

Application Note

AM6xA ISP Tuning Guide



Jianzhong Xu, Gang Hua, and Chau Le

Sitara MPU

ABSTRACT

This application report describes the work flow of tuning the ISP on TI's AM6xA vision processor family for a raw camera in order to obtain optimal image quality. The tuning procedure provided in this report is accompanied with examples using the IMX219 sensor and the AM62A Starter Kit EVM.

Table of Contents

| | |
|---|-----------|
| 1 Introduction | 2 |
| 2 Tuning Overview | 2 |
| 3 Hardware Requirement | 3 |
| 4 Software Requirement | 3 |
| 4.1 Processor SDK Linux | 3 |
| 4.2 TI's Reference Imaging Software | 3 |
| 4.3 ISP Tuning Tool | 4 |
| 5 Sensor Software Development and Integration | 4 |
| 5.1 Adding Sensor Driver to SDK | 4 |
| 5.2 Updating GStreamer Plugins to Support the Sensor | 4 |
| 6 Tuning Procedure | 6 |
| 6.1 Verify Functional Operation of Camera Capturing | 6 |
| 6.2 Enable Camera Streaming with Initial VPAC Configuration | 6 |
| 6.3 Adjust Camera Mounting | 8 |
| 6.4 Capture Raw Images and Perform Basic Tuning | 8 |
| 6.5 Perform Fine Tuning | 17 |
| 7 Summary | 17 |

Trademarks

All trademarks are the property of their respective owners.

1 Introduction

The AM6xA vision processors have a hardware accelerated Image Signal Processor (ISP) which is also referred as the Vision Pre-processing Accelerator (VPAC). With configurable image processing parameters, VPAC is designed to support a wide variety of raw camera modules (a typical raw camera module includes a lens, a filter, a raw image sensor, and sometimes a serializer). To obtain optimal image quality for a specific raw camera module at run-time, the parameters of VPAC needs to be computed and then applied to process the raw sensor images frame by frame. In order to achieve that, optimal VPAC parameters are typically prepared by engineers in an imaging lab under various controlled lighting conditions. Then at run-time, the prepared parameters are referenced and interpolated to fit the run-time lighting environment with the help of software imaging algorithms of Auto Exposure (AE), Auto White Balance (AWB), and dynamic ISP parameter control. The procedure of preparing the optimal VPAC parameters in an imaging lab is referred as ISP tuning in this application report.

The ISP tuning procedure described in this report applies to all SoCs in the AM6xA vision processor family, including AM62A, AM68A, and AM69A. Examples using the AM62A Starter Kit EVM are provided in the report.

For technical details of the ISP (VPAC) on a specific system-on-chip (SoC), see the Technical Reference Manual (TRM) of that SoC.

2 Tuning Overview

The ISP (VPAC) on the AM6xA family SoC is configured through the Dynamic Camera Configuration (DCC) binary files. In Linux based applications, these binary files are provided to VPAC through the popular GStreamer pipeline. The processing blocks of VPAC are encapsulated by the GStreamer pipeline elements (tiovxisp, tiovxldc, tiovxmultiscaler) while all configurable parameters of VPAC are provided as properties. For example, the following GStreamer pipeline performs video streaming from an IMX219 camera to an HDMI display, using VPAC to process the raw images before sending them to the display:

```
gst-launch-1.0 v4l2src device=/dev/video2 io-mode=dmabuf-import ! \
video/x-bayer, width=1920, height=1080, framerate=30/1, format=rggb10 ! \
tiovxisp sink 0::device=/dev/v4l-subdev2 \
sensor-name="SENSOR_SONY_IMX219_RPI" \
dcc-isp-file=/opt/imaging/imx219/dcc_viss_10b.bin \
sink 0::dcc-2a-file=/opt/imaging/imx219/dcc_2a_10b.bin format-msb=9 ! \
video/x-raw, format=NV12, width=1920, height=1080, framerate=30/1 ! \
kmssink driver-name=tidss sync=false
```

In the above pipeline, GStreamer element **tiovxisp** includes VPAC hardware and TI's reference software for imaging algorithms of AE, AWB and ISP parameter control. The VPAC configurations for IMX219 are provided through the following two binary files which are among the properties of **tiovxisp**:

- dcc-isp-file=/opt/imaging/imx219/dcc_viss_10b.bin for tuned ISP parameters
- dcc-2a-file=/opt/imaging/imx219/dcc_2a_10b.bin for AE and AWB algorithm calibration information

These binary files are the output of the ISP tuning, and are also referred as the Dynamic Camera Configuration (DCC) profiles.

As an overview, the ISP tuning procedure includes the following steps (using TI's reference imaging software and tuning tool as an example):

1. **Hardware setup**: Prepare and set up all necessary hardware equipment.
2. **Software setup**: Download and install all necessary software components.
3. **Sensor software development and integration**: Ensure the camera sensor driver is properly integrated in the system and can capture raw images. Add support for this sensor to GStreamer plugins.
4. **Generating Initial ISP configuration**: Generate an initial DCC profile for VPAC configuration. This configuration can enable video streaming with minimum image quality so that next step can be performed.
5. **Adjust camera mounting**: Run live video streaming using the DCC profile generated in the previous step and adjust camera module mounting to ensure images are captured with good positioning, focus, lighting, and so on.
6. **ISP basic tuning**: Capture raw images and perform basic tuning in order to achieve 70%~80% of optimal image quality. Replace the initial DCC profile with new one generated at the end of tuning.
7. **ISP fine tuning**: Run live streaming again with the new DCC profile generated in the previous step. Based on image quality, identify what needs to be improved and perform additional fine tuning if necessary.

This application note uses the AM62A Starter Kit EVM and an IMX219 camera to demonstrate the above listed tuning steps. The principles and procedure of the tuning apply to any custom board and raw camera.

