Allied Vision

APPLICATION NOTE

Setting up i.MX6 Embedded Boards for Alvium MIPI CSI-2 Cameras

V1.0.0

Contents

Introduction . . . 2
- Scope of this document . . . 2
- In a nutshell . . . 2
- Downloading the required components . . . 2

Installing the kernel driver module . . . 3
- Precompiled kernel driver module . . . 3
- Kernel driver module sources . . . 5

Changing default settings . . . 6
- Changing the keyboard layout . . . 6
- Enabling and disabling MIPI CSI-2 camera drivers . . . 6
- Changing memory size reserved for CMA . . . 7
- Changing driver parameters with devicetree . . . 9

Installing V4L2 utilities . . . 11

Tips and Troubleshooting . . . 11

Introduction

Scope of this document

Using Alvium MIPI CSI-2 cameras requires setting up your embedded board. This document describes how to install the kernel including the camera driver module and how to prepare image acquisition.

Supported embedded boards with i.MX6:

- Nitrogen6_MAX with i.MX6QP, Linux kernel version 4.1.15
  - Nit6QP_MAX
  - Nit6Q_MAX
- Wandboard with i.MX6QP, Linux kernel version 4.1.15
  - WB-IMX6QP
  - WB-IMX6Q
  - WB-IMX6U
  - WB-IMX6S

In a nutshell

To get started with the SD card image or the precompiled kernel:

- Wandboard users can download an SD card image, which contains the operating system including Allied Vision's kernel driver module and everything needed to explore the camera.
- Wandboard and Nitrogen6_MAX users can download a tarball, which contains the precompiled kernel including Allied Vision's kernel driver module. For easy installation, a script is included.

We provide code examples on the SD card image and on GitHub.

Downloading the required components

Downloads

SD card image, precompiled kernel driver module for Alvium CSI-2 cameras, sources, and scripts:
https://www.alliedvision.com/en/products/software/embedded-software-and-drivers.html

Code examples, sources, and scripts:
https://github.com/alliedvision

Camera documentation including safety notes, specifications, and instructions for assembling the hardware:
https://www.alliedvision.com/en/support/technical-documentation/alvium-documentation.html

Installing the kernel driver module

Precompiled kernel driver module

As Allied Vision's Alvium MIPI CSI-2 camera is an innovation, its driver is not yet part of the Linux kernel. We provide a script file with all necessary installation steps.

To install the Allied Vision kernel driver module, follow these steps:

Step 1: As a prerequisite, install the operating system of the board manufacturer. Follow the instructions of the board manufacturer.

Nitrogen6_MAX, Linux kernel 4.1.15
https://github.com/boundarydevices/linux-imx6/tree/boundary-imx_4.1.15_2.0.0_ga

SD card image (we recommend using update#2):
https://boundarydevices.com/ubuntu-xenial-mx67-boards-august-2016-kernel-4-1-15

Wandboard, Linux kernel 4.1.15
https://github.com/wandboard-org/linux/tree/wandboard_imx_4.1.15_1.1.0_ga

SD card image:
http://download.wandboard.org/wandboard-imx6/ubuntu-16.04/wandboard-imx6_ubuntu-16.04_sdcard_20171213.zip

Step 2: Increase the partition size of the SD card image with a partition tool like gparted or the command line tool fdisk. Tutorials are available on the internet.

```
# Install gparted and extend the SD card as explained at https://gparted.org/documentation.php
sudo apt-get install gparted
sudo gparted
# Access the GUI
sudo -EH gparted

# Restart the system
```

Listing 1: gparted

Step 3: Download Allied Vision's tarball with the precompiled kernel driver module and copy it to the embedded board, mounting the primary partition first.
https://www.alliedvision.com/en/products/software/embedded-software-and-drivers.html

```
# Copy the tarball to the SD card (for example, Desktop)
cp <name>.tar.gz /mnt/home/ubuntu/Desktop/
```

Listing 2: Copy the tarball to the SD card

Step 4: Extract the tarball and run its included script.

```
# Call the install script
cd /home/ubuntu/Desktop
tar -xf <name>.tar.gz
cd <name>
sudo ./install.sh
```

Listing 3: Install script

Step 5: Follow the instructions of the install script. After the installation, the board will reboot.

Kernel driver module sources

If you don't want to use the precompiled kernel, we also provide the driver sources. To simplify the installation and avoid malfunctions, we offer several scripts.

Figure 1: Scripts for installing Allied Vision's kernel driver module sources

On your Linux PC:

Step 1: Download our kernel driver module sources and the scripts shown in Figure 1. https://github.com/alliedvision

Step 2: Run setup.sh, which installs tools for building the kernel and preparing the environment.

