two times.

  `edp_panel_power` will be called two times during eDP* port driver initialization. This function has a 60 ms delay. Just return if the eDP* interface is already in power-on-state to save an unnecessary delay in `edp_panel_power`.

- Port the original driver to use the V4L2 video buffer 2 framework to support DMA buffer sharing technology.

  The original TW6865/69 PCIe* camera driver is based on video buffer framework, which does not support DMA buffer sharing. Rewrite the driver to support video buffer 2 framework.

- Remove the audio support.

  The original TW6865/69 PCIe* camera driver supports audio, which has nearly 150 ms of delays in total. To avoid this loss of time, remove the audio support because it is not needed in the rear camera presentation.

Optimize the Tizen* User Space

Description of Applied Techniques

- Systemd Services Enabling

The services of `systemd readahead` can reduce libraries loading time during system boot up. Set the `CONFIG_FANOTIFY=y` kernel options in the Linux* kernel.

- Services Tuning

  `Systemd` will detect the ordering cycle and the break ordering cycle by deleting this unit file to make the system run normally; if you manually delete this unit file in your system, `systemd` will not take time to delete it. Delete `smack-default-labeling.service`.

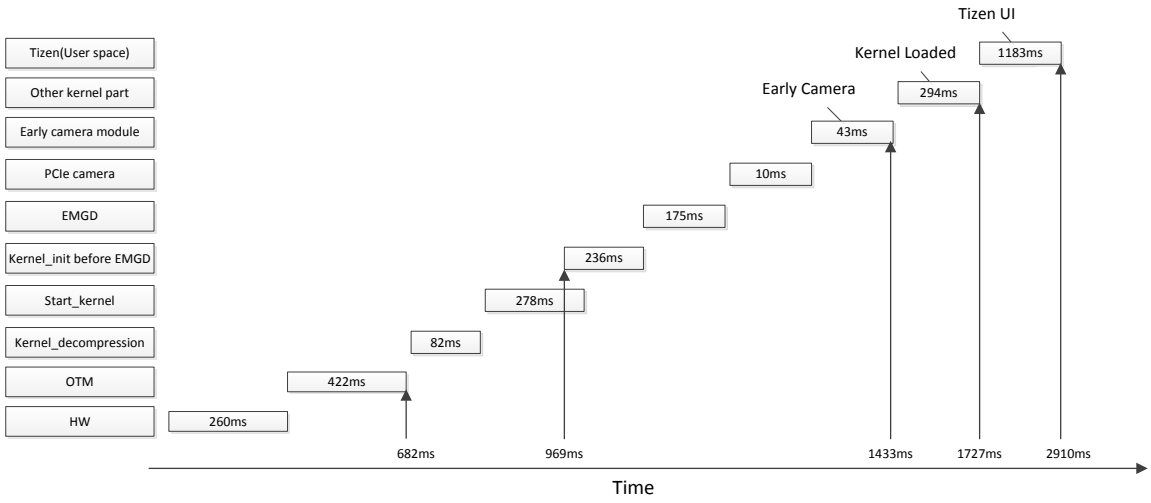
  Some unit files do not exist but are still labeled in `systemd` service or target files. Remove `After = Plymouth-start.service` in `systemd-ask-password-console.service`, `systemd-ask-password-console.path`, and `rescue.service`. Remove `After = syslog.target` in `net-config.service`, `connman.service`, `rygel.service`, `sshd@.service`, `sshd.service`, `dbus.service`, `ofono.service`, and `ambd.service`. Remove `Wants = display-manager.service` in `graphical.target`. Remove `After=Plymouth-quit-wait.service` in `console-getty.service`, `serial-getty@.service`, `getty@.service`, and `console-shell.service`. Remove `After=runlevel1.target runlevel2.target runlevel3.target runlevel4.target runlevel5.target` in `systemd-update-utmp-runlevel.service`. After removal, `systemd` will not take time to load these unit files, and you will not get any errors as these files do not exist.

  These two solutions will shorten system boot up time by about 100 ms.

Results

The figure below shows each component's time cost during boot up. The X axis is the timeline, and the Y axis lists all the components launched after power on. For example, it takes nearly 260 ms for the processor to start the first instruction in OTM after the power-on button is pushed. After 1.4 seconds the system is on, and you will be able to see the screen output of the rear camera. After 2.9 seconds, the Tizen* main user interface pops up.

Figure 4. Time Saved with Early Camera



Conclusion

In this white paper, we introduced a complete early camera presentation mechanism and implementation. The proposed implementation involves the use of a lightweight boot loader, Linux* kernel optimization, and early camera module optimization along with Tizen* user space fine-tuning. With this implementation, the early camera can be displayed at around 1.4 seconds after the power button is pressed, and the Tizen* UI can be displayed in 2.9 seconds. For the next step, we will port the implementation to the IVI dedicated development kit, which is also based on the Intel® Atom™ Processor E38xx series.

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.

Authors

Bruce Liao is a Platform Application Engineer at Intel Corporation.

Zhou Hua is a Platform Application Engineer at Intel Corporation.

Jia James is a Development Manager at Intel Corporation.

Fang Neo is a Senior Software Engineer at Intel Corporation.

Tang Jun is a Senior Software Engineer at Intel Corporation.

Arshad Mehmood is a Senior Software Engineer at Intel Corporation.

Kim HJ is a Technical Marketing Engineer at Intel Corporation.

Acronyms

AHCI	Advanced Host Controller Interface
API	Application Programming Interface
BIOS	Basic Input/Output System
DMA	Direct Memory Access
DRM	Direct Rendering Manager
EDID	Extended Display Identification Data
eDP*	Embedded DisplayPort*
EMGD	Embedded Media and Graphics Driver
GPIO	General Purpose Input Output
GTT	Graphics Translation Table
Ioctl	Input and Output Control
IVI	In-Vehicle Infotainment
MS	Millisecond
OS	Operating System
SATA	Serial Advanced Technology Attachment
SD	Security Digital
SoC	System on Chip
TSC	Time Stamp Counter
UEFI	Unified Extensible Firmware Interface
USB	Universal Serial Bus
UI	User Interface

§