3 Hardware Requirement

The necessary hardware equipment to perform ISP tuning for a raw camera includes:

- [AM62A Starter Kit EVM](#) (SK EVM) or a custom board with the AM62A SOC. For the complete EVM setup, see the [AM62A SK EVM Quick Start Guide](#).
- The camera and all necessary accessories to connect the camera to the AM62A SK EVM. For how to connect an IMX219 camera to the EVM, see the [AM62A Academy](#). From the Academy main page, go to "Evaluating Linux" -> "Tour of TI Linux" -> "Camera".
- A light box:
 - With calibrated light sources, typically ranging from 2700K to 6500K at least
 - More light sources are needed if wider range of color temperatures is required
- An Xrite Color Checker Chart. Different chart sizes may be needed depending on the setup.
- A light meter and/or a chroma meter for monitoring and recording lighting conditions.
- Tripods or mounts as needed to hold the camera securely at a desired position. [Figure 3-1](#) shows an example of such setup.



Figure 3-1. Tuning Hardware and Equipment Setup

4 Software Requirement

4.1 Processor SDK Linux

Download the [Processor Software Development Kit \(SDK\) Linux for AM62A](#).

To boot the EVM to Linux, follow the [AM62A SK EVM Quick Start Guide](#).

4.2 TI's Reference Imaging Software

Clone the imaging software from <https://git.ti.com/cgit/processor-sdk/imaging/> to a Linux host PC:

```
$ git clone git://git.ti.com/processor-sdk/imaging.git --branch main
```

This repo has TI's reference imaging algorithms for AE, AWB, and ISP control or Dynamic Camera Configuration (DCC). It also contains Python scripts that will be needed in the tuning process.

4.3 ISP Tuning Tool

ISP tuning tools and imaging algorithms (for example, AE/AWB/ISP control) are available from several imaging 3rd parties who may support customers for production. TI's TDA4/AM6xA DCC tuning tool and imaging algorithms are also available for design references, and are used as an example in this report. Please contact your local TI FAE to get this tool. Tuning tools from TI's 3rd parties can be used as well in similar ways, but not covered in this report.

5 Sensor Software Development and Integration

In order to tune the ISP for the target sensor using the AM62A SK EVM and Linux SDK, software needs to be developed in order to support the sensor in the SDK. Sensor driver implementation is beyond the scope of this report. The following sections describe how to add the sensor driver to the SDK as well as how to modify the GStreamer plugins to support the sensor.

5.1 Adding Sensor Driver to SDK

Assuming the driver for the sensor has already been implemented, refer to the [AM62A Academy](#) for how to add the sensor driver to the SDK. From the Academy main page, go to "Develop Linux on TI EVM" -> "Use Device Drivers" -> "Use Camera" -> "Enable a New CSI-2 Sensor".

5.2 Updating GStreamer Plugins to Support the Sensor

In addition to adding the sensor driver to the SDK, the GStreamer plugins need to be updated to support the sensor. This involves modifying and rebuilding two components in the target file system on the EVM.

5.2.1 Update TIOVX Modules

5.2.1.1 Source Code Change

In the target file system on the EVM, go to /opt/edgeai-tiovx-modules/src/tiovx_sensor_module.c, and add the sensor's name and id to the list in function tiovx_init_sensor(). For example, IMX219 has the following:

```
// This line defines a new TIOVX camera sensor string ID for IMX219
else if(strcmp(sensorObj->sensor_name, "SENSOR_SONY_IMX219_RPI") == 0)
{ // This line assigns a new numeric DCC ID to IMX219
  sensorObj->sensorParams.dccId=219;
}
```

Both the sensor name and the DCC ID for the IMX219 camera are defined here. The exact name string "SENSOR_SONY_IMX219_RPI" shall be used in the GStreamer pipeline and the DCC sensor ID (219) is used by the tuning tool in future steps.

5.2.1.2 Rebuild Modules

After finishing the source code change, rebuild and reinstall the modules by running script /opt/edgeai-gst-apps/scripts/install_tiovx_modules.sh.

5.2.2 Update GStreamer Plugins

5.2.2.1 Source Code Change

In the target file system on the EVM, go to /opt/edgeai-gst-plugins/ext/tiovx/gsttiovxisp.c, and add the sensor's name to the list (IMX219 shown below as an example):

```
g_object_class_install_property (gobject_class, PROP_SENSOR_NAME,
  g_param_spec_string ("sensor-name", "Sensor name",
    "TIOVX camera sensor string ID. Below are the supported sensors\n"
    "SENSOR_SONY_IMX219_RPI\n"
```

Then add a function to configure the auto exposure parameters for the sensor. For example, IMX219 has the following function.

```
static int32_t get_imx219_ae_dyn_params (IssAeDynamicParams *p_ae_dynPrms)
{
    int32_t status = -1;
    uint8_t count = 0;

    g_return_val_if_fail (p_ae_dynPrms, status);

    p_ae_dynPrms->targetBrightnessRange.min = 40; // min brightness on 0~255 scale
    p_ae_dynPrms->targetBrightnessRange.max = 50; // max brightness on 0~255 scale
    p_ae_dynPrms->targetBrightness = 45; // target brightness
    p_ae_dynPrms->threshold = 5; // difference between target and min/max
    p_ae_dynPrms->enableBlc = 0; // not used

    p_ae_dynPrms->exposureTimeStepSize = 1; // step size for adjusting exposure time in microseconds
    p_ae_dynPrms->exposureTimeRange[count].min = 100; // min exposure time in microseconds
    p_ae_dynPrms->exposureTimeRange[count].max = 33333; // max exposure time in microseconds

    p_ae_dynPrms->analogGainRange[count].min = 1024; // min analog gain (1024 is for 1.0x gain)
    p_ae_dynPrms->analogGainRange[count].max = 8192; // max analog gain (8192 is for 8.0x gain)
    p_ae_dynPrms->digitalGainRange[count].min = 256; // digital gain is not used
    p_ae_dynPrms->digitalGainRange[count].max = 256; // digital gain is not used
    count++;

    p_ae_dynPrms->numAeDynParams = count;
    status = 0;
    return status;
}
```

Finally, add the exposure time and analog gain mappings. For example, IMX219 has the following mapping:

```
if (g_strcmp0 (self->sensor_name, "SENSOR_SONY_IMX219_RPI") == 0)
{
    double multiplier = 0;
    // map AE output exposure time in microseconds to the exposure register value
    *exposure_time_mapped = (1080 * exposure_time / 33);

    // map AE output analog gain in Q10 integer format to the analog gain register value
    multiplier = analog_gain / 1024.0;
    *analog_gain_mapped = 256.0 - 256.0 / multiplier;
}
```

5.2.2.2 Rebuild Plugins

After finishing the source code change, rebuild and reinstall the plugins by running script /opt/edgeai-gst-apps/scripts/install_gst_plugins.sh.