Step 3: Run build.sh, which builds the Linux kernel including Allied Vision's driver for Alvium MIPI CSI-2 cameras. We have tested this script with Ubuntu 16.04 LTS and Ubuntu 18.04 LTS. The script creates several files.

Step 4: Run deploy.sh, which collects these files and puts them into a.tar.gz archive, which is the deliverable.

On your embedded board:

Step 1: Install the supported operating system of the embedded board provided by the board manufacturer. See Precompiled kernel driver module.

Step 2: Copy the AlliedVision*.tar.gz archive to your board.

Step 3: Unpack the AlliedVision*tar.gz file, which contains install.sh.

Step 4: Run install.sh, which installs the kernel including the Allied Vision driver for Alvium MIPI CSI-2 cameras. Follow the instructions of the script.

Changing default settings

Default passwords

Nitrogen6_MAX
Superuser *root* and default user *ubuntu* share the default password *Boundary*.

Wandboard
Superuser *root* has the default password *root.* Default user *arch* has the password *arch*.

Consider that the default keyboard layout is English.

Kernel upgrade not supported

The driver for Allied Vision MIPI CSI-2 cameras doesn't support a kernel upgrade.

Changing the keyboard layout

By default, the keyboard layout is English (US). To change the keyboard layout:

```
sudo apt-get install language-pack-xx // instead of xx, use fr for French, de for German, etc.
sudo dpkg-reconfigure keyboard-configuration
```

Listing 4: Changing the keyboard layout

Enabling and disabling MIPI CSI-2 camera drivers

By default, the camera driver modules of MIPI CSI-2 cameras from Allied Vision and Omnivision are loaded. This section explains how to enable or disable driver modules either permanently or until the next bootup.

Note that Allied Vision's driver `avt_imx6_csi2` is built on NXP's driver `mxc_v4l2_capture`. Therefore, `mxc_v4l2_capture` must be enabled to use Alvium CSI-2 cameras.

To disable or enable a MIPI CSI-2 driver module until the next bootup:

```
// List all loadable modules
lsmod

// Disable modules
sudo rmmod ov5642_camera // Depends on mxc_v4l2_capture, v4l2_int_device
sudo rmmod avt_imx6_csi2 // Depends on mxc_v4l2_capture, v4l2_int_device

// Keep enabled to use your camera
sudo rmmod mxc_v4l2_capture
sudo rmmod v4l2_int_device

// Enable a module
sudo modprobe avt_imx6_csi2
```

Listing 5: Disabling or enabling a driver module until the next bootup

To disable the Omnivision driver module permanently, go to /etc/modprobe.d/blacklist.conf and append a line with the driver module you want to disable:

```
// Go to the end of the .conf file and append a line with the driver module you want to disable
blacklist ov5642_camera
```

Listing 6: Disabling a driver module permanently

Changing memory size reserved for CMA

The Linux kernel automatically allocates a part of its main memory reserved for CMA (Contiguous Memory Allocator). Increasing the memory size reserved for CMA automatically decreases the available user space RAM and vice versa. CMA is used by the DMA API.

Change the default value of memory size reserved for CMA to:

- Reach a higher frame rate by using more buffers (if the current value limits it)
- Increase performance when capturing images with more than 5 megapixels
- Increase available user space RAM for image processing applications (if the memory size reserved for CMA can be reduced for your use case)

When you change the CMA value, be aware that it is also used by other modules than the camera driver, for example, the GPU driver.

As a first step, modify the devicetree file, for example, arch/arm/boot/dts/imx6q.dtsi. If you're unfamiliar with devicetree, see Changing driver parameters with devicetree.

```
// .dtsi file modification to increase CMA reserved memory to approx. 800 MB
--- a/arch/arm/boot/dts/imx6q.dtsi
+++ b/arch/arm/boot/dts/imx6q.dtsi
@@ -93,7 +93,7 @@
          linux,cma {
              compatible = "shared-dma-pool";
              reusable;
-             size = <0x14000000>;
+             size = <0x30000000>;
              linux,cma-default;
          };
```

Listing 7: Modify the .dtsi file

Additionally, modify the value of CONFIG_CMA_SIZE_MBYTES to, for example, 800:

1. Go to the arch/arm/configs/<boardname>_alliedvision_defconfig file.
   By default, the CMA size value is 320 MBytes:
   `CONFIG_CMA_SIZE_MBYTES=320`
2. Change the value.
3. Rebuild the kernel. Use our install and build scripts, see Kernel driver module sources.