5.2.2.3 Verify New Sensor in GStreamer Plugin

After all the above changes, verify the new sensor name is listed in the properties of GStreamer plugin tiiovxisp. For example, IMX219 is shown as below:

```
root@am62axx-evm:~# gst-inspect-1.0 tiiovxisp
...
  sensor-name : TIOVX camera sensor string ID. Below are the supported sensors
                SENSOR_SONY_IMX219_RPI
```

6 Tuning Procedure

This section provides the detailed tuning procedure.

6.1 Verify Functional Operation of Camera Capturing

Assume the camera driver has been integrated into the SDK, and the AM62A SK EVM boots to Linux and can probe the camera. Verify that both "v4l2-ctl" and "media-ctl" commands show expected output as below (with IMX219 as an example):

```
root@am62axx-evm:~# v4l2-ctl --list-devices
j721e-csi2rx (platform:30102000.ticsi2rx):
    /dev/video2
    /dev/video3
...
    /dev/media0

root@am62axx-evm:~# media-ctl -d /dev/media0 -p | grep imx219
    <- "imx219 4-0010":0 [ENABLED,IMMUTABLE]
- entity 13: imx219 4-0010 (1 pad, 1 link, 0 route)
```

Note

Note that "4-0010" in "media-ctl" output is the I2C bus address for the sensor and it may be different for a different SDK release.

Then verify that the camera can be configured to a certain format and raw images can be captured using a GStreamer pipeline. Below is an example, assuming "4-0010" is what's shown by the "media-ctl -p" command as above:

```
root@am62axx-evm:~# media-ctl -V '"imx219 4-0010":0 [fmt:SRGB10_1X10/1920x1080 field:none]'
```

```
root@am62axx-evm:~# gst-launch-1.0 -v v4l2src num-buffers=5 device=/dev/video2 io-mode=dmabuf ! \
video/x-bayer, width=1920, height=1080, framerate=30/1, format=rggb10 ! \
multifilesink location="imx219-image-%d.raw"
```

The captured raw images are in pure Bayer pattern array format (RGGB for IMX219 sensor) without any header or compression. These raw images can be displayed by a raw image viewer or other tools such as ffmpeg. At this stage, the raw images may be either overexposed or underexposed since the default exposure time and gain in the sensor may not match the lighting environment where these images are captured.

6.2 Enable Camera Streaming with Initial VPAC Configuration

After verifying that the camera and its driver work functionally and the GStreamer plugins for VPAC supports the new sensor, this step is to run live streaming with an initial VPAC configuration. The purpose of this step is to make sure that the GStreamer plugins for VPAC work properly. The live streaming will also make next steps easier, for example, for adjusting camera mounting position.

The initial DCC binaries are automatically generated by Python scripts so that both VPAC and Auto Exposure (AE) program are set up properly for the sensor resolution and color format. [TI's reference imaging software](#) provides necessary tools to generate an initial VPAC configuration (or DCC profile), as described in sections below.

6.2.1 Generate Configuration Files

Go to TI's reference imaging software, `imaging/tools/default_DCC_profile_gen/configs`, and create a configuration file for default camera properties. Existing configuration files in this folder can be references.

The following parameters need to be specified in this configuration file:

- **Raw image format information**
 - WIDTH: sensor image width
 - HEIGHT: sensor image height
 - BIT_DEPTH: sensor pixel bit depth
 - BAYER_PATTERN: Bayer pattern, where 0=RGGB; 1=GRBG; 2=GBRG; 3=BGGR

Note

Note that the above configuration information must match what's used in the media control command for configuring the camera format. For example, if the media control command for the IMX219 camera uses "[fmt:SRGGB10_1X10/1920x1080 field: none]", then the following values shall be used in this configuration file:

- WIDTH 1920
- HEIGHT 1080
- BIT_DEPTH 10
- BAYER_PATTERN 0
- **Camera sensor information**
 - SENSOR_ID: DCC ID of the sensor (**this must match the hard coded "dccID" value in tiouv sensor_module.c**, as described in [updating GStreamer plugins for the sensor](#))
 - SENSOR_NAME: name of the sensor (this is for information only and not used by the tool)
 - SENSOR_DCC_NAME: corresponding DCC name (this is for information only and not used by the tool)
- **XML output folder**
 - PRJ_DIR: folder to store the generated .xml files. This folder must be under imaging/sensor_drv/src.

Once the configuration file is created, run the Python script `ctt_def_xml_gen.py` in `imaging/tools/default_DCC_profile_gen/scripts` as below:

```
imaging/tools/default_DCC_profile_gen/scripts$ python ctt_def_xml_gen.py ../configs/<configuration file>
```

The generated xml files will be in folder `dcc_xmls` under the path specified by `PRJ_DIR` in the configuration file. This folder contains two sets of .xml files, with one set for linear mode and the other one for WDR mode. A script file "generate_dcc.sh" will also be generated in each xml folder. For example, below is the content of the xml folder for linear mode (for IMX219):

```
$PRJ_DIR/dcc_xmls/linear$ ls -l
generate_dcc.sh
imx219_awb_alg_ti3_tuning.xml
imx219_cfa_dcc.xml
imx219_h3a_aewb_dcc.xml
imx219_h3a_mux_luts_dcc.xml
imx219_linear_decompand_dcc.xml
imx219_mesh_ldc_dcc.xml
imx219_rgb2rgb_dcc.xml
imx219_viss_blc.xml
imx219_viss_nsf4.xml
```

6.2.2 Generate DCC Binary Files

Go to the folder containing the .xml files generated in previous step, either linear or wdr, and run script `generate_dcc.sh` in the folder, as shown below:

```
$PRJ_DIR/dcc_xmls/linear$ chmod +x ./generate_dcc.sh
$PRJ_DIR/dcc_xmls/linear$ ./generate_dcc.sh
```

This script generates DCC binary files for default configuration in folder `dcc_bins` under the path specified by `PRJ_DIR`.

6.2.3 Stream Video with the Initial Configuration

Copy the DCC binary files generated in the previous step to the file system on the AM62A SK EVM, in folder `/opt/imaging/<camera>`. For example, IMX219 has at least the following pre-built binary files in the SDK:

```
root@am62axx-evm:~# ls /opt/imaging/imx219
dcc_2a_10b.bin dcc_viss_10b.bin ...
```


Then use "media-ctl" command to set the camera format consistent with the properties in the initial configuration file, and run live streaming:

```
root@am62axx-evm:~# media-ctl -V '"imx219 4-0010":0 [fmt:SRGB10_1X10/1920x1080 field:none]'
```

```
root@am62axx-evm:~# gst-launch-1.0 v4l2src device=/dev/video2 io-mode=dmabuf-import ! \
video/x-bayer, width=1920, height=1080, framerate=30/1, format=rggb10 ! \
tiovxisp sink_0::device=/dev/v4l-subdev2 \
sensor-name="SENSOR_SONY_IMX219_RPI" \
dcc-isp-file=/opt/imaging/imx219/dcc_viss_10b.bin \
sink_0::dcc-2a-file=/opt/imaging/imx219/dcc_2a_10b.bin format-msb=9 ! \
video/x-raw, format=NV12, width=1920, height=1080, framerate=30/1 ! \
kmssink driver-name=tidss sync=false
```

Even though the image quality has not been fully tuned when using the initial VPAC configuration, this can validate that the GStreamer plugins for VPAC are working properly for the camera. The output video shall give correct output color (although not yet tuned) and properly exposed video. This can also make it easier to adjust camera mounting in next step.

6.3 Adjust Camera Mounting

Set up the light box and color checker. Point the camera to the color checker while streaming. Adjust camera position to ensure the color checker is fit exactly into the camera's field of view (FOV).

For a certain resolution, the sensor driver may crop the image before sending it to the ISP. For example, at resolution 1920x1080, the IMX219 driver crops the image and the image looks zoomed in on the display. For this case, a color checker of smaller size may work better.

6.4 Capture Raw Images and Perform Basic Tuning

In general, ISP tuning requires users first to capture a set of raw and/or YUV images under controlled lighting conditions and use a tuning tool to adjust ISP parameters and AWB calibration. In this section, the TDA4/AM62A DCC tuning tool from TI is employed for illustrating the VPAC tuning process. Other tuning tools from TI 3rd parties for imaging may be available as well with similar features and even more advanced capabilities and production level support.

6.4.1 Launch the Tuning Tool and Create a Project

Locate and run the installed tuning tool executable in Windows, <installation folder>\bin\DCC.exe. Choose VPAC3L (AM62A) when prompted for the correct VPAC version. After the tool starts, create a project and enter raw image properties. [Figure 6-1](#) shows an example for IMX219 in 1080p streaming mode.

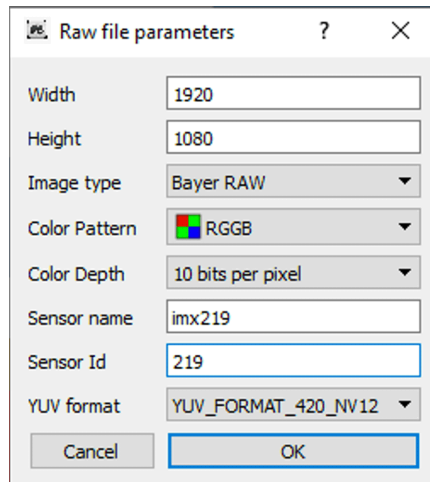


Figure 6-1. Sensor Raw Image Parameters for Tuning

Note

Note that the image properties must match with what will be used for capturing raw images, and the sensor ID must match with the hard coded value in the GStreamer plugins.

Once a new project is created, raw images can be previewed through the tuning tool. Capture a raw image of the color checker with good lighting condition using the command for [Section 6.1](#). Select this image file in the tuning tool's "Browse" window and the image should display in the "Preview window". For example, below is an IMX219 raw image at 1920x1080 with 10 bits per pixel, displayed in the tuning tool.