To apply temporary changes, if you want to try a value:

You can change the kernel behavior with menuconfig (the kernel configuration menu). For temporary changes, rebooting is sufficient. Applying changes permanently requires you to recompile the kernel.

```
// If not already present on your system, install libncurses-dev
sudo apt-get libncurses-dev

// Make menuconfig
make menuconfig
```

Listing 8: Install and make menuconfig

In the kernel configuration menu:

1. Select **Device Drivers** > **Generic Driver Options** and set * for DMA Contiguous Memory Allocator.
2. Select **Size in Mega Bytes**, press the Return key, and enter the desired value.
3. Save.
4. Select **Exit** until the kernel configuration menu closes.
5. Reboot.

Now the new CMA value is applied temporarily.

To apply the new CMA value permanently, rebuild the kernel.

Tested values

We have tested these values for the CMA size:

CMA value	Resolution	Maximum buffers
1200	5632×4096	14
1000	5632×4096	10
800	5632×4096	8
640	5632×4096	5

Table: Tested CMA size values

Changing driver parameters with devicetree

Some use cases require changing camera driver parameters such as the MIPI clock or the number of lanes. To achieve this, devicetree files are used.

Devicetree

Devicetree source files (DTS, file ending .dts) and devicetree source include files (.dtsi) describe a device as a human-readable tree of nodes and their properties. The DTS file is compiled into a binary devicetree blob file (DTB, file ending .dtb) that the bootloader puts into RAM. During the boot process, the kernel takes over the values from the devicetree.

Modifying and compiling the Allied Vision camera devicetree

You can modify the Allied Vision camera devicetree directly on the embedded board:

1. Install the DTC (devicetree compiler) on the board:

```
// Install the DTC
sudo apt install device-tree-compiler
```

Listing 9: Install the DTC

2. Identify the correct devicetree blob for your board:
   - Nitrogen6_MAX: /boot/imx6qp-nitrogen6_max.dtb
   - Wandboard: /media/ubuntu/<boot partition>/imx6qp-wandboard-revd1.dtb
3. Decompile the devicetree blob to devicetree source. In this example, we use Nitrogen6_MAX. Ignore warnings during this procedure:

```
// Decompile the devicetree blob
./dtc -I dtb -O dts -o ./devicetree.dts /boot/imx6qp-nitrogen6_max.dtb
```

Listing 10: Decompile the devicetree blob

4. Make the desired changes to the devicetree source file devicetree.dts:

```
avt_imx6_csi2: avt_imx6_csi2@3C
{
    compatible = "avti,avt_imx6_csi2";
    reg = <0x3C>;                       // I2C camera address
    clocks = <&clks IMX6QDL_CLK_PWM3>; // Pixel clock frequency
    clock-names = "csi_mclk";
    av_cam_i2c_clk = <400000>;          // Host I2C clock for camera

// i.MX6 has two IPUs (IPU0 & IPU1), each IPU has CSI channels CSI0 and CSI1.
// The virtual channel is defined based on the IPU and CSI selection.
// Default: virtual channel 0.

    ipu_id = <0>;           // IPU ID value of the connected camera to select stream and VC
    csi_id = <0>;           // CSI ID value of the connected camera to select stream and VC
    lanes = <4>;            // Lane count. No effect if lanes_auto_conf = <1> (see last line)
    lanes_auto_conf = <1>; // Value <1> sets lane auto negotiation and configuration.
    mclk = <188000000>;     // Master clock
    mclk_source = <0>;
    clk_auto_conf = <1>;    // Value <1> sets clock auto negotiation and configuration
};
```

Listing 11: devicetree.dts

5. Compile the devicetree source to devicetree blob and replace the original one. In this example, we use Nitrogen6_MAX. Ignore warnings during this procedure:

```
// Decompile the devicetree blob
sudo ./dtc -I dts -O dtb -o /boot/imx6qp-nitrogen6_max.dtb ./devicetree.dts
```

Listing 12: Decompile the devicetree blob

6. Reboot the board to apply the changes.

Installing V4L2 utilities

v4l-utils contain useful tools such as a simple viewer application and a command line tool for accessing camera controls.

Install the video4linux support libraries:

```
sudo apt-get install libv4l-dev v4l-utils
```

Tips and Troubleshooting

- Recommended IDE: We recommend using Qt Creator, which is already installed.
- Acquire images with high resolution or high frame rate (image size > 2 MB): See Changing memory size reserved for CMA.
- You always need root privileges for cross-compiling from a Linux PC and for compiling on the embedded board.
- For cross-compiling from a host PC with Ubuntu, we recommend Linaro's cross-toolchain arm-linux-gnueabihf.

Useful commands for SD card handling

```
# List available device its device nodes
ls /dev/sd*
# Alternative command
ls /dev/mmc*

# Before removing the SD card, unmount its device nodes, for example sdh1 and sdh2
sudo umount /dev/sdh1
sudo umount /dev/sdh2
```

Table 13: Commands for SD card handling