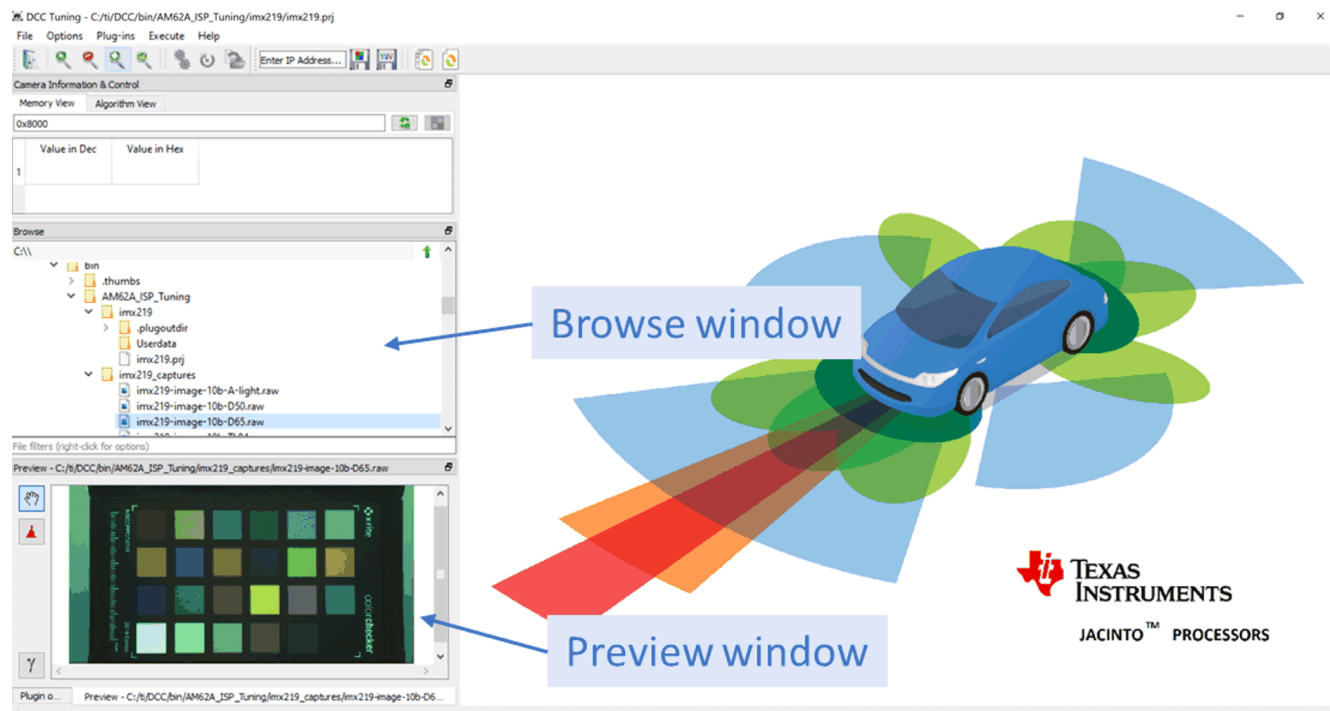


Figure 6-2. Image Preview by DCC Tuning Tool

Note

Only raw images captured with 10/12/16 bits per pixel can be previewed in the tuning tool. If captured with 8 bits per pixel, the image can be converted to 16-bit per pixel first and then previewed and used in the tuning tool.

Now that everything is ready, one can proceed to perform tuning, as described in following sections.

6.4.2 Tuning Order

The AM6xA ISP (VPAC) consists of multiple functional blocks. Raw images are processed by these blocks one after another. The tuning tool allows to tune the ISP blocks in independent groups, with each group containing one or more ISP blocks. The tuning groups are referred as plugins, as shown in the tuning tool menu "Plug-ins":

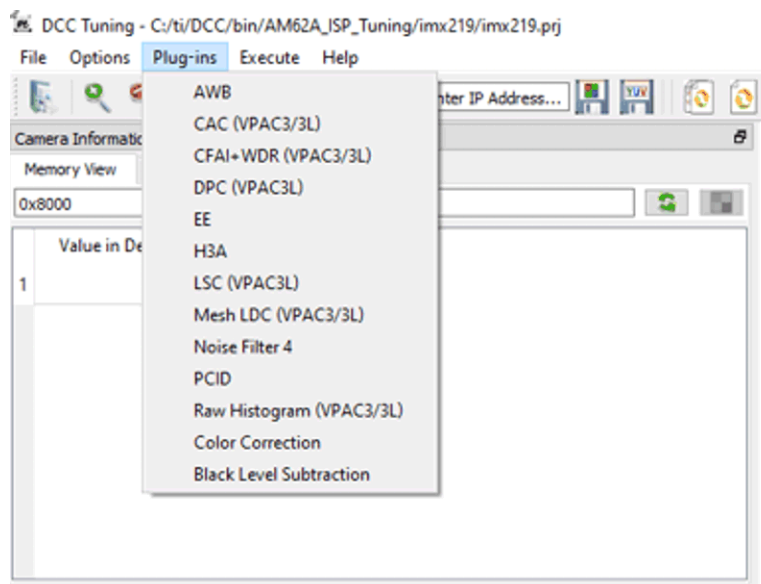


Figure 6-3. Plugins of DCC Tuning Tool

Note

Each plug-in has its own tuning guide which is available from Help -> Documentation drop down menu. These guides should be referenced during the tuning.

In general, these plugins should be tuned in the same order of ISP blocks when processing the raw image. Below is a recommended tuning order:

1. Black Level Subtraction: Data pedestal (sensor black level) is typically specified by the sensor driver and can be calculated here for verification.
2. H3A: Hardware 3A (Auto exposure, Auto focus, Auto white balance) statistics
3. PCID: Pattern Conversion and IR Demosaicing (only for RGB+IR sensors)
4. AWB: Auto White Balance
5. Color Correction
6. EE: Edge Enhancement
7. Noise Filter 4
8. Mesh LDC: Lens Distortion Correction
9. CFAI + WDR: Color Filter Array Interpolation + Wide Dynamic Range
10. LSC: Lens Shading Correction

After tuning each plug-in, a new set of XML files for VPAC configuration will be generated. These new XML files can replace those generated from the [initial configuration](#) to gradually improve image quality for live streaming.

In this application note, the IMX219 sensor in 1080p mode is used as an example for illustrating the tuning and calibration of black level, AWB, and color correction with the TDA4/AM6xA DCC tuning tool from TI. More details about tuning are available in the plug-in guides from the Help menu of the tuning tool. Other versions of ISP tuning tool from TI 3rd parties follow roughly the same procedure.

6.4.3 Black Level Subtraction

For linear sensors including the IMX219, the black level or pedestal shall be subtracted from the raw image pixels before applying any gains (for example, gains for white balance) later in the ISP. Even though the pedestal value is coded in the sensor driver, it is always good to measure its actual value for each sensor working mode supported by the sensor driver. For example, the IMX219 camera has a measured black level around 63 in 10-bit mode (shown in the figure below) and 16 in 8-bit mode.

Follow these steps to tune Black Level Subtraction for the target sensor:

1. Completely cover the camera lens and capture a black RAW image.
2. Choose Black Level Subtraction from the "Plug-ins" drop down menu.
3. Select the RAW image in the "Browse" window and the image should appear black in the "Preview" window.
4. Provide the raw image in the "RAW file" window and click "Process Plugin" as shown below.
5. The measured black level will be displayed in the "Advanced params" tab of the upper right window.

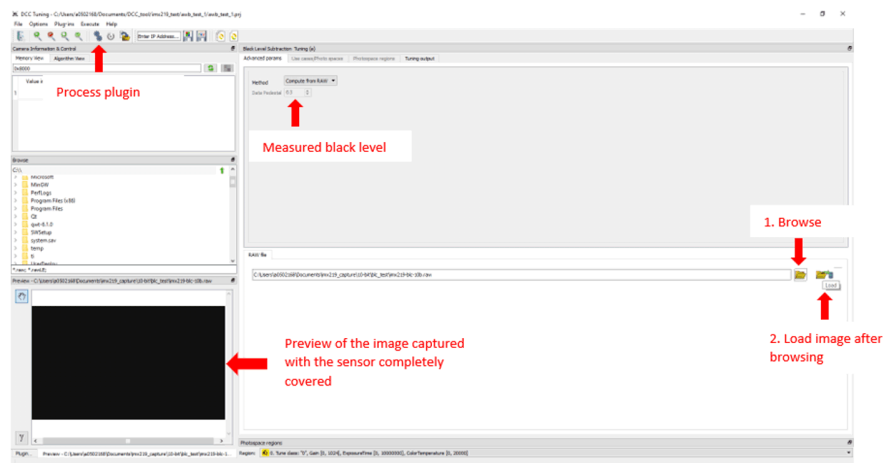


Figure 6-4. Black Level Subtraction Tuning

For WDR sensors, the black level subtraction is usually combined with WDR decompanding and re-compression to generate a single lookup table (LUT). This can be achieved by the "CFA + WDR" plugin in the tuning tool. Please refer to the user guide of the plugin for more details in case of a WDR sensor.

Once tuning for a plug-in is done, click the "Export DCC profile binary" button to generate the output XML files for this plugin. The XML files are located in folder ".plugoutdir\XML" under the project folder created in [Section 6.4.1](#). For Black Level Subtraction, there is only one output XML file: `imx219_viss_blc.xml`. Replace the same XML file generated from [Section 6.2.1](#). Then rerun the Python script to generate new DCC binary files as described in [Section 6.2.2](#). Use the newly generated DCC binaries to improve streaming quality in next step. This should be done after tuning each plug-in in the following sections.

To see the image quality improvement after tuning each plug-in, capture ISP-processed still images with newly generated DCC binary files. For example, use the following GStreamer pipeline with new binary files after tuning the Black Level Subtraction plug-in for IMX219:

```
gst-launch-1.0 -v v4l2src num-buffers=5 device=/dev/video2 io-mode=dmabuf-import ! \
video/x-bayer, width=1920, height=1080, framerate=30/1, format=rggb10 ! \
tiovxisp sink_0::device=/dev/v4l-subdev2 \
sensor-name="SENSOR_SONY_IMX219_RPI" \
dcc-isp-file=/opt/imaging/imx219/dcc_viss_10b.bin \
sink_0::dcc-2a-file=/opt/imaging/imx219/dcc_2a_10b.bin format-msb=9 ! \
video/x-raw, format=NV12, width=1920, height=1080, framerate=30/1 ! \
jpegenc ! multifilesink location="imx219-image-%d.jpg"
```

Figure 6-5 shows an example of the captured images before Black Level Subtraction tuning (using the initial configuration) and Figure 6-6 shows the image after Black Level Subtraction tuning (compare the appearance of black).



Figure 6-5. Image Before Black Level Subtraction



Figure 6-6. Image After Black Level Subtraction

6.4.4 Hardware 3A (H3A)

H3A is a hardware IP block for collecting image statistics for Auto Exposure (AE), Auto White Balance (AWB), and Auto Focus (AF) algorithms. For fixed focus cameras such as IMX219, only AE and AWB are relevant. The "ctt_def_xml_gen.py" script used in [generating initial configuration](#) already configures H3A properly for AE and AWB algorithms. Therefore, no additional tuning steps are needed for H3A for fixed focus cameras.

Note

The Python script takes BIT_DEPTH as an input parameter for IMX219 such that H3A/AE/AWB can be programmed properly. Therefore, this parameter must be set correctly.

The Python script configures the sensor/H3A data flow for linear sensors in the following way:

1. Assume a 0 black level by default (in the "Black Level Subtraction" step above, the actual black level must be measured and updated)
2. Shift the input pixels to the MSBs of the 16-bit ISP internal format given sensor bit depth
3. Send the top (up to) 10 bits of the linear sensor pixels to H3A
4. Configure H3A according to the given sensor resolution
5. Provide the same H3A configuration to AE/AWB algorithms so that AE/AWB can work properly

In case when fine adjustment of H3A settings is required, please follow the SoC's TRM and the user guide in tuning tool.

6.4.5 Auto White Balance (AWB)

6.4.5.1 Capture Raw Images for Different Lighting Conditions

To tune the AWB, raw images need to be captured at different lighting conditions. These images will be used again in next step of color correction tuning. Below are summarized instructions for capturing the required images (please refer to the AWB plugin guide for detailed information):

- Set up lighting conditions to: D65, D50, TL84, and A Light (one at a time)
- Place color checker chart in upright position and center of the camera FOV in the light box
 - Make sure brown patch is at the upper left corner on the first row and black patch is at the lower right corner on the last row
- Capture well exposed raw images for each lighting condition
 - Wait until automatic exposure adjustment stabilizes (live video output on monitor shall be bright enough without apparent saturation/clipping on the white patch. This is usually done in a few seconds.)
 - Run live video streaming with the initial configuration
 - Raw images may be captured by either of the following two ways
- Stop the GStreamer pipeline for streaming. Then run a new GStreamer pipeline to capture a raw image (sensor exposure setting remains unchanged between GStreamer runs).

- If AM62A EVM and PC are connected to the same Ethernet, tuning tool may also capture raw images from EVM directly over Ethernet. From the tool bar of the DCC tuning tool, enter the EVM's IP address and capture raw images as shown below. For more details, follow the tuning tool user's guide.



Figure 6-7. Live Capture Feature of DCC Tuning Tool

Figure 6-8 through Figure 6-11 are examples of well exposed captures for all above mentioned lighting conditions (note the color differences caused by the color of lighting):

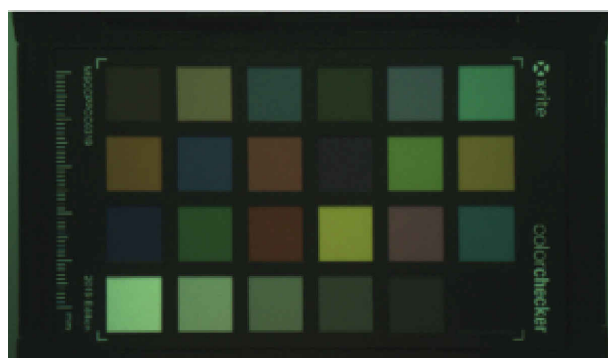


Figure 6-8. Color Chart Image Captured with D50 Lighting

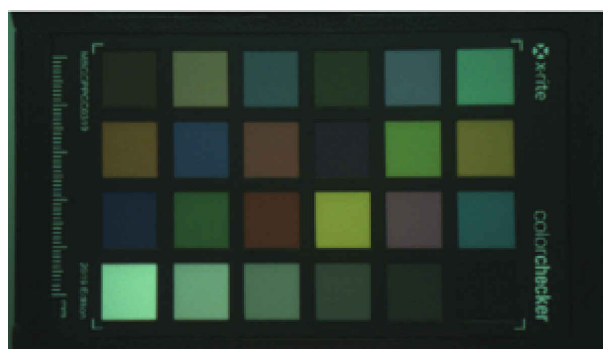


Figure 6-9. Color Chart Image Captured with D65 Lighting

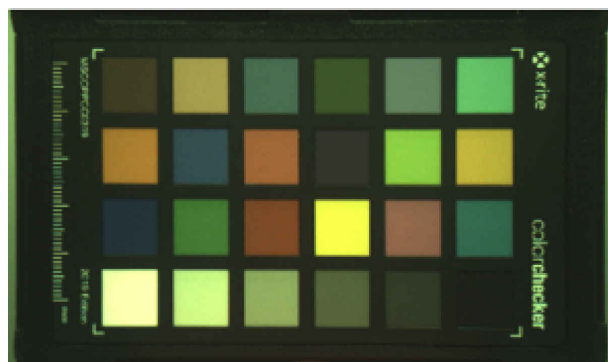


Figure 6-10. Color Chart Image Captured with TL84 Lighting

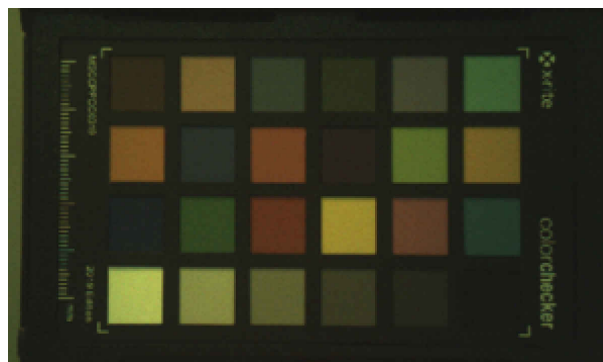


Figure 6-11. Color Chart Image Captured with A-light

6.4.5.2 Tuning AWB

After raw images are captured, start the AWB tuning from the "Plug-ins" drop down menu. Import the raw images one by one in the "Reference files" tab (see the AWB plugin guide for details) and enter the parameter values for each image:

- Color Temperature: this must be what is used when capturing the raw image. For example, the value should be 6500 for D65 lighting condition.
- Exposure, Gain, and Aperture: these values are not really used for AWB and therefore can be ignored.
- Black Level: this must be the pedestal value that was measured in the Black Level Subtraction plugin. In this example, IMX219 has a measured black level of 63 in 10-bit mode.

Please pay special attention when selecting the corners of the color checker chart:

- Starting with the upper left corner, click on the four corners of the color checker chart in clock-wise order.
- After the four corners are selected, the tool automatically identifies the 24 patches and display the selection of each patch as shown below.

Figure 6-12 through Figure 6-13 show an example of importing one raw image for AWB tuning.

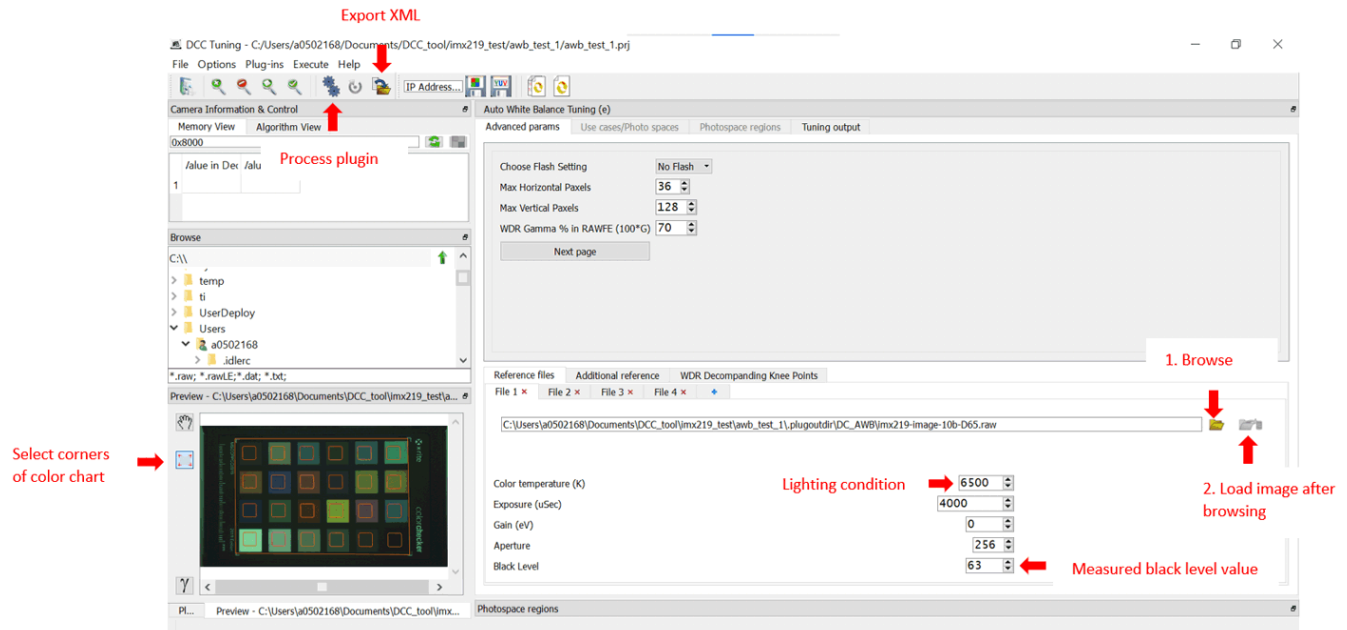


Figure 6-12. Auto White Balance Tuning

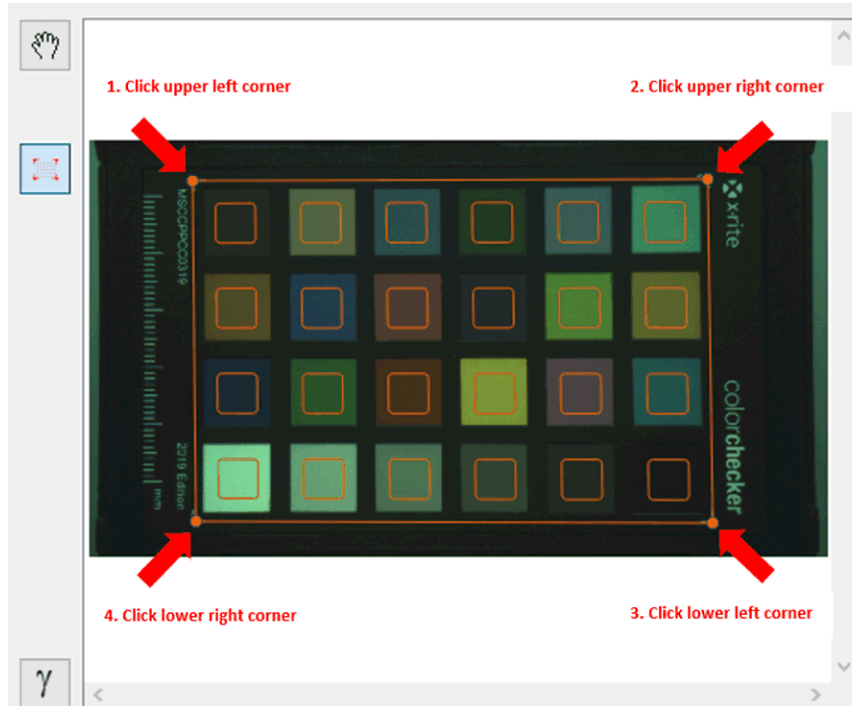


Figure 6-13. Choosing Corners of Color Checker Chart

After importing all raw images, follow the AWB plugin guide to do the tuning. If tuning is successful, reference Cb-Cr plot scheme will be displayed. Below is the result plot using the raw images shown above.

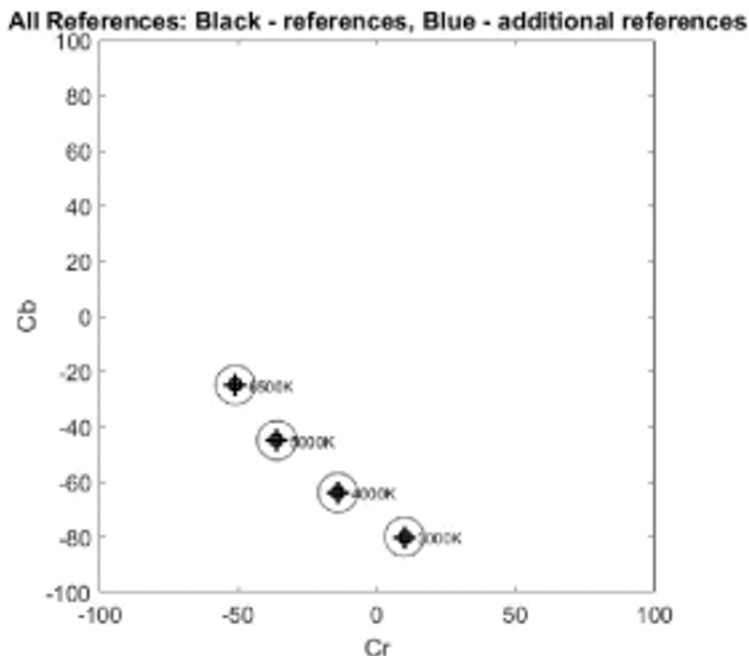


Figure 6-14. Auto White Balance Tuning Results

After tuning is done, generate new output XML files and new binary files as done in tuning Black Level Subtraction. Then use the newly generated DCC binary files to stream and capture. [Figure 6-15](#) and [Figure 6-16](#) show images captured before AWB tuning and after AWB tuning (all gray patches appear neutral in the image after AWB tuning).



Figure 6-15. Image Before Auto White Balance Tuning



Figure 6-16. Image After Auto White Balance Tuning

6.4.6 Color Correction

To get correct color output, the Color Correction plugin should be tuned. The raw images captured for AWB tuning can be used for tuning this plugin. Below is a summary of this tuning step. For more details, see the Color Correction plugin guide from the help menu.

- Load an input RAW image and optionally a reference image. The tuning tool provides a default reference image, but users may use a reference image of their choice.
- Enter the parameter values and select the four corners of the color checker chart as done in AWB tuning.
- Repeat the same process for the reference image.
- "Process" the plugin and "Export DCC profile binary" to generate XML and binary files, as shown in below figures.



As done in AWB tuning, capture images using the newly generated DCC binaries. [Figure 6-19](#) and [Figure 6-20](#) show images before and after Color Correction tuning (note the corrected color in the image after Color Correction tuning).



Figure 6-19. Image Before Color Correction Tuning



Figure 6-20. Image After Color Correction Tuning

6.5 Perform Fine Tuning

Once the Black Level Subtraction, Hardware 3A, Auto White Balance, and Color Correction are tuned properly, the ISP should achieve about 70~80% of the optimal image quality in light box lighting conditions for this IMX219 camera. Other plugins can be tuned additionally following each plugin's tuning guide provided in the DCC tuning tool. For example, following additional plugins can be tuned to achieve even better image quality:

- EE (Edge Enhancement) plugin for better sharpness of the output image
- Noise Filter 4 plugin for the suppression of noise in dark conditions
- Mesh LDC (Lens Distortion Correction) plugin for removing lens distortions
- CFAI + WDR (Color Filter Array Interpolation + Wide Dynamic Range) plugin for demosaicing and WDR sensors LSC
- (Lens Shading Correction) plugin for removing lens shading
- Pattern Conversion and IR Demosaicing (PCID) plugin for RGBIR 4x4 sensors

7 Summary

This application note describes the work flow of the AM6xA ISP tuning. In this version, the ISP is tuned step by step for an IMX219 camera to get the best output color using TI's AM62A Processor SDK, imaging software, and the DCC tuning tool.

Following content will be covered in future revisions:

- Tuning for other image quality factors using more plugins in TI DCC tuning tool
- Live tuning features of the TI DCC tuning tool
- ISP tuning using imaging algorithms and tuning tool from a 3rd party

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to [TI's Terms of Sale](#) or other applicable terms available either on [ti.com](#) or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2023, Texas Instruments Incorporated